Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 3404906, 11 pages
https://doi.org/10.1155/2022/3404906

Research Article

Optimized Query Algorithms for Top-K Group Skyline

Jia Liu,[1] Wei Chen,[1] Ziyang Chen,[2] Lin Liu,[3] Yuhong Wu,[1] Kaiyu Liu,[4] Amar Jain,[5,6] and Yasser H. Elawady[7]

[1]*Department of Information Engineering, Hebei University of Environmental Engineering, Qinhuangdao, China*
[2]*School of Information and Management, Shanghai Lixin University of Accounting and Finance, Shanghai, China*
[3]*Qinhuangdao Vocational and Technical College, Qinhuangdao, China*
[4]*School of Information Science and Engineering, YanShan University, Qinhuangdao, China*
[5]*Research Scholar, Department of Civil Engineering, Faculty of Engineering and Technology, Madhyanchal Professional University, Bhopal, India*
[6]*Sanskriti University, Mathura, India*
[7]*Engineering Dept., Misr Higher Institute of Engineering & Technology, Mansoura, Egypt*

Correspondence should be addressed to Lin Liu; liulin@qvc.edu.cn

Received 22 October 2021; Revised 7 December 2021; Accepted 11 December 2021; Published 4 January 2022

Academic Editor: Deepak Kumar Jain

Skyline query is a typical multiobjective query and optimization problem, which aims to find out the information that all users may be interested in a multidimensional data set. Multiobjective optimization has been applied in many scientific fields, including engineering, economy, and logistics. It is necessary to make the optimal decision when two or more conflicting objectives are weighed. For example, maximize the service area without changing the number of express points, and in the existing business district distribution, find out the area or target point set whose target attribute is most in line with the user's interest. Group Skyline is a further extension of the traditional definition of Skyline. It considers not only a single point but a group of points composed of multiple points. These point groups should not be dominated by other point groups. For example, in the previous example of business district selection, a single target point in line with the user's interest is not the focus of the research, but the overall optimality of all points in the whole target area is the final result that the user wants. This paper focuses on how to efficiently solve top-k group Skyline query problem. Firstly, based on the characteristics that the low levels of Skyline dominate the high level points, a group Skyline ranking strategy and the corresponding SLGS algorithm on Skyline layer are proposed according to the number of Skyline layer and vertices in the layer. Secondly, a group Skyline ranking strategy based on vertex coverage is proposed, and corresponding VCGS algorithm and optimized algorithm VCGS+ are proposed. Finally, experiments verify the effectiveness of this method from two aspects: query response time and the quality of returned results.

1. Introduction

Skyline query are also called maxima or Pareto [1] (to gain optimality without harming the interests of others in the field of business management). It is also a query optimization problem. Skyline query is proposed by Borzsonyi et al. [2], and it is introduced to the database domain at the 2001 ICDE conference at first. From then on, Skyline query attracts extensive attentions of the domestic and foreign researchers and becomes one of the most difficulty and hotspot in database-research field. Skyline query has lots of applications in the field of multidimensional optimization analysis such as choosing petrol stations and hotels in the road network, selecting players in social networks, and determining targets through multiple attribute information.

The Skyline has been differently extended in recent years and becomes an emphasis for research in the database domain. At present, there are still many researches on single point query based on the traditional Skyline, such as the Skyline query on the data stream [3] and on the subspace [4–8]. In the skyline query on the data stream, with the dynamic change of data stream tuples, for a given constraint query,

find the nodes that fall into the valid area or affect the result tuple set. Such queries are often applied to intelligent transportation, online monitoring, and other fields. In the face of massive high-dimensional data, the whole space skyline query has the disadvantages of too large result set and low efficiency; so, the subspace skyline query has more important research significance. To reduce the size of the result set and feedback some representative Skyline points, k -dominated Skyline-defined variant is given by Skyline [9, 10], Top-k Skyline query, distance-based classical Skylines [11], etc. In many cases, what we search is a point group made up of s points not a single point. For example, in the road network query, people want to find adjacent malls which meet their demands. These malls form a cluster and are connected on the route to shopping. In turn, people can recognize hotels and entertainment within the Skyline according to the distribution dense of shopping malls, which is usually called site selection analysis. Liu et al. [12] first extend the Skyline based on an original single point to the Skyline based on the point group and propose the corresponding algorithm for Skyline. In practical applications, such objective optimization problems can also be applied to path optimization [13] to calculate the minimum cost path, mobile trajectory tracking [14, 15] to look for similar trajectories, social networks to find close communities, and graph correlation [16] to get the correlation degree of the target point.

Top-k is a typical query problem in large-scale data processing, which is widely used in daily query, such as the analysis and summary of the top 10 query words in search engines. The Top-k is introduced into the group Skyline query, and each query returns the best k-Skyline point groups to reduce the burden of further selection by setting the measurement index. To solve the group Skyline problem, this paper proposes some efficient algorithms. The main research work of this paper has four points.

(1) Combining with the practical application requirements, this paper introduces the Skyline query problem of Top-k group, makes theoretical analysis and exploration on this problem, and puts forward the criterion to evaluating the quality of the point-group. Taking the number of vertices in the Skyline layer as the basis for sorting the results, an SLGS algorithm is proposed

(2) Aiming at the ranking strategy of skyline layer, the concept of vertex coverage is proposed to deal with the situation that the ranking of result point groups is the same. To avoid blindness in selection, the VCGS algorithm based on vertex coverage is proposed, which further ranks all result sets and returns the Top-k point groups

(3) For optimizing the algorithm and improving efficiency, the VCGS + algorithm is proposed. By pruning Skyline layer, the number of enumerated result sets and redundant traversal operations is significantly reduced, and the efficiency of the algorithm is improved

(4) Some experiments based on multiple real data sets are carried out, and the performance of different methods is compared from query response time and the quality of the returned results. The validity and accuracy of the proposed algorithms are verified

2. Background Knowledge and Related Work

2.1. Background Knowledge

Definition 1. (Dominance). Given a set P which contains n data points in d-dimensional spaces, let p and p' be two different points in the set P. If $p[i] \le p'[i]$ are in all dimensions, and at least one dimensional $p[i] < p'[i]$, $p[i]$ is the i dimension of point p for $1 \le i \le d$, and then p dominates p'.

Definition 2. (Strictly dominance). Given a set P which contains n data points in d-dimensional spaces, let p and p' be two different points in the set P. If $p[i] \le p'[i]$ in all dimensions for $1 \le i \le d$, then p strictly dominates p'.

Definition 3. (Group dominance). Given a set P which contains n data points in d-dimensional spaces, $G = \{p_1, p_2, \cdots, p_s\}$ and $G' = \{p_1', p_2', \cdots, p_s'\}$ are two different point groups with s points of P. We can say that the G group dominates G' if we can find two permutations of the s points for G and G', $G = \{p_{u_1}, p_{u_2}, \cdots, p_{u_s}\}$ and $G' = \{p_{v_1}, p_{v_2}, \cdots, p_{v_s}\}$, such that p_{us} dominates p_{us}' for all i ($1 \le i \le$s), and p_{us} dominates p_{us}' strictly for at least one i.

Definition 4. (Skyline). Given a set P which contains n data points in d-dimensional spaces, Skyline is a set of points that are not dominated by other points in P.

Definition 5. (Group Skyline). Group Skyline is a set of point groups that are not dominated by other point groups.

2.2. Related Work Analysis. This paper mainly focuses on how to obtain the top-k Skyline point groups. Top-k [12, 16–20] Skyline query is a common problem in large-scale data processes. The group Skyline query is to compute the set of point groups which are not dominated by other point groups on a given dataset. It is a further extension of the traditional Skyline query. Up to now, there are few researches on group Skyline query, the group Skyline is put forward to and researched in Ref. [21–24]. In recent years, effective query results [25–27]get more attention. For reducing the size of the query result set and returning more representative Skyline points, the variations of Skyline definitions such as k -dominated Skyline [24], representative Skyline [28], top-k Skyline query, and distance-based representative Skyline [29] are given. Basic algorithms of group Skyline query include algorithm PointWise [12], UnitWise [12], and UnitWise+ [12].

In Ref. [24], the definition of group Skyline is first proposed, and the definition of group domination depends on a certain aggregated point or a representative point in a

point group. Although many aggregation functions, such as the function summation, minimum, and maximum, can be used to calculate aggregation points, finding all group Skyline sets is not easy.

PointWise algorithm enumerates candidate group Skyline by dynamically generating set enumeration tree containing candidate group and pruning off nongroup Skyline group. Firstly, the directed Skyline graph is preprocessed, the redundant nodes are filtered out, and then the remaining points in the graph are enumerated. The pruning strategy: if a point group is not a group Skyline in the enumeration process, then it need not be extended, and the subtree rooted by it can be pruned. Each candidate set corresponds to an extended set of points, which can filter out some points in the set and further reduce the enumeration. The verified point groups are the final point group Skyline.

UnitWise algorithm expands candidate group by adding point groups one by one. Similarly, the candidate groups are enumerated by dynamically generating a set enumeration tree containing candidate groups. Each node in the tree is a set of unit groups. At the same time, the candidate skyline groups are listed by pruning off the other useless point groups to the greatest extent. The pruning strategy: the candidate point group G contains at least s points, then the number of candidate point groups in G's subtree will be larger than s, the subtree can be pruned, and some points in the set of extended points corresponding to candidate point group can be filtered. The algorithm is based on cell group expansion, reduces the number of enumerations, and is more efficient than PointWise.

UnitWise+ algorithm is an improved algorithm based on UnitWise. In order to delete more point groups of nongroup Skyline in advance, the algorithm first processes the high-level points in Skyline layer, enumerates larger candidate point groups in advance, and also filters the set of extended points corresponding to candidate point groups to reduce the size of the set. Moreover, depth-first traversal is used to detect candidate point groups to terminate the algorithm in advance, which can further narrow the query range and improve the effectiveness of the result set.

Although these algorithms reduce the size of candidate sets and the number of enumeration point groups, the result set is still large when the size, dimension, and the number of point groups are enlarged. We need to extract some appropriate point groups as the result to return. To overcome this shortcoming, the SLGS algorithm based on Skyline layer, and VCGS algorithm based on vertex coverage and improved algorithm VCGS+ are proposed.

3. Query Algorithm Based on Skyline Layer

3.1. Ranking Strategy. Firstly, the characteristics of the result set are discussed, and the criteria to measure the quality of the result set are put forward. The following analysis is combined with an example. As shown in Example 3.1, Table 1 is a set of hotel data sets.

The Skyline layer of the point set is constructed as follows: Firstly, 12 points are sorted in ascending order according to their attribute value-distance (users can choose the attribute value according to their preferences), and each point is processed sequentially. Point p_1 is the first point, and the next point p_7 is processed. The other point on the first layer cannot dominate p_7; so, the point p_7 belongs to the first layer. The next Skyline layer is constructed by processing p_4, where the point on the first layer dominates p_4. By analogy, until all points are processed. The results are shown in Figure 1.

Based on the definition of Skyline layer, the directed Skyline graph of the point set is constructed. In Figure 1, index value of the point is omitted (the point index value of point p_1 is 0, the point index value of point p_7 is 1, the point index value of point p_{12} is 2, and so on). According to the Skyline layer, the directed Skyline graph results are numbered sequentially from lower to higher levels, as shown in Figure 2.

According to the definition of Skyline layer, the points on the first layer are defined as Skyline points of the whole point set P, which dominate the points on other layers except the first layer; the points on the second layer are Skyline points of the subset of the set P except the points on the first layer; that is, the points on the second layer dominate the points on other layers except the points on the first layer and the second layer. By analogy, it can be concluded that the point at the lower level dominates the point at the higher level. According to the definition of point domination, the values of low-level points on some attributes are not worse than that of high-level points, and the value of low-level points on at least one attribute is better than that of high-level points. Therefore, the more points from the lower level in a point group, the better the point group.

The number of points from different Skyline layers in the Skyline point group is to identify which are better or worse, so that the Skyline point groups can be sorted and the top-k Skyline point groups can be obtained. From the above analysis, it can be seen that the number of points from the lower Skyline layer in the Skyline point group is a key factor affecting the overall group's quality. Thus, the following definitions about Skyline point group are derived.

Definition 6. (G is better than G'). Given the two point groups G and G' in group Skyline that have not dominant relationship each other, M_i and $M_i{}'$ represent, respectively, the number of points on the i-th skyline layer. If $M_i = M_i{}'$ ($0 \le \mathrm{i} \le \mathrm{laye\,size} - 1$), M_{i+1} and $M_{i+1}{}'$ will be compared, until the number of points from the i-th layer in G is greater than G', or in all the skyline layers, M_i is always equal to $M_i{}'$. If $\forall i(0 \le i \le \mathrm{layer\,size} - 1), \exists M_i > M_i{}'$, then G is better than G'.

Definition 7. (G is equivalent to G'). Given the two point groups G and G' in group Skyline that have not dominant relationship each other, if $\forall$i(0$\le$ i $\le$layers size-1), $\exists M_i = M_i{}'$, then G is equal to G'.

3.2. Algorithmic Description. Through the ranking strategy proposed above, k optimal groups of Skyline points can be obtained by processing the result set calculated by the

Table 1: A set of hotel data.

Hotel	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}
Distance	4	30	24	14	36	26	8	34	20	40	28	16
Price	400	390	380	340	300	280	260	220	210	200	120	60

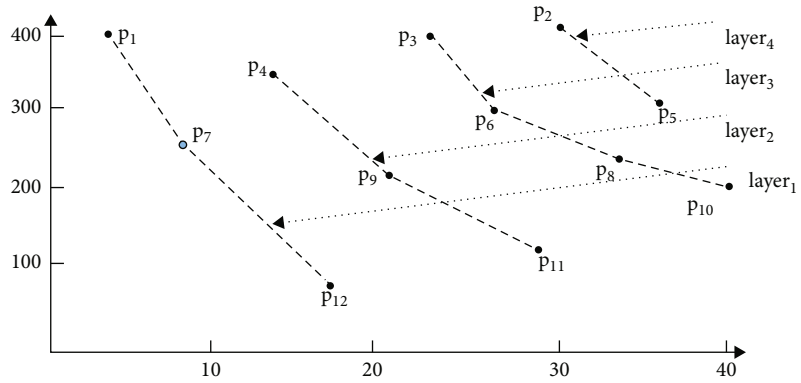

Figure 1: The Skyline layer.

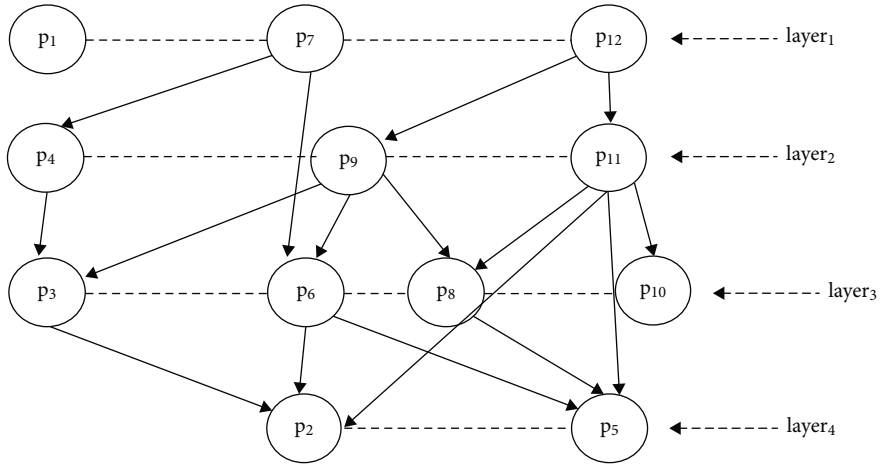

Figure 2: Directed Skyline graph.

UnitWise+ algorithm, but the result set is disorderly. The best point group may be located anywhere in the result set, which brings the adverse effects to the user's choice. Because the first result is not sure to be the best, the whole result set needs to be sorted. Based on the ranking strategy, it can be concluded that there are some equivalent and indistinguishable point groups in the ranking process, which are packaged into a block to distinguish different groups of equivalents. That is to say, after processing the result set, different equivalent point group blocks are formed. The first block includes the best point group, the second block includes the better point group, and the last block naturally contains the worst point group. The results are well organized and hierarchical. Based on this idea, a SLGS query algorithm based on Skyline layer is proposed.

The basic idea of the algorithm is given the result point group R; based on the ranking strategy proposed above, each point group in R is traversed, and the equivalent point groups are divided into blocks; then, each block is sorted, and the blocks are dynamically inserted into the corresponding positions. Finally, k point groups from the block result set are extracted. The following is a simple flow chart of the algorithm SLGS.

Example 8. Given a set of hotel data, let $s = 4$ (the size of the result point group). Based on the constructed directed Skyline graph, all group Skyline point groups can be enumerated, including $a = \{p_6, p_7, p_{12}, p_9\}$, $b = \{p_8, p_{12}, p_9, p_{11}\}$, $c = \{p_1, p_7, p_{12}, p_4\}$, $d = \{p_1, p_7, p_{12}, p_9\}$, $e = \{p_1, p_7, p_{12}, p_{11}\}$, $f = \{p_1, p_{12}, p_9, p_{11}\}$, $g = \{p_7, p_{12}, p_4, p_9\}$, $h = \{p_7, p_{12}, p_4, p_{11}\}$, $i = \{p_7, p_{12}, p_9, p_{11}\}$, $j = \{p_1, p_{12}, p_{11}, p_{10}\}$, $l = \{p_7, p_{12}, p_{11}, p_{10}\}$, and $n = \{p_{12}, p_9, p_{11}, p_{10}\}$.

For the results set $R = \{a, b, c, d, e, f, g, h, i, j, l, n\}$, if the user wants to select 5 optimal groups from 12 given result

sets, that is, set $k = 5$, then the execution process of the algorithm is as follows. First, initializing the tag array mark [] $= \{0\}$, which means that all point groups are not accessed, and the block set C is empty. Second, traversing point group a, it is found that the number of points from the first, second, and third levels is 2, 1, and 1, respectively, and a is equivalent to point group j and l. The corresponding position of tag array *mark* is assigned to 1, and the block C_1 composed of three point groups (a, j, l) is added to the head of C. Then, the next unvisited point group b is processed, and the number of points from the first, second, and third layers is 1, 2, and 1, respectively, which is equivalent to point group n. The corresponding position of tag array *mark* is assigned to 1. Point groups n and b form a block C_2, which is inserted into C. At this time, the block C_1 already exists in the set C. Because the number of point groups from the lower layer is more than that from the upper layer, the midpoint group of C_1 is better than that of C_2. Therefore, the block C_2 is inserted at the end of the C. By analogy, point groups d, e, and c are equivalent, and they are composed of the block C_3. Because C_3 is better than C_1 and C_2, C_3 is inserted into the head of set C. Point group f is equivalent to g, h, and i, and they formed the block C_4. By comparing the number of points from lower level, it is found that C_4 is better than C_1 and C_2, but worse than C_3; so, C_4 should be inserted in front of C_1. At this time, all R point groups have been accessed, and a complete set of 4 blocks $C = \{C_3, C_4, C_1, C_2\}$ has been obtained. Given $k = 5$, because the number of point groups of the first block C_1 in C is 3 and less than k, selecting two point groups from the second block C_4 is needed. Finally, the five point groups of c, d, e, f, and g are returned.

Assuming that there are n elements and e blocks in the result set R, the time complexity of the algorithm SLGS involves two aspects: (1) traversing n elements and realizing the block processing, whose time complexity is $O(n)$. (2) Finding the equivalence point group has to traverse n elements again. At the same time, order each block in the set of blocks by the binary search method. This time, the time complexity is $O(n + \log e)$. Therefore, the overall time complexity of the algorithm SLGS is $O(n(n + \log e))$.

4. Query Algorithm Based on Vertex Cover

4.1. The Basic Idea. According to the directed Skyline graph in Figure 2, it is found that if the number of points that can be dominated by points from other layers is different, then the number of points that can be dominated by different point groups is also different. For example, the size of the given point group $G = \{p_1, p_7, p_{12}, p_4\}$ is 4, and the number of points dominated for p_1, p_7, p_{12}, and p_4 is 0, 5, 8, and 2, respectively. The sum of points that this point group can dominate is 15. Similarly, the sum of points is 18 for another given point group $G' = \{p_1, p_7, p_{12}, p_9\}$. Obviously, the point group G' is better than G. It means that the sum of points that can be dominated by all the point groups also affect the overall quality of the Skyline point group. Therefore, the concept of vertex coverage is proposed.

Definition 9. (Vertex coverage). Given a point group $G = \{p_1, p_2, \cdots p_s\}$ in group Skyline, Let $n_1, n_2, \cdots n_s$ separately represents the number of points dominated by p_i in G, so that $S = \text{sum}(n_1, n_2, \cdots n_s)$ is the number of vertex covers, and we name S as VC (vertex coverage) (G).

According to the ranking strategy of 3.1, the characteristic of the Top-k Skyline point-group is as follows:

(1) There are more points on the Skyline low layer in the point group

(2) The number of vertex cover of the point group is larger

Through the above analysis, the accurate Top-k groups can be obtained by further sorting the partition results of SLGS, and the corresponding VCGS (Vertex Coverage Group Skyline) algorithm is proposed. The basic algorithm idea is given the result point group R, first run the algorithm SLGS, using Skyline layer and the number of vertices in the layer as the basis of the result ranking, get the result C composed of the blocks, and then traverse the point groups in each block of C. While traversing, the better or worse point group between blocks is judged by the size of the vertex cover set. Then, these equivalent point groups are reordered; after each block has been processed, k point groups can be extracted. The algorithm VCGS is shown in Algorithm 2.

Example 10. Given the result set returned by example 8 is $C = \{(c, d, e), (f, g, h, i), (a, l, j), (b, n)\}$, let array store the number of points of each vertex dominants. By traversing c_i in C, we can get $S(\text{VC}(G))$ of each point group in per partition c_i. S of the c, d, and e is 15,18, and 17, respectively. Because d is better than e, and e is better than c, block c_1 is reordered as (d, e, c). In the same way, c_2, c_3, and c_4 are reordered as (i, g, h, f), (a, j, l), and (b, n). Now, $C = \{(d, \text{e}, c), (i, g, h, f), (a, j, l), (b, n)\}$. if $k = 5$, the result will return d, e, c, i, g. if $k = 1$, then the result is d, not c like example 8.

4.2. Algorithm Optimization. The algorithm is very sensitive to the size of block set n and the number of elements m of each block. This is mainly because the values of n and m will be very large when the size of data sets, dimensions, and point groups increases; so, it will take more time to traverse these elements. In fact, it is not necessary to calculate the whole result set. The algorithm UnitWise$^+$ enumerates all the points on Skyline layer. Assuming that the number of Skyline points on the first layer is n_1, we select s points from the n_1 points from enumeration. If the enumeration value e_1 is greater than or equal to k, that is to say, the point group generated by the points on the first layer is enough to find k optimal results, then the Skyline layer higher than the first layer can be pruned, and the point group composed of the points on the first layer can be sorted directly. If e_1 is less than k, indicating that the point groups on the first layer are not enough to find k results, add the second layer and enumerate the Skyline points on the first two layers. If the enumeration number e_2 is greater than or equal to k, the Skyline layer higher than the second layer can be pruned,

```
Input: The number of result k⊠Group-Skyline R.
Output: top-k Group-Skyline groups(S).
Init array mark[] = {0}, C←∅
for each group G_i ∈ Rdo
  ifG_i has visited then
    continue;
for each group G_j ∈ Rdo
  ifG_j is not visited and G_i is equals to G_jthen
    add G_j to a existing chunk c
    else ifG_j is not visited and j == ithen
      add G_i to a new chunk c.
  ifC == ∅then.
    add chunk c to C
  else ifC!= ∅then
      for each c_i in Cdo
      ifc_i better than the head then
        insert c_i into head, break
      else ifc_i worse than the last then
        insert c_i into last, break
      else
        insert c_i into the corresponding position, break
S ← select k Group-Skyline groups from C 1.
returnS1
```

Algorithm 1: SLGS.

```
Input: Group-Skyline R, Directed Skyline graph (DSG), The number of result k
Output: top-k Group-Skyline groups(S)
C ← SLGS(R)
number(i) ← {0}
for each p_i∈do
  number(i) ← the number of p_i's children
for each C_i∈ C do
  for each group G_i∈ C_i do
    use number[] to compute VC for G_i
  use quick sort to sort the group in C_i
S ← select k Group-Skyline groups from C
returnS
```

Algorithm 2: VCGS.

the points on the first two layers can be enumerated, and the result points can be sorted to find k optimal point groups, and so on. In this way, the algorithm UnitWise$^+$ can be judged earlier in the execution process and do not enumerate the invalid point groups. Based on the above analysis, the optimized algorithm VCGS$^+$ is introduced and shown in Algorithm 3.

4.3. Comparison and Analysis of Three Algorithms. Algorithm 1 has the problem of too much enumeration and computation, and in the equivalent tuple, it cannot compare which point in the tuple is better. Algorithm 2 can further distinguish the advantages and disadvantages of points in tuples, but there is still a large amount of calculation, and many useless point groups participate in the calculation. Therefore, Algorithm 3 is optimized from three aspects: enumerator, pruning strategy, and selection of equivalence points in the group.

5. Experiment and Result Analysis

5.1. Experimental Environment. The hardware and software platforms used in the experiment are Intel (*R*) Pentium (*R*) CPU with a main frequency of 2.9 GHz, 1 TB hard disk, 4 GB RAM memory, and 64-bit Windows 7 Professional OS. The experimental programming environment of all algorithms is Microsoft Visual Studio 2010, and the programming language is C++.

In experiment, the parameters d, n, s, and k represent, respectively, the dimension of the data set, the size of the data set, the size of the points group required, and the number of best groups returned.

Input: Directed Skyline graph (DSG), The size of point-group *s*the number of results *k*
Output: top-*k* Group-Skyline groups(*S*)
1 $n \leftarrow$ the number of point on $layer_1$
2 **if***n* > *s***then**
3 $c \leftarrow$ FreeCombination(*n*,*s*)
4 **if***c* ≥ *k***then**
5 *layer*' ← the first layer in DSG
6 ReConstruct_DSG(*layer*',*s*)
7 $G \leftarrow$ UnitWise$^+$(*layer*',*s*)
8 **else**
9 **for** each $layer_i (1 < i \leq s)$ in DSG **do**
10 **for** each $layer_j (j \leq i)$ in DSG **do**
11 *layer*" ← add $layer_j$ to the *layer*"
12 ReConstruct_DSG(*layer*",*s*)
13 ConstructDSG(*layer*',*s*)
14 $G \leftarrow$ UnitWise$^+$(*layer*",*s*)
15 **if** |*G*| ≥ *k***then**
16 **break**;
17 **if***i*==*s* and |*G*| < *k***then**
18 **break**;
19 $C \leftarrow$ SLGS(*G*)
20 *C*' ← VCGS(*C*)
21 S ← select *k* Group-Skyline groups from *C*'
22 **return***S*

Algorithm 3: VCGS+.

Table 2: Information description of data sets.

Dataset	Data number	Attr_1	Attr_2	Attr_3	Attr_4	Attr_5
NBA	2500	Score	Backboard	Assist	Steals	Blockshot
NHL	3000	GoalNumber	Assist	Score	+/-score	WinNumber

Table 3: Information description of data sets.

Parameter description	Default	Minimum value	Maximum value
λ	5	3	8
n	1000	500	3000
s	5	3	6
k	5	1	200

5.2. The Data Set and Evaluation Criteria. The experiment uses the two real datasets for NBA (http://stats.nba.com/leaders/alltime/?ls=iref:nba:gnav) player statistics and NHL (htttps://http://www.nhl.com/player/) player statistics. The information description of each data set is shown in Table 2. The experiment tests and compares the dimension, scale, size of point groups, and the number of returned results. The specific settings are shown in Table 3.

The dimension of data set λ is as follows: the number of attributes contained in the target point set.

Data set size n is as follows: the number of target points.

The size of the point group s is as follows: the number of target points contained in each point group.

Returns the number of optimal point groups k: the number of elements in the result set returned to the user.

5.3. The Performance Comparison and Analysis

5.3.1. The Influence of the Size of the Point Group. As can be seen from Figure 3, the execution time of each algorithm increases with the increase of s. When the s is small, the execution speed of the three algorithms is very fast, and the execution speed of SLGS and VCGS is similar. Compared with SLGS, VCGS takes a little longer to execute because VCGS algorithm is a further sorting processing of point groups on the basis of the calculation results of SLGS algorithm, which increases the execution time. With the increase of s, the execution time of SLGS and VCGS increases exponentially, and the number of points on the former Skyline layer will increase sharply, resulting in the increase of the Skyline point groups and data scale. However, the execution speed of the improved VCGS$^+$ algorithm based on VCGS is better than the other two algorithms, and the best case is 10 times the worst case.

As can be seen from Figure 4, the enumeration results of the three algorithms increase with the increase of s value. When the s value is small, the enumeration results of the three algorithms are not much different, and the

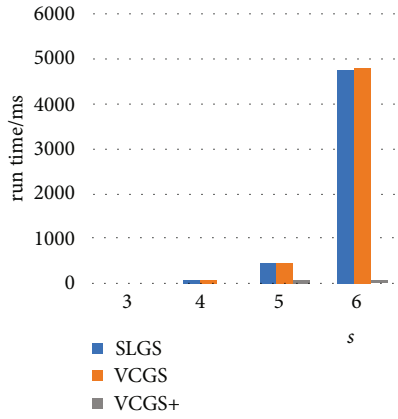

Figure 3: The run time under different s.

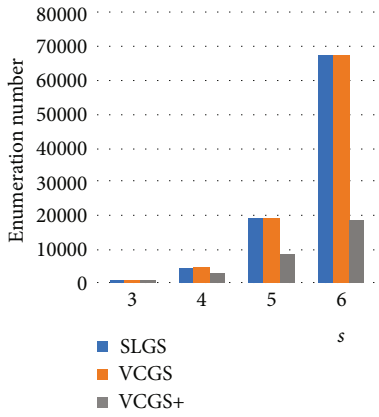

Figure 4: The number of enumeration point groups under different s.

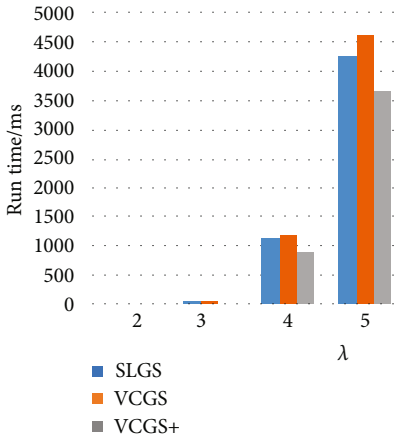

Figure 5: The run time in NBA under different λ.

enumeration number of the algorithm VCGS$^+$ is only a little less than that of the first two algorithms. Moreover, because the algorithm VCGS is a further ranking of equivalent point groups based on SLGS calculation results, the enumeration results of the two algorithms are equal. When the s value

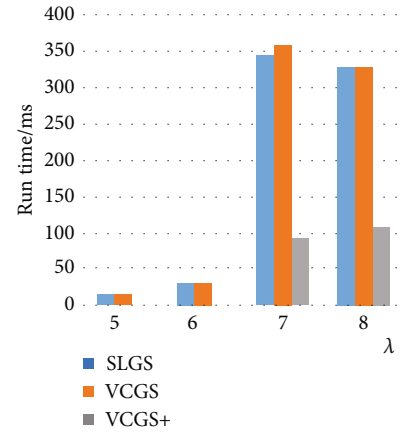

Figure 6: The run time in NFL under different λ.

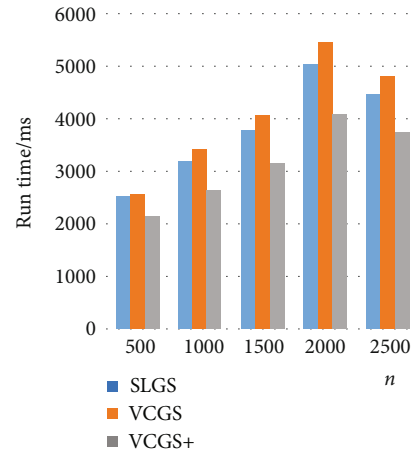

Figure 7: The run time in NBA under different n.

increases gradually, the enumeration number of SLGS and VCGS increases a lot. By pruning Skyline layer, the enumeration result of VCGS$^+$ is less affected by s.

5.4. The Influence of Data Dimensions. In Figures 5 and 6, on two different datasets, we can see that with the growth of data scale λ, the running time of the algorithm also increases, and the efficiency decreases rapidly. When the dimension of NBA dataset rises to 5, and that of NFL dataset rises to 7, the impact of SLGS and VCGS is more severe. The reason is that with the increase of λ, the number of Skyline points on each Skyline layer increases dramatically. These two algorithms need more time to calculate the group of Skyline points; so, the efficiency will become lower. Compared with the other two algorithms, VCGS$^+$ is more efficient and performs better on NFL datasets.

5.5. The Influence of Dataset's Size. In Figures 7 and 8, we can see that with the increase of target data n, the performance of the algorithm is relatively stable. Therefore, the influence of n is not obvious. The running time of the algorithm increases linearly with the increase of n. The main reason is that only the points on the former s Skyline layer are used when computing group Skyline, and the number of

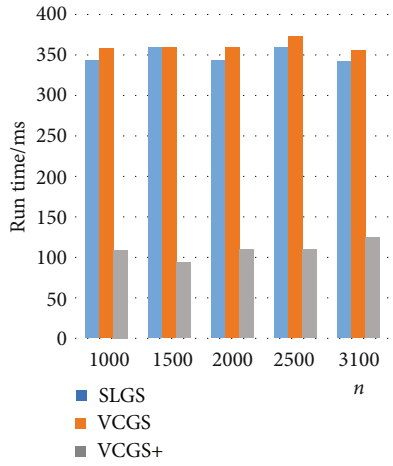

Figure 8: The run time in NFL under different n.

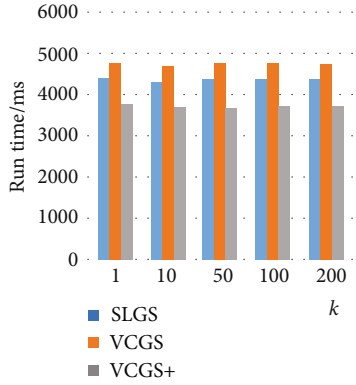

Figure 9: The run time in NBA under different k.

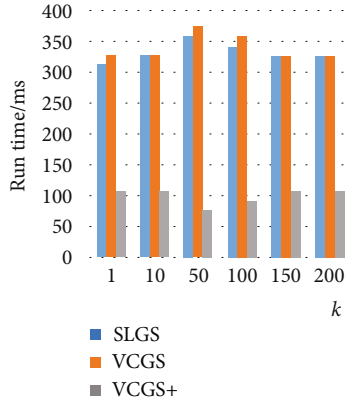

Figure 10: The run time in NFL under different k.

Table 4: The survey results.

$\{p_{11}, p_{12}\}$	$\{p_9, p_{12}\}$	$\{p_4, p_7\}$	$\{p_1, p_7\}$	$\{p_1, p_{12}\}$	$\{p_7, p_{12}\}$
8	7	7	10	12	16

these points is much smaller than the size of data n. For different data sets, the impact of data size on the overall algorithm is different. The running time of VCGS$^+$ is less than that of SLGS and VCGS, and the performance of VCGS$^+$ on NFL datasets is more drastic. The pruning strategy of this algorithm can improve the efficiency of the algorithm by two to three times.

5.6. The Influence of the Number of Point Groups Returned. In Figures 9 and 10, with the growth of the point in the result set, the efficiency of the three algorithms varies steadily and linearly. Because the number of enumerated result point groups is greatly affected by the size of calculated point groups, data dimension, and data set size, it is independent of k value. This can also be explained directly from the time complexity of the algorithm.

6. Conclusions

Aiming at the problem of large result set and low query efficiency in existing group Skyline query algorithms, the following results are obtained.

(1) Aiming at the problem of large result set and large number of meaningless result point groups in existing Skyline algorithm, the Skyline query problem of the top-k group is given, and a SLGS algorithm based on Skyline layer is proposed to return k optimal Skyline point groups. This algorithm combines the structural characteristics of the high-level points dominated by the middle and low-level points in Skyline layer and gives a quantitative criterion to find the better one of two groups. Based on this criterion, the group Skyline results are ranked. and the k results in the top ranking are returned

(2) To solve the problem of the same ranking result in SLGS algorithm, a ranking strategy based on Skyline layer and vertex coverage is proposed. The size of vertex coverage set in the point group is used as the basis of ranking, and the results with the same ranking are further processed. The corresponding VCGS algorithm is proposed to sort all the results, which makes the sorting results more accurate. Because the algorithm adopts traversal strategy, it is inefficient. In order to improve users' satisfaction with the returned results, an improved algorithm VCGS+, which is based on the algorithm VCGS, is proposed. This algorithm provides a pruning strategy of Skyline layer and avoids accessing most Skyline points. Only a few results can be calculated to find top-k groups of Skyline points, reduces the number of results enumerated and the number of points that need to be traversed, and thus improves the efficiency of the algorithm. Meantime, the experimental results show that the algorithm can improve the efficiency about ten times

(3) The proposed algorithm is validated by experiments. The experimental results verify the effectiveness of the proposed method in terms of query response time and the quality of the returned results

(4) In order to verify the effectiveness and feasibility of the algorithm, a simple test was made with the data of 12 hotels in Table 1, and a questionnaire was developed, with 30 members of the research group and 30 family members in the laboratory as the interview targets. Firstly, the skyline point group with size of 2 is listed. The results include 6 point groups: $\{p_{11}, p_{12}\}$, $\{p_9, p_{12}\}$, $\{p_4, p_7\}$, $\{p_1, p_7\}$, $\{p_1, p_{12}\}$, and $\{p_7, p_{12}\}$. Table 4 shows the choices made by the target population

The investigation results are consistent with the algorithm results, which proves the effectiveness of the proposed algorithm in practical application. At the same time, this study can be applied to various site selection analysis, such as school district housing selection, division of business district, and the location of public facilities. It has a high theoretical value in the application of location-based services.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was partially supported by the grants from the Hebei Education Department Key Project (No. ZD2021037) and Hebei University of Environmental Engineering Key Project (No. 2020ZRZD03).

References

[1] V. Pareto, *Cours Deconomie Politique, [M.S. thesis]*, F. Rouge, Lausanne, 1896.

[2] S. Borzsonyi, D. Kossmann, and K. Stocker, "The skyline operator," in *Proceedings 17th International Conference on Data Engineering*, pp. 421–430, Heidelberg, Germany, 2001.

[3] J. Yang, B. Qu, P. Li, and H. Chen, "DC-Tree:.an algorithm for skyline query on data streams," in *Advanced Data Mining and Applications: International Conference on Advanced Data Mining and Applications*, Lecture Notes in Computer Science, pp. 644–651, Springer, Berlin, Heidelberg, 2008.

[4] Y. D. Yuan, X. M. Lin, Q. Liu, W. Wang, J. X. Yu, and Q. Zhang, "Efficient computation of the skyline cube," in *Proceedings of the 31st International Conference on Very Large Data Bases,*, pp. 241–252, Trondheim, Norway, 2005.

[5] L. B. Han, Y. Wu, and G. Nong, "Succinct suffix sorting in external memory," *Information Processing and Management*, vol. 58, no. 1, article 102378, 2021.

[6] J. Pei, W. Jin, M. Ester, and Y. Tao, "Catching the best views of skyline: a semantic approach based on decisive subspaces," in *Proceedings of the 31st International Conference on Very Large Data Bases*, pp. 253–264, Trondheim, Norway, 2005.

[7] M. Olteanu, A. Hazan, M. Cottrell, and J. Randon-Furling, "Multidimensional urban segregation: toward a neural network measure," *Neural Computing and Applications*, vol. 32, no. 24, pp. 18179–18191, 2020.

[8] Y. F. Tao, J. Pei, and X. K. Xiao, "Efficient skyline and top-k retrieval in subspaces," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 8, pp. 1072–1088, 2007.

[9] H. Tian, M. A. Siddique, and Y. Morimoto, "An efficient processing of K-dominant skyline query in mapreduce," in *Proceedings of the First International Workshop on Bringing the Value of Big Data to Users*, p. 29, New York, USA, September 2014.

[10] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang, "Selecting stars: the K most representative skyline operator," in *Proceedings of the 2007 IEEE International Conference on Data Engineering*, pp. 86–95, Istanbul, Turkey, 2007.

[11] Y. F. Tao, L. Ding, X. M. Lin, and J. Pei, "Distance-based representative skyline," in *Proceedings of the 2009 IEEE 25th International Conference on Data Engineering*, pp. 892–903, Shanghai, China, April, 2009.

[12] A. Yu, P. K. Agarwal, and J. Yang, "Top-K preferences in high dimensions," in *Proceedings of the 2014 IEEE 30th International Conference on Data Engineering*, pp. 748–759, Chicago, IL, USA, 2014.

[13] W. Chen, Z. B. Lou, and Q. Z. Yang, "A shortest path query algorithm based on hanging vertex association index," *Journal of Yanshan University*, vol. 42, no. 3, pp. 265–271, 2018.

[14] J. Y. Zhu, C. Zhang, H. Zhang et al., "Pg-causality: identifying spatiotemporal causal pathways for air pollutants with urban big data," *IEEE Transactions on Big Data*, vol. 4, no. 4, pp. 571–585, 2018.

[15] A. Sundareswaran and K. Lavanya, "Real-time vehicle traffic prediction in apache spark using ensemble learning for deep neural networks," *International Journal of Intelligent Information Technologies*, vol. 16, no. 4, pp. 19–36, 2020.

[16] Z. Yan, W. Cao, and J. Ji, "Social behavior prediction with graph U-net+," *Discover Internet of Things*, vol. 1, no. 1, p. 18, 2021.

[17] G. Das, D. Gunopulos, and N. Koudas, "Answering top-k queries using views," in *Proceedings of the VLDB Endowment*, pp. 451–462, Seoul, Korea, September, 2006.

[18] J. Lee, G. W. You, and S. W. Hwang, "Personalized top-k skyline queries in high-dimensional space," *Information Systems*, vol. 34, no. 1, pp. 45–61, 2009.

[19] C. Li, N. Zhang, N. Hassan, N. Hassan, S. Rajasekaran, and G. Das, "On skyline groups," in *Proceedings of the 21st ACM international conference on Information and knowledge management*, pp. 2119–2123, New York, USA, November 2012.

[20] A. Sohail, M. A. Cheema, and D. Taniar, "Social-aware spatial top-k and skyline queries," *The Computer Journal*, vol. 61, no. 11, pp. 1620–1638, 2018.

[21] H. Im and S. Park, "Group skyline computation," *Information Sciences*, vol. 188, no. 1, pp. 151–169, 2012.

[22] M. Magnani and I. Assentt, "From stars to galaxies: skyline queries on aggregate data," in *Proceedings of the 16th International Conference on Extending Database Technology*, pp. 477–488, New York, USA, 2013.

[23] N. Zhang, C. K. Li, N. Hassan, S. Rajasekaran, and G. Das, "On skyline groups," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 4, pp. 942–956, 2013.

[24] C. Wang, C. K. Wang, G. Y. Guo, X. J. Ye, and P. S. Yu, "Efficient computation of G-skyline groups," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 4, pp. 674–688, 2018.

[25] H. Zhu, P. D. Zhu, X. Y. Li, and Q. Liu, "Top-K skyline groups queries," in *Proceedings of the 20th International Conference on Extending Database Technology*, pp. 442–445, Venice, Italy, 2017.

[26] Z. B. Yang, X. Zhou, K. L. Li, G. Q. Xiao, Y. J. Gao, and K. Q. Li, "Efficient processing of top k group skyline queries," *Knowledge-Based Systems*, vol. 182, no. 15, p. 104795, 2019.

[27] H. Y. Zhu, X. Y. Li, Q. Liu, and Z. C. Xu, "Top-kdominating queries on skyline groups," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 7, pp. 1431–1444, 2020.

[28] R. Mao, T. T. Cai, R. H. Li, J. X. Yu, and J. Li, "Efficient distance-based representative skyline computation in 2D space," *World Wide Web*, vol. 20, no. 4, pp. 621–638, 2017.

[29] T. T. Cai, R. H. Li, J. X. Yu, R. Mao, and Y. D. Cai, "Efficient algorithms for distance-based representative skyline computation in 2D space," in *Asia-Pacific Web Conference:Web Technologies and Applications*, Lecture Notes in Computer Science, pp. 116–128, Springer, Cham, 2015.