Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 170 (2020) 249–256

Procedia Computer Science

www.elsevier.com/locate/procedia

The 11th International Conference on Ambient Systems, Networks and Technologies (ANT)
April 6-9, 2020, Warsaw, Poland

A Model for Computing Skyline Data Items in Cloud Incomplete Databases

Yonis Gulzar[a,*], Ali A. Alwan[b], Abedallah Zaid Abualkishik[c], Abid Mehmood[a]

[a]*Department of Management Information Systems, King Faisal University, Al-Ahsa, 31982, Saudi Arabia*
[b]*Kulliyyah of Information and Communication Technology, International Islamic University Malaysia,53100, Kuala Lumpur, Malaysia*
[c]*American University in the Emirates, Dubai, United Arab Emirates*

Abstract

Skyline queries intend to retrieve the most superior data items in the database that best fit with the user's given preference. However, processing skyline queries are expensive and uneasy when applying on large distributed databases such as cloud databases. Moreover, it would be further sophisticated to process skyline queries if these distributed databases have missing values in certain dimensions. The effect of data incompleteness on skyline process is extremely severe because missing values result in un-hold the *transitivity property* of skyline technique and leads to the problem of *cyclic dominance*. This paper proposes an efficient model for computing skyline data items in cloud incomplete databases. The model focuses on processing skyline queries in cloud incomplete databases aiming at reducing the domination tests between data items, the processing time, and the amount of data transfer among the involved datacenters. Various set of experiments are conducted over two different types of datasets and the result demonstrates that the proposed solution outperforms the previous approaches in terms of domination tests, processing time, and amount of data transferred.

Keywords: Skyline queries; incomplete data; query processing; cloud databases; distributed database

* Corresponding author. Tel.: +966-135896205.
E-mail address: ygulzar@kfu.edu.sa

1. Introduction

Skyline query operator returns only non-dominated data items from the database, which are not worse than any of the data items in the database [1-5]. For example, a tourist is seeking for a hotel that is closest to a beach and at the same time is cheapest in price, among the set of available hotels in the hotel database, skyline technique would only retrieve the hotels that best meet the preferences of the tourist.

Skyline queries have been implemented in both complete [1, 6] and incomplete databases [2-5, 7-8, 17]. Computing skylines in complete database is not challenging in comparison to incomplete databases. This is due to the fact that in complete databases the values of each dimension in the database are always present. Thus, the issue of losing *transitivity property* of skyline technique and the problem of *cyclic dominance* are not exist. However, computing skylines in incomplete database would be further complicated. This is because in incomplete database data items might have dimensions with missing values. The missing values in the dimensions of the data items add more challenges when processing skyline queries. This is because the data incompleteness turn the process of skyline computation to be sophisticated and increase the amount of domination tests between the data items. Values missing in the dimensions of the database, not only increase the number of pairwise comparisons between data items, but it also might compromise losing the *transitivity property* of skyline technique.

Transitivity property can be defined as: for instance, there are 3 data items (d_1, d_2, d_3) and these data items contain 3 dimensions (a_1, a_2, a_3). It is assumed that there are some dimensions missing (-) in each data item d_1 (4, -, 5), d_2 (-, 6, 3), d_1 (7, 4, -). While comparing these data items to identify skylines it can be noticed that data item d_1 is dominating d_2 in dimension a_3 (max values is considered better), whereas other two dimensions are incomparable due to missing values. Comparing d_2 with d_3 we noticed that d_2 is dominated d_3 at dimension a_2. Finally, when we compared d_1 with d_3, and noticed that d_1 did not dominate d_3 in any dimension. Such kind of scenarios void transitivity property. In addition to that, we found out that d_3 dominated d_1 at dimension a_1. Therefore, these data items are dominated by one another (d_1 dominates d_2, d_2 dominates d_3, d_3 dominates d_1), which led to cyclic dominance. So, in conclusion none of these data items are considered as skylines.

The implementation of skylines queries have been incorporated in may databases including but not limited to distributed databases [9] and cloud databases [10]. In cloud databases data is spread over various different data centers at different remote locations. In order to derive the skyline data items in cloud incomplete databases, data items needs to be transferred from one data center to other in order to compute the skyline data items of the entire database. It is impractical to transfer all data items from one datacenter to another without prior pruning [9-10]. Because transferring data items from one location into another without filtration is an expensive and time consuming process. It is desirable in cloud incomplete databases to apply a data pruning technique that leads to eliminate the unnecessary data items from the skyline process which in turn result into avoid many unwanted pairwise comparisons between data items. This paper presents a model for computing skyline data items in cloud incomplete database. The proposed model incorporates an efficient approach to avoid involving the unwanted data items from further processing. Doing so leads to a great benefits by simplifying the domination tests process between data items and also reducing the total processing time of skyline operation. This paper concentrates on deriving skyline data items in a cloud incomplete database with horizontal partitioning of data spread over different datacenters.

The reminder of this paper is orderly as follows. Section 2 discusses the previous works relevant to skyline queries in both complete and incomplete databases. Section 3 describes the basic necessary definitions and notations related to skyline queries. The detail steps of the proposed approach are explained in section 4. The experiment results that demonstrate the effectiveness and the efficiency of the proposed approach are presented and explained in section 5. Lastly, section 6 outlines the conclusion of the paper.

2. Related Work

Numerous approaches have been proposed since the first introduction of skyline queries in database systems. The literature of the database reported that the work in [1] is the first that highlights the issue of skyline queries in database systems. The proposed skyline operator introduced in [1] has been designed based on the concept of Block-Nested-Loop (BNL) and Divide-and-Conquer (D&C) [1]. BNL compares one data item with the rest and this process is continued until whole dataset is scanned. Whereas, in D&C dataset is divided into 2 distinct subsets and after identifying the skyline data items from each subset, the skyline data items of these subsets are further evaluated with each other to identify the skyline data items of the entire dataset. After the introduction of BNL and D&C many other algorithms were proposed aiming at improving the computation of skyline queries over a database with complete data. Among the remarkable skyline techniques are SFS [11], LESS [12], SaLSa [6], BSkyTree [13] and many others. These approaches focus on designing skyline technique that avoid scanning the entire database when deriving the skyline data items. The work in [3] is the first that discussed that problem of skyline queries computation in a database with incomplete data. They have proposed an approach called ISkyline that attempt to handle the issue of data incompleteness in the database when deriving skyline data items. Many other skyline approaches have been proposed to process skyline queries in incomplete data. These approaches are either sorting-based technique or partitioning-based technique. For instance, SIDS [4], framework [2], SOBA [14], SPQ [15], SCSA [5] are incorporate either sorting or partitioning technique to compute the skyline data items in incomplete data.

To our best knowledge, the most recent work skyline queries in cloud incomplete data presented in [10]. The work in [10] presents and approach named ICS that incorporate a sorting technique to determine the skyline data items in cloud incomplete databases. ICS starts by sorting each dimension values in descending order (higher value is better) and *n* number of arrays are constructed to store *id* of each sorted data items (*n= number of dimensions in dataset*). Then a scanning process is carried out to filter out these data items that do not have potential to be part of final skylines. After that local skylines are identified by comparing the data item with one another. At the end, local skylines of each datacenter are transferred only to the query submitted datacenter where whole process is getting repeated to identify final skylines of entire data. However, in ICS algorithm losing of *transitivity property* is not properly well addressed. No measures have been taken in order to hold *transitivity property*. Moreover, no additional optimization technique has been used to reduce amount of data items before applying skyline process.

3. Definitions

This section gives the basic necessary definitions and annotations relevant to skyline queries in the context of incomplete database to facilitate understanding the detail steps of the proposed approach of processing skyline queries in cloud incomplete database.

Definition 1, *Incomplete Database*: Given a database D (R_1, R_2, ..., R_n), where R_i is a relation denoted by R_i (d_1, d_2, ..., d_m), D is said to be incomplete if (and only if) it contains at least a data item p_j with missing values in one or more dimensions d_k (attributes); otherwise, it is complete.

Definition 2, *Dominance on Incomplete Database*: Given two data items p_i and $p_j \in D$ incomplete database with d dimensions, p_i dominates p_j (denoted by $p_i \succ p_j$) if (and only if) the following three conditions hold:

1. The values of d_k and $d_l \in d$ for p_i and p_j must be non-missing and

2. $\forall\ d_k \in d, p_i.d_k \geq p_j.d_k$ and

3. $\exists\ d_l, \in d, p_i.d_l > p_j.d_l$

Definition 3, *Skyline Queries on Incomplete Database*: Select a data item p_i from the set of D incomplete database if (and only if) p_i is as good as p_j (where $i \neq j$) in all common non-missing dimensions and strictly better than p_j in at least one common non-missing dimension. We use *IncoDskyline* to denote the set of skyline data items on an incomplete database, *IncoDskyline* = ($p_i\ \forall\ p_i, p_j \in D, p_i \succ p_j$).

4. The Proposed Model for Computing Skyline Data Items in Cloud Incomplete Databases

This section elaborates the detail steps of the proposed model for computing skyline data items in cloud incomplete databases. The proposed model is made of three modules, namely: (i) Identify local skyline data items of each datacenter, (ii). Join local skyline data items, and (iii). Evaluate final skyline data items.

4.1. Identifying Local Skyline Data Items of Each Datacenter

This module identifies the local skyline data items of all involved datacenters to help in removing the dominated data items from the join operation. In order to achieve this goal this module is further divided into five submodules, which are sorting and filtering, clustering and grouping, identifying local skyline data items, selecting superior local skyline data items, and retrieving skyline data items.

4.1.1. Sorting and Filtering

In this submodule the data items are sorted in non-ascending order for each dimension and a set of array list is created to save the indices (ID's) of the sorted data items. Then, a scanning process is carried out to read all the data items ID's present in array list in a round robin fashion. The number of occurrences of each data item ID is calculated and stored in 2D array list. Scanning processes is stopped only when all data items present in the relation of the datacenter are read at least once. Furthermore, an optimization technique called filtering is implemented in order to filter out non-dominant data items before applying skyline technique. A user defined threshold value is defined and the occurrence value (count) of each data items from 2D array is compared with the threshold. If the count is less than the threshold, then those data items are eliminated from further processing.

4.1.2. Clustering and Grouping

The aim of this submodule is to ensure that the transitivity property of skyline technique is hold and the problem of cyclic dominance is avoided. This is achieved by dividing the remaining data items present in 2D into distinct clusters based on the 'count' value of each data item. All these data items with the same count value will be placed in one cluster. Then each cluster is further subdivided into distinct groups based on the bitmap representation of the data items. Dividing of data items into clusters and groups guarantee that transitivity property of skyline is hold.

4.1.3. Identifying Local Skyline Data Items

This submodule identifies skylines of each cluster of the relation. This is performed by comparing the data items present in each group against each other. Dominated data items will be removed from further processing while the remaining data items present in each group will be further compared with the data items of other groups (cross comparison is performed) present in the same cluster. At the end of the process dominated data items will be removed and remaining data items present in cluster will be reported as local skyline data items of the cluster. It is important to notice that the process of identifying local skyline data items of clusters will be executed simultaneously for each cluster. The idea behind this is to reduce the processing time to identify the local skyline data items.

4.1.4. Selecting Superior Skyline Data Items

This submodule incorporate an optimization technique that helps in further pruning the local skyline data items from further processing. This is performed by selecting only those data items in each cluster that have highest value(s) in any dimension than other data items. Doing so, helps to further eliminate non-dominant data items that in turn reduce the number of data items considered for next processes.

4.1.5. Retrieving Skyline Data Items

In this phase, the final skyline data items of each relation is identified. It attempts to return the skyline data items of all involved datacenters. This is achieved by comparing local superior skyline data items of each cluster against each other. At the end of the process all dominated data items in all clusters of a relation are eliminated and only the remaining data items are considered as skyline of the relation. The process of identifying local skylines of each datacenter is run parallelly for each datacenter in order to reduce the processing time. At the end of, the local skyline data items of all relations ($R1, R2, R3$) stored at different datacenters ($DC1, DC2, DC3$) involved are identified.

4.2. Joining Local Skyline Data Items

This is the second module of our proposed model for computing skyline data items in cloud incomplete databases. The aim of this module is to reduce the amount of data transfer from one datacenter to another. To do so, the local skyline data items of all involved relations (R_1, R_2, R_3) present at different datacenters are joined. The skyline data items from all relations are combined in one relation R. The data items of relation R are called candidate skyline data items. These candidate skyline data items will be further processed to identified the final skyline data items.

4.3. Evaluating Final Skyline Data Items

This model helps in identifying final skyline data items after the join operation has been performed on the joined local skyline data items. This is achieved by repeating the steps present in the first module of our proposed model. The process will start by sorting the candidate skyline data items in non-increasing order for each dimension of relation *R* as mentioned in module 1 above. Array lists are constructed for each dimension to store *ID's* of sorted data items. Then the scanning process begins by reading each sorted data item *ID's* in round robin fashion and the presence of each data item will be counted and stored in 2D array. Filtering process will be applied again as mentioned in module 1 to eliminate the data items that do not have a potential to be part of the final skyline data items for entire database. After filtration process, clustering and grouping takes place to ensure that the cyclic dominance would not be occurred and the *transitivity property* of skyline remain hold. Next, local skyline data items are being identified from each cluster and dominated data items are being eliminated from further processing. Optimization technique of selecting superior skyline data items is applied again to prune the remaining data items. At the end, the remaining data items in each cluster are compared against the data items of other clusters to eliminate the dominated data items and the remaining data items will be considered as final skyline data items of the entire database.

5. Experimental Environment

The proposed model has been compared with the most recent works such as SCSA [5], ICS [10], *IncoSkyline* [16], and SIDS [4]. Since skyline computation is CPU intensive [1-5, 7-8, 10, 16-17] and high number of domination tests between data item is required. Therefore, this work focuses on measuring the efficiency of the proposed model with respect to domination tests between data items, processing time and amount of data transfer between datacenters. These are considered the most influenced parameters in skyline query processing. The number of pairwise comparisons and processing time are been calculated with respect to number of dimensions as well as dataset size of cloud incomplete database. Two types of datasets (synthetic and real) have been used to evaluate the proposed model. For synthetic dataset a corelated and independent datasets have been used while one real dataset (NBA) has been used [1-5, 7-8, 16-17]. For simplicity and without losing generality, we assumed that the query statement attempts to retrieve the skyline data items with the highest values. Table 1 summarizes the parameter settings for synthetic and real datasets used in the experiments.

5.1. Experimental Results

This section highlights the experimental results performed on the synthetic and real datasets for our proposed model of processing skyline queries in cloud incomplete databases. In this section, we attempt to investigate the impact of database dimensionality (number of dimensions) and the influence of database cardinality (dataset size) on the process of pairwise comparison and the processing time for skyline evaluation and amount of data transfer from one datacenter to another. We argue that these are the most crucial parameters that influence the skyline query processing[1-5,7-10, 16].

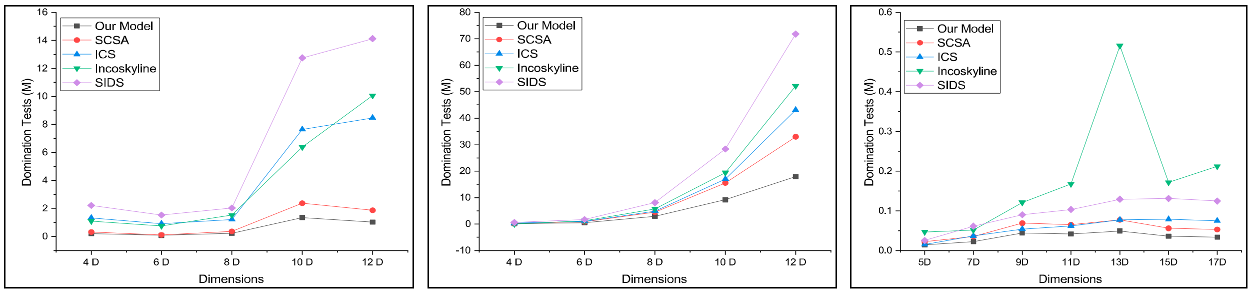
Table 1. The parameter settings of synthetic and real datasets

Parameter Settings	**Dataset Name**		
	Correlated	***Independent***	***NBA***
No. of Dimensions	5, 7, 9, 11, 13	5, 7, 9, 11, 13	6, 8, 10, 12, 14, 16, 18
No. of Dimensions with missing value	4, 6, 8, 10, 12	4, 6, 8, 10, 12	5, 7, 9, 11, 13, 15, 17
Dataset Size (KB)	100, 200, 300, 400, 500, 600	100, 200, 300, 400, 500, 600	40, 80, 120, 160, 200
Datacenters	3	3	3

5.1.1. Effect of Number of Dimensions

This set of experiments assess the impact of number of dimensions on the process of domination tests to identify skyline in cloud incomplete data. In the first part of this experimental results we calculate pairwise comparisons taken between data items in order to identify final skylines. In the second part of this experimental results we calculate the processing time taken by all approaches to identify final skylines in cloud incomplete database. In this set of experiments, the size of dataset is fixed and number of dimensions are varying.

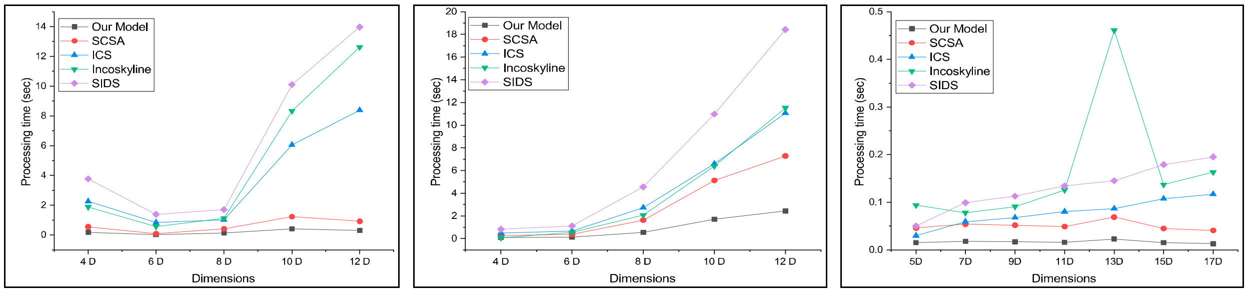
Fig. 1a and 1b illustrate the results of correlated and independent synthetic dataset respectively. In which number of dimensions are varying from 4 to 12 and sized is fixed to 300KB. From the figure it can be noticed that our model is persistently outperforming the *SCSA*, *CIS*, *Incoskyline*, and *SIDS*. Furthermore, Fig. 1c represents the results of NBA real dataset in which dataset size of fixed to 120 KB and number of dimensions are varying from 5 to 17. It can be clearly seen that our model better than *SCSA*, *CIS*, *Incoskyline*, and *SIDS*. It is due to using of two optimization techniques which helps to eliminate dominated data items as early as possible.



a) Correlated b) Independent c) NBA

Fig. 1. The effect of number of dimensions on the number of pairwise comparisons.

Fig. 2a, 2b and 2c represents the result of correlated and independent and NBA datasets where processing time is being calculated of our model and existing approaches. This set of experiment have same parameter setting as of previous experiments for synthetic and real datasets. From the figures it is clearly seen that our model is taking less processing time when it comes to identify skylines in cloud incomplete data. our model is outperforming *SCSA*, *CIS*, *Incoskyline*, and *SIDS* approaches.



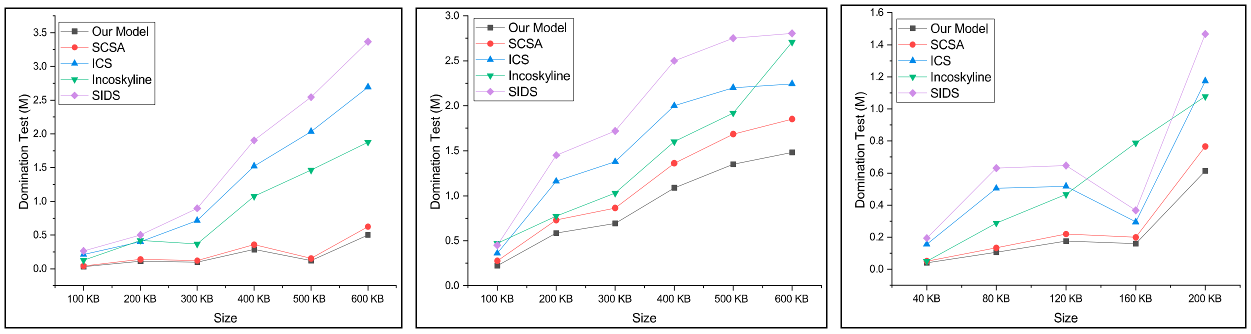
a) Correlated b) Independent c) NBA

Fig. 2. The effect of number of dimensions on processing time.

5.1.2. *Effect of Dataset Size*

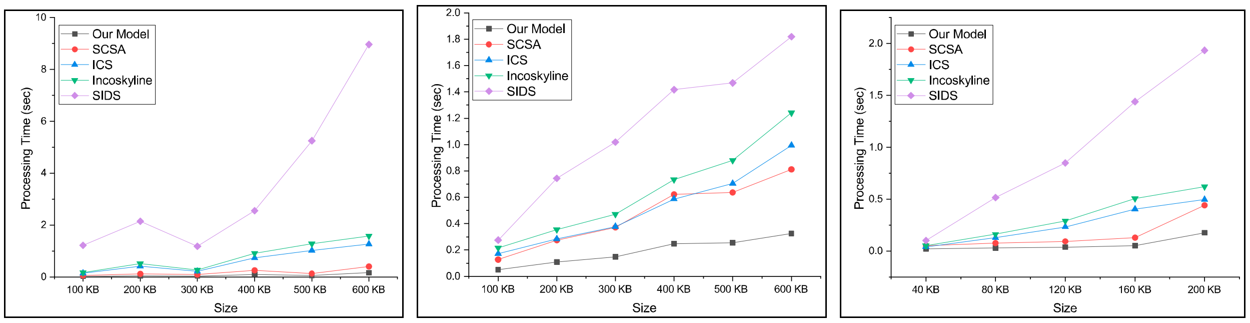
Fig. 3a, 3b and 3c explain the results of number of domination tests that have been assessed while identifying skylines in correlated, independent and NBA dataset respectively. in this set of experiment, the dataset dimensions are fixed, and dataset size is varying. For correlated and independent dataset, the number of dimensions is fixed to 6 and dataset size is varying from 100KB to 600KB. For NBA dataset the number of dimensions is fixed to 7 and dataset size is varying from 40KB to 200KB. From the figures, it can be noticed that our model is outperforming *SCSA*, *CIS*, *Incoskyline*, and *SIDS* approaches. This is achieved by using sorting and filtering technique to filter out those data items that do not have potential to be part of final skyline.

Fig. 4a, 4b, and 4c explains the results of processing time taken by all approaches to identify final skylines in correlated, independent and NBA datasets respectively. For this set of experiments the parameter settings are same as previous experiment. From the fig. 4 we conclude that our model is outperforming all other approaches proposed for skyline query processing in incomplete data. this is achieved by using clustering and grouping technique, which identifies local skylines of each datacenter parallelly and simultaneously. Doing so helps in reducing processing time while assessing skylines in cloud incomplete data.



a) Correlated b) Independent c) NBA

Fig. 3. The effect of dataset size on the number of pairwise comparisons.



a) Correlated b) Independent c) NBA

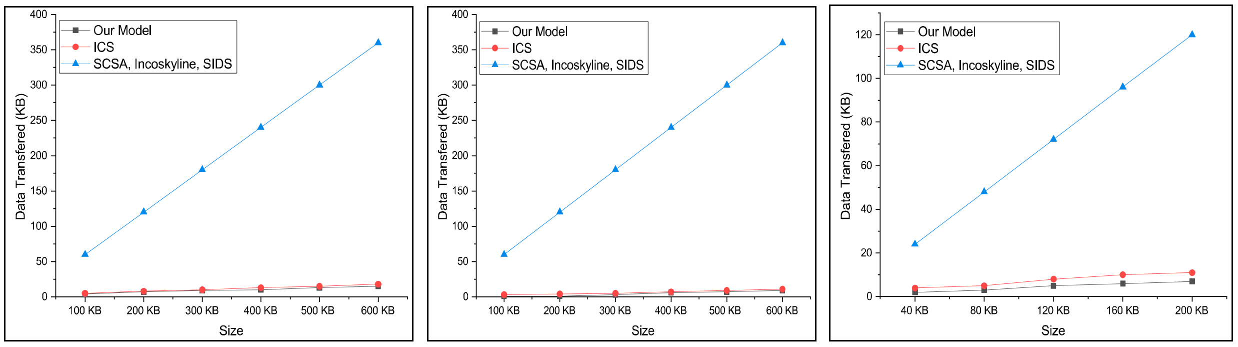
Fig. 4. The effect of dataset size on processing time

5.1.3. *Effect of Amount of Data Transfer*

In this set of experiment, the amount of data transfer from one datacenter to another is evaluated. The amount of data transfer indicated that the number of data items that need to be transferred from one datacenter to another. In our proposed model only local skylines of each realtion involved which are present at different datacenters located at remote locations are transferred. Fig. 5a, b and c represents the amount of data tranfered from one datacenter to another of correlated, indepdent and NBA datasets respectively. In our propsoed model it has been assumed that three datacenters have been used where data is distributed horizontally. From Fig. 5 it can be noticed that our proposed model reduces the amount of data tranfered to 96% to 98%, whereas the existing approaches (SCSA, *Incoskyline*, and SIDS) send whole dataset from one datacenter to another in order to identify the final skylines. Our proposed model also outperforms ICS approach it is because of using optimization technique called selecting superior local skylines which reduces the total amount of local skylines to be transferred from one datacenter to another which in turn reduces the network cost.

6. Conclusion

This paper presents a model for computing skyline data items in cloud incomplete databases. We described the detail components of the model and explained how it managed to identify the skyline data items over a cloud databases with incomplete data. Two optimization techniques have been proposed aiming at reducing the domination tests and decrease the amount of data transfer from one datacenter to another during the skyline query process. Several experiments have been conducted and the results show that our model has significantly outperformed the previous existing approaches.



a) Correlated b) Independent c) NBA

Fig. 5. Amount of data transfer

References

[1] S. Borzsony, D. Kossmann, and K. Stocker, "The Skyline operator," in *Proceedings 17th International Conference on Data Engineering*, Cancun, Mexico, 2001, pp. 421-430.

[2] Y. Gulzar, A. A. Alwan, N. Salleh, I. F. A. Shaikhli, and S. I. M. Alvi, "A Framework for Evaluating Skyline Queries over Incomplete Data," *Procedia Computer Science,* vol. 94, pp. 191-198, 2016/01/01/ 2016.

[3] M. E. Khalefa, M. F. Mokbel, and J. J. Levandoski, "Skyline Query Processing for Incomplete Data," in *IEEE 24th International Conference on Data Engineering*, Cancun, (Mexico). PP. 556-565, 2008, pp. 556-565.

[4] R. Bharuka and P. S. Kumar, "Finding skylines for incomplete data," presented at the Proceedings of the 24th Australasian Database Conference - Volume 137, Adelaide, Australia, 2013.

[5] Y. Gulzar, A. A. Alwan, R. M. Abdullah, Q. Xin, and M. B. Swidan, "SCSA: Evaluating skyline queries in incomplete data," *Applied Intelligence,* vol. 49, pp. 1636-1657, May 01 2019.

[6] I. Bartolini, P. Ciaccia, and M. Patella, "SaLSa: computing the skyline without scanning the whole sky," presented at the Proceedings of the 15th ACM international conference on Information and knowledge management, Arlington, Virginia, USA, 2006.

[7] M. B. Swidan, A. A. Alwan, S. Turaev, and Y. Gulzar, "A Model for Processing Skyline Queries in Crowd-sourced Databases," *Indonesian Journal of Electrical Engineering and Computer Science,* vol. 10, pp. 798-806, 2018.

[8] Y. Gulzar, A. A. Alwan, H. Ibrahim, and Q. Xin, "D-SKY: A Framework for Processing Skyline Queries in a Dynamic and Incomplete Database," presented at the Proceedings of the 20th International Conference on Information Integration and Web-based Applications & Services, Yogyakarta, Indonesia, 2018.

[9] A. A. Alwan, H. Ibrahim, N. I. Udzir, and F. Sidi, "Processing skyline queries in incomplete distributed databases," *Journal of Intelligent Information Systems,* vol. 48, pp. 399-420, April 01 2017.

[10] Y. Gulzar, Ali A. Alwan, N. Salleh, and I. F. Al Shaikhli, "Identifying Skylines In Cloud Databases With Incomplete Data," *Journal of ICT,* vol. 18, pp. 19-34, 2019.

[11] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting," in *Proceedings 19th International Conference on Data Engineering (ICDE03)*, Bangalore (India), 2003, pp. 717-719.

[12] P. Godfrey, R. Shipley, and J. Gryz, "Maximal vector computation in large data sets," presented at the Proceedings of the 31st international conference on Very large data bases, Trondheim, Norway, 2005.

[13] J. Lee and S.-w. Hwang, "Scalable skyline computation using a balanced pivot selection technique," *Information Systems,* vol. 39, pp. 1-21, 2014/01/01/ 2014.

[14] J. Lee, H. Im, and G.-w. You, "Optimizing Skyline Queries over Incomplete Data," *Information Sciences,* vol. 361, pp. 14-28, 2016.

[15] Y. Wang, Z. Shi, J. Wang, L. Sun, and B. Song, "Skyline Preference Query Based on Massive and Incomplete Dataset," *IEEE Access,* vol. 5, pp. 3183-3192, 2017.

[16] A. A. Alwan, H. Ibrahim, N. I. Udzir, and F. Sidi, "An Efficient Approach for Processing Skyline Queries in Incomplete Multidimensional Database," *Arabian Journal for Science and Engineering,* vol. 41, pp. 2927-2943, 2016.

[17] Y. Gulzar, Ali A. Alwan, and S. Turaev, "Optimizing Skyline Query Processing in Incomplete Data," *IEEE Access,* vol. 7, pp. 178121-178138, 2019.