An Algorithm for Retrieving Skyline Points based on User Specified Constraints using the Skyline Ordering

Jasna S
MTech Student
TKM College of engineering
Kollam

Manu J Pillai
Assistant Professor
TKM College of engineering
Kollam

ABSTRACT
Given a multidimensional data set, a skyline query returns the interesting points that are not dominated by other points. The actual cardinality (s) of a skyline query result may vary substantially from the desired result cardinality (k). An approach called skyline ordering is used that forms a skyline based partitioning of a given data set, it provides an ordering among the partitions. The constrained skyline query results the skyline points that may be too small in some cases. The paper proposes a new method for finding the arbitrary number of points for the constrained skyline query. The skyline ordering algorithm and size constrained skyline ordering algorithm are used for developing the algorithm. The results of experiments of algorithm show that the proposed scheme yields an efficient and scalable resolution of arbitrary size constraints on constrained skyline queries. By comparing the existing and proposed system, the proposed system is efficient in returning the arbitrary number of skyline points.

General Terms
Data Base Management, Query Processing

Keywords
Constrained Skyline Query, Skyline Queries, Skyline Order

1. INTRODUCTION
The Skyline query has received considerable attention from the database community, because of its importance in many applications like decision making, data mining etc. The Skyline query returns all interesting points that are not dominated by any other points. A point p dominates another point q if p is not worse than q in every single dimension but better than q in atleast one dimension.

So the skyline queries have the powerful capability of retrieving points from a large multidimensional data set. Skyline queries have the ability for multi-criteria decision making and user preference applications. For example(fig 1), a tourist can issue a skyline query on a hotel relation in order to get the hotels with high facilities and having cheap prices. This work focuses on the constrained skyline query. The constrained skyline query returns the interesting points from the data set if the constraints specified by the user are satisfied. For example, a tourist wants to get only the hotels within a specified price range.

The well-known weakness of the skyline queries is that the result cardinality may vary. The result cardinality can be large when the dimensionality of the data set is high. This problem also arises in the case of constrained skyline queries.

In order to deal with the large skyline results, there are four approaches. In the first approach called pointwise ranking, all the points are ordered based on some mapping function and only top points are returned. In the second approach called subspace reference, subspaces of full d-dimensional space are investigated and points preferred in subspaces are favoured in the result. In the third approach, called set-wide maximization, a subset of skyline points is selected such that a collective quantity based objective. In the fourth approach, called approximate selection, a predefined threshold is given so that the points are compared with this such that more points are identified as being dominated.

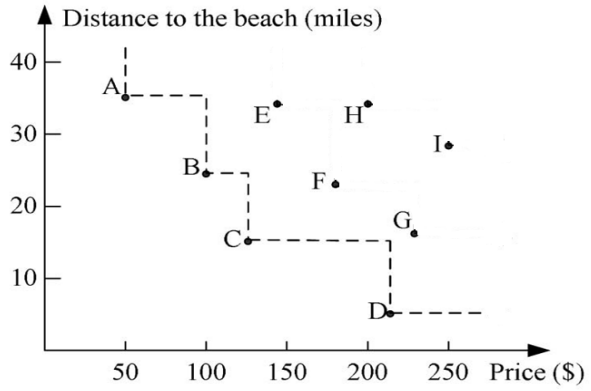

Fig 1. Skyline of Hotels

The small skylines are also a problem. Suppose the tourist specifies a parameter k, the number of interesting hotels that satisfies the constraint to be returned. The conventional constrained skyline query, usually, if not always, fails to return k hotels. So in order to get the arbitrary number of skyline points from a d-dimensional data set, the approach called skyline ordering is used. This approach combines the point-wise ranking and the set-wide maximization techniques. Skyline ordering provides a skyline-based partitioning of a given data set and these partitions provides an ordering. To this skyline order the size constrained skyline query is applied in order to get the arbitrary number of points.

The experiments on these methods show that the proposed constrained skyline query with skyline ordering returns the arbitrary number of points that is obtained using the conventional constrained skyline query.

This article is organized as follows. Section II introduces definitions and motivating examples. Literature survey is discussed in section III. Section IV explains the algorithms for skyline order computation, constrained skyline query with skyline ordering. Section V presents the comparison of existing and proposed methods. These methods are compared using large datasets. Section VI contains the conclusions and directions for future work.

2. DEFINITIONS

The definition for skyline ordering, size constrained skyline query and constrained skyline query with skyline ordering are defined below. Table 1 lists the notation used in this paper.

2.1 Skyline Order

Skyline order of a set of d-dimensional points Q is a sequence < S1,S2,.....Sn> defined as :

1. S1 is the skyline of Q (data set).
2. for all i, 1< i ≤ n, Si is the skyline of Q
3. $\bigcup_{i=1}^{n} Si$ = Q.

Each Si is called as skyline subset in the skyline order or a skyline order subset. The n is called as the skyline order length.

Skyline ordering introduces a skyline-based partitioning of a given data set, it provides an ordering among the partitions. In particular, given the partitions in skyline order, the following hold:

1) No point can dominate any other point in the same partition or in a previous partition;

2) Any point in a partition, except in the first partition, must be dominated by some point(s) in the previous partition.

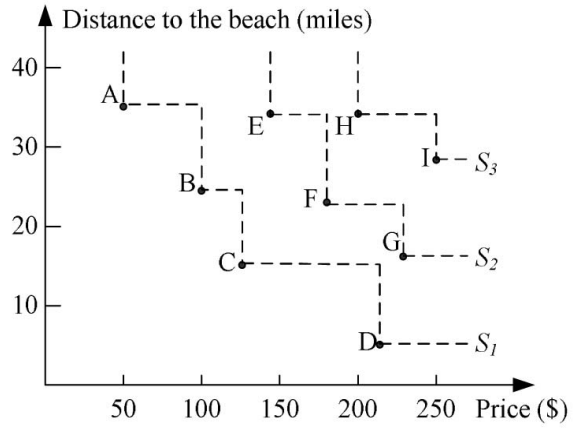

Fig.2: Skyline order

2.2 Size Constrained Skyline Query

Given a set of d-dimensional points P and the skyline order S=<S1,S2,….Sn> then the size constrained skyline query Ssoscs is defined as:

$$Ssoscs = (\bigcup_{i=1}^{l} Si) \cup S'_{l+1}$$

where l is defined such that $\sum_{i=1}^{l} |Si| \leq k < \sum_{i=1}^{l+1} |Si|$, and S'_{l+1} S_{l+1} such that $|S_{soscs}| = k$.

S'_{l+1} is selected from S_{l+1} using the set-wide maximization approach.

2.3 Constrained Skyline query with skyline ordering

The constrained skyline query with skyline ordering returns the arbitrary number of skyline points based on the constraints specified by the user. In order to get the arbitrary number of points the skyline ordering concept is used. The skyline order partition the data set based on the dominance relation and from that the points satisfying the constraints are retrieved. Then the size constrained skyline query is applied to this to get the k number of skyline points as specified by the user.

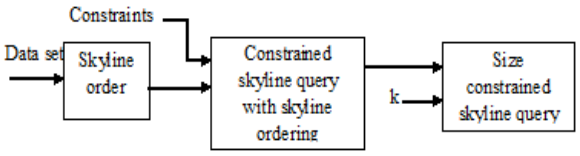

Fig.3: Steps for obtaining constrained skyline query with skyline ordering

Table 3.1 Table of notations

Notation	Description
Q	Argument data set
N	Cardinality of Q
d	Dimensionality of Q
S_p	Skyline of Q
s	Size of S_q
Q_k^{scs}(P)	A k-size constrained skyline query on Q
S_p	Skyline order of Q
S_i	i-th skyline order subset of S_q
n	Skyline order length
C_p	Skyline points satisfying constraints
$D_p^{<}$ (p)	All points in P that are dominated by p

3. LITERATURE SURVEY

3.1 Skyline Algorithms

There are two categories of skyline algorithms. In the first category, it does not rely on indexes on the data set. Theoretical algorithms for maximal vector computation[1],[3],[19]fall into this first category. The skyline query is introduced into database by Borzonyi [5],defines Block Nested Loop(BNL) and Divide-and-Conquer(D&C) algorithms. A variant of BNL is sort-filter-skyline (SFS)[10]. Godfrey et al. [12] proposes a comprehensive analysis of the non index- based algorithms.Two progressive algorithms are proposed by Tan.et.al [25],they are Bitmap and Index. The Bitmap method represents points by means of bit vectors, and employs bitwise operations, while the Index method utilizes data transformation and B+ tree indexing. Kosmann.et.al proposes a Nearest Neighbour method. This method identifies skyline points by recursively invoking R*-tree based depth-first NN search over different data portions. Papadias et al. [22] propose a BBS method that is based on the best first nearest neighbor algorithm. Zhang et al. [30] propose dynamic indexing tree for the skyline points that helps to reduce the CPU costs in sort-based algorithms[10],[12],[2].

A constrained skyline query [23] returns the most interesting points in the data set defined by the constraints. The constrained skyline query first considers the whole database that satisfies the specified constraint.

The weakness of the constrained query algorithm is that it does not return the arbitrary number of skyline points. So a new method is needed to return the required result. The

skyline ordering concept and the size constrained skyline query introduced in [31] is used to solve the weakness of constrained skyline query.

3.2 Dominance-Based Skyline Query Derivatives

Several skyline query derivatives have been proposed based on the dominance relation. Its methodological nature is considered in all approaches, and also its applicability to the case where the number of points expected (k) is less than the skyline size(s) and the case where k>s, and its relevant algorithm. Based on the nature of an approach four types are considered, they are point-wise ranking, subspace reference, set-wide maximization and approximate selection.

The top-k dominating query [22] that retrieves points that dominate the largest number of points. Yiu and Mamoulis [28] proposes an efficient algorithm for this problem.The top-k dominating query is a point-wise ranking type. Chan.et.al [7] proposes a top-k ranking problem that gives the priority to points that appear more frequently in subspace skylines. The concept of strong skyline points proposed by Zhang.et.al [29] which appear frequently in small sized subspace skylines in high dimensional spaces. It does not return a fixed number of points in the case of k<s and not applicable to case of k>s. A k-dominant skyline for high-dimensional space is proposed by Chan.et.al [6] does not return a fixed number of points in the case of k<s, and not applicable for k>s.

Lin et al. [21] propose the top- k representative skyline points problem. It selects a portion of points from the skyline points that maximizes the total number of dominated points. In this the results turn out not to be representative. So Tan et al. [24] propose the techniques that minimize the distance between a non representative skyline point and its nearest representative point in the selection. These approaches are the examples of set-wide maximization.

Approximately dominating representatives are proposed by Koltun and Papadimitriou [17] which produces smaller but not fixed size called Ɛ-ADR skyline**.** For k>s this approach is not applicable. Xia et al. [27] define Ɛ-skyline in which user specified weights are allowed on each dimension. The size of the Ɛ-skyline can be increased and decreased by varying Ɛ and the weights.

4. ALGORITHMS

4.1 Constrained Skyline Query Computation

The algorithm scans the input data set P. For each point p in P, the constraint is applied and if p satisfies the constraint it is added into the set W. Then the skyline points are retrieved from the set W. Each point q in W is compared with each point r in W. If r dominates q then deletes q from W. Otherwise if q dominates r then delete r from W. Finally, the set W contain the skyline points satisfying the constraints.

Algorithm 1: ConstrainSkyQry (dataset P, constraints c)

```
 1: W=< >
 2: for each point p in P do
 3: begin
 4:     if p satisfies the constraints c then
 5:         add p to W
 6: end
 7: for each point q in W
 8: begin
 9:     for each point r in W
10:     begin
11:         if r < q then
12:             delete q from W
13:         else
14:             if q < r then
15:                 delete r from W
16:     end
17: return W
18: end
```

4.2 Skyline order computation

The algorithm takes input data set Q. For each point q in Q, q is checked against each subset in the current skyline order and put it into a specified subset or a new subset is created for it. All subsets of the current skyline order is maintained by the algorithm. The q s added to the appropriate subset, during this time some of the points in the skyline order subset is changed their membership, producing a new subset in the skyline order.

Algorithm 2: SkyOrdScan (data set Q)

```
 1: S = < >
 2: for each point q in Q do
 3: begin
 4:     for each Si from S1 to Sn in S do
 5:     begin
 6:       isSky = TRUE; Stmp = NULL
 7:       for each point t in Si do
 8:       begin
 9:         if t < q then
10:           isSky = FALSE; break
11:         else
12:           if q < t then
13:             move t from Si to Stmp
14:       end
15:       if isSky then
16:       begin
17:         if Si =NULL then
18:           Si = {q}; insert Stmp into S after Si
19:         else
            begin
20:           if Stmp ≠ NULL then
21:             AdjustSkyOrd(Stmp, S, i+1)
22:           add q to Si
23:         end
24:         break
25:       end
26:     end
27:       if q does not belong to any Si then
28:       append {q} to S
29: end
30. return S
```

Initially the skyline order *S* is empty. Each point q in Q is compared to each point t in each Si from s1 to sn. If q is dominated by t, break from the loop and go to next subset in skyline order. If t is dominated q, t will be moved to a temporary list Stmp. Stmpcontains all points that come from current Si, the actions are taken to update the skyline order. If Si is empty, which means that q dominates all points in Si, {q} is inserted into new subset and Stmp is inserted into S after the new Si. If Si is not empty, the AdjSkyOrd is called to adjust the subsets after current Si in S, this is done only if Stmp is not empty and q will be added into Si. If q doesnot belong to any Si, {q} will be added to S.

The algorithm for AdjSkyOrd is given below. Three inputs are taken:Stmp the temporary list contains points excluded from the current skyline subset, current skyline order *S*, next subset index. Starting from Snext, it loops on each subset Si in *S*. If q is not dominated by any point in Stmp then each point q in Si is moved to temporary list temp. If the temp is empty,Stmp is

added to S immediately before Si. If Si is empty after all the points in it are checked, Stmp replaces it in D. Otherwise,Stmp and Si are swapped, and continues to next subset in S. When the loop ends, if Stmp till contains the points, then it is added to S.

Algorithm 3: AdjSkyOrd (Temporary list Stmp, Current Skyline order *S*, Next subset index next)

```
1: for each subset Si from Snext to Sn in S do
2: begin
3:        temp = NULL
4:        for each point p in Si do
5:        begin
6:                  if ∄q Ɛ Stmp s.t. q < p then
7:                            move p from Si to temp
8:         end
9:         if temp==NULL then
10:          add Stmp to S immediately before Si; return
11:       merge temp to Stmp
12:       if Si== NULL then
13:                 replace Si in S with Stmp; return
14:        swap Si and Stmp
15: end
16: if Stmp ≠ NULL then
17:   append Stmp to S
```

4.3 User Specified Constrained Skyline Query Algorithm

If the skyline order *S* of the data set P has been computed before a user specified constrained skyline query is issued, *S* can be used to facilitate the query processing**.**. The algorithm takes skyline order *S* and constraints specified by the user, c as input. First the set *R* is initialized to be empty. Then each point p in each Si sequentially from S1 to the current Sn is checked with the constraints specified by the user. If the point p satisfies the constraints, it is added into the set *R*. Otherwise the point is not added to the set. The pseudocode for user specified constrained skyline query is described below.

Algorithm 4: USConstrskyord(Skyline order *S*, constraints c)

```
1: R =<>
2: for each subset Si from S1 to Sn in S do
3: begin
4:    for each point p in Si do
5:    begin
6:         if p satisfies the constraints c then
7:              append {p} to R
8:        else
9:             continue
10:   end
11: end
12: return R
```

4.4 Size Constrained On User Specified Constrained Skyline Query Algorithm

The algorithm takes constraint satisfying points *R* and the number of points to be retrieved, k as input. First, the result set R is initialized to be empty, and a count variable cnt is set to k (line 1). Then, each subset R_i in *R* is checked sequentially. If the cardinality of R_i equals the current count in cnt, R_i is merged into R, and the loop stops (lines 3-4). If the cardinality of R_i is smaller than cnt, R_i is merged into R and the loop continues with an updated cnt (lines 5-6). Otherwise, rSKY is called to select the last cnt points from R_i, and the loop stops (line 8). The set-wide maximization approaches can be used for implementing rSKY. The pseudocode for size constrained on user specified constrained skyline query is described below.

Algorithm 5: USCskyord(constrained points *R*, number of points to retrieve k)

```
1: R = Ø ; cnt = k
2: for each subset Ri from R 1 to R t in R do
3: begin
4:    if | R i | == cnt then
5:          R = R U R i; break
6:    else if  | R i | < cnt then
7:          R = R U R i; cnt = cnt - | R i |
8:    else
9:          R = R U rSKY(R i ,cnt) ;break
10: end
11: return R
```

5. EXPERIMENTAL EVALUATION

5.1 Data Sets

The NBA(National Basketball Association) data set is used for analyzing the algorithms. NBA data set contains regular season statistics of 19112 NBA players (i.e., data objects) and generally it follows a correlated data distribution. This data set contains Player regular season stats, player regular season career totals, player playoff stats, player playoff career totals, player all-star game stats, team regular season stats, complete draft history. In order for the query to be meaningful, only few important attributes are selected for the NBA players: games played (gp), points (pts), rebounds (reb), and assists (ast).

5.2 Comparison of Constrained Skyline Query with Skyline Ordering and without Skyline Ordering

For the given data set the constrained skyline query with skyline ordering is executed. The tables of different cardinality are used for the comparison. For the same table, depends on the constraint applied the number of skyline points it returned varies. In constrained skyline query with skyline ordering the arbitrary number of skyline points are obtained. But there is no provision to arbitrary number of points in constrained skyline query with skyline ordering.

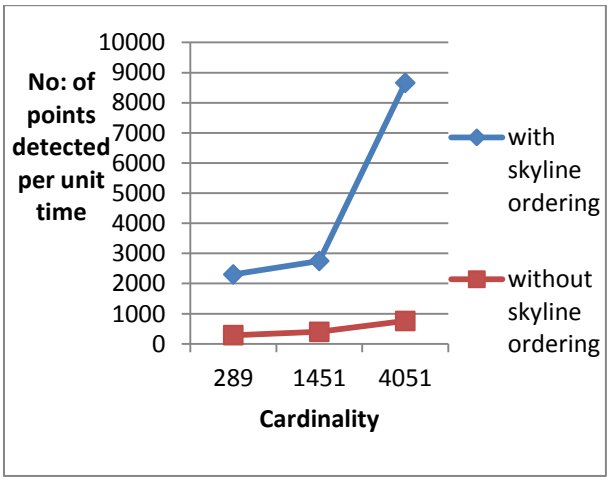

Fig. 4: Comparison of Constrained Skyline Query Algorithm with Skyline Ordering and without Skyline Ordering

Figure.4 shows the comparison existing and proposed system. From the graph it is clear that the constrained skyline query with skyline ordering gives more number of skyline points than the constrained skyline query. By comparing the two systems the proposed system ie, constrained skyline query

with skyline ordering can give the skyline points based on the user interests. The computation time increases for both the systems as the cardinality increases. For the constrained skyline query with skyline ordering takes more time constrained skyline query because of the skyline ordering computation. But in terms of number of skyline points retrieves, the constrained skyline query with skyline ordering method is better than the constrained skyline query without skyline ordering.

5. CONCLUSION

Skyline queries find the most interesting points in the multidimensional data sets. The previous works focus on getting the arbitrary number of skyline points in normal skyline query. In order to get the arbitrary number of skyline points in constrained skyline query, the skyline ordering is used. The existing technique does not give the sufficient number of skyline points in some cases**.** The skyline order concept is used, in which data set is partitioned using the skyline dominance relationship. An order among different partitions is provided by the skyline ordering. An algorithm is developed for computing the arbitrary number of points for the constrained skyline query using the concept of skyline ordering and size constrained skyline query.

By analyzing the proposed method, it is found that it returns the arbitrary number of points, that provision is not in the existing method. The algorithm does not give sufficient number of points in case of small databases. The work can be extended for the uncertain data.

7. REFERENCES

[1] J.L. Bentley, K.L. Clarkson, and D.B. Levine, "Fast Linear Expected-Time Algorithms for Computing Maxima and Convex Hulls," Proc. First Ann. ACM-SIAM Symp. Discrete Algorithms (SODA), pp. 179-187, 1990.

[2] I. Bartolini, P. Ciaccia, and M. Patella, "Efficient Sort-Based Skyline Evaluation," ACM Trans. Database Systems, vol. 33, no. 4, pp. 1-49, 2008.

[3] J.L. Bentley, H.T. Kung, M. Schkolnick, and C.D. Thompson, "On the Average Number of Maxima in a Set of Vectors and Applications," J. ACM, vol. 25, no. 4, pp. 536-543, 1978.

[4] H. Blunck and J. Vahrenhold, "In-Place Algorithms for Computing (Layers of) Maxima," Proc. Scandinavian Workshop Algorithm Theory (SWAT), pp. 363-374, 2006.

[5] S. Borzonyi, D. Kossmann, and K. Stocker, "The Skyline Operator," Proc. Int'l Conf. Data Eng. (ICDE), pp. 421-430, 2001.

[6] C.-Y. Chan, H. Jagadish, K.-L. Tan, A.K. Tung, and Z. Zhang, "Finding K-Dominant Skylines in High Dimensional Space," Proc. ACM SIGMOD, pp. 503-514, 2006.

[7] C.-Y. Chan, H. Jagadish, K.-L. Tan, A.K. Tung, and Z. Zhang, "On High Dimensional Skylines," Proc. Int'l Conf. Extending Database Technology (EDBT), pp. 478-495, 2006.

[8] Y.-C. Chang, L. Bergman, V. Castelli, C.-S. Li, M.-L. Lo, and J.R. Smith, "The Onion Technique: Indexing for Linear Optimization Queries," Proc. ACM SIGMOD, pp. 391-402, 2000.

[9] J. Chomicki, "Preference Formulas in Relational Queries," ACM Trans. Database Systems, vol. 28, no. 4, pp. 427-466, 2003.

[10] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with Presorting," Proc. Int'l Conf. Data Eng. (ICDE), pp. 717-719, 2003.

[11] B. Cui, H. Lu, Q. Xu, L. Chen, Y. Dai, and Y. Zhou, "Parallel Distributed Processing of Constrained Skyline Queries by Filtering," Proc. Int'l Conf. Data Eng. (ICDE), pp. 546-555, 2008.

[12] P. Godfrey, R. Shipley, and J. Gryz, "Maximal Vector Computation in Large Data Sets," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 229-240, 2005.

[13] G. Hjaltason and H. Samet, "Distance Browsing in Spatial Database," ACM Trans. Database Systems, vol. 24, no. 2, pp. 265- 318, 1999.

[14] Z. Huang, C.S. Jensen, H. Lu, and B.C. Ooi, "Skyline Queries against Mobile Lightweight Devices in MANETs," Proc. Int'l Conf. Data Eng. (ICDE), p. 66, 2006.

[15] W. Jin, M. Ester, and J. Han, "Efficient Processing of Ranked Queries with Sweeping Selection," Proc. European Conf. Principles and Practice of Knowledge Discovery in Databases (PKDD), pp. 527- 535, 2005.

[16] W. Jin, J. Han, and M. Ester, "Mining Thick Skylines over Large Databases," Proc. European Conf. Principles and Practice of Knowledge Discovery in Databases (PKDD), pp. 255-266, 2004.

[17] V. Koltun and C.H. Papadimitriou, "Approximately Dominating Representatives," Proc. Int'l Conf. Data Theory (ICDT), pp. 204-214, 2005.

[18] D. Kossmann, F. Ramsak, and S. Rost, "Shooting Stars in the Sky: An Online Algorithm for Skyline Queries," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 275-286, 2002.

[19] H.T. Kung, F. Luccio, and F.P. Preparata, "On Finding the Maxima of a Set of Vectors," J. ACM, vol. 22, no. 4, pp. 469-476, 1975.

[20] P. Larson and H.Z. Yang, "Computing Queries from Derived Relations," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 259- 269, 1985.

[21] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang, "Selecting Stars: The k Most Representative Skyline Operator," Proc. Int'l Conf. Data Eng. (ICDE), pp. 86-95, 2007.

[22] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An Optimal and Progressive Algorithm for Skyline Queries," Proc. SIGMOD, pp. 467-478, 2003.

[23] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "Progressive Skyline Computation in Database Systems," ACM Trans. Database Systems, vol. 30, no. 1, pp. 41-82, 2005.

[24] Y. Tao, L. Ding, X. Lin, and J. Pei, "Distance-Based Representative Skyline," Proc. Int'l Conf. Data Eng. (ICDE), pp. 892-903, 2009.

[25] K.L. Tan, P.K. Eng, and B.C. Ooi, "Efficient Progressive Skyline Computation," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 301-310, 2001.

[26] A. Vlachou, C. Doulkeridis, K. Nørva°g, and M. Vazirgiannis, "Skyline-Based Peer-To-Peer Top-k Query Processing," Proc. Int'l Conf. Data Eng. (ICDE), pp. 1421-1423, 2008.

[27] T. Xia, D. Zhang, and Y. Tao, "On Skylining with Flexible Dominance Relation," Proc. Int'l Conf. Data Eng. (ICDE), pp. 1397- 1399, 2008.

[28] M.L. Yiu and N. Mamoulis, "Efficient Processing of Top-k Dominating Queries on Multi-Dimensional Data," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 483-494, 2007.

[29] Z. Zhang, X. Guo, H. Lu, A.K. Tung, and N. Wang, "Discovering Strong Skyline Points in High Dimensional Spaces," Proc. ACM Int'l Conf. Information and Knowledge Management (CIKM), pp. 247- 248, 2005.

[30] S. Zhang, N. Mamoulis, and D.W. Cheung, "Scalable Skyline Computation Using Object-Based Space Partitioning," Proc. SIGMOD, pp. 483-494, 2009.

[31] Hua Lu, Christian S. Jensen, Zhenjie Zhang. "Flexible and Efficient Resolution of Skyline Query Size Constraints." IEEE Transactions on Knowledge and Data Engineering (TKDE).23(7): 991.1005, 2011.