Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186–5140
Volume 5, Number 1, pages 64–67, January 2016

Find top-k representative skyline points by using grid data structure

Cheng Chang, Md. Anisuzzaman Siddique, Asif Zaman, and Yasuhiko Morimoto
Graduate School of Engineering, Hiroshima University
Kagamiyama 1-7-1, Higashi-Hiroshima 739-8521, Japan
Email: {m145661@, siddique@, d140094@, morimoto@mis.}hiroshima-u.ac.jp

***Abstract*—To analyze a database more intensively, we need some representative records, such as, the cheapest one, the most popular one, the most convenient one and so on. Skyline query and its variant are popular functions to find such representative objects from a numerical database. However, a skyline query often retrieves too many points to analyze intensively especially for high-dimensional dataset. Therefore, we put our focus on selecting top-k points that represents a database. We select the k points so that many points are dominated by the k points. This paper proposes an efficient grid-based algorithm to find such top-k representative skyline points. We conducted a set of experiments to show the effectiveness and the scalability of the propose algorithm.**

***Keywords*—Skyline; Representative points, Grid-based; Database;**

I. Introduction

Representative records, such as, the cheapest one, the most popular one, the most convenient one and so on are one of important clues to understand a database. Skyline query and its variant are popular functions to find such representative objects from a numerical database. Given a set of d-dimensional points D, the skyline consists of the points, called "skyline points", which are not dominated by another point. Let p,q be two vectors, and let $p[i]$ be the $i-th$ dimension's value of p. A point $p = (p[1], p[2], \cdots, p[d])$ dominates another point $q = (q[1], q[2], \cdots, q[d])$ if $p[i] \leq q[i]$ for $(1 \leq i \leq d)$ and there is at least one dimension j such that $p[j] < q[j]$.

Figure 2 shows an example of skyline that is calculated from a database in Figure 1. The table in Figure 1 is a list of phones, each of them contains two numerical attributes: power and price, for online booking. A user chooses a phone from the list according to her/his preference. In this situation, her/his choice usually comes from the phones in skyline. In this example, the skyline of the phones is $\{1, 2, 3, 4, 5, 6, 7\}$ (see Fig. 2).

As illustrated in the example, without any preferences the skyline query is capable to find a common subset of non-dominated points for all linear scoring functions. This intuitive nature of the query formulation has been a key strength of skyline queries. On the down side, we cannot control the number of retrieved points. Skyline queries may retrieve too many objects especially in high dimensional databaes. It is non-trivial to identify truly interesting points from large skyline result set. In the above example, it may be hard for users to make a good, quick selection by referencing all points from skyline that consists of many phones. Therefore, we put our focus on the top-k representative skyline points selection problem. We select the k points so that many points are dominated by the k points.

phone	Power(hours)	Price(thousand)
1	1	3.6
2	1.5	3.5
3	1.6	3.4
4	2	2.7
5	2.5	2.6
6	4.3	1.2
7	4.8	1
8	2.1	3.2
9	2.3	3.5
10	3	3.7
11	3.6	3.5
12	3.6	3.2
13	3.6	2.7
14	4.2	3
15	4.4	2.8
16	4.4	2.3
17	4.4	3.9
18	4.8	2.6
19	4.8	2.4
20	4.8	2.1
21	5	4
22	5.2	3.9
23	5.3	3.5
24	5.3	2.7
25	5.3	2.2
26	5.2	1.1
27	5.5	2
28	5.8	2.2
29	5.7	3.5
30	6	2.5
31	5.9	3.1
32	5.9	3.9

Fig. 1. A list of phone

In this paper, we used greedy algorithm to solve this problem. For example, the top-1 representative skyline of Figure 2 is $\{6\}$, because the number of dominated points by $\{6\}$ is the largest. After removing the dominated points by $\{6\}$, $\{4\}$ dominates the largest number of points. Therefore, the top-2 representative skyline points are $\{6,4\}$. We repeat the greedy process until we select k points.

Assume that a user want to find her/his best phone to buy from the database in Figure 1. Since it contains 32 different phones, the selection takes time. If we compute the skyline query, the number of candidates is reduced to 7. However, it is still relatively large. Moreover, as we mentioned, we cannot

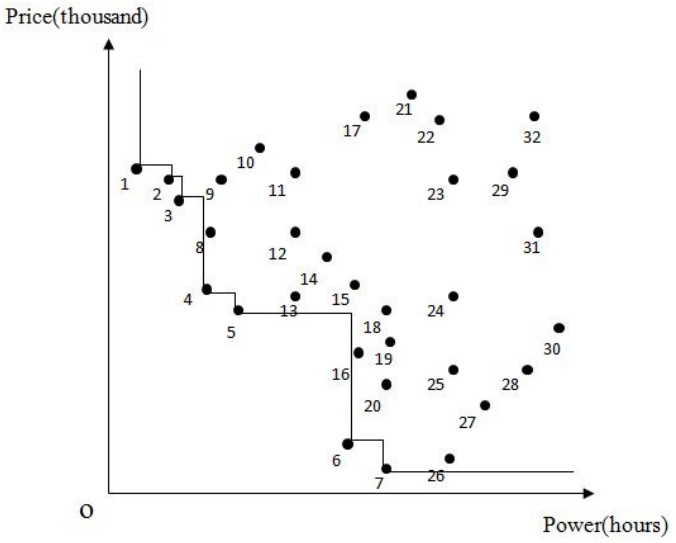

Fig. 2. A skyline example

control the size of candidates.

Our proposed approach outputs the top-k representative skyline. If $k = 5$, we can output {6,4,7,5,3} as the top-5 representative skyline. The selection problem from the top-5 representative skyline is much more easier and faster.

In summary, the contributions of this paper include following aspects:

- ☐ We studied the top-k representative skyline query.
- ☐ We developed a grid based algorithm to select the top-k representative skyline.

The rest of this paper is organized as follows. Section II reviews related work. Section III presents the notions and properties of key personnel computation using skyline query. We provide detailed examples and analysis of our algorithm in Section IV.Then, we show the experimental evaluation of our algorithm in Section V. After that, we show a short summary in Section VI. Finally, Section VII concludes the paper

II. Related Work

Skyline Query Processing

Borzsonyi et al. first introduced the skyline operator over large databases and proposed three algorithms: $Block\text{-}Nested\text{-}Loops(BNL)$, $Divide\text{-}and\text{-}Conquer$ $(D\&C)$, and B-tree-based schemes [1]. Kossmann et al. followed to improved the $D\&C$ algorithm, and proposed nearest neighbor (NN) algorithm for efficiently pruning out dominated objects by partitioning the data space iteratively based on the nearest objects in the space [5].

Similarly, Chomicki et al. improved BNL by presorting, they proposed $Sort\text{-}Filter\text{-}Skyline(SFS)$ as a variant of BNL [3]. Godfrey et al. proposed linear elimination sort for skyline $(LESS)$ algorithm for efficient computation of skyline queries [4]. Among index-based methods, Tan et al. proposed two progressive skyline computing methods Bitmap and Index [8]. The current most efficient method is $Branch\text{-}and\text{-}Bound$ $Skyline(BBS)$, proposed by Papadias et al., which is a progressive algorithm based on the *best-first nearest neighbor (BF-NN)* algorithm [7]. Instead of searching for nearest neighbor repeatedly, it directly prunes using the R*-tree structure. Meanwhile, there have been efforts for designing effective skyline structures for high-dimensional data. Yuan et al. proposed $skycube$ structure, which amortizes the cost of computing skylines over all possible subspaces [11]. Xia et al. improved $skycube$ structure and proposed CSC structure as a more concise alternative to remove duplicated skylines in the skycube by storing each skyline object only to its $minimum$ $subspace$ [10].

The first piece of work on representative skylines is paper [6]. The work of [6], however, adopts a definition of representativeness that sometimes returns skyline points that are barely representative. It is a poor choice because it has a high possibility that all the points belongs to the same set and does not indicate the trade-offs provided by other sets. Another work of representative skylines is paper [9], this paper try to use Distance-based method to find top-k. But obviously, they dont consider the dominate power between different skyline points. They just choose the points which can cover the largest distance, even if these points dominate power is low. To the best of our knowledge, the latest work about skyline operator is [2]. They present that regret minimizing sets are a recent approach to representing a dataset by a small subset R of size r of representative data points. we introduce the relaxation to k-regret minimizing sets, whereby a top-1 query on R returns a result imperceptibly close to the top-k on dataset. We introduce and use set-cover problem based on this paper.

III. Preliminaries

In this section, we present some definitions and basic properties of our algorithm.

A. Set Cover Problem

Definition 1 (Set Cover). Given a set of n elements $E = \{e1, e2, \cdots, en\}$ and a set of m subsets of E, $S = \{S1, S2, \cdots, Sm\}$, find the least number collection C of sets from S such that C contains all elements in E. That is, $\cup_{Si \in C} Si = E$.

For example, consider a universe $\cup$ contains seven elements, that is $\cup = \{p1, \cdots, p7\}$ and the set of sets $S = \{\{p1, p2, p3, p4\}, \{p2, p4\}, \{p3, p4, p5\}, \{p5, p6, p7\}\}$. Clearly the union of S is $\cup$ and we can cover all of the elements with the following, smaller number of sets: $\{\{p1, p2, p3, p4\}, \{p5, p6, p7\}\}$.

B. Dominance and Skyline

Definition 2 (Domination). For two points $p, p' in D$, point p is said to *dominate* the point p', denoted by $p \prec p'$, if $p.a_s \leq p'.a_s$ for all attributes $(s = 1, \cdots, d)$ and $p.a_x < p'.a_x$ for at least one attribute $(1 \leq x \leq d)$. We call such p as *dominant* point and such p' as *dominated* point between the two points. If p dominates p', then p is preferable than p'.

In Figure 2 point 8 dominates point 11 ($8 \prec 11$). This is because point 8 has smaller value in both attributes price and power than point 11.

Definition 3 (Domination Set). Domination set for a point p is a set of all points that are dominated by p. The domination set of p, denoted by $dom(p)$. For example from Figure 2 $dom(1) = \{10, 17, 21, 22, 32\}$.

Definition 4 (k-most Valuable Set Cover). Given a positive integer k and a dataset D, the k-most valuable set cover returns k non dominant points. And these k points can dominate largest number of different points. k-most valuable set cover of a dataset D is denoted as $S_{kmv}(D)$. If we set $k = 2$, then 2-most valuable set cover for Figure 2 is $S_{2mv}(D) = \{4, 6\}$.

Definition 5 (Skyline). A point $p \in D$ is in skyline of D (i.e., a skyline point in D) if p is not dominated by any other point in D. The skyline of D, denoted by $Sky(D)$, is the set of skyline points in D.

For the dataset in Figure 2, points $1, 2, 3, 4, 5, 6$, and 7 can dominate all other points and they are not dominated by another point. Thus, skyline query for this dataset will retrieve $\{1, 2, 3, 4, 5, 6, 7\}$.

C. Grid Construction

Our propose calculation is based on a grid computation technique. At first we randomly choose q points from dataset D. Next based on the coordinate of these q points we create the grid partition for D. Then we can get the grid partition.(see Figure 3)

Each grid cell is denoted as $C(x, y)$, where x represents grid coordinate x value and y represents grid coordinate y value. To find valuable grid we have to search those grid that contains skyline points. This is because, skyline retrieves all non dominant points. Thus cell with skyline points has better possibility to dominate many points than other cells.

Definition 6 (Skyline Grid Cell). Grid cell with skyline points are called as skyline grid cell.

For example in Figure 3 Grid C(2, 3) is an example of skyline grid cell.

A cell $C(x, y)$ has a maximum and minimum dominate power. We define two different functions to compute maximum and minimum dominate power. For a grid cell denoted as $C(x', y')$, where x' represents grid coordinate x value and y' represents grid coordinate y value. They are as follows:

Definition 7 (Max Dominate Power Function). The $MaxC(x, y)$ functions computes the maximum number of point dominated by a point $p \in C(x, y)$. Mathematically define as-

$$MaxC(x, y) = \sum |C(x', y')| - 1, (x', y')|x' \geq x, y' \geq y$$

For a grid cell $C(x, y)$ the maximum number of points are dominated by the min-corner (lower left corner). But the number should minus one, because we consider each point in this cell, so for each point in grid $C(x, y)$ can not dominate itself. It means that possible for each point can dominate all the other points in this cell, just except itself. For example in Figure 3, the grid cell C(5,1) dominates at most 11 points, so it's max dominate power is 11.

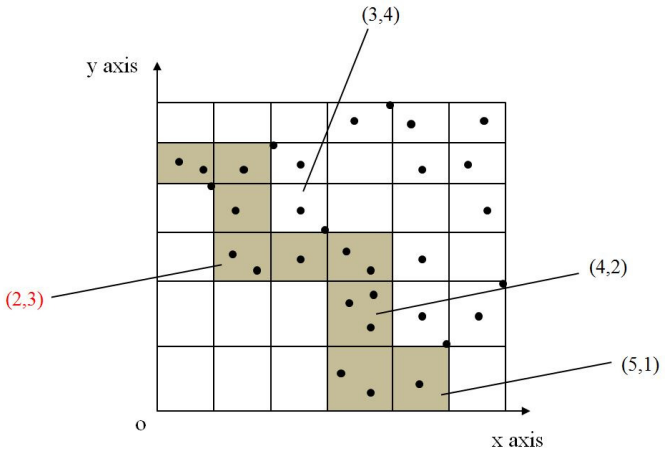

Fig. 3. The data partition.

Definition 8 (Min Dominate Power Function). The $MinC(x, y)$ functions computes the minimum number of point dominated by a point $p \in C(x, y)$. Mathematically define as-

$$MinC(x, y) = \sum |C(x', y')|, (x', y')|x' \geq x, y' \geq y$$

For example in Figure 3, the grid cell C(5,1) dominates at least 6 points, so it's min dominate power is 6.

For a grid cell $C(x, y)$ the minimum number of points are dominated by the max-corner (upper right corner). For two grid cells $C(x, y)$ and $C(x', y')$, if $max \mid C(x, y) \mid < \mid C(x', y') \mid$, then we can remove $C(x', y')$.

From Figure 3 the max dominate power of C(5,1) is 11 and the min dominate power of C(5,1) is 6. The max dominate power of C(2,3) is 19 and the min dominate power of C(2,3) is 11. We can see that the max-corner of grid cell C(2, 3) can dominates the min-corner of grid cell C(5, 1). Thus we can remove grid cell C(5, 1) from non-dominated grid, although it succeed to dominate 10 points.

IV. Algorithm

In this part, we proposed an algorithm, it is used for calculate the top-k representative skyline. We use the same example to show. After we get the non-dominated grid, we need to calculate the max dominate power and the min dominate power for every non-dominated grid.(based on definition 7 and definition 8) Then we start the first prunning.(see Figure 4)

After first prunning, we use the max dominate power sort. Then we calculate the skyline point for the first grid. In this example, it means the skyline point of C(4,1), they are 6 and 7. Then we calculate the dominate power for these skyline points. Use these dominate power for second prunning. We repeat these steps until we find the top-k representative points.

V. Experimental Evaluation

This section reports our experimental results to validate the effectiveness and efficiency of proposed method.We set up

Grid number	Min-d power	Max-d power
(1, 5)	5	11
(2, 3)	11	18
(2, 4)	8	13
(2, 5)	8	9
(3, 3)	8	14
(4, 1)	11	20
(4, 2)	7	16
(4, 3)	6	9
(5, 1)	6	10

→

Grid number	Min-d power	Max-d power
(2, 3)	11	18
(2, 4)	8	13
(3, 3)	8	14
(4, 1)	11	20
(4, 2)	7	16
(5, 1)	6	10

Fig. 4. Result of prunning

Algorithm top-k GRS-2D

Input: the integer k, the two dimension dataset D.
Output: the top-k representative points.

1. Randomly choose m points in D.
2. Get the split point array in every dimension. Denote L_x and L_y.
3. Establish a grid data structure based on these m points.
4. Calculate the non-dominated grid.
5. Calculate the max and min dominate power functions of every RS-grid, and then pruning.
6. Order by the max-dominate power, get the maximum grid.
7. Calculate the skyline of the first grid, use the dominate power of these skyline points pruning again.
8. Get the top-1 representative point. Then k=k-1.
9. Remove the points that dominated by top-1. Back to step5, until k=0.

Fig. 5. Algorithm for top-k represnetative skyline

an commodity PC in high speed Gigabit network, which has an Intel Core 2 Duo E8500 3.16GHz CPU, 8GB memory.To conduct experiments we used synthetic datasets and each experiment is repeated five times and the average result is considered for performance evaluation.

We study the effect of calculation on proposed technique. We vary the data cardinality from 100k to 400k. Also we vary the number of k from 2 to 5. The run-time result for this experiment is shown in Figure 8.

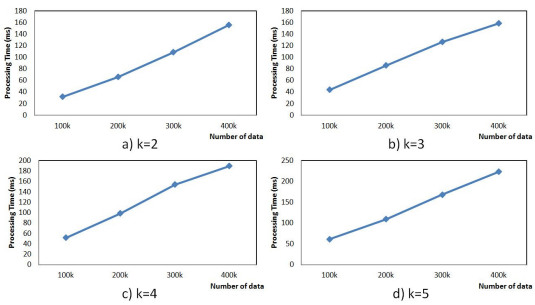

Fig. 6. Time Efficiency in 2-D space

VI. Summary

As a short summary, our performance evaluation indicates that GRS is quite efficient in 2-d space. When the data size becomes larger, the grid-based structure will be more efficiently.

VII. Conclusion

The skyline of a dataset may have a large number of points. Returning all of them may make it difficult for a user. A better approach is to present only a few representative points that represent the entire skyline.

In this paper, we investigate the problem of computing the top-k representative skyline points without calculating the skyline of the database. After introducing the novel skyline operator: top-k representative skyline points, we present an efficient dynamic programming based algorithm for a 2d-space in which an exact solution can be achieved.

Acknowledgment

This work is supported by KAKENHI (23500180, 25.03040) Japan. A. Zaman is supported by Japanese Government MEXT Scholarship.

References

[1] S. Borzsonyi, D. Kossmann, and K. Stocker. The skyline operator. In *Proceedings of ICDE*, pages 421–430, 2001.

[2] S. Chester, A. Thomo, S. Venkatesh, and S. Whitesides. Computing k-regret minimizing sets. *Proc. VLDB Endow.*, pages 389–400, 2014.

[3] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with presorting. In *Proceedings of ICDE*, pages 717–719, 2003.

[4] P. Godfrey, R. Shipley, and J. Gryz. Maximal vector computation in large datasets. In *Proceedings of VLDB*, pages 229–240, 2005.

[5] D. Kossmann, F. Ramsak, and S. Rost. Shooting stars in the sky: an online algorithm for skyline queries. In *Proceedings of VLDB*, pages 275–286, 2002.

[6] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang. Selecting stars: the k most representative skyline operator. In *Proceedings of ICDE*, pages 86–95, 2007.

[7] D. Papadias, Y. Tao, G. Fu, and B. Seeger. Progressive skyline computation in database systems. *ACM Transactions on Database Systems*, pages 41–82, 2005.

[8] K.-L. Tan, P.-K. Eng, and B. C. Ooi. Efficient progressive skyline computation. In *Proceedings of VLDB*, pages 301–310, 2001.

[9] Y. Tao, L. Ding, X. Lin, and J. Pei. Distance-based representative skyline. In *Proceedings of ICDE*, pages 892–903, 2009.

[10] T. Xia and D. Zhang. Refreshing the sky: the compressing skycube with efficient support for frequent updates. In *Proceedings of SIGMOD*, pages 491–502, 2006.

[11] Y. Yuan, X. Lin, Q. Liu, W. Wang, J. X. Yu, and Q. Zhang. Efficient computation of the skyline cube. In *Proceedings of VLDB*, pages 241–252, 2005.