An Efficient Algorithm to Compute Compositional Skyline

Xiaowei Cui$^1$ and Leigang Dong $^{1,2}$

$^1$Computer Science and Information Technology Department, Daqing Normal University, Daqing 163712, China
$^2$College of Information Science and Technology, Donghua University, Shanghai 201620, China
Email: Leigang Dong.Lgdong010@163.com

Abstract. In recent years, finding outstanding groups from the dataset has become an important research emphasis of skyline query. In this paper, we propose efficient algorithms to compute the compositional skyline (C-skyline). Firstly, we present dominant scope which can be used to prune useless groups, based on which, we present the basal algorithm (BA) to compute the C-skyline (2). Secondly, in order to circularly compute the C-skyline ($k$) when $k>2$, we present a sharing theorem and propose the superior algorithm (SA). Our algorithms can increase the pruning rate and reduce the data operand obviously. In our experiments, synthetic data are used to test the algorithms, and the experiment results show the usefulness of C-skyline and the efficiency of our algorithm.

1. Introduction

With the important usefulness in multi-criteria analysis and decision making, the skyline query has been paid more and more attention. For a point $p$ of dataset, assume that there are not two same points in dataset, if there is no any other point which is better than $p$, we can say $p$ is skyline point, and the skyline of a dataset consists of all such points. These skyline points maybe not the best points, but they are the ones the user (or customer) are most interested in.

For instance, Apple Company intends to advertise new goods for sale on TV. As shown in table 1, if there are 15 TVs with two attributes, the price indicates the money to pay, and the anti-rating represents how much percent of people which are not watching TV. It is obvious that for both attributes of a TV, the less the better.

Table 1. TV station information.

| TV  | Price | Anti-Rating |
|-----|-------|-------------|
| a   | 150   | 55          |
| b   | 200   | 52          |
| c   | 240   | 56          |
| d   | 270   | 53          |
| e   | 315   | 55          |

If Apple Company plans to advertise on one TV, which one should be selected? This is a multi-criteria decision problem, and we can compute the skyline as candidates for customer to select a suitable one. As shown in figure 1, each point represents a TV with price and anti-rating. Point $p$, $q$ and $n$ in figure 1 are the points which the Apple company are most interested in, because every point of them represents a TV with lower price and lower anti-rating.
However, if Apple Company considers advertising on 2 TVs, what should we do? We can regard each two points as a composition and do the traditional skyline query on all the compositions. If we add the values of same attributes of 2 points to form a composition, each composition has a new price and anti-rating. For example, we use \( g_1 \) to represent the composition of two points \( a \) and \( b \), then \( g_1 \) can be written as \( g_1(150+200, 55+52) \). For these 15 points, the number of all the compositions of 2 points is \( \binom{15}{2}=(15*14)/(2*1)=105 \). All the compositions with 2 points in table 1 are shown in table 2. In this paper, we use summation as the function to compute the price and anti-rating of a composition. Obviously, doing the traditional skyline query on such a dataset (especially when the number of points is large) will need much more time cost and storage cost.

### Table 2. Compositions of two ad slots.

| Id | Price | Anti-rating |
|----|-------|-------------|
| \( g_1 \) (p & a) | 250 | 105 |
| \( g_2 \) (p & b) | 300 | 102 |
| \( g_3 \) (a & b) | 350 | 107 |
| ...... | ...... | ...... |
| \( g_5 \) (n & k) | 610 | 70 |
| ...... | ...... | ...... |

2. **Related Work**

The skyline query can return the outstanding items of the dataset, it is very useful in big data or cloud computing, especially in multi-criteria decision. The skyline operator was firstly studied in paper [1], then many algorithms for different applications of skyline[2-6] were proposed.

In recent years, the problem of group skyline query has been widely studied. Paper [7, 8] studied the top-k combinatorial skyline which returns the best \( k \) groups in the queue. Papers [9-14] discussed the characteristics of existing algorithms and provided guidelines on selecting algorithms for different situations. Paper [15] revealed the parallelization of the group skyline computation.

3. **Definitions**

**Definition 1.** (Composition \( c_k \)). We can use a 5-tuple \( (D, d, n, k, F) \) to express the composition \( c_k \), where, \( D \) is a dataset, \( d \) is the dimension size of \( D \), \( n \) is the number of points of \( D \), \( k \) is the size of composition, \( F \) is the functions to form the composition. It can be written as \( F=\{f_1, f_2, ..., f_d\} \), where \( f_i \) is a function to compute all the \( i \)-th attribute values of \( k \) points to the \( i \)-th attribute values of \( c_k \) (1\( \leq \)d).\n
**Definition 2 (C-Skyline).** A composition \( c_k \) belongs to the C-Skyline(k) if and only if there is no any other composition can dominate \( c_k \). The size of composition is \( k \).

For convenience, we use \( p \preceq q \) to indicate that point \( p \) dominates point \( q \), and we use \( p \oplus c_k \) to indicate composing the point \( p \) and the composition \( c_k \).

**Theorem 1.** There are two different compositions \( c_k \) and \( c'_k \) with the same size, and \( c'_k \) is dominated by \( c_k \). If there is a point \( p \) which is neither in \( c_k \) nor in \( c'_k \), we can infer that \( c_k \oplus p \) dominates \( c'_k \oplus p \).

**Proof.** Let \( c_k=[t_1, t_2, ..., t_k] \) and \( c'_k=[t'_1, t'_2, ..., t'_k] \), thus \( c_k \oplus p \) and \( c'_k \oplus p \) can be respectively written as \([t_1, t_2, ..., t_k, p]\) and \([t'_1, t'_2, ..., t'_k, p]\). The precondition that \( c_k \) dominates \( c'_k \) suggests \( f(t_i, t_2, ..., t_k) \leq f(t'_j, t'_2, ..., t'_k) \) where \( 1 \leq i \leq k \), and there must exist \( j, f(t_i, t_2, ..., t_k) \preceq f(t'_j, t'_2, ..., t'_k) \) where \( 1 \leq j \leq k \). Because \( f_i \) is a monotone function, so we can infer that \( f(t_i, t_2, ..., t_k, p) \leq f(t'_j, t'_2, ..., t'_k, p) \) where \( 1 \leq i \leq k \), and there must exist \( j, f(t_i, t_2, ..., t_k, p) \preceq f(t'_j, t'_2, ..., t'_k, p) \) where \( 1 \leq j \leq k \). So we can say \( c_k \oplus p \) dominates \( c'_k \oplus p \).

**Theorem 2.** There are two different points \( p \) and \( q \) of \( D \), and \( c_k \) is a composition. If \( p \preceq q \), and these two points do not belong to \( c_k \), we will infer that \( c_k \oplus p \preceq c_k \oplus q \).

**Proof.** We can easily prove this theorem based on theorem 1.
Theorem 3. If $c_k$ belongs to $C$-skyline$(k)$, for each point $p$ in $c_k$, we can infer that all of the points dominating $p$ are also in $c_k$.

Proof. If there is a point $q$ which dominates $p$ and is not in $c_k$, we can easily find a composition containing $p$ to dominate $c_k$, so $c_k$ does not belong to $C$-skyline$(k)$ any more.

4. The Basal Algorithm of C-skyline

4.1. The definitions of scope

Definition 3. (dominant scope) Given a point $p$, the set of all the points dominated by $p$ is called the dominant scope of $p$, denoted as $S(p)$, such as $A \cup B$ in figure 2.

Definition 4. (independent scope) Given a point $p$, the set of all the points only dominated by $p$ is called the independent scope of $p$, denoted as $IS(p)$, such as $B$ in figure 2.

Definition 5. (overlap scope) If all of points in a scope are dominated by more than one point (such as $p$ and $q$), we can call it overlap scope, denoted as $OS(p, q)$, such as $A$ in figure 2.

Figure 2. Dominant scope and independent scope.

Figure 3. Query window.

How to compute the IS($q$) while $q$ is a skyline point? As shown in figure 3, we find that the IS($q$) is not only relevant to $p$ but also $n$. Here we use $q_{front}$ (precursor) to indicate the point which is the most close to $q$ and is smaller than $q$ in $x$-dimension value, and we also use $q_{next}$ (successor) to indicate the point which is the most close to $q$ and is larger than $q$ in $x$-dimension value.

In order to find the IS of each skyline point, all of points of $D$ are organized into a R-tree and sorted with increasing $x$-coordinate. For the first skyline point $t$ in array, we set its precursor as $t_{front} = (t_x, \max(a_y))$ while $a$ belongs to $D$, and for the last skyline point $t'$ in array, we set its successor as $t'_{next} = (\max(b_x), t'_y)$ while $b$ belongs to $D$. For instance, if we intend to find the IS of a skyline point $p$, we firstly construct a query window with the minimum corner $p$ and the maximum corner ($t_{next}.x$, $t_{front}.y$). The IS of $p$ consists of all the points located in this query window. As shown in figure 3, the gray box is a query window, and the solid point indicates the maximum corner of the query window, however, this point maybe a virtual point.

4.2. The Basal Algorithm

According to the scope definitions, we can divide a dataset $D$ with 2 dimension size into some scopes. For instance, figure 3 shows all the points of table 1 and the scopes. We can find that $p$, $q$ and $n$ are the traditional skyline points, the whole dataset can be divided into six scopes while 1, 3, and 6 are independent scopes and 2, 4, 5 are overlap scopes.

Pruning strategy: When we consider computing $C$-skyline$(2)$, all the compositions with 2 points can be classified into different categories as follows according to the location of point.

1) both points are traditional skyline points, such as $p \oplus q$.
2) one point belongs to traditional skyline while another point belongs to its independent scope, such as $p \oplus a$.
3) others except for (1) and (2).

Prove: We analyze these three kinds of compositions to see whether they can be pruned safely as soon as possible.
The first kind of compositions. Let \( c_k = [t_1, t_2] \) while \( t_1, t_2 \) are skyline points. As there is no point which dominates \( t_1 \) or \( t_2 \), so we cannot directly find a composition \( c'_k \) such that \( c'_k < c_k \). So this kind of compositions cannot be pruned directly.

The second kind of compositions. Let \( c_k = [t_1, t_2] \) while \( t_1 \) is skyline point and \( t_2 \) belongs to the independent scope of \( t_1 \). Thus, we cannot find a point which dominates \( t_2 \), and we cannot always find a point which dominates \( t_2 \), so this kind of compositions cannot be pruned directly.

The third kind of compositions. Let \( c_k = [t_1, t_2] \) while \( t_1 \) and \( t_2 \) are two different points, we can easily find that there is at least one point either dominating \( t_1 \) or \( t_2 \). Assume that these two points are both non-skyline points, then there must exist a skyline point \( t \) dominating \( t_1 \), according to theorem 2 we can infer that \( c_k = [t_1, t_2] \) must dominated by \( c'_k = [t, t_2] \). Assume that \( t_1 \) is skyline point while \( t_2 \) is non-skyline point and is not in the independent scope of \( t_1 \), then there must exist a skyline point \( t \) dominating \( t_2 \), so we can conclude that \( c_k = [t_1, t_2] \) must dominated by \( c'_k = [t_1, t] \). Finally, we can infer that this kind of compositions can be pruned safely and indirectly.

Thus, only the first two kinds of compositions need to be examined further to compute the C-skyline of a dataset. The main procedure of the basal algorithm is shown in Algorithm 1.

---------------------------------------------------------------------------------------------------------------------------

Algorithm 1: basal algorithm of C-skyline in two-ds

---------------------------------------------------------------------------------------------------------------------------

Input: a dataset \( D \) with \( n \) points in two-ds
Output: C-skyline(2)
1 init \( C1=\emptyset \), \( C2=\emptyset \), \( C=\emptyset \);
2 compute skyline(\( D \));
3 for each point \( q\in \text{skyline}(D) \)
4 computing the IS(\( q \));
5 \( C1=\{ \text{compositions formed by } q \text{ and one point in } \text{IS}(q) \} \)
6 \( C= C \cup C1 \);
7 \( C2=\{ \text{all the compositions formed by any two skyline points } \} \)
8 \( C=C1 \cup C2 \)
9 C-skyline(2) = skyline(C);

---------------------------------------------------------------------------------------------------------------------------

5. The Superior Algorithm of C-skyline

**Theorem 4.** If \( c_{k+1} \) belongs to C-skyline(\( k+1 \)), there must exist a \( c_k \in C_{k+1} \), while \( c_k \) belongs to C-skyline(\( k \)).

**Prove:** For conveniently describing the process, we set \( k+1=4 \), then \( k=3 \). Assume that \( c_{k+1} = \{ t_1, t_2, t_3, t_4 \} \) is a C-skyline composition, then for each \( t_i \), \( c_{k+1} \) must contain all the points dominating \( t_i \), if not, we can easily find a composition dominating \( c_{k+1} \) according to theorem 2. We can proceed by contraposition. Suppose there is no C-skyline(3) composition burnt from \( c_4 \) which is a composition of C-skyline(4), then we can infer that:

If \( c_4 = \{ a, b, c \} \) is dominated by a \( c'_3 \), then there will be a point dominating one of points in \( c_3 \). However, according to theorem 3 we know that each point \( t_i \) of \( c_4 \) and the points dominating \( t_i \) are in \( c_3 \), so we can infer that \( d \) must dominate one of point in \( c_3 \).

If \( c_4 = \{ a, c, d \} \) is dominated by a \( c'_3 \), then we can infer that \( b \) must dominate one point in \( c_3 \).

If \( c_4 = \{ a, b, d \} \) will be dominated by a \( c'_3 \), then we can infer that \( c \) must dominate one point in \( c_3 \).

If \( c_4 = \{ b, c, d \} \) will be dominated by a \( c'_3 \), then we can infer that \( a \) must dominate one point in \( c_3 \).

Thus, each point of \( c_4 \) dominates another point in \( c_3 \), it is not true. So we are sure to find a composition skyline \( c_3 \) from \( c_4 \).

Using this theorem, we can compute the C-skyline(\( k+1 \)) based on the C-skyline(\( k \)), as each composition in C-skyline(\( k+1 \)) is expanded from a composition in C-skyline(\( k \)).

**Pruning strategy:** When each composition \( c_k \) of C-skyline(\( k \)) is expanded to candidates of C-skyline(\( k \)), only two kinds of points should be considered: (1) the skyline points not in \( c_k \); (2) the
points in the independent scope of each skyline point in $c_k$. (3) the points in each overlap scopes of some adjacent skyline points in $c_k$.

**Prove:** If there is a point $p$ which belongs to these kinds of points, we cannot certainly find another $c_{k+1}$ which can dominate $c_k \oplus p$, so this kind of compositions need to be checked further. However, for a point $q$ not in these three kinds, we can easily find a point $t \prec q$, and then we can find that $c_k \oplus q$ is dominated by $c_k \oplus t$ certainly according to theorem 1.

-----------------------------------------------------------------------------------------------------------------
Algorithm 2: superior algorithm of C-skyline in two-ds
-----------------------------------------------------------------------------------------------------------------

Input: a dataset $D$ with $n$ points in two-dimension size
Output: C-skyline($k$)
1  compute C-skyline(2);
2  for $i=2$ to $k-1$
3    for each composition $c \in$ skyline($i$)
4      compute the independent scopes and the overlap scope of all skyline points in $c$ as union $SP(c)$
5      $C_1=\{\text{compositions formed by } c \text{ and each point in } SP(c)\}$
6      $C_2=\{\text{compositions formed by } c \text{ and each skyline point not in } c\}$
7      $C_3=\{\text{compositions formed by } c \text{ and each one in } OP(\cdot) \text{ of the adjacent skyline points in } c\}$
8      $C=C_1 \cup C_2 \cup C_3$
9      C-skyline($i+1$) = skyline($C$);
10  return C-skyline($k$).

-----------------------------------------------------------------------------------------------------------------

6. Experiments

As paper [12] firstly and formally presented combinatorial skyline query, and our algorithms are much more motivated by this paper, so in the experiments, we compare our SA algorithm with DA algorithm[12] based on three kinds of synthetic datasets which are produced based on the methods in paper[1]: independent data (IND), correlated data (CORD) and anti-correlated data (ANTID).

Figure 4 (a) shows the performance on the independent dataset. We find that when the dataset size is small, the SA algorithm performs better than DA algorithm, such that the time cost of SA is nearly half of that of DA. With the dataset size increasing, the SA performs more better than DA, as the pruning strategy can prune most useless data, and the calculating process is less than DA. When the dataset size is $10^7$, the time cost of SA is two orders of magnitude less than that of DA.

Figure 4 (b) shows the performance on the correlated dataset. We can see that the time cost of SA is a little more than the time cost on IND, as this kind of dataset has less skyline points and there are much more points in the scope of each skyline point, so much cost will be needed to calculate these points. In this case, SA also performs better than DA.

Figure 4 (c) shows the performance on the anti-correlated dataset. We can see that the time cost of SA is more less, as this kind of dataset has more skyline points and there are fewer points in the scope of each skyline point, so the time cost is better than that of other two datasets. We also find that SA performs better than DA.

![Figure 4](image-url)

**Figure 4.** Computing C-skyline with different $n$. 
Figure 5 shows the influence of varying dimension size on the algorithms, where n=2000, k=3. We find that the time cost of SA increases obviously when the dimension size increases. The reason is that with the dimension size increasing, the chance of one point dominating other points becomes smaller, so there will be more skyline points, and then much more time will be needed to compute the candidates. However, with d increasing, SA performs better than DA more and more, and the time cost of SA is nearly three orders of magnitude less than that of DA.

![Graphs showing the influence of dimension size on algorithms](image1)

**Figure 5. Computing C-skyline with different d.**

Figure 6 shows the influence of varying composition size on the algorithms, where n=2000, d=5. We find that when the composition size k is larger than d, the time cost increase sharply. The reason is that most C-skyline compositions maybe contain all the skyline points if k>d, so the number of candidates will be large and much time will be needed for computation. The time cost of SA is nearly two orders of magnitude less than that of DA.

![Graphs showing the influence of composition size on algorithms](image2)

**Figure 6. Computing C-skyline with different k.**

7. **Conclusions**
In this paper, we proposed the problem of composition skyline query which returns the outstanding groups. Firstly, BA algorithm was proposed based on the dominant scope to compute the C-skyline(2). Then, we presented a theorem to compute the C-skyline (k) by expanding the compositions in C-skyline(k-1), and the superior algorithm (SA) is proposed to compute the C-skyline(k) circularly. We evaluate our algorithm based on three kinds of synthetic datasets, and the experimental results indicate the efficiency of our algorithm. In the future, we will consider introducing the cluster analysis on the composition to dig more useful information.

8. **Acknowledgments**
This work is supported by the Youth Foundation of DaQing Normal University (No.15ZR07).

9. **References**
[1] Borzonyi S, Kossmann D and Stocker K. The skyline operator[C]. ICDE, 2001. Heidelberg, Germany, 2001, pp 421–430.
[2] J Lee and SW Hwang. Toward efficient multidimensional subspace skyline computation[J]. Vldb journal, 2014,23(1): pp 129–145.

[3] T. Xia and D. Zhang. Refreshing the sky: the compressed skycube with efficient support for frequent updates[C]. SIGMOD, 2006, pp 491–502.

[4] Yuanyuan Li, Zhiyang Li and Mianxiong Dong. Efficient subspace skyline query based on user preference using MapReduce[J]. Ad Hoc Networks, 2015(35): pp 105–115.

[5] C.Y. Chan, H.V. Jagadish and K.L. Tan, et al. Finding k-dominant skylines in high dimensional space[C]. SIGMOD, 2006, pp 503–514.

[6] Xiaoye Miao, Yunjun Gao, et al. k-dominant skyline queries on incomplete data[J]. Information Sciences, 2016, 367, pp 990–1011.

[7] IF Su ,YC Chung and C Lee. Top-k combinatorial skyline queries. DSAA, 2010, 5982: pp 79–93.

[8] W Son, F Stehn and C Knauer. Top-k Manhattan Spatial Skyline Queries. Information Processing Letters, 2017, 123 (1) : pp 27–35.

[9] X Guo , H Li and A Wulamu. Efficient processing of skyline group queries over a data stream. Tinghua Science and Technology, 2016, 21(1): pp 29–39.

[10] C. Li, N. Zhang, N. Hassan and S. Rajasekaran. On skyline groups. In CIKM, 2012, pp 2119–23.

[11] N. Zhang, C.Li, N. Hassan and S.Rajasekaran. On skyline groups. IEEE trans. Knowl. Data Eng. 2014, 26(4): pp 942–956.

[12] Yu-Chi Chung, I-Fang Su and Chiang Lee. Efficient computation of combinatorial skyline queries. Information System, 2013, pp 369–387.

[13] MY Lin ,CW Yang and SC Hsueh. Efficient Computation of Group Skyline Queries on MapReduce. Journal on Computing, 2016, 5, pp 69–76.

[14] JinFei Liu, LiXiong and JianPei. Finding Pareto Optimal Groups: Group-based Skyline[C]. VLDB, 2015, 8(13): pp 2086–97.

[15] H Zhu, P Zhu and X Li. Computing skyline groups: an experimental evaluation. Acm Turing Celebration Conference-china, 2017: p 48

[16] H Zhu, P Zhu and X Li. Parallelization of group - based skyline computation for multi - core processors. Concurrency & Computation Practice & Experience, 2017,29 (18) : pp 4195–4215.