An Efficient Grid Method for Continuous Skyline Computation over Dynamic Data Set

He Li
Department of Information and Communication Engineering
Chungbuk National University, Cheongju, 361-763, Korea

Su-Min Jang
Department of Information and Communication Engineering
Chungbuk National University, Cheongju, 361-763, Korea

Kwan-Hee Yoo
Department of Computer Education
Chungbuk National University, Cheongju, 361-763, Korea

Jae-Soo Yoo
Department of Information and Communication Engineering
Chungbuk National University, Cheongju, 361-763, Korea

ABSTRACT

Skyline queries are an important new search capability for multi-dimensional databases. Most of the previous works have focused on processing skyline queries over static data set. However, most of the real applications deal with the dynamic data set. Since dynamic data set constantly changes as time passes, the continuous skyline computation over dynamic data set becomes ever more complicated. In this paper, we propose a multiple layer grids method for continuous skyline computation (MLGCS) that maintains multiple layer grids to manage the dynamic data set. The proposed method divides the work space into multiple layer grids and creates the skyline influence region in the grid of each layer. In the continuous environment, the continuous skyline queries are only handled when the updating data points are in the skyline influence region of each layer grid. Experiments based on various data distributions show that our proposed method outperforms the existing methods.

Keywords*: skyline query, continuous skyline query, dynamic data, grid method.*

1. INTRODUCTION

In the recent years, the interests on skyline query processing have been significantly increased since the skyline results can be used in many applications with multi-dimensional data set. Given a data set P containing objects {P1, P2 … Pn}, the skyline operator returns all objects Pi such that Pi is not dominated by another object Pj. A common example for the use of a skyline operator assists a tourist to choose a set of interesting hotels from a large set of candidate hotels. We assume that each hotel has two attributes such as its distance to the tourist and its price. It is said that hotel A dominates hotel B if A is at least as close as B and at least as cheap as B. We recommend the hotel A that offers a better price or is closer than B.

The previous works of the skyline operation have concentrated on the efficient evaluation of skyline queries in static data set [1]-[4] and distributed systems [5]. However, in most of the real applications, the data set is dynamic, i.e. the data set changes constantly as time passes. For the static data set, the skyline algorithms are evaluated only once. But for the dynamic data set, the current skyline should be re-computed constantly as the data point changes. Recently, several methods are proposed for continuous skyline computation on sliding window data streams [6]-[8] and moving objects [9]. The continuous skyline computation on sliding window streams [6]-[8] assumes that each data point has an arrival time and an expiration time associated with it, and the first-in first-out character must be satisfied. The continuous skyline queries

* *Corresponding author. E-mail : yjs@chungbuk.ac.kr*
Manuscript received Jan. 20, 2010 ; accepted Mar. 15, 2010

(CSQ) method for moving objects [9] assumes that objects move in a smooth manner and each object have a single dynamic attribute (the location) and one or more static dimensions, i.e. only one dimension of the data is dynamic.

In this paper, we study continuous skyline computation over dynamic data set and data stream without any restricted conditions, i.e. each object can change in an unrestricted fashion and can have an arbitrary number of dynamic attributes.

If we consider the familiar example of choosing hotels, the distance from the tourist to hotels may change as the tourist moves, and the price of the hotel will also be reviewed with the drastic competition in the market, or in some case, the hotel has no vacancies. The interesting hotels need to be re-computed whenever the attributes of the data are changed. To handle these issues, the grid index based algorithm for continuous skyline computation (GICSC) has been proposed in [10]. It employs the grid structure to manage the data set and creates a skyline influence region in the work space. When the data set is updated, the skyline queries are handled only when the updating data points are in the skyline influence region. This method is efficient as most of the irrelevant data points can be filtered out. However, for the highly skewed data set, since a large number of data points fall into one cell, if the cell is in skyline influence region, most of the irrelevant data points can't be filtered out. Then this technique is inappropriate for such dynamic data set.

In this paper, we propose a multiple layer grids method (MLGCS) for efficiently processing the continuous skyline queries. The proposed method maintains multiple layer grids to manage the data set. It creates the skyline influence region in the grid of each layer. In the continuous environment, the skyline queries are only handled when the updated data points are in the skyline influence region of each layer grid. By doing it, even for the highly skewed data set, the irrelevant data points can be filtered out well, the proposed method improves the performance.

The remainder of this paper is organized as follows. Related work is covered in section 2. We present our novel algorithm in section 3. Experiment evaluations are presented in section 4, and finally, section 5 contains our conclusions and future work.

2. RELATED WORK

In [12], the skyline algorithm was first proposed by Kung et al. Borzsonyi et al. first introduced the skyline operation in database systems and proposed two solutions based on Block Nested Loop (BNL) and Divide and Conquer (D&C) in [1]. Since both of the two methods may incur much iteration, they are unsuitable for continuous skyline queries. SFS (Sort First Skyline) as the variation of BNL was proposed by Chomicki et al. in [13]. The nearest neighbor (NN) algorithm [14] indexes the data set with an R-tree. NN utilizes nearest neighbor queries to find the skyline results. In [3] and [4], the branch and bound skyline (BBS) algorithm was proposed. BBS is also based on nearest neighbors search and outperforms the NN approach. Recently, BBS is regarded as the most efficient method of skyline computation over static data set.

In [7], Lin et al. proposed n-of-N skyline queries against the most recent n of N elements to support on-line computation against sliding windows over a rapid data stream. Yufei Tao et al. proposed two algorithmic frameworks called Lazy and Eager for continuously maintaining sliding window skylines on data stream in [8]. Morse et al. proposed a scalable LookOut algorithm for updating the continuous time-interval skyline efficiently in [6]. The LookOut algorithm is more general than the sliding window method. Huang et al. has given a method to compute continuous skyline for moving objects in [9]. It addresses the problems of continuous skyline query processing, where the skyline query point is a moving object and the skyline changes continuously due to the query point's movement. Tian Li et al. proposed the grid index based algorithm (GICSC) for continuous skyline computation over highly dynamic moving objects in [10] and [11]. It employs the grid structure to manage the data set and is efficient for the uniformly distributed data set. However, when the data set is skewed or distributed unevenly within the work space, GICSC significantly degrades the performance.

3. THE PROPOSED METHOD

As mentioned above, the GICSC algorithm cannot perform well when the data set is skewed or the data points are distributed unevenly within the grid. The cell size of the grid significantly impacts the performance of the GICSC algorithm. Since a smaller cell size can decrease the number of the data points in each cell and further reduce the total area of the skyline influence region, more irrelevant data points can be filtered out, which improves the performance by decreasing the volume of data processed. However, a large number of additional cells are generated and processing these cells will cause large computation costs and memory requirements. In contrast, a bigger cell size cannot well incarnate the advantage of the grid index method. As an example, when a large number of data points fall into one cell and the cell is in skyline influence region, the irrelevant data points in this cell cannot be filtered out, or in extreme cases, the work space has only one cell, then the GICSC algorithm is similar to the BNL algorithm. As we know BNL algorithm is unsuitable for continuous skyline queries. Therefore, our objective is finding a new approach that can well decide the cell size of the grid and filter out more irrelevant data points.

Before the proposed method is explained in detail, we first define three dominance relationships between the cell and a given data point. We assume the point (x,y) denotes the attributes of a 2-dimensional data point. If the point (x,y) is greater than the value of the top right corner coordinate of the cell, we define that the data point is dominated by the cell. If it is greater than the value of the lower left corner coordinate and smaller than the value of the top right corner coordinate of the cell, we define that the data point is contained in the cell. If it is smaller than the value of the lower left corner coordinate of the cell, we define that the data point dominates the cell.

We assume that the data set is composed of tuples with d-dimension. The proposed method maintains a d-dimensional grid with equal-sized cells to manage the data set. Given the attributes of an object, it is easy to calculate the cell that it

belongs to. When the number of data points in one cell grows too large, it can be handled by creating a second layer grid based on the high density cell. The skyline influence regions of the first and second layer grids are created according to the existing skyline data. In the continuous skyline environment, since the non-skyline influence region is dominated by the existing skyline data, the skyline queries are only handled when the data points in the skyline influence region of each layer grid are changed. According to the data distribution, three or more layer grids can be generated. And as time passes, the low density cell can become high data density and vice verse. Therefore, the multiple layer grids structure is variable.

The proposed method consists of the initialization module and the maintenance module. We illustrate it by means of an example depicted in figure 1. We assume that both the first and the second layer grids consist of 4*4 cells, where the shaded area corresponds to the skyline influence region. The initialization module is responsible for initializing the skyline influence region (IR) and skyline list (SL) based on the initial static data set. It adopts queues to extend cells and begins by inserting the left-bottom corner cell $C^1(0,0)$ ($C^i(x,y)$ represents the correspond cell of the ith layer grid) into the $queue^1$($queue^i$ represents the queue of the ith layer grid). During each circle, it gets the first entry $C^1(x,y)$ of the $queue^1$, then extends its neighbor cells ($C^1(x,y+1)$ and $C^1(x+1,y)$, if valid) and inserts them into the tail of $queue^1$, provided that they have not been extended before and are not dominated by any skyline points. The skyline data included in $C^1(x,y)$ are verified according to the BNL algorithm [1] and then $C^1(x,y)$ is set to the skyline influence region. Note that, when $C^1(1,1)$ is popped from the $queue^1$, since it contains more data points than the predefined threshold value t (here we assume t is 5), it is switched to a second layer grid. In the second layer grid, another queue $queue^2$ is initialized and $C^2(0,0)$ is inserted into it. The rest operation of the second layer grid is the same as the first layer grid. Until $queue^2$ becomes empty, the data points B, C and G included in $C^1(1,1)$ are verified as skyline points. And then the execution of the first layer grid is continued. In the end, the initialization module terminates as $queue^1$ becomes empty.

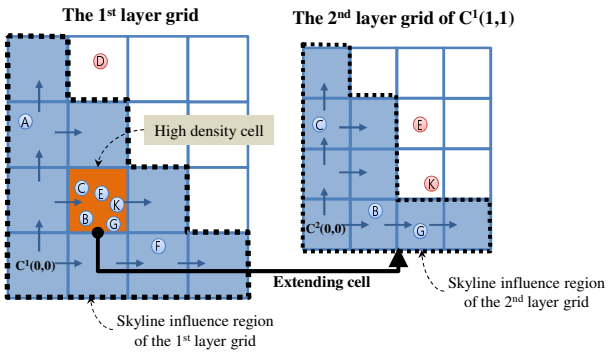


Fig.1. An example of MLGCS method

After obtaining the initial results, the maintenance module is used for updating SL and IR when new data points arrive or existing data points expire. If a new data point arrives, the SL and IR are updated only when its corresponding cell is in the skyline influence region of each layer grid. For instance, as shown in figure 1, if the new data point E arrives in $C^1(1,1)$, although $C^1(1,1)$ is in the skyline influence region of the first layer grid, it is not processed as the corresponding cell $C^2(1,2)$ of the second layer grid is not in the skyline influence region of the second layer grid. If the new data point is the skyline point, the skyline influence region which is dominated by the new skyline point needs to be updated. When an existing data point expires, it is deleted from the system. SL and IR are updated only when the expiring data point is skyline data as the expiration of the other points do not affect the skyline list and the skyline influence region. The procedure of processing the expiring data is similar to the initialization module, but it begins from the corresponding cell of the expiring data point.

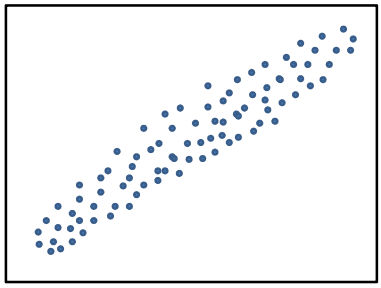
4. PERFORMANCE EVALUATION

In this section, we present the experimental results comparing GICSC algorithm [10] with our proposed method MLGCS. We examine the performance of GICSC and MLGCS by varying the number of the data and the dimensions.

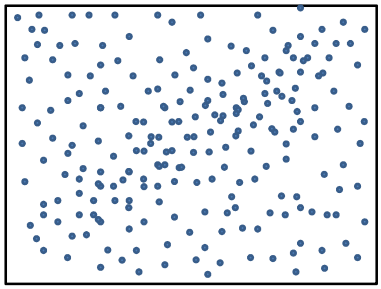
4.1 Experimental environment

We conducted our experiments on a desktop PC running on Windows XP professional. The PC has a Pentium IV 3.20GHz CPU and 1GB memory. All the experiments were coded in Java.

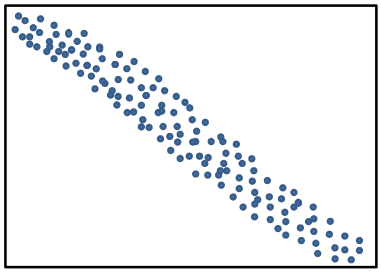
There are three critical types of data distributions that stress the effectiveness of skyline methods proposed in [1]. These three data distributions are correlated, anti-correlated, and independent. The correlated data represent that the data points that are good in one dimension are also good in the other dimensions. It is considered the easiest case for skyline computation as only a small portion of the data set will be considered during the skyline evaluation. The anti-correlated data represent that the data points that are good in one dimension are bad in one or all of the other dimensions. It is considered the most challenging of the three data types for skyline computation. This is because the data points in the skyline dominate only a small portion of the entire dataset. And the independent data set is the data distributed in the arbitrary work space. The 2-dimensional data set of the three data types are shown in figure 2. All of the experiments below are evaluated based on these three most popular synthetic data sets.



(a) Correlated data



(b) Independent data

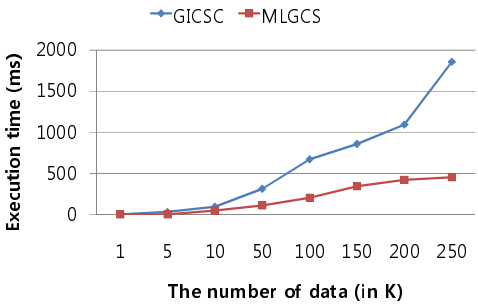


(c) Anti-correlated data

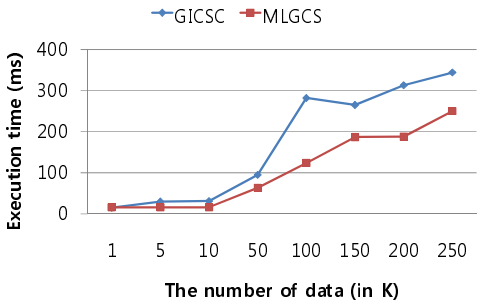
Fig.2. The three critical types of data distributions

4.2 Experimental results

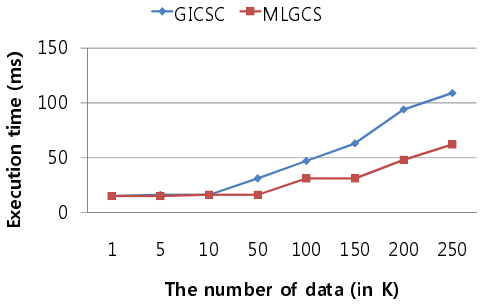
In the first experiment, we examine the performance of the methods by varying the number of data tuples from 1K to 250K. We set the number of queries is 2000 and the grid size is 10 cells per dimension. Performance is measured by CPU time. Figure 3 shows the results for the execution time of the skyline queries with respect to the number of data tuples. The execution time of skyline queries in the anti-correlated data set is higher than those of skyline queries in independent data set and correlated data set. The reason is that the skyline influence region of anti-correlated data set is larger than those of independent and correlated data sets. MLGCS outperforms GICSC in all of the three data sets since more irrelevant data points are filtered without computation by the skyline influence region of multiple layer grids.



(a) Anti-correlated data



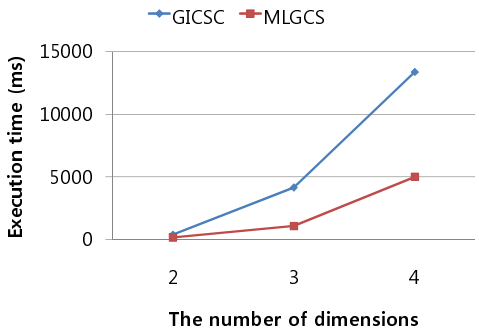
(b) Independent data



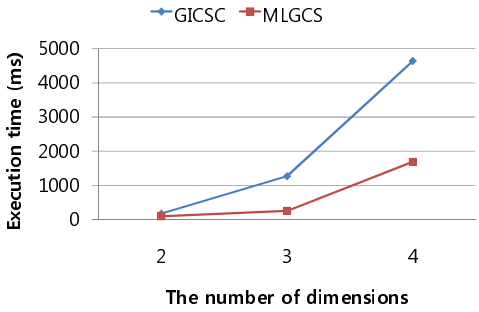
(c) Correlated data

Fig.3. Execution times according to data set size

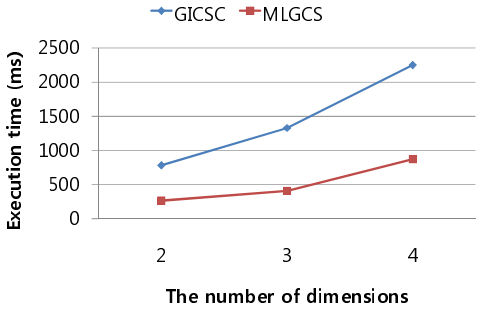
In the second experiment, we show the execution time with respect to the number of dimensions. The data set size is 10,000 and the number of queries is 1,000. The number of dimensions is varied from 2 to 4. The results for this experiment are shown in figure 4. Notice that the number of the cells in the grid is exponential to the dimensions of grid and so the number of the cells increases sharply with the growth of the dimensions. From the results, we can see that the execution time increase fast with the growth of the dimensions. The reason is that the skyline data points increase fast with the increase of the dimensions. Since more data points and cells need to be computed, their performance degrades. For the anti-correlated data set, the execution time is more expensive than the other two data sets. This is because the skyline influence region of the anti-correlated data set is larger than that of the other two data sets. Therefore, more data points and cells need to be computed. We find that MLGCS method is faster than GICSC in all of the three data types. This is because the skyline influence region of MLGCS is more accurate than that of GICSC and more irrelevant data points can be filtered out, which significantly reduced the execution time.



(a) Anti-correlated data



(b) Independent data



(c) Correlated data

Fig.4. Execution times according to dimensions

5. CONCLUSIONS

In this paper, we have proposed a novel method towards efficiently processing the continuous skyline queries. The proposed (MLGCS) method divides the work space into multiple layer grids and creates the skyline influence region of each layer grid. For the dynamic data set or data stream, the skyline queries are handled only when the dynamic data change in the skyline influence region of each layer grid. By doing this, since more irrelevant data points are filtered without computation, the performance of the proposed method is significantly improved. The experimental results have shown that the proposed method is more efficient than the existing methods.

In the future, we will improve the efficiency of our method on high dimensional spaces.

REFERENCES

[1] S. Borzsonyi, D. Kossmann and K. Stocker, "The skyline operator," *Proc. ICDE*, 2001, pp. 421-430.

[2] K.L. Tan, P.K. Eng and B.C. Ooi, "Efficient Progressive Skyline Computation," *Proc. VLDB*, 2001, pp. 301-310

[3] D. Papadias, Y.F. Tao, G. Fu and B. Seeger, "An optimal and progressive algorithm for skyline queries," *Proc. SIGMOD*, 2003, pp.467-478.

[4] D. Papadias, Y.F. Tao, G. Fu and B. Seeger, "Progressive skyline computation in database system," *ACM Journal, vol.30, no.1*, Mar. 2005, pp. 41-82.

[5] W.T. Balke, U. Guntzer and J.X. Zheng, "Efficient distributed skyline for web information systems," *Proc. EDPT'04*, 2004, pp.256-273.

[6] M. Morse, J.M. Patel and W.I. Grosky, "Efficient Continuous Skyline Computation," *Proc. ICDE*, 2006, pp. 108-108.

[7] X.M. Lin, Y.D. Yuan, W. Wang and H.J. Lu, "Stabbing the Sky: Efficient Skyline Computation over Sliding Windows," *Proc. ICDE*, 2005, pp. 502-513.

[8] Y.F. Tao and D. Papadias, "Maintianing sliding window skylines on data streams," *IEEE Journal, vol.18, no.3*, Jan. 2006, pp. 377-391.

[9] Z.Y. Huang, H. Lu, B.C. Ooi and Anthony K.H. Tung, "Continuous skyline queries for moving objects," *IEEE Journal, vol.18, no.12*, 2006, pp. 1645-1658.

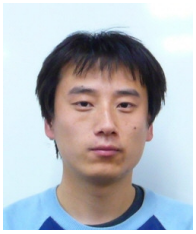
[10] L. Tian, L. Wang, P. Zou, Y. Jia and A. Li, "Continuous Monitoring of Skyline Query over Highly Dynamic Moving Objects," *Proc. MobiDE'07*, 2007, pp. 59-66.

[11] L. Tian, P. Zou, A. Li and Y. Jia, "Grid Index Based Algorithm for Continuous Skyline Computation," *Chinese Journal of Computers, vol. 6, no.6*, Jun. 2008, pp.998-1012.

[12] H.T. Kung, F. Luccio and F.P. Preparata, "On finding the maxima of a set of vectors," *ACM Journal, vol.22, no.4*, 1975, pp. 469-476.

[13] Jan Chomicki, Parke Godfrey, Jarek Gryz, and Dongming Liang, "Skyline with presorting," *Proc. ICDE*, 2003, pp. 717-719.

[14] D. Kossmann, F. Ramsak and S. Rost, "Shooting Stars in the Sky: an Online Algorithm for Skyline Queries," *Proc. VLDB*, 2002, pp.275-286.



He Li

He received the B.S. in computer science from Yunnan University, China in 2006. Since then, he has been a M.S. course student in the department of Information and Communication Engineering, Chungbuk National University, Korea. His main research interests include database system, distributed computing system, and wireless sensor network.

Su-Min Jang

He received his B.S. in Computer Science and Statistics from Mokpo National University, Korea in 1997 and he received M.S. and Ph.D. in Information and Communication Engineering from Chungbuk National University, Korea in 1999 and 2007, respectively. He is now the postdoctoral in the Dept. of Chungbuk National University Industry Academic Cooperation Foundation, Korea. His main research interests include distributed virtual environments system, location based services, distributed computing system.

Kwan-Hee Yoo

He is a professor of computer education and IIE(information industrial engineering) at Chungbuk National University, Korea. He received the B.S. in computer science from Chonbuk National University, Korea in 1985, and also received M.S., Ph.D. in computer science from KAIST(Korea Advanced Institute of Science and Technology), Korea in 1988 and 1995, respectively. His research interests include computational geometry, computer graphics, 3D character animation, dental/medical applications.

Jae-Soo Yoo

He received his B.S. in Computer Engineering from Chonbuk National University, Korea in 1989, and he also received his M.S. and Ph.D. in Computer Science from the Korean Advanced Institute of Science and Technology, Korea in 1991 and 1995. He is now a professor in Information and Communication Engineering, Chungbuk National University, Korea. His main research interests include database systems, sensor data management, location based services, distributed computing and storage management system.