How to cite this paper:

Ghazaleh Babanejad, Hamidah Ibrahim, Nur Izura Udzir, Fatimah Sidi, & Ali Amer Alwan. (2017). Deriving skyline points over dynamic and incomplete databases in Zulikha, J. & N. H. Zakaria (Eds.), Proceedings of the 6th International Conference of Computing & Informatics (pp 77-83). Sintok: School of Computing.

# DERIVING SKYLINE POINTS OVER DYNAMIC AND INCOMPLETE DATABASES

**Ghazaleh Babanejad[1],Hamidah Ibrahim[1], NurI zura Udzir[1] Fatimah Sidi[1],Ali Amer Alwan[2]**

*[1]Universiti Putra Malaysia, Ghazaleh.babanejd@upm.student.edu.my, {hamidah.ibrahim,izura,fatimah}@upm.edu.my*
*[2]International Islamic University Malaysia, aliamer@iium.edu.my*

**ABSTRACT**. The rapid growth of data is inevitable, and retrieving the best results that meet the user's preferences is essential. To achieve this, skylines were introduced in which data items that are not dominated by the other data items in the database are retrieved as results (skylines). In most of the existing skyline approaches, the databases are assumed to be static and complete. However, in real world scenario, databases are not complete especially in multidimensional databases in which some dimensions may have missing values. The databases might also be dynamic in which new data items are inserted while existing data items are deleted or updated. Blindly performing pairwise comparisons on the whole data items after the changes are made is inappropriate as not all data items need to be compared in identifying the skylines. Thus, a novel skyline algorithm, DInSkyline, is proposed in this study which finds the most relevant data items in dynamic and incomplete databases. Several experiments have been conducted and the results show that DInSkyline outperforms the previous works by reducing the number of pairwise comparisons in the range of 52% to 73%.

**Keywords**: skyline queries, preference queries, incomplete database, dynamic database.

## INTRODUCTION

Finding the best option between different choices based on user's preferences is not easy. Realizing the best and accurate results is the ultimate goal of skyline query processing. Consider a user who wanted to attend a conference and thus choose a hotel with the following preferences: hotel that is near to the conference venue and with cheap price. However, hotels that are near to a conference venue are more expensive than those that are far away from the conference venue. Relying on the traditional query processing which is based on exact match between the preferences and the data in the database is inappropriate, as it may not return any results, i.e. if there is no hotel which is the nearest to the conference venue (minimum distance) and at the same time with the cheapest price (minimum price), then no results will be returned. Hence, skyline which works based on domination has been proposed, in which data items that are not dominated by other data items in the database are returned as results. Database can either be in a complete or incomplete state. Most of the existing works assumed that databases are complete and static. Obviously in real world databases are incomplete with

changing states. For example, consider an investor who wants to invest a share in a stock exchange. When the investor queries for the best stock, the stocks keep on changing due to new stocks are added, while existing stocks are deleted or updated. Some of the stocks may have incomplete data in its dimensions. For such case finding skylines is challenging. Hence, we propose an algorithm that attempts to find skylines for dynamic and incomplete databases. Our aim is to minimize the number of comparisons needed during the skyline query processing. This is achieved by comparing only the needed data items instead of comparing the whole database.

The rest of the paper is organized as follows. The following section discusses the previous approaches on skylines and its applications, which is then followed by problem formulation. Next, the proposed algorithm is presented followed by the results of the experiments that we have conducted. The last part of this paper concludes the work.

## RELATED WORKS

There are many techniques for preference queries, which include top-k (Chaudhuri and Gravano,1999), k-dominance (Chan *et al*., 2006), top-k dominating (Yiu and Mamoulis, 2007), k-frequency (Papadias *et al*., 2005), skylines (Borzsony *et al*., 2001), multi objective skyline (Balke and Guntzer, 2004), and ranked skylines (Lee *et al*., 2009). All of these techniques attempt to find the best results that meet the user's preferences. Also, there are varieties of skyline techniques that have been proposed such as Block-Nested-Loop (Borzsony *et al*., 2001), Divide and Conquer (D&C) (Borzsony *et al*., 2001), Bitmap and Index (Tan *et al*., 2001), Sort-Filter-Skyline (SFS) (Chomicki *et al*., 2001), Nearest Neighbor (Kossmann *et al*., 2002), Branch-Bound-Skyline (BBS) (Papadias *et al*., 2003), Linear Elimination Sort for Skyline (LESS) (Godfrey *et al*., 2005), and Sort and Limit Skyline algorithm (SaLSa) (Bartolini *et al*., 2006). However, these techniques are proposed to tackle skyline computation on complete databases. The early research on databases with incomplete data items is conducted by Khalefa *et al*. (2008), which proposed two algorithms, namely: Replacement and Bucket. These algorithms use traditional skyline algorithm. Later, they introduced the ISkyline algorithm. To overcome the cyclic dominance relations, virtual points and shadow skylines are introduced. Bharuka and Kumar (2013) proposed the Sort-based Incomplete Data Skyline (SIDS) algorithm which works by pre-sorting the data items in descending order of each dimension. Alwan *et al*. (2013) introduced an algorithm named Incoskyline. It has four phases, namely: clustering, grouping and finding local skylines, deriving k-dom skylines, and retrieving skylines. Gulzar *et al.* (2016) proposed a framework for evaluating skyline queries in incomplete data which consists of four steps, namely: Data Sorting and Array Constructor, Data Filter, Candidate Skyline Identifier, and Final Skyline Identifier. Kontaki et al. (2010) worked on top-k and top-k dominating and reviewed algorithms for evaluating continuous preference queries under sliding window streaming model. Also k-dominant skyline is presented in (Cui *et al*., 2011). In their work when the dataset is changed, the existing k-dominant skylines are compared to the new k-dominant data items to derive the results.

## PROBLEM FORMULATION

In this section we explain the concepts related to the skyline queries for dynamic and incomplete database. We assume that data items with the highest value are preferable.

*Definition* 1, *Dominance Relation*: Given a database $D$ with data items $P_i$, $i = 1, 2, \ldots, m$, and $n$ dimensions $d = \{d1, d2, \ldots, dn\}$. Let two d-dimensional data items $P_k = (U1, U2, \ldots, Un)$ and $P_l = (S1, S2, \ldots, Sn)$, $P_k$ dominates $P_l$ denoted by $P_k \succ P_l$ if and only if the following condition holds: $\forall di \in d$ , $U_i \geq S_i \wedge \exists d_j \in d$, $U_j > S_j$.

*Definition* 2, *Skyline Point*: Given a set of data items in a database *D*, a data item $P_i \in D$ is a skyline if there is no other data items $P_j \in D$ that dominates $P_i$. Data items that are not dominated by the other data items in the database *D* are the skylines. Skylines hold the transitivity property that means if $P_i$ dominates $P_k$ and $P_k$ dominates $P_l$ it leads to $P_i$ dominates $P_l$ (Borzsony *et al*., 2001).

*Definition* 3, *Incomplete Database*: A database *DI* is incomplete if and only if it contains at least a data item with missing values in one or more of its dimensions. There are many reasons for having missing values in databases like mistake in data entry, inaccurate data from heterogeneous data sources and integrating heterogeneous schemes (Khalefa *et al*., 2008).

*Definition* 4, *Dynamic Database*: A database *DD* is said to be dynamic if the data items in the database keep on changing in which new data items are inserted, while existing data items are deleted and updated. With these definitions, we can now formally define the problem that we are focusing in this paper.

*Problem Definition*: Given an incomplete database, *DI*, it may change to a new state, $D_{new}$, due to the following operations:

- Insert Operation: $D_{new} = DI \cup D_{<insert>}$ where $D_{<insert>}$ is the new set of data items to be inserted into the initial database, *DI*.
- Delete operation: $D_{new} = DI - D_{<delete>}$ where $D_{<delete>}$ is the set of data items to be deleted from the initial database, *DI*.
- Update operation: $D_{new} = (DI - D_{<delete>}) \cup D_{<insert>}$ where an update operation is considered as a delete operation followed by an insert operation.

$P_i \in D_{new}$ is a skyline if there is no data item in $D_{new}$ that dominates $P_i$. Finding the set of skylines in $D_{new}$ should incur the least number of comparisons between the data items which will indirectly incur the least processing time. The data item $P_i \in DI$ may have missing values in one or more of its dimensions (Babanejad *et al*., 2014).

## THE PROPOSED ALGORITHM

As explained earlier, there are three algorithms which focus on deriving skyline points on incomplete and static databases. The proposed algorithm, DInSkyline, attempts to find skyline points over incomplete and dynamic database. The main components of DInSkyline are the Domination History (DH) table that keeps track of domination relations, the Bucket Dominating (BDG) that stores the data items that dominates the other data items, and Bucket Dominated (BDD) that stores the data items that are dominated by the other data items. The aim is to prevent exhaustive comparisons, thus decreases the number of comparisons. Hence, performing comparisons on the whole database can be avoided. For proving the correctness of DInSkyline algorithm, the ISkyline, SIDS and Incoskyline algorithms are applied on $D_{new}$ to produce the skyline points, *S*, and these points are then compared to the skyline points produced by our algorithm, *S*''. If the results are the same i.e. $S = S''$, then we can conclude that our algorithm is correct.

The DInSkyline works as follows. The first step is to group the data items based on missing values, all data items with the same missing values (bitmap representation) are grouped in the same bucket as shown in Figure 1. Next, the skylines of each bucket are derived and stored in the Bucket Skylines (BS) (see Table 2). This is performed by comparing the data items of each bucket. During these comparisons the domination relations are kept in the DH table (see Table 7). Then the bucket skylines of each bucket (see Table 3a) are compared to each other. Those bucket skylines that dominate other bucket skylines are listed in the BDD while those that are being dominated are stored in the BDD (see Table 3b). During the comparisons between the bucket skylines some data items may be considered as bucket dominat-

ing items but later can be dominated by the other data items hence these items are stored in both the BDG and BDD. Finally, the data items that appeared in both the BDG and BDD are removed from the BDG and the remaining data items in the BDG are the final skylines.

Given a set of data items to be inserted, $D_{<insert>}$, the data items are grouped based on the bitwise representation (see Table 4), then the skylines of each bucket are determined and stored in the Temp Bucket Skyline (TBS) (see Table 5). As mentioned above, the data items which dominate other data items are kept in the BDG. The data items of BDG are compared to the data items of TBS (see Table 6a). After comparing these two sets of data items, the new skylines are derived, *S''* and the BDG is updated (See Tables 6b and 6c).

For deletion, the data items to be deleted, $D_{<insert>}$, are checked against the DH table (see Table 7a). If the deleted data item is dominated by the bucket skylines, then this item is deleted from the database and the DH and no further action is needed. However, if the deleted data item is one of the bucket skylines, then all the data items which are dominated by this item are retrieved (see Table 7b). These data items are then compared together and if they dominate the other data items then they are considered as the BDG items. The remained items are then compared to the BDG items to find the new skyline points. Assume that we want to delete $w_4$ and $y_6$ from the initial database. First the DH table is checked (Table 7). Referring to the DH, $y_6$ is dominated by the bucket skyline $y_2$, thus $y_6$is deleted from the database and the DH because data items that are dominated by $y_6$ are also dominated by $y_2$ as they are in the same bucket. On the other hand $w_4$ is bucket skylines hence all the data items which are dominated by $w_4$ are retrieved and pairwise comparisons between them are performed. $w_4$ is then deleted from the DH and BDG.

**Tables 1. Bitmap Representation of Dataset**

| $w_i$ | $d_1$ | $d_2$ | $d_3$ | $d_4$ |
|---|---|---|---|---|
| $w_1$ | - | 5 | 3 | 3 |
| $w_2$ | - | 1 | 3 | 1 |
| $w_3$ | - | 3 | 2 | 3 |
| $w_4$ | - | 6 | 6 | 6 |
| $w_5$ | - | 3 | 6 | 6 |
| $w_6$ | - | 4 | 3 | 5 |
| $w_7$ | - | 3 | 2 | 2 |
| $w_8$ | - | 5 | 5 | 5 |
| $w_9$ | - | 6 | 4 | 4 |
| $w_{10}$ | - | 2 | 1 | 1 |

Bucket 1 (0111)

| $y_i$ | $d_1$ | $d_2$ | $d_3$ | $d_4$ |
|---|---|---|---|---|
| $y_1$ | 2 | 3 | - | 2 |
| $y_2$ | 6 | 4 | - | 6 |
| $y_3$ | 4 | 4 | - | 5 |
| $y_4$ | 1 | 3 | - | 3 |
| $y_5$ | 3 | 5 | - | 5 |
| $y_6$ | 6 | 3 | - | 4 |
| $y_7$ | 5 | 4 | - | 5 |
| $y_8$ | 3 | 3 | - | 2 |
| $y_9$ | 3 | 6 | - | 5 |
| $y_{10}$ | 2 | 1 | - | 2 |

Bucket 2 (1101)

| $x_i$ | $d_1$ | $d_2$ | $d_3$ | $d_4$ |
|---|---|---|---|---|
| $x_1$ | 7 | - | 6 | 6 |
| $x_2$ | 3 | - | 5 | 4 |
| $x_3$ | 5 | - | 7 | 4 |
| $x_4$ | 5 | - | 3 | 3 |
| $x_5$ | 1 | - | 1 | 1 |
| $x_6$ | 2 | - | 1 | 2 |
| $x_7$ | 7 | - | 4 | 6 |
| $x_8$ | 5 | - | 5 | 5 |
| $x_9$ | 1 | - | 3 | 3 |
| $x_{10}$ | 3 | - | 3 | 1 |

Bucket 3 (1011)

**Tables 2. BS for each Bucket**

| $w_i$ | $d_1$ | $d_2$ | $d_3$ | $d_4$ |
|---|---|---|---|---|
| $w_4$ | - | 6 | 6 | 6 |

| $y_i$ | $d_1$ | $d_2$ | $d_3$ | $d_4$ |
|---|---|---|---|---|
| $y_2$ | 6 | 4 | - | 6 |
| $y_9$ | 3 | 6 | - | 5 |

| $x_i$ | $d_1$ | $d_2$ | $d_3$ | $d_4$ |
|---|---|---|---|---|
| $x_1$ | 7 | - | 6 | 6 |
| $x_3$ | 5 | - | 7 | 4 |

**Tables 3. a) BS** **b) BDG and BDD** **c)Final Skylines**

| $w_4$ | - | 6 | 6 | 6 |
|---|---|---|---|---|
| $x_1$ | 7 | - | 6 | 6 |
| $x_3$ | 5 | - | 7 | 4 |
| $y_2$ | 6 | 4 | - | 6 |
| $y_9$ | 3 | 6 | - | 5 |

| BDG | $w_4$ | $x_1$ | $x_3$ |
|---|---|---|---|
| BDD | $y_2$ | $y_9$ | |

| $w_4$ | $x_1$ | $x_3$ |
|---|---|---|

**Table 4. D <insert>** **Table 5. TBS**

| $z_i$ | $d_1$ | $d_2$ | $d_3$ | $d_4$ |
|---|---|---|---|---|
| $z_1$ | 5 | 3 | 5 | - |
| $z_2$ | 4 | 5 | 5 | - |

| | | | | |
|---|---|---|---|---|
| $z_3$ | 6 | 7 | 5 | - |
| $z_4$ | 7 | 7 | 6 | - |
| $z_5$ | 3 | 2 | 2 | - |
| $z_6$ | 1 | 1 | 3 | - |
| $z_7$ | 2 | 2 | 2 | - |
| $z_8$ | 7 | 4 | 4 | - |
| $z_9$ | 1 | 3 | 2 | - |
| $z_{10}$ | 1 | 2 | 3 | - |

| $z_i$ | $d_1$ | $d_2$ | $d_3$ | $d_4$ |
|---|---|---|---|---|
| $z_4$ | 7 | 7 | 6 | - |

**Tables 6. a) Comparing TBS and BDG** **b) BDG and BDD** **c) Final Skylines**

| | | | | |
|---|---|---|---|---|
| $w_4$ | - | 6 | 6 | 6 |
| $x_1$ | 7 | - | 6 | 6 |
| $x_3$ | 5 | - | 7 | 4 |
| $z_4$ | 7 | 7 | 6 | - |

| BDG | $w_4$ | $x_1$ | $x_3$ | $z_4$ |
|---|---|---|---|---|
| BDD | $y_2$ | $y_9$ | $w_4$ | |

| | | |
|---|---|---|
| $x_1$ | $x_3$ | $z_4$ |

**Tables 7. a) DH** **b) Retrieved Data Items** **c) BDG**

| Dominating | Dominated |
|---|---|
| $w_4$ | $w_5$ , $w_6$ , $w_8$, $w_9$ , |
| $w_4$ | $y_2$, $y_9$ |
| $y_2$ | $y_6$ |

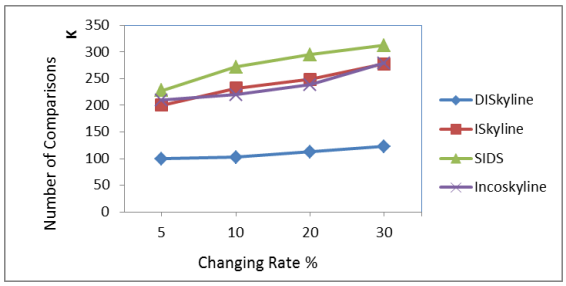
| | | | | |
|---|---|---|---|---|
| $w_5$ | - | 3 | 6 | 6 |
| $w_6$ | - | 4 | 3 | 5 |
| $w_8$ | - | 5 | 5 | 5 |
| $w_9$ | - | 6 | 4 | 4 |
| $y_2$ | 6 | 4 | - | 6 |
| $y_9$ | 3 | 6 | - | 5 |

| BDG |
|---|
| $x_1$ |
| $x_3$ |
| $z_4$ |

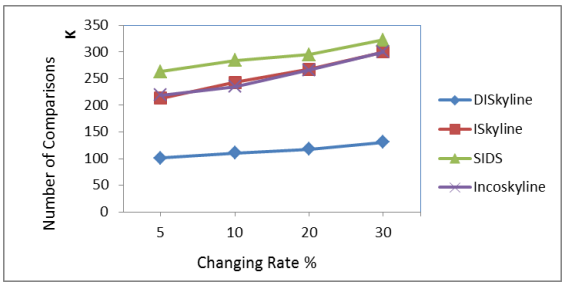
Based on DH, w4 dominates w5, w6, w8, w9, y2, and y9. Between these data items y2 and y9 dominate w5, w6 and w8, w9, respectively. y2 and y9 are then compared to the BDG data items, i.e.x1, x3, and z4 which lead to y2 and y9 being dominated. Thus, the final skylines are x1, x3, and z4.

## EXPERIMENTS

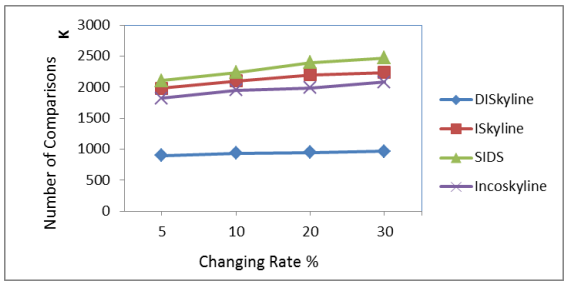
In this section, we present the results of the experiment that we have conducted. All the experiments are conducted on Intel Core i7 3.6GHz processor with 8GB of RAM. Figure 8 presents the results of the number of pairwise comparisons for processing skylines over the synthetic, NBA, stock exchanges, and MovieLens datasets with dataset size fixed to 100K, 120K, 500K and 1200K respectively; the rate of insertion and deletion are 5% to 30% while 50% of the dataset is incomplete. For ISkyline, SIDS, and Incoskyline algorithms the skyline computation is directly performed on $D_{new}$. Based on the results it can be observed that changing rate have no significant impact on the performance of our approach. This is because many of the unnecessary data items are not considered for comparisons. Also, the performance of DInSkyline algorithm outperforms ISkyline, SIDS, and Incoskyline. DInSkyline incurred 52% to 73% number of comparisons compared to the other algorithms.



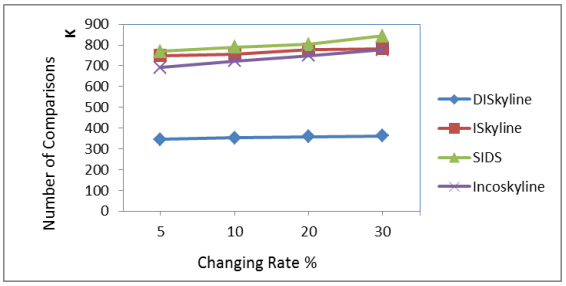

Synthetic dataset with 50% incompleteness




NBA dataset with 50% incompleteness




MovieLens dataset with 50% incompleteness




Stock exchange dataset with 50% incompleteness

**Figure 8. The Effect of Insertion and Deletion the Number of Pairwise Comparisons**

## CONCLUSION

In this study, we proposed the DInSkyline algorithm to derive skyline points over dynamic and incomplete database. The proposed algorithm works by utilizing the Bucket Skylines (BS), Bucket Dominating (BDG), and Bucket Dominated (BDD) as points of comparison as well as the Domination History (DH) table that keeps track of the domination relations. Huge amount of pairwise comparisons can be avoided as clearly shown by the results of the experiments. The performance of proposed algorithm is superior to other previous works.

## REFERENCES

Alwan, A., Ibrahim, H., Udzir, N.I., and Sidi, F. (2013). Estimating Missing Values of Skylines in Incomplete Database. *Proceedings of the 2th International Conference on Digital Enterprise and Information Systems*, pp. 220-229

Babanejad, Gh., Ibrahim, H., Udzir, N.I., Sidi, F., and Alwan, A.A. (2014). Finding Skyline Points over Dynamic Incomplete Database. *Proceedings of the Malaysian National Conference on Databases*, pp. 60-64

Balke, W.T. and Güntzer, U. (2004). Multi-objective Query Processing for Database Systems. *Proceedings of the13th International Conference on Very Large Data Bases*. pp. 936-947

Bartolini, I., Ciaccia, P. and Patella, M. (2006). Computing the Skyline without Scanning the Whole Sky. *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, pp. 405-414

Bharuka, R. and Kumar, P.S. (2013). Finding Skylines for Incomplete Data. *Proceedings of the 24th Australasian Database Conference*, pp. 109-117

Borzsony, S., Kossmann, D., and Stocker, K. (2001). The Skyline Operator. *Proceedings of the17th IEEE International Conference on Data Engineering*, pp. 421-430

Chan, C.Y., Jagadish, H.V., Tan, K.L., Tung, A.K., and Zhang, Z. (2006). Finding K-dominant Skylines in High Dimensional Space. *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pp. 503-514

Chaudhuri, S. and Gravano, L. (1999). Evaluating Top-k Selection Queries. *Journal of Very Large Data Bases*, pp. 99, 397-410

Chomicki, J., Godfrey, P., Gryz, J., and Liang, D. (2001). Skyline with Presorting: Theory and Optimizations. *Proceedings of the Intelligent Information Processing and Web Mining.* LNCS, pp. 595-604.

Cui, X.W., Dong, L.G., Zou, H., and X.M. (2011). Notice of Retraction Finding K-dominant Skyline in Dynamic Dataset. *Proceedings of the 7th IEEE International Conference on Natural Computation*, pp. 1247-1250

Fang, Y. and Chan, C.Y. (2010). Efficient Skyline Maintenance for Streaming Data with Partially-ordered Domains. *Proceedings of the Database Systems for Advanced Applications.* LNCS, pp. 322-336.

Godfrey, P., Shipley, R., and Gryz, J. (2005). Maximal Vector Computation in Large Data Sets. *Proceedings of the 31st International Conference on Very Large Data Bases*, pp. 229-240

Gulzar, Y., Alwan, A. A., Salleh, N., Al Shaikhli, I. F., & Alvi, S. I. M. (2016). A framework for evaluating skyline queries over incomplete data. *Proceedings of the 13th International Conference on Mobile Systems and Pervasive Computing*. *Procedia Computer Science*, pp. 191-198.

Khalefa, M.E., Mokbel, M.F., and Levandoski, J.J. (2008). Skyline Query Processing for Incomplete Data. *Proceedings of the 24th IEEE International Conference on Data Engineering*, pp. 556-565

Kontaki, M., Papadopoulos, A.N., and Manolopoulos, Y. (2010). Continuous Processing of Preference Queries in Data Streams. *Proceedings of the Theory and Practice of Computer Science*. LNCS, pp. 47-60.

Kossmann, D., Ramsak, F., and Rost, S. (2002). Shooting Stars in the Sky: An Online Algorithm for Skyline Queries. *Proceedings of the 28th International Conference on Very Large Data Bases*, pp. 275-28

Lee, J., You, G.W., and Hwang, S.W. (2009). Personalized Top-k Skyline Queries in High-Dimensional Space. *Journal of Information Systems. 34(1)*, 45-61

Papadias, D., Tao, Y., Fu, G., and Seeger, B. (2003). An Optimal and Progressive Algorithm for Skyline Queries. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 467-478

Papadias, D., Tao, Y., Fu, G., and Seeger, B. (2005). *Progressive Skyline Computation in Database Systems. ACM Transactions on Database Systems*, pp. 41–82

Tan, K.L., Eng, P.K., and Ooi, B.C. (2001). Efficient Progressive Skyline Computation. *Journal of Very Large Data Bases*, pp. 301-310

Yiu, M.L. and Mamoulis.N. (2007). Efficient Processing of Top-k Dominating Queries on Multi-dimensional Data. *Proceedings of the 33rd International Conference on Very Large Data Bases*, pp. 483-494