Efficient Approaches to *k* Representative G-Skyline Queries

XU ZHOU and KENLI LI, Hunan University
ZHIBANG YANG, Changsha University
YUNJUN GAO, Zhejiang University
KEQIN LI, Hunan University and State University of New York

The G-Skyline (GSky) query is a powerful tool to analyze optimal groups in decision support. Compared with other group skyline queries, it releases users from providing an aggregate function. Besides, it can get much comprehensive results without overlooking some important results containing non-skylines. However, it is hard for the users to make sensible choices when facing so many results the GSky query returns, especially over a large, high-dimensional dataset or with a large group size. In this article, we investigate *k* representative G-Skyline (*k*GSky) queries to obtain a manageable size of optimal groups. The *k*GSky query can also inherit the advantage of the GSky query; its results are representative and diversified. Next, we propose three exact algorithms with novel techniques including an upper bound pruning, a grouping strategy, a layered optimum strategy, and a hybrid strategy to efficiently process the *k*GSky query. Consider these exact algorithms have high time complexity and the precise results are not necessary in many applications. We further develop two approximate algorithms to trade off some accuracy for efficiency. Extensive experiments on both real and synthetic datasets demonstrate the efficiency, scalability, and accuracy of the proposed algorithms.

CCS Concepts: • **Information systems** → *Data management systems*; *Database management system engines*; *Database query processing*; *Query operators*;

Additional Key Words and Phrases: Approximate algorithms, data management, group skyline query, Progressive algorithms

ACM Reference format:
Xu Zhou, Kenli Li, Zhibang Yang, Yunjun Gao, and Keqin Li. 2020. Efficient Approaches to *k* Representative G-Skyline Queries. *ACM Trans. Knowl. Discov. Data* 14, 5, Article 58 (July 2020), 27 pages.
https://doi.org/10.1145/3397503

This research was supported by the National Key R&D Programs of China (Grant No. 2018YFB1003401), the Programs of National Natural Science Foundation of China (Grant Nos. 61772182, 61802032, 61806077, 61902120, and 61872127), and the Key Area Research Program of Hunan (2019GK2091).

Authors' addresses: X. Zhou, College of Information Science and Engineering, Hunan University, Changsha, China; email: zhouxu2006@126.com; K. Li (corresponding author), College of Information Science and Engineering, Hunan university, Lushan Road (S), Yuelu District, Changsha, Hunan Province, China; email: lkl@hnu.edu.cn; Z. Yang, College of Computer Engineering and Applied Mathematics, Changsha University, No. 98, Hongshan Road, Kaifu District, Changsha, China; email: yangzb@hnu.edu.cn; Y. Gao, College of Computer Science, Zhejiang University, Hangzhou, China; email: gaoyj@zju.edu.cn; K. Li, Department of Computer Science, State University of New York, Science Hall 249, 1 Hawk Drive, New Paltz, New York, USA; email: lik@newpaltz.edu.

1556-4681/2020/07-ART58 $15.00
https://doi.org/10.1145/3397503

1 INTRODUCTION

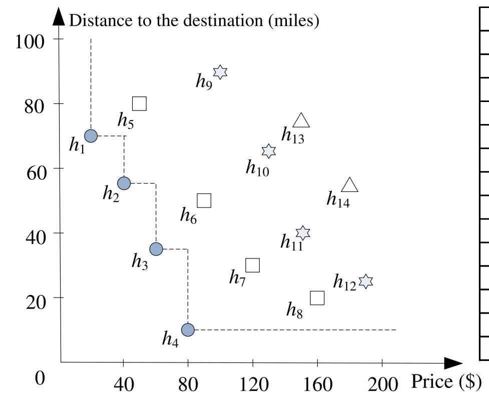
1.1 Background

The skyline query is a famous tool for the scenarios of multi-criteria optimization to filter out many incompetent objects [Zhao et al. 2017]. A point p' is uncompetitive with comparison to another point p if it is dominated by the point p. Here, p dominates p' if and only if p is not worse than p' in all the dimensions and is better than p' in at least one dimension [Börzsönyi et al. 2001]. The skyline query [Börzsönyi et al. 2001; Papadias et al. 2005; Lee et al. 2010] and its variants [Zhang et al. 2009; Gao et al. 2015b; Zhou et al. 2016b; Lu et al. 2011; Tao et al. 2009; Lin et al. 2007; Magnani et al. 2014] are useful to analyze individual points. However, they are inadequate to many real-world applications that call for optimal groups of points.

In Chung et al. [2013], Im and Park [2012], and [Zhang et al. 2014], group skyline queries were researched to find optimal groups not dominated by any other group of equal size. For given groups G and G', most of these group skyline queries first compute the aggregation points p and p' of G and G' separately. Then we have G dominates G' if p dominates p'. Here, the attribute values of the points p and p' are the aggregations over the points in G and G'. However, it is difficult for users to offer an appropriate aggregate function, and get all the optimal groups what the users expect [Liu et al. 2015]. To address this issue, Liu et al. [2015] proposed a new definition of group dominance which takes into account the dominance relationship between points of different groups. Under the new group dominance operator, they formulated the G-Skyline (GSky) query which can release users from specifying the aggregate function and can get more comprehensive results.

Consider a real-life application that a travel agency selects some hotels for cooperation. In Figure 1, it depicts a typical example with a hotel set $H = \{h_1, h_2, \ldots, h_{14}\}$ containing 14 candidate hotels (data points). We take into account the following two attributes: distance to the destination (distance for short) and price. Without loss of generality, the hotels cheap and close to the destination are preferred. As shown in Figure 1, the hotels h_1, h_2, h_3, and h_4 are skylines since they are not dominated by any other hotel according to the traditional dominance relationship [Börzsönyi et al. 2001]. The hotels h_5 to h_{14} are non-skylines because of being dominated by at least one hotel in Figure 1. For instance, the hotel h_5 is dominated by the hotel h_1 since it is more expensive than h_1 and its distance is farther than h_1. Assume that the travel agency needs to pick out three hotels. The right table in Figure 1 shows all the GSky query results (G-Skylines) as well as the non-skylines contained in each result. The hotel groups $\{h_1, h_2, h_3\}$, $\{h_1, h_2, h_4\}$, $\{h_1, h_3, h_4\}$, and $\{h_2, h_3, h_4\}$ merely consist of skylines. Furthermore, there are also many G-Skylines such as $\{h_1, h_2, h_5\}$ containing a hotel h_5 that is a non-skyline. This GSky $\{h_1, h_2, h_5\}$ is considered better to the travelers who use the price as the main criterion, e.g., students with low travel budget. This is because the hotels h_1, h_2, h_5 have the first three lowest prices. If the hotels h_1, h_2 are fully-booked, then the hotel h_5 is a good choice since it is the cheapest hotel that the traveler can book. The hotel groups, not shown in Figure 1, are not the GSky query results for being dominated by at least one hotel group of the same size. For instance, the hotel group $\{h_1, h_2, h_6\}$ is dominated by $\{h_1, h_2, h_3\}$ because h_6 is dominated by h_3. After getting all the G-Skylines, the travel agency can make a choice according to its personalized preference.

In addition to the above example of the travel agency, the GSky query can also offer powerful decision support in many other real-life applications. When a coach builds an NBA team of five players, the GSky query can help to find the teams not only consisting of skyline players but also including non-skyline players who are only dominated by another player in the same team [Wang et al. 2018]. For an advertising company, the G-Skyline query is also a useful tool to help select billboards to serve its advertisements [Wang et al. 2018].



Hotel	Price	Distance
h_1	20	70
h_2	40	55
h_3	60	35
h_4	80	10
h_5	50	80
h_6	90	50
h_7	120	30
h_8	160	20
h_9	100	90
h_{10}	130	65
h_{11}	150	40
h_{12}	190	25
h_{13}	150	75
h_{14}	180	55

G-Skylines	Non-Skylines
$\{h_1,h_2,h_3\}$	
$\{h_1,h_2,h_4\}$	
$\{h_1,h_3,h_4\}$	
$\{h_2,h_3,h_4\}$	
$\{h_1,h_2,h_5\}$	h_5
$\{h_3,h_4,h_6\}$	h_6
$\{h_1,h_4,h_7\}$	h_7
$\{h_2,h_4,h_7\}$	h_7
$\{h_3,h_4,h_7\}$	h_7
$\{h_1,h_4,h_8\}$	h_8
$\{h_2,h_4,h_8\}$	h_8
$\{h_3,h_4,h_8\}$	h_8
$\{h_4,h_7,h_8\}$	h_7,h_8
$\{h_4,h_8,h_{12}\}$	h_8,h_{12}

Fig. 1. The GSky query example of a hotel set.

As mentioned above, the GSky query is significant and practical for many applications that need to compute optimal groups. However, over datasets with large cardinality, high dimensionality, or large group sizes, the GSky query faces the problem of combinatorial explosion and the sizes of the GSky query results may become excessively large. As pointed out in Liu et al. [2015], the GSky query results consist of points within the first l skyline layers where l is the group size. Assume that there are n_l points in the first l skyline layers. There can be $\binom{n_l}{l}$ different GSky query results where $l \leq n_l$. For instance, suppose a travel agency needs to select 3 hotels from 50 candidate hotels where each candidate hotel is not dominated by any other one. By the GSky query, it reports $\binom{50}{3} = 19{,}600$ hotel groups. The travel agency has to compare all the hotel groups manually and make a decision. Apparently, large-scale results of the GSky query prevent the travel agency from making a quick and rational decision [Zhao et al. 2017].

1.2 Our Contributions

In this article, we firstly investigate a new variant of the GSky query, namely, k representative G-Skyline (kGSky), which can not only overcome the limitation of the GSky query but also inherit its advantages. The kGSky query has the nice properties of controllable number of results and the results have representativeness. Moreover, it inherits the advantage of the GSky query which is to get comprehensive and diversified results. It can also output some G-Skylines containing non-skylines. Second, we propose some pruning strategies to reduce the search space and present a grouping strategy, a hybrid strategy, and a layered optimization strategy to boost query performance. Finally, we develop approximate algorithms to trade off some accuracy for efficiency. The main contributions of our work are summarized as follows.

— We formulate the kGSky query to retrieve a manageable size of G-Skylines.
— We exploit the properties of the kGSky query and propose some pruning strategies to reduce the search space.
— We develop three exact algorithms for the kGSky query and two of them can report results progressively.
— We present approximate algorithms for the kGSky query which exchange accuracy for higher efficiency.
— We perform an extensive experimental study with both synthetic and real datasets.

The rest of the article is organized as follows. In Section 2, we review the related work. In Section 3, we introduce the kGSky query and its properties. In Section 4, we design exact algorithms for the *k*GSky query. In Section 5, we present two approximate algorithms. In Section 6, we evaluate the performance of the proposed algorithms by extensive experiments. In Section 7, we conclude the article and also expatiate the directions for future work.

2 RELATED WORK

In this section, we review the related work about representative skyline queries and group skyline queries.

2.1 Representative Skyline Queries

The traditional skyline queries always retrieve large-scale results [Magnani et al. 2014]. Inspired by this, many representative skyline queries were proposed. Papadias et al. [2005] presented a top *k* dominating query to find *k* points with the largest dominance sizes. This query can release users from specifying a ranking function and can return a manageable size of results. Lin et al. [2007] researched the problem of selecting *k* skylines such that the number of points dominated by at least one of these *k* skylines is maximized. Tao et al. [2009] presented a distance-based skyline query which aims to minimize the distance between a non-representative skyline and its nearest representative one. Lu et al. [2011] were concerned the case when the actual cardinality of skyline results is less than the desired cardinality *k*. The above representative skyline queries only highlight some features which are stability, scale invariance, diversification of the results, and partial knowledge of the record scoring function [Magnani et al. 2014]. Magnani et al. [2014] investigated the representative skyline queries with taking into account both the significance and diversity of results. In Zhou et al. [2016a], we formulated a top *k* favorite probabilistic products (TFPP) query which can help users find *k* products meeting the preferences of different customers. After that, in Zhou et al. [2019], we investigated the constrained optimal product combination problem under price promotion.

2.2 Group Skyline Queries

Group skyline queries are useful tools for many applications to compute optimal groups of points. Most of the group skyline queries compute the optimal groups based on the dominance relationship between corresponding aggregate-based points [Chung et al. 2013; Im and Park 2012; Magnani and Assent 2013; Zhang et al. 2014; Wan et al. 2009]. However, it is hard for users to specify an appropriate aggregate function and it may lose significant results that include non-skylines [Liu et al. 2015]. To resolve this problem, Liu et al. [2015] proposed the pareto group-based skyline (GSky) query. Recently, Wang et al. [2018] developed the minimum GSky support structure for the GSky query which is without redundant points.

The literature [Wang et al. 2018; Zhou et al. 2018] point out that the GSky query faces the problem of a prohibitive large number of query results. This will make the users overwhelmed and have a negative impact on the decision-making process. The work in Zhu et al. [2019] is significant to release the burden on users for processing so many G-Skylines manually. In Zhu et al. [2019], Zhu et al. researched the top *k* dominating queries on skyline groups where the score of each group is measured by the number of points it dominates. Besides, Yu et al. [2019] presented a *k* RG-Skyline query which aims to compute *k* representative G-Skylines. Here all the G-Skylines are clustered into *k* clusters, and the *k* cluster centers are returned as final results.

Similarly to Zhu et al. [2019], our *k*GSky query also computes *k* representative G-Skylines. However, different from Zhu et al. [2019], we define the dominance size based on the volume of the dominated space which can reflect the dominance number to a certain degree and has satisfactory representativeness [Bai et al. 2016]. Furthermore, as well as static datasets, this definition is appropriate to data streams and frequently updated datasets [Bai et al. 2016].

3 THE kGSKY QUERY

In this section, we introduce some basic concepts and the presented kGSky query.

3.1 The GSky Query

Assume that we have a given dataset D with d dimensions. A point $p \in D$ is denoted as $<p[1], p[2], \ldots, p[d]>$ where $p[i]$ is the ith dimensional value of p for $1 \leq i \leq d$. Without loss of generality, the small value is preferred in each dimension. A point p is dominated by another point p', denoted as $p' \prec p$, if and only if for all i, $p'[i] \leq p[i]$ and for at least one i, $p'[i] < p[i]$ for $1 \leq i \leq d$ [Börzsönyi et al. 2001]. The points are called skylines if they are not dominated by any other point in the given dataset.

In the following, we introduce the basic concepts of the skyline layer and the group dominance operator.

Definition 3.1 (Skyline Layer). For a given dataset D, the ith skyline layer SL_i is a set of skylines in $D - \cup_{j=1}^{i-1} SL_j$, i.e., $SL_i = \{t \in D - \cup_{j=1}^{i-1} SL_j | \nexists t' \in D - \cup_{j=1}^{i-1} SL_j, t' \prec t\}$.

In the layered minimum dominance graph (LMDG) in Section 5.2, we organize the points as different skyline layers. As shown in Figure 3(b), the first skyline layer $SL_1 = \{h_1, h_2, h_3, h_4\}$ contains all the hotels that are not dominated by any other hotel, the second skyline layer $SL_2 = \{h_5, h_6, h_7, h_8\}$ only includes the hotels dominated by at least one hotel within SL_1. For the hotels within SL_3, it contains only one hotel h_{12} which are dominated by at least one hotel within SL_1 and SL_2, respectively. Specifically, h_{12} is dominated by $h_4 \in SL_1$ and $h_8 \in SL_2$.

Definition 3.2 (Group Dominance [Liu et al. 2015]). For given l-point groups $G, G' \subseteq D$, G g-dominates G', denoted as $G \prec_g G'$, if and only if there are two permutations of G and G', $G = \{p_1, p_2, \ldots, p_l\}$ and $G' = \{p'_1, p'_2, \ldots, p'_l\}$, satisfying $p_i \preceq p'_i$ for $1 \leq i \leq l$ and $p_i \prec p'_i$ for at least one i. Here, $p_i \preceq p'_i$ means that $p_i \prec p'_i$ or p_i is equal to p'_i.

Definition 3.3 (G-Skyline [Liu et al. 2015]). The l-point group G is called GSky if it is not g-dominated by another group of size l.

According to Definition 3.2, for two given groups $G, G' \subseteq D$, $G \prec_g G'$ if and only if there exist two permutations of G and G' satisfying $G \prec_g G'$. For instance, consider two hotel groups $G = \{h_1, h_2, h_3\}$ and $G' = \{h_2, h_3, h_5\}$ in Figure 1. By reordering the points within G and G', we have $G = \{h_1, h_2, h_3\}$ and $G' = \{h_5, h_2, h_3\}$. Since $h_1 \prec h_5$, it holds that $G \prec_g G'$ based on Definition 3.2. The group G is a GSky, as it is not g-dominated by any other group of size 3. Besides, the group G' dominated by G is not a GSky.

In Liu et al. [2015], Liu et al. proposed the GSky query for the first time. It aims to compute the G-Skylines that are not g-dominated by any other group of equal size [?]. Compared to the traditional group skyline query in Chung et al. [2013], Im and Park [2012], Magnani and Assent [2013], Zhang et al. [2014], and Wan et al. [2009], it gets comprehensive and diversified results which can meet the personalized preference of more users. Nevertheless, the GSky query faces the limitation of prohibitively large number of query results which makes the query results less informative. Besides, it brings a big challenge for users to make a good decision fast enough. To avoid this issue, the efficient way is to compute a manageable size of results with good representativeness. This is also the focus of this article.

3.2 The kGSky Query

In this section, we present the kGSky query to compute k representative G-Skylines with the highest dominance capacity.

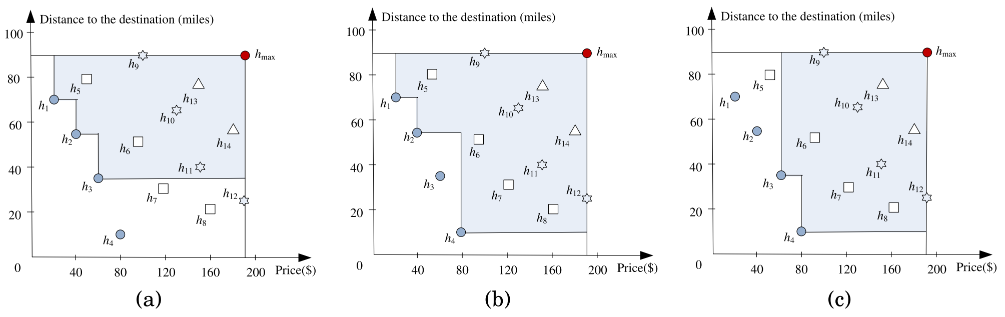


Fig. 2. Dominance space of G-Skylines. (a) $\{h_1, h_2, h_3\}$. (b) $\{h_1, h_2, h_4\}$. (c) $\{h_3, h_4, h_6\}$.

It is generally accepted that a query result with large dominance capacity has high representativeness [Bai et al. 2016]. In this article, we also choose to measure the representativeness of the G-Skylines by their dominance capacity.

At present, there are two popular definitions of the dominance capacity. In Miao et al. [2015], Tiakas et al. [2016], Amagata et al. [2018], Kontaki et al. [2011], and Santoso and Chiu [2014], the dominance capacity is defined based on the number of points dominated. Moreover, in Bai et al. [2016], the authors present a new definition of the dominance capacity, namely, dominance size, on the basis of the volume of the dominated space. As introduced in Bai et al. [2016], this definition can reflect the dominance number to a certain degree and has satisfactory representativeness. Furthermore, this definition is more general. As well as for static datasets, this definition is appropriate to data streams and frequently updated datasets. Following the work in Bai et al. [2016], in the proposed kGSky query, we also compute the representative G-Skylines due to their dominance sizes.

Definition 3.4 (Maximum Boundary Point). For a given dataset D, the maximum boundary point is denoted as $p_{max} = <p_{max}[1], p_{max}[2], \ldots, p_{max}[d]>$ where $p_{max}[j] = \max_{p \in D} p[j]$ for $1 \leq j \leq d$.

Definition 3.5 (Dominance Space). For a given dataset D and a point p, the dominance space of the point p is denoted as $DomSp(p) = ([p[1], p_{max}[1]], [p[2], p_{max}[2]], \ldots, [p[d], p_{max}[d]])$ where p_{max} is the maximum boundary point. Moreover, for a point group G, its dominance space is the union of dominance space of all the points $p \in G$, i.e., $DomSp(G) = \bigcup_{p \in G} DomSp(p)$.

Figure 2 illustrates the dominance space (marked in blue) of the G-Skylines $\{h_1, h_2, h_3\}$, $\{h_1, h_2, h_4\}$, and $\{h_3, h_4, h_7\}$. The maximum boundary point $h_{max} = (190, 90)$. For the point $h_1 = (20, 70)$, we have $DomSp(h_1) = ([20, 190], [70, 90])$. As depicted in Figure 2(a), the dominance space of the GSky $\{h_1, h_2, h_3\}$ is $DomSp(\{h_1, h_2, h_3\}) = DomSp(h_1) \cup DomSp(h_2) \cup DomSp(h_3)$.

There may be overlapping of the dominance space of points within a group G. Consequently, we formulate an intersection dominance size to evaluate the overlapping space of points within G.

Definition 3.6 (Intersection Dominance Size). For a given dataset D and a group G, the intersection dominance size of the group G is defined as $IntSize(G) = \prod_{j=1}^{d} (p_{max}[j] - \max_{p \in G} p[j])$. Here, p_{max} is the maximum boundary point of the multiple dimensional space of the dataset D.

For the hotel group $\{h_1, h_2\}$ in Figure 1, we have $IntSize(\{h_1, h_2\}) = (190 - \max\{20, 40\}) \times (90 - \max\{70, 55\}) = 3{,}000$.

Table 1. Dominance Sizes of the G-Skylines Over the Dataset in Figure 1

G-Skyline G	$DomSize(G)$	G-Skyline G	$DomSize(G)$
$\{h_1, h_2, h_3\}$	8,250	$\{h_1, h_2, h_4\}$✓	10,600
$\{h_1, h_3, h_4\}$✓	10,700	$\{h_2, h_3, h_4\}$✓	10,600
$\{h_1, h_2, h_5\}$	5,650	$\{h_3, h_4, h_6\}$	9,900
$\{h_1, h_4, h_7\}$	10,000	$\{h_2, h_4, h_7\}$✓	10,200
$\{h_3, h_4, h_7\}$	9,900	$\{h_1, h_4, h_8\}$	10,000
$\{h_2, h_4, h_8\}$✓	10,200	$\{h_3, h_4, h_8\}$	9,900
$\{h_4, h_7, h_8\}$	8,800	$\{h_4, h_8, h_{12}\}$	8,800

Definition 3.7 (Dominance Size). For a given point p, its dominance size is computed as $DomSize(p) = \prod_{j=1}^{d}(p_{max}[j] - p[j])$. Besides, for a group G, its dominance size is:

$$DomSize(G) = \sum_{p \in G} DomSize(p) + \sum_{i=2}^{k}(-1)^{i-1} \sum_{G_i \in \{G' | G' \subseteq G, |G'|=i\}} IntSize(G_i). \quad (1)$$

Based on Definition 3.7, the kGSky query is formulated as follows.

Definition 3.8 (k Representative Group Skyline Query, kGSky). Given a dataset D, a group size l, and a parameter k, the kGSky query returns k representative G-Skylines G of size l with the largest dominance sizes. Here, for a GSky G, its dominance size is computed due to Equation (1). In case G-Skylines with the same dominance size are tie at rank k-th, only some of them are returned.

We consider the example in Figure 1 with $l = 3$. For the GSky $\{h_1, h_2, h_3\}$, we have:

$$\begin{aligned} &DomSize(\{h_1, h_2, h_3\}) \\ &= (DomSize(h_1) + DomSize(h_2) + DomSize(h_3)) \\ &\quad + (-1) \times (IntSize(\{h_1, h_2\}) + IntSize(\{h_1, h_3\}) + IntSize(\{h_2, h_3\})) \\ &\quad + (-1)^2 \times IntSize(\{h_1, h_2, h_3\}) = 8250. \end{aligned}$$

Similarly, we can get the dominance sizes of other G-Skylines as illustrated in Table 1. The G-Skylines with large dominance sizes have nice representativeness. For example, consider the hotel groups $\{h_1, h_2, h_4\}$ and $\{h_1, h_2, h_3\}$ as shown in Figure 2. The hotels in $\{h_1, h_2, h_4\}$ can well represent the hotels h_5 to h_{14} because they are dominated by at least one hotel within $\{h_1, h_2, h_4\}$. For the hotel group $\{h_1, h_2, h_3\}$, it can represent 7 hotels, $h_5, h_6, h_9, h_{10}, h_{11}, h_{13}, h_{14}$. Obviously, the dominance size of $\{h_1, h_2, h_4\}$ is larger than that of $\{h_1, h_2, h_3\}$, and it has nicer representativeness. Based on the dominance size, the hotel groups $\{h_2, h_3, h_4\}$, $\{h_1, h_3, h_4\}$, $\{h_1, h_2, h_4\}$, $\{h_2, h_4, h_7\}$, and $\{h_2, h_4, h_8\}$ with the highest dominance sizes are returned as the final results of the kGSky query.

The symbols frequently used in this article are shown in Table 2.

4 APPROACHES TO THE kGSKY QUERY

In this section, we propose three exact algorithms which introduce a grouping strategy, a layered optimum strategy, and a hybrid strategy to efficiently process the kGSky query, respectively.

4.1 The Group-based Algorithm (GA)

The first exact algorithm is proposed based on the grouping strategy.

Table 2. Symbols to be Used

Notation	**Definition**
D	The dataset
G	The group of points
l	The group size
k	The number of returned results
SL_i	The ith skyline layer
$AncSet(p)$	The ancestor set of p
$DesSet(p)$	The descendant set of p
$DDesSet(p)$	The direct descendant set of p
$DomSize(G)$	Dominance size of the G-Skyline G
$DomSize^{+}(G)$	Upper bound of $DomSize(G)$
$p.layer$	The skyline layer that the point p belongs to
$max\_layer(G)$	The maximum layer of the points within G

4.1.1 The Minimum Dominance Graph. Wang et al. [2018] introduced the minimum dominance graph (MDG) which is the minimum GSky support structure for the GSky query without redundant point. The structure of each node is [*layer index*, *point index*, *ancestor set*, *and descendant set*] where the layer index is the skyline layer that the point lies on and the point index identifies the point uniquely. The edges in MDG represent the dominance relationship between points. Given a point p, the descendant and ancestor sets are defined as follows.

Definition 4.1 (Descendant Set, DesSet). For a given point $p \in D$, the descender set of p is defined as $DesSet(p) = \{p' \in D | p \prec p'\}$. In addition, for a group $G \subseteq D$, $DesSet(G) = \cup_{p \in G} DesSet(p)$.

Definition 4.2 (Ancestor Set, AncSet). For a given point $p \in D$, the ancestor set of p is $AncSet(p) = \{p' \in D | p' \prec p\}$. For a group $G \subseteq D$, $AncSet(G) = \cup_{p \in G} AncSet(p)$.

Formally, we can define MDG as follows.

Definition 4.3 (Minimum Dominance Graph (MDG)). A MDG is a directed graph $G = (V, E)$ over a given dataset D. With a group size l, each vertex $v \in V$ represents a point $p \in D$ that is dominated by at most $l - 1$ points in the given dataset. Besides, each edge $e \in E$ represents the dominance relationship between corresponding points. The MDG is organized as two parts where the left part contains all the vertexes representing the skylines, and the right part contains the vertexes representing all the points dominated by at least one another point in the given dataset. Here the skylines are the points not dominated by any other point.

Figure 3 shows the MDG over the hotel set in Figure 1. The MDG contains two parts where the left part contains the skyline hotels h_1, h_2, h_3, and h_4 and the right part includes the non-skyline hotels h_5, h_6, h_7, h_8, and h_{12}. Besides, it only includes the hotels dominated by less than $l = 3$ hotels.

4.1.2 The Group-based Algorithm (GA). In this subsection, we propose the GA algorithm based on the following lemmas.

Property 1. Given a point group $\mathrm{G} \subseteq D$ *and a point* $p \in D$, *it holds that* $\mathrm{DomSize}(G \cup \{p\}) = \mathrm{DomSize}(G)$ *if the point* p *is dominated by a point* $p' \in G$.

LEMMA 4.4. *Given a GSky* $\mathrm{G} \subseteq D$ *with* $G_1 = G \cap \mathrm{SL}_1$, *it holds that* $\mathrm{DomSize}(G) = \mathrm{DomSize}(G_1)$. *Here* G_1 *only contains the points of* G *that belong to the first skyline layer* SL_1.

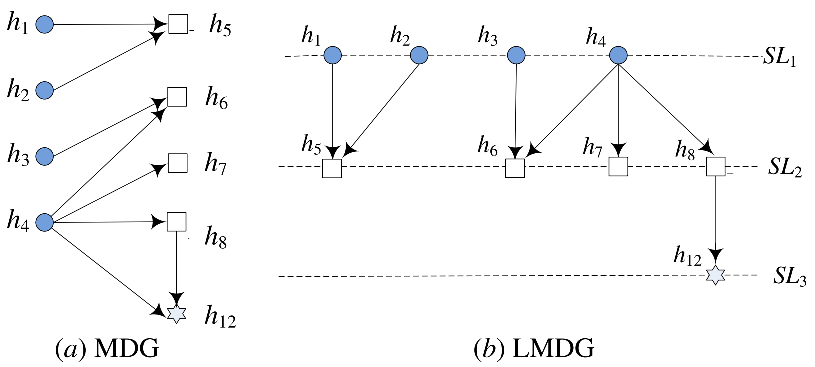


Fig. 3. The MDG and LMDG of the hotel set in Figure 1.

Proof. For a non-skyline point p in the GSky G, all the points p' that dominate p are contained in G [Liu et al. 2015]. The dominance space of p is covered by that of p' according to Definition 3.5, and $DomSize(\{p, p'\}) = DomSize(\{p'\})$.

From Definition 3.1, any non-skyline point $p \in G - G_1$ is dominated by at least a point $p'' \in G_1$ for $G_1 = SL_1 \cap G$. Besides, we have $DomSize(G = G_1 \cup (G - G_1)) = DomSize(G_1)$ based on Property 1. Therefore, this lemma holds. □

On the basis of Lemma 4.4, in Figure 2(c) for the GSky $\{h_3, h_4, h_6\}$, we have $DomSize(\{h_3, h_4, h_6\}) = DomSize(\{h_3, h_4\})$.

Due to Lemma 4.4, for a given GSky G, its dominance size is closely related to $G \cap SL_1$. Therefore, we can compute $DomSize(G \cap SL_1)$ instead of computing $DomSize(G)$. Moreover, to check and prune unqualified results as soon as possible, we develop the following upper bound pruning strategy.

Definition 4.5 (Upper bound of DomSize(G), DomSize$^+(G)$*).* For a given group G, the upper bound of $DomSize(G)$ is $DomSize^+(G) = \sum_{p \in G \cap SL_1} DomSize(p)$.

On the basis of Definition 4.5, we have the upper bound pruning strategy as follows.

Lemma 4.6. *For a given GSky G, G can be pruned safely if* DomSize$^+(G) \le r$ *where r is the current kth largest dominance size.*

Proof. Since $DomSize(G) < DomSize^+(G)$ based on Equation (1) and $DomSize^+(G) \le r$, it holds that $DomSize(G) < r$, and G is not a result of the kGSky query due to Definition 3.8. □

As illustrated in Algorithm 1, the GA algorithm first computes the G-Skylines (line 1) by the G-MDS algorithm in Wang et al. [2018] which is the state-of-the-art algorithm for the GSky query. Then, it initializes l sets V_i to store the G-Skylines containing i points within SL_1 for $1 \le i \le l$ (Line 2). Next, the G-Skylines G are divided due to $|G \cap SL_1|$ and stored in sets V_i for $1 \le i \le l$ (Lines 3 to 5). This means that the set V_i contains i points within SL_1. In the left part of the GA algorithm, the G-Skylines containing more points within the first skyline layer SL_1 are given priority to be processed. In lines 6 and 7, we initialize a set $RSky$ to store the final results of the kGSky query and r to represent the current kth highest dominance size. Lines 8 to 19 are a for-loop that computes k G-Skylines with the highest dominance sizes. Here, i varies from l to 1. Noticeably, by the grouping strategy, we could give higher priority to process the G-Skylines that may have large dominance sizes. If V_i is not empty, lines 9 to 18 are executed to compute the G-Skylines $G \in V_i$ with dominance sizes exceed r. For each GSky G within V_i, line 11 checks whether its upper

ALGORITHM 1: Group-based Algorithm (GA) for the kGSky query

```
Require: A MDG $MD$ over $D$, a group size $l$, and the number of results $k$
Ensure: $k$GSky query results
 1: Compute G-Skylines by G-MDS($D$, $l$, $MD$)
 2: Initialize sets $V_i \leftarrow \emptyset$ for $1 \leq i \leq l$
 3: for each G-Skyline $G$ do
 4:     Add the G-Skyline $G$ with $|G \cap SL_1| = i$ to the set $V_i$
 5: end for
 6: Initialize a set $RSky$ of size $k$ to store $k$ G-Skylines with highest dominance sizes
 7: Initialize $r \leftarrow 0$ //the $k$th highest dominance size
 8: for $i \leftarrow l$ to 1 do
 9:     if $V_i$ is not empty then
10:         for each G-Skyline $G \in V_i$ do
11:             if $DomSize^+(G) > r$ then
12:                 $DomSize(G) \leftarrow DomSize(G \cap SL_1)$ //Lemma 4.4
13:                 if $DomSize(G) \geq r$ then
14:                     Update $RSky$ and $r$ using $<G, DomSize(G)>$
15:                 end if
16:             end if
17:         end for
18:     end if
19: end for
20: Return the G-Skylines within $RSky$
```

bound of the dominance size is larger than r. If it returns 'no', G is pruned due to Lemma 4.6. Otherwise, line 14 refreshes r and updates the set $RSky$ by adding the new GSky G and removing the old G-Skylines whose dominance sizes are less than r. After the for-loop, $RSky$ contains all the kGSky query results.

Example. We continue the example in Figure 1 with $l = 3$ and $k = 5$. After getting all the G-Skylines, we part them into 3 groups due to $|G \cap SL_1|$ and gain the three sets $V_1 = \{\{h_4, h_7, h_8\}, \{h_4, h_8, h_{12}\}\}$, $V_2 = \{\{h_1, h_2, h_5\}, \{h_3, h_4, h_6\}, \{h_1, h_4, h_7\}, \{h_2, h_4, h_7\}, \{h_3, h_4, h_7\}, \{h_1, h_4, h_8\}, \{h_2, h_4, h_8\}, \{h_3, h_4, h_8\}\}$, and $V_3 = \{\{h_1, h_2, h_3\}, \{h_1, h_2, h_4\}, \{h_1, h_3, h_4\}, \{h_2, h_3, h_4\}\}$. For $k = 5 > |V_3|$, all the G-Skylines within V_3 are inserted into the set $RSky$ as candidate results. Here, we have $r = 8250$. Next, the GSky $\{h_1, h_2, h_5\} \in V_2$ is accessed and added to the set $RSky$ because the size of $RSky$ is less than $k = 5$. The current kth highest score is $r = DomSize(\{h_1, h_2, h_5\}) = 5650$. After accessing other G-Skylines within V_2, $\{h_2, h_4, h_7\}$ and $\{h_2, h_4, h_8\}$ are inserted into $RSky$. Now, r is modified as $DomSize(\{h_2, h_4, h_8\}) = 10{,}200$ and the set $RSky$ is refreshed as $\{\{h_1, h_2, h_4\}, \{h_1, h_3, h_4\}, \{h_2, h_3, h_4\}, \{h_2, h_4, h_7\}, \{h_2, h_4, h_8\}\}$ by removing the G-Skylines $\{h_1, h_2, h_5\}$ and $\{h_1, h_2, h_3\}$ with dominance sizes less than $r = 10{,}200$. Finally, the G-Skylines $\{h_4, h_8, h_{12}\}$ and $\{h_4, h_7, h_8\}$ in V_1 are pruned directly as their dominance sizes are both less than $r = 10{,}200$. Finally, the G-Skylines within $RSky$ are reported as the final results.

Complexity. The time complexity of the G_MDS algorithm in Wang et al. [2018] is $O(\sum_{i=1}^{l} \binom{\hbar}{i}) \leq O(2^{\hbar})$ to calculate the unit group sets of size i for $1 \leq i \leq l$. Here, $\hbar = | \cup_{i=1}^{l} SL_i|$. Suppose the number of the G-Skylines is $\hbar_g$. Then, it divides the G-Skylines into different sets V_i with a time complexity of $O(\hbar_g)$. For all the G-Skylines, in worst case it takes $O(\hbar_g \times 2^l)$ to compute their dominance sizes. Therefore, the time complexity of the GA algorithm is $O(2^{\hbar} + \hbar_g + \hbar_g \times 2^l) = O(2^{\hbar} + \hbar_g \times 2^l)$.

4.2 The Layered Unit-based (LU) Algorithm

As pointed out in Papadias et al. [2005], a skyline query algorithm is progressive if the first result can be reported to the user as soon as possible and the output size increases gradually. The progressive property is needed in many preference queries such as the skyline query [Papadias et al. 2005; Ding and Jin 2012; Zhou et al. 2016b] and the top k dominating query [Tiakas et al. 2016; Han et al. 2017; Amagata et al. 2018].

In the GA algorithm, the kGSky query results are computed and reported at the end of the query procedure. Therefore, the users cannot get any results until the algorithm terminates, and there may be a long latency delay between issuing the kGSky query and getting any results. Inspired by this, we propose a layered unit-based algorithm to report the query results progressively.

4.2.1 The Layered MDG. The DSG [Liu et al. 2015] is a layered index that organizes the points within the first l skyline layers. Because there are many redundant points dominated by at least l points in DSG, MDG is proposed in Wang et al. [2018]. When building MDG, only the points dominated by less than $l-1$ points are considered. Now, MDG is considered the minimum GSky support structure without redundant points. However, there are redundant edges in MDG. In Figure 3(a), as well as the edges $h_4 \rightarrow h_8$ and $h_8 \rightarrow h_{12}$, MDG also stores the edge $h_4 \rightarrow h_{12}$ to show the dominance relationship between h_4 and h_{12}.

In this article, we introduce the layered MDG, namely, LMDG, that combines the advantages of both DSG and MDG. Different from MDG in Wang et al. [2018], LMDG proposed divides the points dominated by less than l points into different layers. Compared to DSG and MDG, LMDG is a more simplified index without redundant points and redundant edges.

Definition 4.7 (LMDG). A LMDG is a directed graph $G = (V, E)$ over a given dataset D. With a group size l, each vertex $v \in V$ represents a point $p \in D$ that is dominated by at most $l-1$ points in the given dataset. Besides, each edge $e \in E$ represents the dominance relationship between corresponding points in adjacent layers. The LMDG is organized as l layers and the ith layer contains the vertexes corresponding to the points within the ith skyline layer SL_i for $1 \leq i \leq l$.

Figure 3(b) shows the LMDG over the hotel dataset in Figure 1 with the group size $l = 3$. The LMDG organizes the points dominated by less than $l-1 = 2$ points as three skyline layers, $SL_1 = \{h_1, h_2, h_3, h_4\}$, $SL_2 = \{h_5, h_6, h_7, h_8\}$, and $SL_3 = \{h_{12}\}$. All the hotels within the LMDG are dominated by at most $l-1 = 2$ hotels. The hotels $h \in SL_1$ are skylines that are not dominated by any other hotel in Figure 1. The edges represent the dominance relationship between hotels. For instance, the edge $h_1 \rightarrow h_5$ represents that the hotel h_5 is dominated by the hotel h_1.

4.2.2 The LU Algorithm. In this subsection, we develop the layered unit-based (LU) algorithm which introduces a layered strategy and based on the following lemmas.

Property 2 [Wang et al. 2018]. For a given group G, it can be returned as a G-Skyline directly if it consists of l points within the first skyline layer SL_1.

LEMMA 4.8. *A GSky G contains at most $l-i+1$ points within the ith skyline layer* SL_i *for* $1 \leq i \leq l$.

PROOF. For the GSky G, we get a group $G' = SL_i \cap G$. Both G' and $AncSet(G')$ are contained in G, and $G' \cup AncSet(G') \subseteq G$. Therefore, $|G' \cup AncSet(G')| = |G'| + |AncSet(G')| \leq l$.

According to Definition 3.1, a point $p \in SL_i$ is dominated by at least one point $p' \in SL_j$ for $1 \leq j < i$. Therefore $|AncSet(G')| \geq i-1$. For $|G'| + |AncSet(G')| \leq l$ and $|AncSet(G')| \geq i-1$, it is easy to get $|G'| \leq l - |AncSet(G')| \leq l-i+1$. Besides, when all the points within $G' \subseteq SL_i \cap G$ have a same ancestor point within SL_j for $1 \leq j < i$, $|G'|$ reaches its maximum value that is equal to $l-i+1$. Therefore, this lemma holds. □

From Lemma 4.8, we have any GSky of size l includes at most $l - i + 1$ points within the ith skyline layer.

Definition 4.9 (Maximum Layer). Given a group G, its maximum layer is $max\_layer(G) = \max_{p \in G} p.layer$. Here, $p.layer$ is the skyline layer that the point p belongs to.

For the hotel group $\{h_4, h_8\}$ in Figure 1, we have $max\_layer(\{h_4, h_8\}) = \max\{h_4.layer, h_8.layer\} = 2$.

Lemma 4.10. *Given two point groups G_{maxl_i} and G_{maxl_j} whose maximum layers are i and j, respectively, G_{maxl_i} only may be g-dominated by G_{maxl_j} if $j \leq i$.*

Proof. Assume that $G_{maxl\_j} \prec_g G_{maxl\_i}$ and $j > i$. Due to Definition 3.2, if $G_{maxl\_j} \prec_g G_{maxl\_i}$, for each point $p' \in G_{maxl\_j}$, p' dominates or is just equal to some point $p \in G_{maxl\_i}$, and for at least one point p', $p' \prec p$. However, since $j > i$, there is at least a point $p \in G_{maxl\_j} \cap SL_j$ that is dominated by or incomparable to some points in $G_{maxl\_i} \cap SL_i$ due to Definition 3.1. This contradicts to the assumption. It is worth to notice that in case $i = j$, G_{maxl_j} may be dominated by G_{maxl_i} only it satisfies $G_{maxl\_j} \cap SL_j = G_{maxl\_i} \cap SL_i$. Therefore, this lemma holds. □

In Figure 1, for $G = \{h_1, h_3, h_5\}$ where $max\_layer(G) = 2$, it is g-dominated by $\{h_1, h_2, h_3\}$ where each hotel is within SL_1. The hotel groups of size 3, which contain hotels within SL_i for $i > 2$, cannot g-dominate $\{h_1, h_3, h_5\}$. For instance, $\{h_1, h_3, h_5\}$ is incomparable to $\{h_1, h_3, h_8\}$ since h_5 is incomparable to h_8. Moreover, for two given hotel groups $G = \{h_4, h_6, h_8, h_{12}\}$ and $G' = \{h_3, h_4, h_8, h_{12}\}$ with $max\_layer(G) = max\_layer(G') = 3$, we can also have $G' \prec G$. This is because $h_6 \in G$ is dominated by $h_3 \in G'$ and other hotels in the two groups are the same.

Based on Lemma 4.10, any group G of size l whose maximum layer is i may be only g-dominated by the groups $G' \subseteq \bigcup_{j=1}^{i} SL_j$. Besides, the groups of l points within the first skyline layer SL_1 could be returned as G-Skylines, directly, since they are dominated by no groups of the same size.

Definition 4.11 (Unit Group [Liu et al. 2015]). For a point $p \in D$, its unit group is $u_p = \{p\} \cup AncSet(p)$.

Theorem 4.12 (Verification of GSky [Liu et al. 2015]). *For a group $G = \{p_1, p_2, \ldots, p_l\}$, it is a GSky, if and only if the unit group set $S = \cup\{u_{p_i} | i \in [1, l]\}$ includes l points.*

Lemma 4.13. *Given a unit group set U with $1 \leq |U| \leq l$, $G = \bigcup_{u \in U} u$ is a GSky if and only if $|G| = l$, and G with $|G| > l$ could be pruned safely.*

Proof. This lemma is correct due to Theorem 4.12. □

Lemma 4.14. *Assume that the groups G_{maxl_i} are the G-Skylines with* $\mathrm{max\_layer}(G_{maxl\_i}) = i$*. For a GSky G_{maxl_j} with* $\mathrm{max\_layer}(G_{maxl\_j}) = j$*, if $j > i$, then* $\mathrm{DomSize}(G_{maxl\_j}) < \max \mathrm{DomSize}(G_{maxl\_i})$*. Here* $\max \mathrm{DomSize}(G_{maxl\_i})$ *is the maximum dominance size for all G_{maxl_i}.*

Proof. Consider a GSky G_{maxl_j} with $j > i$. Due to Property 1, we have $DomSize(G_{maxl\_j}) = DomSize(G')$ for $G' = G_{maxl\_j} - (G_{maxl\_j} \cap \cup_{t=i+1}^{j} SL_t)$. By selecting $|G_{maxl\_j} \cap \cup_{t=i+1}^{j} SL_t|$ points within $SL_1 - G_{maxl\_j}$ and adding them to G', we get a new GSky G'_{maxl_i} with $max\_layer(G'_{maxl\_i}) = i$. Since $G_{maxl\_j} \cap SL_1 \subset G'_{maxl\_i}$, it holds that $DomSize(G_{maxl\_j}) < DomSize(G'_{maxl\_i})$ on the basis of Property 1. Because $DomSize(G'_{maxl\_i}) < \max DomSize(G_{maxl\_i})$, we have $DomSize(G_{maxl\_j}) \leq \max DomSize(G_{maxl\_i})$ and this lemma holds. □

In Figure 1, consider the G-Skylines G_{maxl_1}, which are $\{h_1, h_2, h_3\}$, $\{h_1, h_2, h_4\}$, $\{h_1, h_3, h_4\}$, and $\{h_2, h_3, h_4\}$. These G-Skylines G_{maxl_1} only consist of points within the first skyline layer, and the maximum layers of these G-Skylines are 1. The maximum dominance size of the groups G_{maxl_1} is equal to max $DomSize(\{h_1, h_3, h_4\}) = 10700$. Based on Lemma 4.14, for any GSky G_{maxl_j} with $j > 1$, it holds that $DomSize(G_{maxl\_j}) <$ max $DomSize(G_{maxl\_1}) = 10700$. This means that the dominance size of any GSky G_{maxl_j} for $j > 1$ is less than max $DomSize(G_{maxl\_1}) = 10700$. Accordingly, the GSky $G_{maxl\_1} = \{h_1, h_3, h_4\}$ with the maximize dominance size 10700 could be outputted directly.

Based on Lemma 4.14, we have the maximum dominance size of the G-Skylines G_{maxl_i} with maximum layer i for $1 \le i < l$ is always larger than the G-Skylines whose maximum layers are larger than i. Accordingly, we introduce the layered strategy into the kGSky query, and present the LU algorithm depicted in Algorithm 2.

ALGORITHM 2: Layered Unit-based (LU) Algorithm for the kGSky query

```
Require: A LMDG $LM$ over $D$, a group size $l$, and number of results $k$
Ensure: $k$GSky query results
 1: Initialize a set $RSky \leftarrow \emptyset$ that stores the final results of the $k$GSky query
 2: Initialize the $k$th highest dominance size $r \leftarrow 0$
 3: for $i \leftarrow 1$ to $l$ do
 4:     $MaxDomSize \leftarrow 0$
 5:     Initialize a set $CanSky$
 6:     $CanSky \leftarrow$ GSkySearcher($LM$, $i$, $l$)
 7:     for each $G \in CanSky$ do
 8:         if $DomSize^+(G) > r$ then
 9:             $DomSize(G) \leftarrow DomSize(G \cap SL_1)$ //Lemma 4.4
10:             if $DomSize(G) \ge r$ then
11:                 Update $RSky$ and $r$ using $<G, DomSize(G)>$
12:                 if $DomSize(G) > MaxDomSize$ then
13:                     $MaxDomSize \leftarrow DomSize(G)$
14:                 end if
15:             end if
16:         end if
17:     end for
18:     Report the G-Skylines $G \in RSky$ whose dominance sizes exceed $MaxDomSize$ due to Lemma 4.14
19: end for
20: Return the G-Skylines within $RSky$
```

As shown in Algorithm 2, it takes the LMDG LM, the group size l, and the number of results k as the inputs. Line 1 first defines the set $RSky$ which is the final result set of the kGSky query and the current kth highest dominance size r is set to 0 (line 2). Lines 3–19 are a for-loop which is executed to compute k G-Skylines with the highest dominance sizes. Here, the G-Skylines are generated based on unit groups within different skyline layers, respectively. To report the final results in a progressive manner, line 4 defines $MaxDomSize$ to represent the maximum dominance size of the G-Skylines generated in each iteration. In the first iteration, it builds the G-Skylines G that consist of just l points within SL_1 by invoking GSkySearcher(LM, 1, l), selects k G-Skylines with the highest dominance sizes, and finally adds them to the set $RSky$. The G-Skylines within $RSky$ whose dominance sizes exceed $MaxDomSize$ are returned and removed from $RSky$. Thereafter, the unit groups within the ith skyline layer for $2 \le i \le l$ are taken into account. It builds the G-Skylines that contain just j points within SL_i for $1 \le j \le l - i + 1$ by invoking the GSkySearcher algorithm in Algorithm 3. Lines 8–16 identify the G-Skylines G with $DomSize(G) \ge r$ and use them to update $RSky$, r, and $MaxDomSize$. Due to Lemma 4.14, the maximum dominance size of the G-Skylines generated based on the unit groups $u \in U_i$ is always larger than the dominance sizes of the G-Skylines G that contain at least one point within SL_j for $j > i$. Therefore, the G-Skylines within $RSky$ whose dominance sizes exceed $MaxDomSize$ are final results of the kGSky query and

ALGORITHM 3: GSkySearcher

```
Require: A LMDG $LM$, layer $i$, and a group size $l'$
Ensure: A point group set $CanSet$
 1: Initialize a set $CanSet \leftarrow \emptyset$
 2: if $i = 1$ then
 3:     Add groups $G \subseteq SL_1$ of size $l'$ to $CanSet$ //Property 2
 4: else
 5:     Compute a unit group set $U_i \leftarrow \{u_p | p \in SL_i\}$ which includes the unit groups of the points $p \in SL_i$
 6:     Generate unit group sets $U' \subseteq U_i$ of size $j$ for $1 \leq j \leq l' - i + 1$ // Lemma 4.8
 7:     for each unit group set $U'$ do
 8:         Generate a new group $G' \leftarrow \bigcup_{u \in U'} u$
 9:         if $|G'| = l'$ then
10:             Insert the group $G'$ to $CanSet$ //Lemma 4.13
11:         else
12:             if $|G'| < l'$ then
13:                 Compute a LMDG $LM'$ that consists of points within $\cup_{t=1}^{i-1} SL_t - G'$
14:                 for $j \leftarrow 1$ to $i - 1$ do
15:                     Generate groups $G'' \leftarrow$GSkySearcher($LM', j, l' - |G'|$) //Lemma 4.8
16:                     for each group $G''$ do
17:                         if $G' \cup G'' = l'$ then
18:                             Insert the group $G' \cup G''$ to $CanSet$ due to Lemma 4.13
19:                         end if
20:                     end for
21:                 end for
22:             end if
23:         end if
24:     end for
25: end if
26: Return $CanSet$
```

can be returned as soon as possible. It is because these G-Skylines are with the current largest dominance sizes and their dominance sizes are larger than all the unseen G-Skylines generated in the remaining iterations.

As listed in Algorithm 3, the GSkySearcher algorithm takes the LMDG LM, the layer i, and the group size l' as inputs. When taking into account the unit groups within the first skyline layer, it creates the groups G that just contain l' points within SL_1, and adds them to the set *CanSet*. Otherwise, lines 5–24 are executed. Line 5 computes the unit group set U_i that consists of all the unit groups of the points within SL_i. For example, in Figure 1, $SL_2 = \{h_5, h_6, h_7, h_8\}$, $u_5 = \{h_1, h_2, h_5\}$, $u_6 = \{h_3, h_4, h_6\}$, $u_7 = \{h_4, h_7\}$, and $u_8 = \{h_4, h_8\}$. Therefore, $U_2 = \{u_5, u_6, u_7, u_8\} = \{\{h_1, h_2, h_5\}, \{h_3, h_4, h_6\}, \{h_4, h_7\}, \{h_4, h_8\}\}$. Line 6 builds the unit group sets $U' \subseteq U_i$ of size j for $1 \leq j \leq l' - i + 1$ due to Lemma 4.8. For each unit group set U', a new candidate group G' is generated (line 8). If $|G'| > l'$, it could be pruned directly. The group G' of size l' is added to the set *CanSet*. For the group G' whose size is less than l', lines 13–21 are executed to generate new groups of size l' by invoking the GSkySearcher algorithm recursively. It computes the LMDG LM' over the points within $\cup_{t=1}^{i-1} SL_t - G'$. Next, for the unit groups within the first $i - 1$ skyline layers, GSkySearcher($LM', j, l' - |G'|$) is executed to compute the groups G'' which are combined with G' to build new groups $G' \cup G''$. The groups $G' \cup G''$ of size l' are inserted to the set *CanSet*. Finally, *CanSet* contains all the G-Skylines that contains at least a point within SL_i.

Example. Consider the example in Figure 1 with $l = 3$ and $k = 5$. We first compute the G-Skylines $\{h_1, h_2, h_3\}, \{h_1, h_2, h_4\}, \{h_1, h_3, h_4\}$, and $\{h_2, h_3, h_4\}$ that only consist of the points within SL_1. The above G-Skylines are all added to *RSky*, and the GSky $\{h_1, h_3, h_4\}$ with the highest dominance size is returned. For the unit group set $U_2 = \{u_5, u_6, u_7, u_8\}$, unit group sets $U' \in \{\{u_5\}, \{u_6\}, \{u_7\}, \{u_8\}, \{u_5, u_6\}, \{u_5, u_7\}, \{u_5, u_8\}, \{u_6, u_7\}, \{u_6, u_8\}, \{u_7, u_8\}\}$ are generated. Taking into account each unit group set U', we could get new candidate groups. For $U' = \{u_5\}$, a new GSky $G = \{h_1, h_2, h_5\}$ is generated and added to the set *RSky*. Now r is modified as $DomSize(\{h_1, h_2, h_5\}) = 5650$. Next, based on the unit group set $U' = \{u_6\}$, we get the GSky $\{h_3, h_4, h_6\}$, and $RSky = \{\{h_1, h_2, h_3\}, \{h_1, h_2, h_4\}, \{h_2, h_3, h_4\}, \{h_3, h_4, h_6\}\}$. The fifth highest dominance size is $r = 8250$. For the unit group set $U' = \{u_7\}$, we have a candidate group $\{h_4, h_7\}$ and the GSkySearcher algorithm is applied to compute G-Skylines based on it. By combining $\{h_4, h_7\}$ with groups $G' \subseteq SL_1 - \{h_4\} = \{h_1, h_2, h_3\}$ of size $l - 2 = 1$, we gain new G-Skylines $\{h_4, h_7\} \cup \{h_1\} = \{h_1, h_4, h_7\}$, $\{h_4, h_7\} \cup \{h_2\} = \{h_2, h_4, h_7\}$, and $\{h_4, h_7\} \cup \{h_3\} = \{h_3, h_4, h_7\}$. The G-Skylines $\{h_1, h_4, h_7\}$ and $\{h_2, h_4, h_7\}$ are utilized to refresh the set *RSky* and r. After computing the G-Skylines containing $\{h_4, h_8\}$, *RSky* is updated as $\{\{h_1, h_2, h_4\}, \{h_2, h_3, h_4\}, \{h_2, h_4, h_7\}, \{h_2, h_4, h_8\}\}$, $r = DomSize(\{h_2, h_4, h_7\}) = 10{,}200$, and $MaxDomSize = 10{,}200$. For the unit groups $\{u_5, u_6\}, \{u_5, u_7\}, \{u_5, u_8\}, \{u_6, u_7\}$, and $\{u_6, u_8\}$, the sizes of the new candidate groups generated based on it are all larger than $l = 3$, and they are pruned directly. For the last unit group $\{u_7, u_8\}$ within U_2, we get a new GSky $u_7 \cup u_8 = \{h_4, h_7, h_8\}$. Since $DomSize(\{h_4, h_7, h_8\}) = 8800 < r = 10{,}200$, it is pruned directly. After taking into account all the unit group sets $U' \subseteq U_2$, the G-Skylines $\{h_1, h_2, h_4\}, \{h_2, h_3, h_4\}, \{h_2, h_4, h_7\}, \{h_2, h_4, h_8\}$ within *RSky* whose dominance sizes are no less than *MaxDomSize* are returned and $RSky = \emptyset$. Consider the unit group set u_{12} in SL_3. We have a new GSky $\{h_4, h_8, h_{12}\}$. Since its dominance size is less than $r = 10{,}200$, it is not added to the set *RSky*, and there is no new result generated.

Complexity. As pointed out in Liu et al. [2015], the G-Skylines only consist of the points within the first l skyline layers. Here, l is the group size. Assume that $\hbar_i$ denotes the size of SL_i and $\hbar = \sum_{i=1}^{l} \hbar_i$ for $1 \le i \le l$. The maximum cardinality of the G-Skylines is $\binom{\hbar}{l}$. Due to Lemma 4.8, the candidate groups G including more than $l - i + 2$ points $p \in SL_i$ are not G-Skylines. The size of candidate groups generated in the LU algorithm is:

$$\binom{\hbar}{l} - \sum_{i=2}^{l} \sum_{j=l-i+2}^{k} \binom{\hbar_i}{j} \times \binom{\hbar - \hbar_i}{l - j} < \binom{\hbar}{l} - \sum_{i=2}^{l} \binom{\hbar_i}{k}.$$

In the worst case, the LU algorithm computes the dominance sizes for all the candidate groups with the time complexity $O((\binom{\hbar}{l} - \sum_{i=2}^{l} \binom{\hbar_i}{k}) \times 2^l)$. As analyzed above, the time complexity of the LU algorithm is

$$O\left(\left(\binom{\hbar}{l} - \sum_{i=2}^{l} \binom{\hbar_i}{k}\right) \times 2^l\right) < O\left(\binom{\hbar}{l} \times 2^l\right).$$

4.3 The Top-down Point-based (TP) Algorithm

The GA and LU algorithms both compute the G-Skylines first and then identify the k G-Skylines with the highest dominance sizes. In this subsection, we propose the top-down point-based algorithm, which only generates the G-Skylines that are most likely to be the final results of the kGSky query. In addition, we redefine the structure of the node in the LMDG as [*layer index*, *point index*, *ancestor set*, and *direct descendant set*]. Here, each node stores the direct descenders instead of all the descenders.

Definition 4.15 (Direct descender set). Given a point p', p' is a direct descendant of p if in the LMDG there is an edge connecting the nodes of p and p' directly. For a given point p, its direct descendant set $DDesSet(p, i)$ contains all the direct descendants of p in the ith skyline layer. Besides, for a given group G, we have the direct descendant set $DDesSet(G, i) = \cup_{p \in G} DDesSet(p, i)$.

Considering the hotel h_4 in Figure 1, we have $DDesSet(h_4, 2) = \{h_6, h_7, h_8\}$. For the hotel group $\{h_2, h_3\}$, it holds that $DDesSet(\{h_2, h_3\}, 2) = \{h_5, h_6\}$.

Property 3. For a given GSky G of size l, G contains i points within SL_1 *for* $1 \leq i \leq l$.

Lemma 4.16. *For a GSky G, it is not a final result of the kGSky query if* $\mathrm{DomSize}(G \cap \mathrm{SL}_1) < r$. *Here, r is the current kth highest dominance size.*

Proof. Due to Property 1, dominance size of the GSky G is computed as $DomSize(G) = DomSize(G \cap SL_1)$. On the assumption that $DomSize(G \cap SL_1) < r$, it holds that $DomSize(G) < r$. Therefore, G is not a final result of the kGSky query due to Definition 3.8 and this lemma holds. □

Lemma 4.17. *For G-Skylines G_i that contain i points within* SL_1, *we have* $\mathrm{DomSize}(G_j) \leq \max \mathrm{DomSize}(G_i)$ *for all G_j with* $|G_j \cap \mathrm{SL}_1| = j < i$.

Proof. Based on Lemma 4.4, the dominance size of a GSky G depends on the dominance size of $G \cap SL_1$. Since $|G_j \cap SL_1| = j < i$, similarly to Lemma 4.14, it holds that there is at least a GSky G_i satisfying $DomSize(G_j) < DomSize(G_i)$, and $DomSize(G_j) < \max DomSize(G_i)$ accordingly for all G_i. Therefore, this lemma holds. □

Due to Lemma 4.17, the G-Skylines containing more points within SL_1 may have large dominance sizes.

Lemma 4.18. *Assume that the G-Skylines with more points within the first skyline layer* SL_1 *are generated and processed in prior, the G-Skylines G_i contain i points within* SL_1 *and* $\mathrm{DomSize}(G_i) > r$. *Here r is the current kth highest dominance size. Considering the G-Skylines G_{i-1} that include $i-1$ points within* SL_1, *the G-Skylines G_i could be reported as final results of the kGSky query if their dominance sizes exceed* $\max \mathrm{DomSize}(G_{i-1}) \geq r$ *for all G_{i-1}. Furthermore, the G-Skylines G_{i-1} whose dominance sizes are just equal to* $\max \mathrm{DomSize}(G_{i-1})$ *can also be reported directly as final results.*

Proof. For the G-Skylines G_{i-1} and G_j with $j < i-1$, we have $DomSize(G_j) < \max DomSize(G_{i-1})$ due to Lemma 4.17. Since the G-Skylines G_i satisfy $DomSize(G_i) \geq \max DomSize(G_{i-1})$, it holds that $DomSize(G_i) > DomSize(G_j)$ for $j \leq i-1$. The G-Skylines with more points within SL_1 are generated and given priority to be processed. This means that $DomSize(G_i)$ and $\max DomSize(G_{i-1})$ are larger than those of the left G-Skylines that have not been accessed. For $\max DomSize(G_{i-1})$ is larger than r which is the current kth highest dominance size, the G-Skylines G_i whose dominance sizes exceed $\max DomSize(G_{i-1})$ could be returned as the final results of the kGSky query. Moreover, the G-Skylines G_{i-1} whose dominance sizes are exactly equal to $\max DomSize(G_{i-1})$ are final results of the kGSky query and can be reported directly too. □

Consider the example in Figure 1 with $l = 3$ and $k = 5$. Let G_3 and G_2 denote the hotel groups which are G-Skylines containing three and two hotels within the first skyline layer SL_1, respectively. First, we compute the hotel groups $\{h_1, h_2, h_3\}, \{h_1, h_2, h_4\}, \{h_1, h_3, h_4\}, \{h_2, h_3, h_4\}$, which are G-Skylines consisting of $l = 3$ hotels within the first skyline layer SL_1. As shown in Table 1, the G-Skyline $\{h_1, h_3, h_4\}$ is with the maximum dominance size $\max DomSize(G_3) = 10{,}700$, and it can be reported as a final result directly due to Lemma 4.18. Second, the hotel groups $\{h_1, h_2, h_5\}, \{h_1, h_4, h_7\}, \{h_1, h_4, h_8\}, \{h_2, h_4, h_7\}, \{h_2, h_4, h_8\}, \{h_3, h_4, h_6\}, \{h_3, h_4, h_8\}$ including two hotels within SL_1 are generated. The maximum dominance size of these groups is $\max DomSize(G_2) = 10{,}200$. On the basis of Lemma 4.18, the hotel groups G_3, which are $\{h_1, h_2, h_4\}$ and $\{h_2, h_3, h_4\}$, can be reported as final results because their dominance sizes exceed 10,200. In addition, considering the hotel groups $\{h_2, h_4, h_7\}$ and $\{h_2, h_4, h_8\}$ which include two hotels within SL_1 can also be returned as results of the kGSky query. This is because their dominance sizes are equal to $\max DomSize(G_2) = 10{,}200$.

Based on Lemmas 4.16–4.18, we develop the top-down point-based algorithm as follows.

In Algorithm 4, it first initializes a set *RSky* to store the final results of the kGSky query (line 1). Line 2 initializes the kth highest dominance size r to 0. Then, line 3 computes the G-Skylines G directly due to Property 2 and adds them to a set V_0. These G-Skylines consist of just l points belonging to SL_1. Next, k G-Skylines $G \subseteq V_0$ with the highest dominance sizes are computed and inserted to the set *RSky* (Lines 4 to 10). The left part of Algorithm 4 (Lines 11–43) generates other G-Skylines whose dominance sizes may exceed r. In each iteration, it generates groups G_1^i that consist of i points within SL_1. Here i is reduced from $l-1$ to 1 due to Lemma 4.17. By this way, the G-Skylines most likely to be the final results are generated in priority. In line 13, the groups G_1^i whose dominance sizes exceed r are added to the set V_i. Lines 14 to 18 define *MaxDomSize* as the maximum dominance size of the G-Skylines generated based on the groups $G' \in V_i$. If V_i is empty, then *MaxDomSize* is set to r. In line 19, the G-Skylines $G \in RSky$ with $DomSize(G) \geq MaxDomSize$ are returned as final results. Lines 20–42 are a while-loop to generate the G-Skylines based on each candidate group within V_i. If V_i is not empty, the groups within V_i are sorted in non-increasing order of their dominance sizes (Line 21). For each candidate group $G' \in V_i$ with $DomSize(G') \geq r$, it computes the maximum layer of points $p \in G'$ (line 24). Considering each nonempty subset $DC' \subseteq DDesSet(G', max\_layer + 1)$ of size no more than $l - |G'|$, line 26 checks whether G' contains $AncSet(DC')$. This ensures G' contains all the ancestors of points within DC'. If it returns “yes,” a new group G'' is generated. The new group G'' with $|G''| < l$ is added to the set V_i, and the ones of size l are utilized to refresh the set *RSky* and r. Line 37 removes the group G' that has been processed from the set V_i. If the dominance size of the group $G' \in V_i$ is less than r, then the left groups within V_i can be overlooked by modifying V_i as $\emptyset$ in line 39. This is because the groups within V_i are sorted in non-increasing order of their dominance sizes and the dominance sizes of the groups ranked after G' are all less than r if $DomSize(G') < r$. At the end of the TP algorithm, the G-Skylines within *RSky* are reported.

Example. Going back to the example in Figure 1, the TP algorithm first generates the G-Skylines $\{h_1, h_2, h_3\}, \{h_1, h_2, h_4\}, \{h_1, h_3, h_4\}$, and $\{h_2, h_3, h_4\}$ which consist of $l = 3$ points within SL_1. The above G-Skylines are all inserted into the set *RSky*. Here $r = 0$. Next, the groups G_1^2 that contain 2 points within SL_1 are generated and we gain $V_2 = \{\{h_1, h_2\}, \{h_1, h_3\}, \{h_1, h_4\}, \{h_2, h_3\}, \{h_2, h_4\}, \{h_3, h_4\}\}$. On account of $MaxDomSize = DomSize(\{h_2, h_4\}) = 10{,}200$, we report the G-Skylines $\{h_1, h_2, h_4\}$, $\{h_1, h_3, h_4\}$, and $\{h_2, h_3, h_4\}$ within the set *RSky* and $RSky = \{\{h_1, h_2, h_3\}\}$. After sorting the groups within V_2, $V_2 = \{\{h_2, h_4\}, \{h_1, h_4\}, \{h_3, h_4\}, \{h_2, h_3\}, \{h_1, h_3\}, \{h_1, h_2\}\}$. Then we build the G-Skylines based on the groups $\{h_2, h_4\}$. For the group $\{h_2, h_4\}$, its direct descendant set consisting of the points belonging to SL_2 is $\{h_5, h_6, h_7, h_8\}$. Since $AncSet(h_7) \subseteq \{h_2, h_4\}$ and $AncSet(h_8) \subseteq \{h_2, h_4\}$, we get two new G-Skylines $\{h_2, h_4\} \cup \{h_7\} = \{h_2, h_4, h_7\}$ and $\{h_2, h_4\} \cup \{h_8\} = \{h_2, h_4, h_8\}$. Here, we have $RSky = \{\{h_2, h_4, h_7\}, \{h_2, h_4, h_8\}\}$, $r = 10{,}200$, and $\{h_1, h_2, h_3\}$ is removed from *RSky* because its dominance size is less than r. For the next group $\{h_1, h_4\} \in V_2$, $DomSize(\{h_1, h_4\}) = 10{,}000 < r = 10{,}200$. Therefore, it is unnecessary to process $\{h_1, h_4\}$ and other groups within V_2. Moreover, the groups $\{h_1\}, \{h_2\}, \{h_3\}$, and $\{h_4\}$ that contain only one point within SL_1 are not taken into account. This is because the dominance sizes of the G-Skylines generated based on them are all less than $r = 10,200$. At last, the G-Skylines $\{h_2, h_4, h_7\}$ and $\{h_2, h_4, h_8\}$ within *RSky* are reported.

ALGORITHM 4: Top-down Point-based (TP) Algorithm for the kGSky query

```
Require: A LMDG $LM$ over $D$, a group size $l$, and the number of results $k$
Ensure: $k$GSky query results
 1: Initialize a set $RSky \leftarrow \emptyset$ that stores the final results of the $k$GSky query
 2: Initialize $r \leftarrow 0$ //the $k$th highest dominance size
 3: Compute point groups $G \subseteq SL_1$ of size $l$ and add them to a set $V_0$ //Property 2
 4: for each G-Skyline $G \in V_0$ do
 5:     if $DomSize^+(G) > r$ then
 6:         if $DomSize(G) \geq r$ then
 7:             Update $RSky$ and $r$ using $<G, DomSize(G)>$
 8:         end if
 9:     end if
10: end for
11: for $i \leftarrow l - 1$ to 1 do
12:     Compute groups $G_1^i \subseteq SL_1$ of size $i$ based on Property 3
13:     Add the groups $G_1^i$ with $DomSize(G_1^i) \geq r$ into $V_i$
14:     if $V_i$ is not empty then
15:         $MaxDomSize \leftarrow \max_{G \in V_i} DomSize(G)$
16:     else
17:         $MaxDomSize \leftarrow r$
18:     end if
19:     Report G-Skyline $G \in RSky$ as a final result if $DomSize(G) \geq MaxDomSize$ due to Lemma 4.18
20:     while $V_i$ is not empty do
21:         Sort groups $G \in V_i$ in non-increasing order of the dominance sizes
22:         for each candidate group $G' \in V_i$ do
23:             if $DomSize(G') \geq r$ then
24:                 $max\_layer \leftarrow \max_{p \in G'} p.layer$
25:                 for each $DC' \subseteq DDesSet(G', max\_layer + 1)$ with $|DC'| \leq l - |G'|$ do
26:                     if $\text{AncSet}(DC') \subseteq G'$ then
27:                         Generate a new group $G'' \leftarrow G' \cup DC'$
28:                         if $|G''| < l$ then
29:                             Add $G''$ into $V_i$
30:                         else
31:                             if $|G''| = l$ then
32:                                 Update $RSky$ and $r$ using $<G'', DomSize(G'')>$
33:                             end if
34:                         end if
35:                     end if
36:                 end for
37:                 $V_i \leftarrow V_i - \{G'\}$
38:             else
39:                 $V_i \leftarrow \emptyset$
40:             end if
41:         end for
42:     end while
43: end for
44: Return G-Skylines $G \in RSky$
```

Complexity. The TP algorithm first takes $O(\binom{\hbar_1}{l} \times 2^l)$ to compute the groups $G \subseteq SL_1$ of size l and get their dominance sizes. Then, it takes $O(\binom{\hbar_1}{i} \times 2^i)$ to generate the G-Skylines consisting of i points within SL_1 and compute their dominance sizes. After that, for each group within V_i, new candidate groups are generated by adding corresponding direct descendants with the time complexity $O(\binom{\hbar_1}{i} \times |DC'|)$. Suppose that $|DC'| \leq \partial$. The time complexity of the TP algorithm is:

$$O\left(\sum_{i=1}^{l}\binom{\hbar_1}{i} \times 2^i + \sum_{i=1}^{l-1}\binom{\hbar_1}{i} \times |DC'|\right) \leq O(2^{h_1} \times (2^l + \partial)).$$

4.4 Discussion

In this subsection, similarly to Gao et al. [2015a], we first prove the correctness of the proposed exact algorithms, CA, LU, and TP, in theory. Moreover, we also offer theoretical proofs about the progressiveness of the LU and TP algorithms referring to Tiakas et al. [2016].

THEOREM 4.19. *The GA, LU, and TP algorithms can return all the exact results of the kGSky query.*

PROOF. This theorem means that there is no unqualified group returned (i.e., no false positive) and there is no result missed (i.e., no false negative).

Assume that there is an unqualified group G returned as a final result. Since G is an unqualified result, we have $DomSize(G) < r$, and in the GA, LU, and TP algorithms, it will be pruned by the pruning strategies, Lemmas 4.6 and 4.16, or Definition 3.8. Moreover, if some G-Skylines are returned as the final results of the kGSky, it holds that their scores exceed r. This contradicts to the assumption. Besides, in the GA, LU, and TP algorithms, the pruning strategies (Lemmas 4.6 and 4.14) are utilized to identify and prune unqualified groups whose dominance sizes are less than r. This can ensure no false negative. □

THEOREM 4.20. *The LU algorithm can return the results of the kGSky query in a progressive manner.*

PROOF. In the LU algorithm, the G-Skylines are generated based on the unit groups of points within different skyline layers, respectively. We first generate the G-Skylines G based on the unit groups of points $p \in SL_i$ for $1 \leq i < l$. Then, we compute *MaxDomSize* as the maximum dominance size of these G-Skylines G. Consider the unseen G-Skylines G' generated based on the unit groups of points within SL_j for $j > i$. We have $DomSize(G') < MaxDomSize$ on the basis of Lemma 4.14. Accordingly, in the LU algorithm, the G-Skylines within *RSky* whose dominance sizes exceed the current maximum dominance size *MaxDomSize* are final results and can be returned as soon as possible. As analyzed above, this theorem holds. □

THEOREM 4.21. *The TP algorithm can report the results of the kGSky query progressively.*

PROOF. The TP algorithm gives priority to generate the G-Skylines that have higher chance to be the final results of the kGSky query. The candidate groups $G_1^i \subseteq SL_1$ for $1 \leq i \leq l$ are computed and the groups G_1^i containing i points within SL_1 are stored in the set V_i. Then, *MaxDomSize* is defined as the maximum dominance size of the G-Skylines generated based on the groups $G' \in V_i$. Due to Lemma 4.18, *MaxDomSize* is larger than the dominance sizes of other unseen G-Skylines which will be generated with considering the groups $G'' \in V_j$. Here, $1 \leq j < i$. Therefore, the G-Skylines $G \in RSky$ with $DomSize(G) \geq MaxDomSize$ can be returned as final results before computing other unseen G-Skylines, and this theorem holds. □

5 APPROXIMATE ALGORITHMS FOR THE kGSKY QUERY

The precise results are not necessary in many applications. In this section, we introduce some approximate techniques to the TP algorithm, which has the best performance among the proposed exact algorithms, at the expense of accuracy of the results to get better efficiency.

5.1 The βTP Algorithm

In this subsection, we introduce the N-consider technique [Corral and Vassilakopoulos 2005] to the kGSky query, and develop an approximate algorithm, namely, βTP.

The time complexity of the TP algorithm is prohibitively high because it needs to process a large size of candidate groups, and for each candidate group, it costs $O(2^l)$ to compute its dominance size. In the βTP algorithm, we introduce the N-consider technique to cut down the size of candidate groups.

For the TP algorithm, it generates the groups $G \subseteq SL_1$ of sizes no more than l (lines 3 and 12, Algorithm 4). Based on these groups, the G-Skylines that are likely to be the final results are created. Different from the TP algorithm, the βTP algorithm first sorts points $p \in SL_1$ in descending order of their dominance sizes. Then, we select a specified portion (top β percentage for $0 < \beta \leq 1$) of the points within SL_1 and store these points in a new set SL_1'. In lines 3 and 12 of the βTP algorithm, it only takes into account the points $p \in SL_1'$. By this way, $|V_i|$ for $1 \leq i \leq l$ can be decreased dramatically, and the size of the candidate groups is reduced in turn.

5.2 The αTP Algorithm

In Gao et al. [2015a], the α-allowance technique is developed to boost the query performance of the distance-based queries. The quality of the approximate results for the α-allowance method is very high. Therefore, based on the α-allowance technique utilized in Gao et al. [2015a], we propose an approximate algorithm, namely αTP.

The αTP algorithm is similar to the TP algorithm. Different from the TP algorithm, it takes the parameter α as an important input. Moreover, we modify the upper bound pruning strategy and line 5 of the TP algorithm is adjusted to "$DomSize^+(G) \times (1 - \alpha) \geq r$." Accordingly, there are much more candidate groups that could be pruned by the adjusted upper bound pruning strategy.

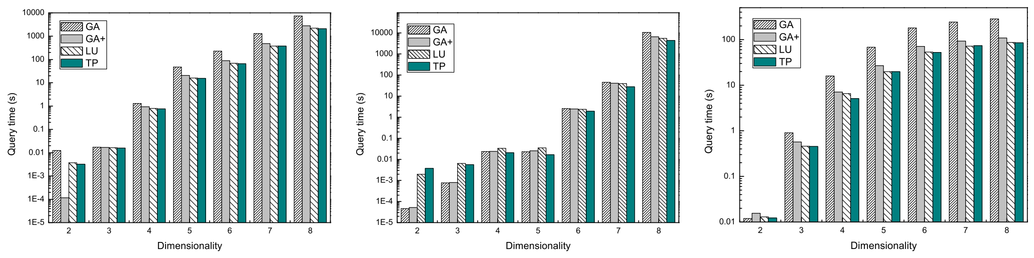
In the αTP algorithm, we adjust $DomSize^+(G)$ by multiplying $1 - \alpha$ because $DomSize^+(G)$ is always much larger than $DomSize(G)$. By reducing $DomSize^+(G)$ to a certain extent, it can prune many unqualified results without significantly reducing the accuracy of the results.

6 PERFORMANCE EVALUATION

In this section, we evaluate the performance of the proposed algorithms on both synthetic and real datasets. All our experiments were conducted on a PC with Intel^{TM} Core^{TM} I7-6700T 2.81 GHz CPU (contains four cores), 8 GB main memory, and under the Microsoft Windows 7 operating system.

Datasets. We have conducted experiments on synthetic datasets with three popular distributions: Independent (Ind), Correlated (Cor), and Anti-correlated (Ant), following the work in Börzsönyi et al. [2001]. Furthermore, we employ the real dataset, NBA, which is also utilized in the related work [Liu et al. 2015; Liu et al. 2018; Wang et al. 2018]. Specifically, NBA was obtained from http://stats.nba.com. It contains 5,000 players and each player has 5 attributes, points, rebounds, assists, steals, and blocks.

The GA, GA+, LU, and TP algorithms for the kGSky query have been implemented. Here, GA and GA+ are the group-based algorithm in Section 4.1 without and with the upper bound pruning strategy (Lemma 4.6), respectively. By removing line 11 which checks if $DomSize^+(G)$ exceeds the current kth highest score r from Algorithm 1, we have the GA algorithm.



(a) N=10^4, l=3, k=40 (b) N=10^5, l=5, k=40 (c) N=10^3, l=3, k=40

Fig. 4. Experimental results vs. dimensionality d. (a) Ind; (b) Cor; and (c) Ant.

It is worth to notice that for the kGSky query, LU and TP can report the results in a progressive manner as proven in Theorems 4.20 and 4.21. This is different from the GA algorithm which cannot get any results before completing the calculation of the entire result set.

We analyze the effect of the important parameters that are dimensionality d, cardinalities of datasets N, group size l, and number of results returned k, on the proposed algorithms, respectively. In each experiment, we vary one parameter and other parameters are fixed. The fixed values of the parameters for the Ind, Cor, and Ant datasets may be different due to the query time. Given the same parameters, the processing time of the proposed algorithms over the Cor datasets may be too small and has no significant difference. For the Ant datasets, some algorithms may take prohibitively long time.

6.1 Performance of the Exact Algorithms

In this subsection, we evaluate the performance of the proposed algorithms in processing the kGSky query.

6.1.1 Effect of Dimensionality d. In the first set of experiments, we study the influence of the dimensionality d on the performance of the proposed algorithms.

As shown in Figure 4, the dimensionality d has a great impact on the performance of the four algorithms. The query time of GA, GA+, LU, and TP all increases with the growth of d in most cases. This is because the query time of the above algorithms is closely related to the number of points $p \in SL_i$ for $1 \leq i \leq l$ which grows exponentially with the increase of d. Assume that the points within the given dataset D are uniformly distributed in each dimension, there are no points $p \in D$ sharing the same value in any dimension. The expected cardinality of points $p \in SL_i$ is:

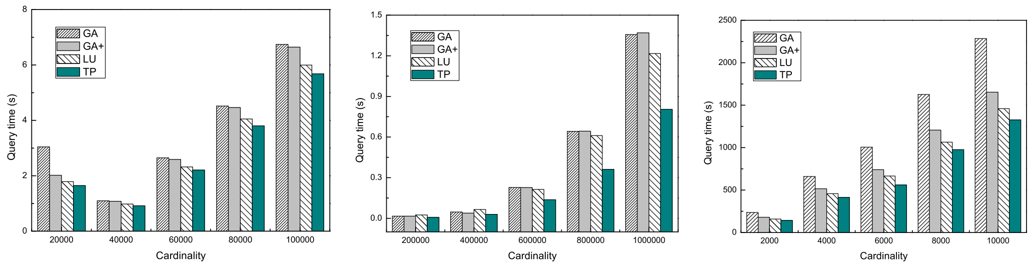
$$|SL_i| \approx \frac{(\ln N_i + \gamma)^{d-1}}{(d-1)!}.$$

Here, $N_1 = N$, $N_i = N - \sum_{j=1}^{i-1} |SL_i|$ for $2 \leq i \leq k$, and γ is equal to 0.577 [Ding and Jin 2012]. Accordingly, we have:

$$\sum_{i=1}^{l} |SL_i| \approx \sum_{i=1}^{l} \frac{(\ln N_i + \gamma)^{d-1}}{(d-1)!}.$$

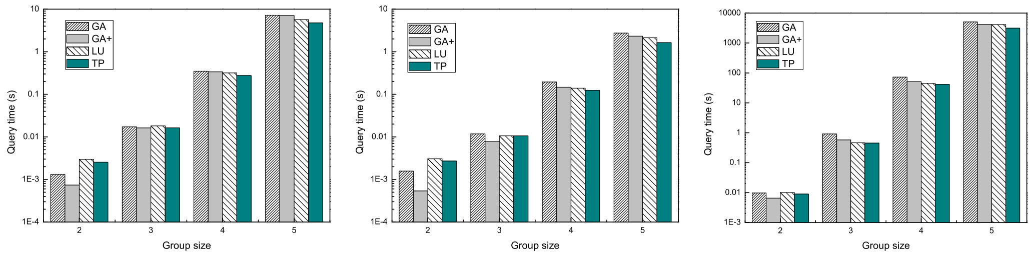
By applying the upper bound pruning strategy, GA+, LU, and TP all outperform GA, and TP gets the best performance in most cases. This is as expected since TP only generates partial G-Skylines that may be the potential results of the kGSky query while GA, GA+, and LU need to compute all the G-Skylines, and then pick out the best k G-Skylines according to their dominance sizes.

6.1.2 Effect of Cardinality N. In this second set of experiments, we demonstrate the impact of the cardinality N on the proposed algorithms.



(a) d=3, l=4, k=40 (b) d=4, l=5, k=40 (c) d=3, l=4, k=40

Fig. 5. Experimental results vs. cardinality N. (a) Ind; (b) Cor; and (c) Ant.



(a) d=3, N=10^4, k=40 (b) d=6, N=10^5, k=40 (c) d=3, N=10^3, k=40

Fig. 6. Experimental results vs. group size l. (a) Ind; (b) Cor; and (c) Ant.

Figure 5 depicts the query time of the proposed algorithms with the growth of N. In general, the query time of the proposed algorithms increases as the cardinality N grows. From Figure 5, GA+ outperforms GA which proves the effectiveness of the upper bound pruning strategy (Lemma 4.6). Among the four algorithms, TP needs the least query time.

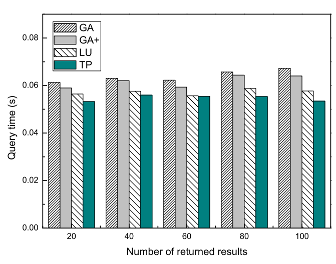
Figure 5(a) also illustrates that it needs less query time to process the Ind dataset with $N = 40{,}000$ by the four algorithms than to handle the Ind dataset with $N = 20{,}000$. This is reasonable since the query time of these algorithms is closely related to the size of the index, MDG or our LMDG, but not the cardinality of the dataset [Wang et al. 2018]. MDG and LMDG both only contain the points in the first l skyline layers. Generally, we have $\sum_{i=1}^{l} |SL_i|$ is far less than N, and it may reduce with the growth of N.

6.1.3 Effect of Group Size l. In this third set of experiments, we study the effect of the group size l on the performance of the four algorithms.

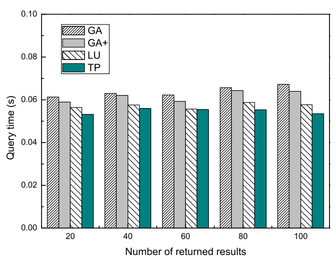
Figure 6 shows query time of the GA, GA+, LU, and TP algorithms when l increases over the Ind, Cor, and Ant datasets, respectively. Similarly to the dimensionality d, the group size l affects the performance of the four algorithms significantly. This is also as expected. As pointed out in Liu et al. [2015], the G-Skylines only contain the points within the first l skyline layers. As l goes up, the cardinality of candidate groups sharply goes up. Besides, the larger l, the more G-Skylines. The time cost to compute dominance sizes of the G-Skylines increases accordingly.

Again, GA+ has better performance than GA. Although TP and LU require comparable time, TP is slightly better than LU in most cases. This is because LU needs to compute all the G-Skylines while TP only computes the ones whose dominance sizes exceed r.

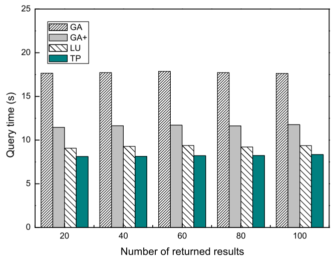
6.1.4 Effect of Number of Returned Results k. In the fourth set of experiments, we evaluate the performance of the proposed algorithms by varying the number of returned results k.



(a) d=3, N=2 $*$ 10^4, l=3

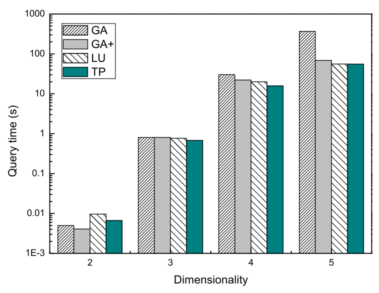


(b) d=4, N=10^5, l=6

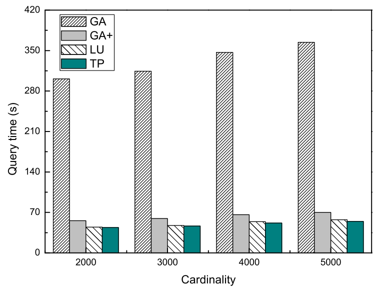


(c) d=3, N=2 $*$ 10^3, l=3

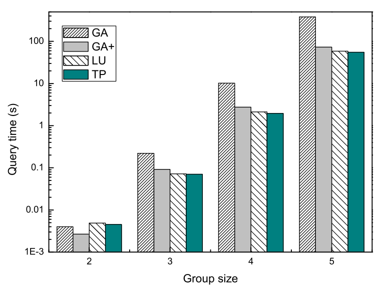
Fig. 7. Experimental results vs. number of returned results k. (a) Ind; (b) Cor; and (c) Ant.



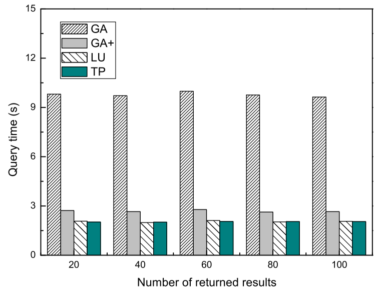
(a) N=5 $*$ 10^3, l=5, k=40



(b) d=5, l=5, k=40



(c) N=5 $*$ 10^3, d=5, k=40



(d) N=5 $*$ 10^3, d=5, l=5

Fig. 8. Experimental results over NBA. (a) d. (b) N. (c) l (d) k.

As shown in Figure 7, all the four algorithms are insensitive to different k values. This is similar to the top k dominating query in Miao et al. [2015]. Form Figure 7, GA+ needs less query time with comparing to GA. As k increases, TP and LU are both better than GA+, and TP gets the best performance again.

6.1.5 Performance Over the NBA Dataset. In this subsection, we present the experimental results over the real dataset, NBA, which is a popular dataset also utilized in Liu et al. [2015] and Wang et al. [2018].

Figure 8 shows experimental results over NBA by varying different parameters. From Figure 8, the conclusions gained from synthetic datasets can also be verified. The query time of the proposed algorithms increases sharply with the increase of d or l. Besides, TP is most friendly for large d or large group size l. Over NBA, the four algorithms are insensitive to the parameters N and k. As the results over synthetic datasets, TP is better than LU and always requires the least query time among the proposed algorithms.

Table 3. Experimental Results vs. Cardinality N over Ant ($l = 4, k = 40, d = 3$)

N	Processing Time (s)					Approximate Ratio			
	TP	βTP(0.9)	βTP(0.8)	βTP(0.7)	βTP(0.6)	βTP(0.9)	βTP(0.8)	βTP(0.7)	βTP(0.6)
2000	144.587	107.015	62.656	34.723	17.569	1.00	0.70	0.40	0.18
4000	439.103	299.039	165.935	95.073	49.751	1.00	0.80	0.55	0.35
6000	569.443	411.377	233.977	125.139	67.231	1.00	1.00	0.50	0.00
8000	976.491	678.733	381.542	224.797	120.198	1.00	1.00	0.00	0.00
10000	1316.960	1003.340	560.908	319.984	181.381	0.98	0.93	0.03	0.00

Table 4. Experimental Results vs. k over Ant ($N = 10^3, l = 4, d = 3$))

k	Processing Time (s)					Approximate Ratio			
	TP	βTP(0.9)	βTP(0.8)	βTP(0.7)	βTP(0.6)	βTP(0.9)	βTP(0.8)	βTP(0.7)	βTP(0.6)
20	40.861	31.002	17.570	10.205	5.452	1.00	1.00	0.45	0.40
40	41.616	34.219	17.601	10.302	5.403	1.00	1.00	0.53	0.50
60	41.717	31.866	17.566	10.207	5.537	1.00	1.00	0.58	0.52
80	42.115	32.270	18.545	10.301	5.399	0.99	0.99	0.59	0.54
100	43.169	32.515	18.318	11.229	8.729	0.99	0.99	0.56	0.52

Table 5. Experimental Results vs. Dimensionality d over Ant ($N = 10^3, l = 4, k = 40$)

d	Processing Time (s)					Approximate Ratio			
	TP	βTP(0.9)	βTP(0.8)	βTP(0.7)	βTP(0.6)	βTP(0.9)	βTP(0.8)	βTP(0.7)	βTP(0.6)
2	0.162	0.106	0.065	0.041	0.022	0.95	0.53	0.38	0.28
3	41.495	31.184	17.753	10.305	5.237	1.00	1.00	0.53	0.500
4	839.272	596.882	338.869	179.850	96.369	1.00	1.00	1.00	0.28
5	4412.000	3635.390	2276.360	1290.470	704.826	1.00	0.93	0.08	0.58

6.2 Performance of the Approximate Algorithms

In this subsection, we evaluate the βTP and αTP algorithms by comparing them with the TP algorithm, which is the proposed exact algorithm having the best performance, in term of processing time and approximate ratio (AR). Here, the AR is the percentage of the approximate results returned by βTP or αTP that also exist in the exact results. The larger the AR, the better the quality of the approximate results.

We report experimental results of the two approximate algorithms over the Ant datasets. Noteworthy, the conclusions from the experiments over the Ind and Cor datasets are consistent with the ones obtained from the Ant datasets.

Tables 3–6 depict results of βTP by varying β from 0.9 to 0.6. With the reduction of β, the processing time of βTP decreases. It is because only β percent of the points within SL_1 are considered, and a partial candidate groups are generated. When β is set to no less than 0.8, βTP can get a good AR that is mostly close to 1.00.

Tables 7–10 report results of αTP by varying α from 0.2 to 0.5. As α increases, the processing time of αTP dramatically reduces in most cases. This is as expected because the adjusted upper bound pruning strategy could identify and prune much more unqualified groups with the growth of α. As shown in Tables 7–10, the AR of αTP are almost all close to 1.00 when α is not more than

Table 6. Experimental Results vs. Group Size l over Ant ($N = 10^3, d = 3, k = 40$)

l	Processing Time (s)					Approximate Ratio			
	TP	βTP(0.9)	βTP(0.8)	βTP(0.7)	βTP(0.6)	βTP(0.9)	βTP(0.8)	βTP(0.7)	βTP(0.6)
2	0.011	0.009	0.007	0.007	0.005	1.00	0.93	0.73	0.65
3	0.4532	0.369	0.245	0.164	0.102	1.00	1.00	0.65	0.55
4	41.865	31.574	18.209	10.117	5.437	1.00	1.00	0.53	0.50
5	3065.820	2100.220	1045.090	530.928	243.324	1.00	1.00	0.60	0.58

Table 7. Experimental Results vs. Cardinality N over Ant ($l = 4, k = 40, d = 3$))

N	Processing Time (s)					Approximate Ratio			
	TP	αTP(0.2)	αTP(0.3)	αTP(0.4)	αTP(0.5)	αTP(0.2)	αTP(0.3)	αTP(0.4)	αTP(0.5)
2000	144.587	89.699	59.897	42.737	38.720	1.00	1.00	0.98	0.13
4000	439.103	278.964	184.743	122.412	109.851	1.00	1.00	0.95	0.13
6000	569.443	354.847	239.293	171.250	156.660	1.00	1.00	1.00	0.00
8000	976.491	614.838	413.710	286.902	258.883	1.00	1.00	1.00	0.05
10000	1316.960	810.072	547.860	399.189	382.478	1.00	1.00	1.00	0.13

Table 8. Experimental Results vs. k over Ant ($N = 1000, l = 4, d = 3$))

k	Processing Time (s)					Approximate Ratio			
	TP	αTP(0.2)	αTP(0.3)	αTP(0.4)	αTP(0.5)	αTP(0.2)	αTP(0.3)	αTP(0.4)	αTP(0.5)
20	40.861	24.117	16.552	12.851	12.161	1.00	1.00	1.00	0.35
40	41.616	25.660	16.994	13.087	12.241	1.00	1.00	1.00	0.35
60	41.717	25.176	17.184	13.096	12.274	1.00	1.00	0.98	0.28
80	42.115	25.768	17.584	13.292	12.397	1.00	1.00	0.98	0.30
100	43.169	26.827	18.298	16.875	13.645	1.00	1.00	0.98	0.31

Table 9. Experimental Results vs. Dimensionality d over Ant ($N = 1000, l = 4, k = 40$))

d	Processing Time (s)					Approximate Ratio			
	TP	αTP(0.2)	αTP(0.3)	αTP(0.4)	αTP(0.5)	αTP(0.2)	αTP(0.3)	αTP(0.4)	αTP(0.5)
2	0.162	0.148	0.143	0.127	0.081	1.00	1.00	1.00	0.75
3	41.495	24.696	16.919	13.046	12.178	1.00	1.00	1.00	0.35
4	839.272	591.335	558.586	552.578	551.521	1.00	1.00	0.78	0.05
5	4412.000	4058.140	4035.590	3979.430	3964.244	1.00	1.00	1.00	0.05

Table 10. Experimental Results vs. Group Size l over Ant ($N = 1000, d = 3, k = 40$))

l	Processing Time (s)					Approximate Ratio			
	TP	αTP(0.2)	αTP(0.3)	αTP(0.4)	αTP(0.5)	αTP(0.2)	αTP(0.3)	αTP(0.4)	αTP(0.5)
2	0.011	0.008	0.008	0.0083	0.008	0.95	0.48	0.20	0.08
3	0.453	0.313	0.277	0.260	0.252	1.00	1.00	0.30	0.05
4	41.865	25.095	17.270	13.382	12.434	1.00	1.00	1.00	0.35
5	3065.820	2125.560	1387.990	755.180	514.248	1.00	1.00	1.00	0.80

0.4. This means that the approximate results reported by the αTP are very close to the exact results and in some cases, just the same. However, when varying α from 0.4 to 0.5, the processing time of αTP gains a little reduction but the AR reduces significantly. This is because many G-Skylines with large dominance sizes are pruned by the adjusted upper bound pruning and the threshold r cannot be refreshed in time. As analyzed above, by taking into account the processing time and the AR, α is better to be set not more than 0.4.

In summary, αTP can get better AR with comparing to βTP. However, βTP is much more friendly to large or high-dimensional datasets. Besides, with a large group size, βTP performs better than αTP.

7 CONCLUSIONS

In this article, we formulate the kGSky query which aims to compute the k G-Skylines with the highest dominance sizes. Besides, we propose some pruning strategies and introduce several new techniques such as the grouping strategy, the layered optimum strategy, and the hybrid strategy to boost the query performance. In addition, as well as the exact algorithms, we propose two approximate algorithms to get better efficiency with sacrificing some accuracy of the results. The efficiency, scalability, and accuracy of the proposed algorithms have been demonstrated by abundant experiments. As our future work, we will study the kGSky query under multi-core environment or on the GPUs. Moreover, it is also interesting to research the batch G-Skyline query with different group sizes.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable and helpful comments on improving the manuscript. Kenli Li is the corresponding author of this article.

REFERENCES

Daichi Amagata, Takahiro Hara, and Makoto Onizuka. 2018. Space filling approach for distributed processing of top-k dominating queries. *IEEE Transactions on Knowledge and Data Engineering* 30, 6 (2018), 1150–1163.

Mei Bai, Junchang Xin, Guoren Wang, Luming Zhang, Roger Zimmermann, Ye Yuan, and Xindong Wu. 2016. Discovering the k representative skyline over a sliding window. *IEEE Transactions on Knowledge and Data Engineering* 28, 8 (2016), 2041–2056.

Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. 2001. The skyline operator. In *Proceedings of the 17th International Conference on Data Engineering*. 421–430.

Yu-Chi Chung, I-Fang Su, and Chiang Lee. 2013. Efficient computation of combinatorial skyline queries. *Information Sciences* 38, 3 (2013), 369–387.

Antonio Corral and Michael Vassilakopoulos. 2005. On approximate algorithms for distance-based queries using r-trees. *Computer Journal* 48, 2 (2005), 220–238.

Xiaofeng Ding and Hai Jin. 2012. Efficient and progressive algorithms for distributed skyline queries over uncertain data. *IEEE Transactions on Knowledge and Data Engineering* 24, 8 (2012), 1448–1462.

Yunjun Gao, Lu Chen, Xinhan Li, Bin Yao, and Gang Chen. 2015a. Efficient k-closest pair queries in general metric spaces. *VLDB Journal* 24, 3 (2015), 415–439.

Yunjun Gao, Qing Liu, Baihua Zheng, Li Mou, Gang Chen, and Qing Li. 2015b. On processing reverse k-skyband and ranked reverse skyline queries. *Information Sciences* 293 (2015), 11–34.

Xixian Han, Jianzhong Li, and Hong Gao. 2017. Efficient top-k dominating computation on massive data. *IEEE Transactions on Knowledge and Data Engineering* 29, 6 (2017), 1199–1211.

Hyeonseung Im and Sungwoo Park. 2012. Group skyline computation. *Information Sciences* 188 (2012), 151–169.

Maria Kontaki, Apostolos N. Papadopoulos, and Yannis Manolopoulos. 2011. Continuous top-k dominating queries. *IEEE Transactions on Knowledge and Data Engineering* 24, 5 (2011), 840–853.

Ken C. Lee, Wang Chien Lee, Baihua Zheng, Huajing Li, and Yuan Tian. 2010. Z-SKY: An efficient skyline query processing framework based on Z-order. *VLDB Journal* 19, 3 (2010), 333–362.

Xuemin Lin, Yidong Yuan, Qing Zhang, and Ying Zhang. 2007. Selecting stars: The k most representative skyline operator. In *Proceedings of the IEEE 23rd International Conference on Data Engineering*. IEEE, 86–95.

Jinfei Liu, Li Xiong, Jian Pei, Jun Luo, and Haoyu Zhang. 2015. Finding pareto optimal groups: Group-based skyline. *Proceedings of the VLDB Endowment* 8, 13 (2015), 2086–2097.

Jinfei Liu, Juncheng Yang, Li Xiong, Jian Pei, and Jun Luo. 2018. Skyline diagram: Finding the voronoi counterpart for skyline queries. In *Proceedings of the IEEE 34th International Conference on Data Engineering*. 653–664.

Hua Lu, Christian S. Jensen, and Zhenjie Zhang. 2011. Flexible and efficient resolution of skyline query size constraints. *IEEE Transactions on Knowledge and Data Engineering* 23, 7 (2011), 991–1005.

Matteo Magnani and Ira Assent. 2013. From stars to galaxies: Skyline queries on aggregate data. In *Proceedings of the 16th International Conference on Extending Database Technology*. 477–488.

Matteo Magnani, Ira Assent, and Michael L. Mortensen. 2014. Taking the big picture: Representative skylines based on significance and diversity. *VLDB Journal* 23, 5 (2014), 795–815.

Xiaoye Miao, Yunjun Gao, Baihua Zheng, Gang Chen, and Huiyong Cui. 2015. Top k dominating queries on incomplete data. *IEEE Transactions on Knowledge and Data Engineering* 28, 1 (2015), 252–266.

Dimitris Papadias, Yufei Tao, Greg Fu, and Bernhard Seeger. 2005. Progressive skyline computation in database systems. *ACM Transactions on Database Systems* 30, 1 (2005), 41–82.

Bagus Jati Santoso and Ge Ming Chiu. 2014. Close dominance graph: An efficient framework for answering continuous top-k dominating queries. *IEEE Transactions on Knowledge and Data Engineering* 26, 8 (2014), 1853–1865.

Yufei Tao, Ling Ding, Xuemin Lin, and Jian Pei. 2009. Distance-based representative skyline. In *Proceedings of the IEEE 25th International Conference on Data Engineering*. IEEE, 892–903.

Eleftherios Tiakas, George Valkanas, Apostolos N. Papadopoulos, Yannis Manolopoulos, and Dimitrios Gunopulos. 2016. Processing top-k dominating queries in metric spaces. *ACM Transactions on Database Systems* 40, 4 (2016), 1–38.

Qian Wan, Raymond Chi-Wing Wong, Ihab F. Ilyas, M. Tamer Özsu, and Yu Peng. 2009. Creating competitive products. *Proceedings of the VLDB Endowment* 2, 1 (2009), 898–909.

Changping Wang, Chaokun Wang, Gaoyang Guo, Xiaojun Ye, and Philip Yu. 2018. Efficient computation of g-skyline groups. *IEEE Transactions on Knowledge and Data Engineering* 30, 4 (2018), 674–688.

Wenhui Yu, Jinfei Liu, Jian Pei, Li Xiong, Xu Chen, and Zheng Qin. 2020. Efficient contour computation of group-based skyline. *IEEE Transactions on Knowledge and Data Engineering* 32, 7 (2020), 1317–1332.

Nan Zhang, Chengkai Li, Naeemul Hassan, Sundaresan Rajasekaran, and Gautam Das. 2014. On skyline groups. *IEEE Transactions on Knowledge and Data Engineering* 26, 4 (2014), 942–956.

Zhenjie Zhang, Laks V. S. Lakshmanan, and Anthony K. H. Tung. 2009. On domination game analysis for microeconomic data mining. *ACM Transactions on Knowledge Discovery from Data* 2, 4 (2009), 1–27.

X. Zhao, Y. Wu, W. Cui, X. Du, Y. Chen, Y. Wang, D. L. Lee, and H. Qu. 2017. SkyLens: Visual analysis of skyline on multi-dimensional data. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2017), 246–255.

Xu Zhou, Kenli Li, Guoqing Xiao, Yantao Zhou, and Keqin Li. 2016a. Top k favorite probabilistic products queries. *IEEE Transactions on Knowledge and Data Engineering* 28, 10 (2016), 2808–2821.

Xu Zhou, Kenli Li, Zhibang Yang, and Keqin Li. 2019. Finding optimal skyline product combinations under price promotion. *IEEE Transactions on Knowledge and Data Engineering* 31, 1 (2019), 138–151.

Xu Zhou, Kenli Li, Zhibang Yang, Guoqing Xiao, and Keqin Li. 2018. Progressive approaches for pareto optimal groups computation. *IEEE Transactions on Knowledge and Data Engineering* 31, 3 (2018), 521–534.

X. Zhou, K. Li, Y. Zhou, and K. Li. 2016b. Adaptive processing for distributed skyline queries over uncertain data. *IEEE Transactions on Knowledge and Data Engineering* 28, 2 (2016), 371–384.

Haoyang Zhu, Xiaoyong Li, Qiang Liu, and Zichen Xu. 2020. Top-k dominating queries on skyline groups. *IEEE Transactions on Knowledge and Data Engineering* 32, 7 (2020), 1431–1444.

Received August 2019; revised March 2020; accepted April 2020