2024 IEEE 9th International Conference on Engineering Technologies and Applied Sciences (ICETAS)

2024 IEEE 9th International Conference on Engineering Technologies and Applied Sciences (ICETAS) | 979-8-3503-6314-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICETAS62372.2024.11119942

The Conceptual Framework of Knowledge of Large Scale and Incomplete Graphs of Skyline Queries Optimization Using Machine Learning

Ubair Noor
Department of Computer Science, Kulliyah of Information and Communication Technology International Islamic University Malaysia Kuala Lumpur, Malaysia
ubairnoor@gmail.com

Raini Hassan
Department of Computer Science Kulliyah of information and Communication Technology International Islamic University Malaysia Kuala Lumpur, Malaysia
hrai@iium.edu.my

Dini Oktarina Dwi Handayani
Department of Computer Science Kulliyah of Information and Communication Technology International Islamic University Malaysia Kuala Lumpur, Malaysia
dinihandayani@iium.edu.my

***Abstract*— Ever since its introduction into the database community, skyline queries have been widely adopted in a range of contemporary database applications. Skyline technique relies on the concept of Pareto-optimal in which a data item from the set of dataset D is identified as skyline if and only if it is not worse than other data items in all dimensions (attributes) and strictly better in at least one dimension. Most of the previous skyline solutions have been designed for conventional databases for complete, incomplete, and uncertain data. However, not much attention has been paid to issues related to skyline query processing over knowledge of large-scale incomplete graph databases. Most recently, graphs have become prevalent data structures to model complex information networks for various real-life contemporary applications such as social networks, knowledge bases, pattern recognition, and the World Wide Web. It is also important to note that, generally graphs are big structures with very big data and this change often due to updates. These continuous updates makes the graph to be highly dynamic, where nodes/edges are added to or removed from the graph always. However, the issue of data incompleteness when processing skyline queries in large-scale graph databases has not been considered by previous works. The research aims at proposing a new model for processing skyline queries in an incomplete graph database. The research methodology includes reviewing the related literature of skyline queries in incomplete graph databases. Then, propose an method for handling skyline queries within an incomplete graph database followed by designing and implementing a model to evaluate the efficiency and effectiveness of the proposed approaches. The preliminary results using the K means Clustering Algorithm showed that the conceptual framework successfully grouped similar data points, facilitating the identification of skyline points. The implemented algorithm to perform such operation was far more efficient, faster and accurate as compared to conventional methods. This research will ultimately benefit a wide range of applications involving decision-making, decision support, social network, and recommendations aspects by developing a tool that incorporates the proposed approaches.**

Keywords—Skyline Query, Graph Database, Machine Learning

I. INTRODUCTION

A skyline query is used in database applications to retrieve the non-dominated tuples from a database, known as skylines. The main idea of the skyline queries is to identify the nodes not dominated by any other node in the graph database, based on certain criteria or preferences [1].Skyline queries are often used in numerous modern database applications for decision-making process, multi-criteria, road networks, web-based businesses, crowd-sourcing databases and e-commerce.

A major challenge in skyline queries, data incompleteness leads to the loss of the transitivity property, causing the dominant relationships between data items to become cyclic. Skyline queries have shown to be useful and practical tool in many real-world database applications. Processing skyline queries in graph databases poses a significant challenge in database management. Innovative techniques are required for effective and precise skyline computation due to incomplete and uncertain graphs, which are characterized by rapid changes and missing values within tuples or nodes. The aim is to reduce the search space and minimize the cost of computing and the time complexity to identify the skyline of the graph.

Currently, the issue of dynamic skyline queries in uncertain graphs is tackled by the researcher. They emphasize the issue of identifying the superior data vertices concerning the query vertices based on two distance measures (expected distance and majority distance) that fit the uncertain graphs [2]. Other researchers attempt to investigate the problem of continuous subgraph multi-queries processing over graph streams. Based on a literature review insufficient attention has been directed towards addressing the issue of incomplete data within graph databases when handling skyline queries. This incompleteness poses a novel challenge when attempting to process such queries within graph databases.

As it pertains to real-world use, graphs are quite dynamic; new nodes are introduced or nodes are removed. If these graphs were represented as relational tables, they would typically exhibit sparsity with numerous dimensions. Graphs inherently accommodate a wealth of attributes, necessitating the development of efficient indexing methods to enhance the computational efficiency of identifying skyline entities [3], [4], [5], [6]. Despite the potential presence of numerous numeric attributes, indicating rare dominant relationships within knowledge graphs, limited efforts have been allocated towards resolving the challenges associated with processing skyline queries within graph databases.

The effects of data incompleteness are thus worsened by the dynamism of graphs, where nodes and edges can be added or removed arbitrarily. It shows a possibility of graphs' sparseness because nodes often contain much information while they have few links to other nodes, regarding the representation of graphs as relational tables. Due

to the low number of connections, there are challenges in identifying the skyline entities in this sparse connectivity.

Moreover, the practice of applying data pruning before executing skyline queries adds new complications related to partial information about nodes and edges to the list of challenges of using graph databases. Solving these problems requires a collective approach towards devising approaches to manage data incompleteness and enhance skyline query processing for graph databases.

The objectives of this work are to understand and analyze knowledge of large graphs complete and incomplete graphs of skyline queries and to design and develop an approach efficient data pruning technique that best works over incomplete graph database using machine learning models.

II. Related Works

To solve dynamic skyline queries in large graph databases, data points that are independent of any other points across all dimensions have been defined, known as skylines of the graph. This method aims at trying to find the skyline by trying to reduce the search space and any computational cost which is incurred in the process [7].

A new method, already advanced in similarity skyline, has been made to address the problem of multi-measure similarity search in a graph database. In this approach, it focuses on identifying graphs that are similar to a graph query, and similarity between graphs are in terms of scalars or multiple measures. The similarity skyline of a graph query is defined as a subset of graphs from the target database that are the most similar to the query in the Pareto sense [8].

The subgraph skyline problem over large graph data deals with the issue of finding a subgraph within a large graph (G), such that a subgraph (g) is graph isomorphic to a query (q) and is not dominated by any other subgraphs. Three different challenges have been outlined: firstly, dynamic skyline computation, which computes skyline to the numeric attributes specified in query graphs; secondly, efficient querying on graphs, which involves identifying skylines in the knowledge graph and requires checking for structural constraints before outputting the true answers [9] and thirdly, reducing expensive storage costs, which involves selecting numeric and structural features in a very large graph, a process that is costly and consumes a large space, potentially impacting the pruning process and storage space.

Skyline queries in graph databases involve algorithms [10] such as divide and conquer and nested loop which is already adapted by the relational database. Skyline queries in graph databases involve using two algorithms: nested loops and divide-and-conquer. The nested loops algorithm compares each node against all other nodes to determine domination, with a time complexity of $O(n^2)$. On the other hand, the divide and conquer splits the data into parts, compares the nodes within the partitions and then merges the outcomes, thereby providing better time complexity of n log^(d-1) n. From the performance analysis, we conclude that the divide-and-conquer algorithm takes less time than the nested loops algorithm for numerous graph database sizes and query complications.

A method to estimate missing values of the skyline is developed through four phases [11] to find missing values, the method generates Attribute Functional Dependencies (AFDs) by dividing skylines with missing values into two sets: The second skyline is one that contains elements with missing values in the target dimension, and the second skyline contains all the other skylines. It identifies the effect of one dimension on the other, in order to construct an AFD, which depicts the interconnection of these dimensions. The resulting values are then used for the measure of correlation strength, which in turn enables the estimation of the missing values by approximate ones. Last of all, the skylines are sorted by the strength with which probability dependencies have been identified thereby providing high quality skylines to the user.

The path skyline query problem in bicriteria networks, which involves finding all skyline paths from a starting at an origin node and reaching a target node, is crucial for optimizing dual criteria simultaneously [12]. The proposed method, PSQ+, builds on the PSQ algorithm by starting with the initial node in a queue. It processes elements by checking if the current path is dominated by the last skyline path. If it is, the path is added to the current node's skyline, followed by edge relaxation for neighboring nodes. This goes on until the queue becomes empty, and all skyline paths are identified. Compared to the previous version, PSQ+ increases the performance since the algorithm does not need to handle non-skyline curves.

The primary concern in skyline query processing with incomplete data can be attributed to large cardinality and high dimensionality databases [13]. First, the data items they are grouped because this alleviates the problem of cyclic dominance. Subsequently, sorting and filtering get rid of dominated items, which in return minimize the frequency of carrying out domination tests. Thus, specific local skylines for each candidate list are determined to achieve parallel processing which in turn quickens the process of skyline retrieval. This means that the problems that can be associated with cyclic dominance as well as the transitivity property are accounted for, even in the presence of missing values. Last of all, the method acquires the whole set of skyline data items by comparing local skyline of every candidate list.

Moreover, the processing of skyline queries in incomplete datasets focuses on the SCSA algorithm [14] which efficiently addresses situations where data values are missing. The algorithm begins by arranging the data items according to dimensions values in descending order. In addition, it accumulates the domination power of each item by scanning the sorted lists, enabling effective filtration to prune dominated items. Furthermore, the remaining data items are partitioned into clusters according to their domination power and then divided into smaller groups with identical bitmap representations. By running the algorithm in parallel on these groups, unwanted data items are eliminated effectively. Finally, the algorithm compares local cluster skylines to return only those not surpassed by any others in all dimensions.

Furthermore, the focus on skyline nearest neighbor search in multi-layer graphs presents significant advancements. The proposed algorithm incorporates an early-termination condition, which enables the computation of shortest distances to stop once a vertex has been visited across all layers [15]. This approach effectively reduces unnecessary computations. In addition, optimization strategies, such as refining the search order, are employed to further enhance the algorithm's efficiency, ensuring a more streamlined process in identifying skyline nearest neighbors.

Moreover, the exploration of dynamic skyline queries on uncertain graphs introduces a systematic processing method comprising three key steps: Pruning, Distance Computation, and Skyline Vertex Set Generation [16] Pruning occurs in two phases: the first computes path lengths, and the second calculates distances between candidate skyline vertices and query vertices using direct and expected distance measures. During the Skyline Vertex Set Generation phase, the block nested loop (BNL) algorithm is utilized. This combined approach effectively prunes candidate vertices, computes required distances and generates the skyline vertex set based on the results.

Besides, the IDSA algorithm addresses skyline queries in dynamic and incomplete databases through seven phases. It begins with a Pruning Process to identify new skylines, reducing domination tests by leveraging existing skylines before INSERT/UPDATE operations. Step 2 minimizes further domination tests by selecting superior local skylines [17] Steps 3 and 4 involve adding and removing tuples from the database. Step 5 reviews prior methods and challenges. Step 6 produces new candidate tuples on domination power; Step 7 evaluates

the performance of the algorithm experimentally to demonstrate that the skylines after database changes are accurate.

The algorithm presented in [18] optimizes skyline query processing by sorting dataset points into distinct lists based on each dimension and accessing them in a round-robin manner. This indexing technique improves the access efficiency by using the extent component and density aspect best represented by the no. of complete dimensions of any point as the basis for dominance.

Moreover [19] encompasses a two-phase process: modeling and crowdsourcing. In the modeling phase, a Bayesian network is trained to capture data correlations, and a c-table is constructed to represent objects and their associated conditions. This c-table is essential for determining the likelihood of each object being a query result. Furthermore, the crowdsourcing phase involves selecting tasks for crowd workers to resolve missing data issues, employing strategies such as Frequency Based Strategy (FBS), Uncertainty Based Strategy (UBSand Hybrid Heuristic Strategy (HHS). Specifically, FBS prioritizes the most frequent expressions, UBS targets tasks with high uncertainty, and HHS combines elements of both to enhance efficiency. This iterative framework refines query results based on crowd responses until the query is successfully resolved.

In addition, [20] tackles incomplete skyline queries with a two-stage processing method for efficient computation on large datasets. In Stage 1, TSI performs a sequential scan to identify candidate tuples, discarding dominated ones. Stage 2 further refines these candidates with another scan. To boost efficiency, TSI uses a pruning bit-vector (PRB) to streamline the process, significantly cutting execution costs by skipping many tuples in Stage 1. However, some candidates may still require removal in the second stage. Overall, TSI shows high efficiency in computing skylines on large incomplete datasets.

Furthermore [21] explores skyline path queries over temporal graphs with labels, proposing an efficient index-based method for processing these queries. The paper introduces the Main Point (MP) index, which includes phases for MP discovery and Mout set construction, enabling effective handling of skyline path queries by identifying key points in the graph. In addition, the TMP algorithm, built on the MP index, employs a bidirectional topology strategy to address skyline path queries while accommodating multiple constraints related to temporal and label elements. Through experiments and comparisons with other algorithms, the proposed methodology demonstrates significant performance improvements and effectiveness in managing skyline path queries in complex temporal graph environments.

Recently, [22] introduces the ProbSky framework, which utilizes the MapReduce paradigm to efficiently evaluate probabilistic skyline queries on large, uncertain datasets. The methodology begins with slab-based partitioning of the dataset, allowing for the computation of local skyline points within each partition. Furthermore, the framework calculates skyline probabilities of uncertain objects using reference points, which accelerates the evaluation process. To enhance efficiency and scalability in a distributed computing environment, three optimization techniques are incorporated: dominant instance pruning to eliminate unqualified objects early, slab-based partitioning to balance workload and minimize communication costs, and reference point-based acceleration to avoid unnecessary dominance tests. Collectively, these techniques significantly improve the performance of the framework.

Finally [23] focuses method for classifying incomplete data using a weighted classification tree, where missing values are marked as 0 and non-missing as 1. Skyline queries are performed within each class to identify local skyline points. It introduces optimal virtual points, representing the maximum local skyline values, which help minimize comparisons across classes. Points dominated by these virtual points are shifted to a shadow skyline. Global skyline points are determined by comparing candidates in each class with the shadow skylines of others, eliminating dominated points. This approach efficiently handles multidimensional incomplete data, enhancing classification efficiency and reducing comparisons, as shown by improved experimental results.

III. METHODOLOGY

The framework consists of 5 components: sorting and filtering, Creating a cluster, local skylines identifier, and final skyline. Fig. 1 illustrates the proposed framework of skyline queries in the incomplete graph database. Those components are further explained as follows.

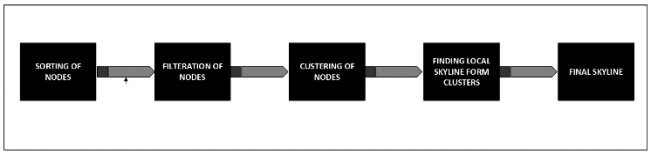

Fig. 1. Proposed Methodology Framework

A. Sorting of Nodes

This phase endeavours to arrange the data nodes within the dataset in descending order according to the domination power of each node. The process begins by organizing the nodes within each distinct list based on the values of properties within each node. The interaction between the nodes which ultimately leads to domination power is done on a round-robin basis. This procedure begins and goes on until all nodes within the initial dataset are visited at least one time. The goal of this step, the node items in the dataset to be sorted with decreasing domination power. Further, it tries to prune out nodes that are dominated by other nodes having lesser values of domination power. As a result, data items with low or equal domination power are considered not to be effective to the extent of contributing to the skyline results. Therefore, the elimination of them before applying the skyline technique would give way to substantial reduction of useless comparisons and the load of the skyline process.

B. Filtering of Nodes

In the filtration stage, the domination power of each data nodes is calculated and any data node with domination power less than the provided threshold are pruned from further processing. This decision stems from the understanding that nodes with a domination power below than the threshold lack the potential to be included in the skyline result, as their domination power suggests they excel in no more than one dimension. The core concept behind the filtration process hinges on leveraging domination power values to streamline the skyline process within an incomplete graph database.

C. Clustering of Nodes

Specifically, the purpose of this phase is to enhance the skyline computation within a dataset by splitting the data nodes by the missing values of the item. Here again, data items with similar missing value is grouped into one cluster by the help of k means clustering algorithm. Consequently, it gives rise to the creation of several numbers of separate clusters. Eliminating many unwanted pairwise comparisons as reduction in the number of data items does not have a negative implication on the skyline result.

D. Identify local Skyline

This component aims to capture the local skylines of each built cluster. There are many advantages to obtaining local skyline of each cluster before obtaining the final skyline. First, it eliminates the processing of numerous dominated data items before reaching the next stage which to some extent reduces the amount of time taken. Second, it makes sure the transitivity property of skyline technique always hold since all data items in one cluster belong to the same namespace. It will be in all the clusters parallelly which will reduce the processing time of the between the data elements.

E. Final Skyline

This component is in our proposed framework for handling skyline queries in an incomplete graph database.it is responsible to determine the final skyline of the incompelete database. The process starts by comparing those local skylines generated from the previous component and retrieving those undominated data items as the final skylines of the entire incomplete graph database. This component ensures that any reported global skyline are the skylines are the skylines over the entire database and no other data items might dominate them.

IV. PRELIMINARY RESULTS

Skyline queries have tremendous benefits on many contemporary database applications that personalize the query results on the given user preferences. Due to its practical use, skyline queries have been adopted in many multi-criteria applications such as decision-making, decision support, recommendation systems, e-commerce, and data mining. The literature is rich with numerous numbers of skyline approaches that process skyline queries in the complete database in which data are present during skyline process. Nevertheless, very few numbers of skyline techniques have been developed aiming at processing skyline queries in a database with incomplete data. The incompleteness of the data adds more challenges to processing skyline queries due to the issue of cyclic dominance and lack of the transitivity property of skyline technique. It is prohibitive to apply skyline techniques tailored for complete data on a database with incomplete data. This is due to the exhaustive unnecessary pairwise comparisons between dimension values, particularly for a database with a large number of dimensions and a high volume of the data. Therefore, an efficient approach using machine learning is proposed for handling skylines queries in the incomplete graph database.

To the further illustrate, consider a general scenario of a tourist is seeking for a hotel near the beach. The network of available hotels is depicted in Figure 2, where each node represents a hotel, with the connections showing possible paths between them.

In the graphical representation, Hotel 1 is denoted as Node 1, Hotel 2 as Node 2, and so forth, up to Hotel 11. Each node contains four values (e.g., 2, 8, -,4 for Hotel 1), representing a hotel's score on key dimensions, such as distance from the beach, scenic view, and price level, rating.

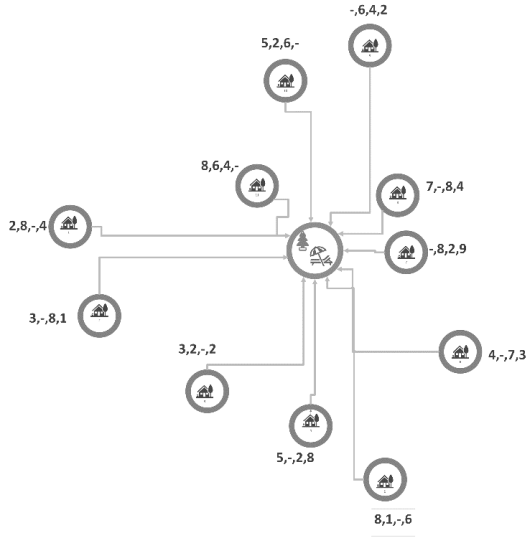

Fig. 2. Hotel database

Table I Provides the corresponding data for each hotel node, illustrating their characteristics and allowing for skyline queries to help the tourist make an informed decision by considering only the non-dominated hotels, i.e., those that offer the best trade-offs in terms of multiple factors.

TABLE I. HOTEL DATABASE

NODES	P1	P 2	P 3	P4
NODE 1	2	8	-	4
NODE 2	3	-	8	1
NODE 3	3	2	-	2
NODE 4	5	-	2	8
NODE 5	8	1	-	6
NODE 6	4	-	7	3
NODE 7	-	8	2	9
NODE 8	7	-	8	4
NODE 9	-	6	4	2
NODE 10	5	2	6	-
NODE 11	8	6	4	-

A. Sorting of Nodes

This phase tries to order them in a manner that they are arranged in the descending order of the domination power of the nodes. Then filter out those data items which overall have a low domination power. The low domination power of data items is unfavorable for attaining the skyline results. Therefore, calling the skyline technique after the removal of these items can help in avoiding a considerable amount of unnecessary pairwise comparisons lower the overall overhead of the skyline process significantly. First of all, it is necessary to sort the elements in every particular list by the values of each dimension in the set.

Node 5 in dimension 1 has 7 scores, Node 2 has 6 scores in dimension 3, Node 11 has 7 scores in dimension 1 and Node 7 has 8 scores in dimension 8 in the Table. II. The elements of constructing lists' data nodes are screened and matched in the round-robin procedure in order to define domination power. This continues until all the nodes have a domination score of the sort.

TABLE II. DOMINANCE SCORE

Node 1	0	5	0	4
Node 2	1	0	6	0
Node 3	1	1	0	0
Node 4	4	0	0	7
Node 5	7	0	0	6
Node 6	3	0	5	3
Node 7	0	5	0	8
Node 8	6	0	6	4
Node 9	0	3	2	1
Node 10	4	1	4	0
Node 11	7	3	2	0

B. Filtering of Nodes

With the help of sorting, the domination score of every node in each dimension was obtained. After the filtration process, all data items having a domination power less than the specified threshold (th) are not processed further. This is because data nodes with low domination

power have no potential to be part of the skyline result, as their domination indicates that these items are good in no more than one dimension. The primary concept of the filtration process relies on exploiting the domination power value to further simplify the skyline process in incomplete databases with a high number of dimensions and large dataset sizes. Therefore, these data items can be safely removed, guaranteeing that the eliminated items do not affect the skyline results.

It is clear that Node 5, Node 11, and Node 8 in Table III, Node 1, Node 7, Node 9, and Node 11 in Table IV, Node 2, Node 8, and Node 6 in Table V, and Node 4 in Table VI Are the nodes with domination scores greater than the threshold value.

TABLE III. SORTED P1

Node 5	7	0	0	6
Node 11	7	3	2	0
Node 8	6	0	6	4
Node 4	4	0	0	7
Node 10	4	1	4	0
Node 6	3	0	5	3
Node 2	1	0	6	0
Node 3	1	1	0	0
Node 1	0	5	0	4
Node 7	0	5	0	8
Node 9	0	3	2	1

TABLE IV. SORTED P2

Node 1	0	5	0	4
Node 7	0	5	0	8
Node 9	0	3	2	1
Node 11	7	3	2	0
Node 3	1	1	0	0
Node 10	4	1	4	0
Node 2	1	0	6	0
Node 4	4	0	0	7
Node 5	7	0	0	6
Node 6	3	0	5	3
Node 8	6	0	6	4

TABLE V. SORTED P3

Node 2	1	0	6	0
Node 8	6	0	6	4
Node 6	3	0	5	3
Node 10	4	1	4	0
Node 9	0	3	2	1
Node 11	7	3	2	0
Node 1	0	5	0	4
Node 3	1	1	0	0
Node 4	4	0	0	7
Node 5	7	0	0	6
Node 7	0	5	0	8

TABLE VI. SORTED P4

Node 7	0	5	0	8
Node 4	4	0	0	7
Node 5	7	0	0	6
Node 1	0	5	0	4
Node 8	6	0	6	4
Node 6	3	0	5	3
Node 9	0	3	2	1
Node 2	1	0	6	0
Node 3	1	1	0	0
Node 10	4	1	4	0
Node 11	7	3	2	0

C. Clustering of Nodes

This phase focuses on further simplify the skyline process in a database containing incomplete data by dividing the data items into separate clusters in Table VII. Utilizing of a machine learning algorithm, clusters of datasets are developed based on their similarities. Specifically, the K-means Clustering Algorithm is used to group similar nodes, including those with empty values. This clustering approach allows for efficient management and processing of the data, ultimately enhancing the accuracy and speed of skyline queries.

TABLE VII. CLUSTERING OF NODES

P1	P2	P3	P4
Node 5	Node 1	Node 2	Node 7
Node 11	Node 7	Node 8	Node 4
Node 8	Node 9	Node 6	Node 5
Node 4	Node 11	Node 10	Node 1
Node 10	Node 10	Node 9	Node 8
Node 6	Node 3	Node 11	Node 6
Node 2	Node 2	Node 1	Node 9
Node 4	Node 3	Node 2	Node 2
Node 3	Node 4	Node 3	Node 2
Node 1	Node 5	Node 4	Node 3
Node 7	Node 6	Node 5	Node 10
Node 9	Node 8	Node 7	Node 11

D. Identify Local Skyline

This phase aims at identifying the local skylines of each of the identified clusters. Also, it eliminates all dominated data items from further processing, which decreases the number of pairwise comparisons required to determine the final skyline of the dataset. It starts with determining the skylines of each group within each of the novel clusters formed. This step provides vital support to the pairwise comparison process as all group data items should have similar bitmaps. This approach assists in eliminating difficulties associated with the loss of the transitivity property and cyclic dominance. After that, the results obtained from the computations made on those sets are compared to identify the local skyline. This process is important as it will ensure that only the non-dominated data items will appear in the next phase.

Hence, the Skyline process will be made easier. According to the running example, Node 7, Node 8, Node 5, and Node 11 are the local skyline in Table VII. When comparing Node 7 with Node 9 in Table VIII Node 7 has more domination power than Node 9, whereas Node 11 has no comparison in Table IX. Furthermore, Node 8, when compared with Node 2 and Node 6 in Table X, exhibits greater domination power. Lastly, Node 5 dominates Node 1 in Table XI.

TABLE VIII. CLUSTER 1

Node 7	-	8	2	9
Node 9	-	6	4	2

TABLE IX. CLUSTER 2

Node 8	7	-	8	4
Node 9	3	-	7	3
Node 6	4	-	7	3

TABLE X. CLUSTER 3

Node 5	8	1	-	6
Node 1	2	8	-	4

TABLE XI. CLUSTER 4

Node 11	8	6	4	-

TABLE XII. LOCAL SKYLINE

Node 5	8	1	-	6
Node 7	-	8	2	9
Node 11	8	6	4	-
Node 8	7	-	8	4

E. Final Skyline

This completes the last phase of the proposed method offering suggestions for skyline query processing on incomplete graph data. The goal of this phase is to select the final skyline of the entire dataset. This is done by checking the skyline clusters we formed against each other for the final skyline. The precise definition of this phase is actually identical to the definition of determining the local skyline of each. Here we have a comparison of Node 7 and Node 11 with Node 8 and Node 5 in the running example. The last two skylines of the study are Node 7 and Node 11 as presented in the Table XII.

TABLE XIII. FINAL SKYLINE

Node 7	-	8	2	9
Node 11	8	6	4	-

V. CONCLUSION

Thus, the proposed approach is capable of overcome the challenges of skyline queries in incomplete graph databases. The approach includes sorting, filtering, clustering, finding local skylines, and having finally been able to capture the final skyline as it were, has been proven to lower avoiding unnecessary comparisons and, therefore, the overall process is further enhanced. By leveraging machine learning algorithms, the clustering phase allows for efficient grouping, and identifying local skylines before determining the final skyline further simplifies the procedure.

This framework enables the accurate retrieval of skyline results even in the presence of incomplete data, ensuring that dominated data items are removed, and only significant ones contribute to the outcome. Through systematic phases, the process ensures efficiency in computational overhead while maintaining the integrity of the skyline result. Ultimately, this method holds promise for improving decision-making and query optimization in various applications involving incomplete and uncertain graph databases.

VI. ACKNOWLEDGEMENTS

The Fundamental Research Grant Scheme (FRGS) with the Reference Code support this research FRGS/1/202/ICT01/ UIAM/02/2 from the Ministry of Higher Education (MOHE) Malaysia.

REFERENCES

[1] S. Börzsönyi, D. Kossmann, and K. Stocker, "The skyline operator," in *Proceedings - International Conference on Data Engineering*, 2001, pp. 421–430. doi: 10.1109/icde.2001.914855.

[2] Z. Yang, X. Yang, and X. Zhou, "Uncertain dynamic skyline queries for uncertain databases," in *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2015*, 2016, pp. 1797–1802. doi: 10.1109/FSKD.2015.7382219.

[3] Mohamed E. Khalefa, *Skyline query Processing for incomplete Data*. IEEE Xplore, 2008.

[4] W. Ren, X. Lian, and K. Ghazinour, "Skyline queries over incomplete data streams," *VLDB Journal*, vol. 28, no. 6, pp. 961–985, Dec. 2019, doi: 10.1007/s00778-019-00577-6.

[5] X. Miao, Y. Gao, G. Chen, B. Zheng, and H. Cui, "Processing incomplete k nearest neighbor search," *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 6, pp. 1349–1363, 2016, doi: 10.1109/TFUZZ.2016.2516562.

[6] D. Amr and N. El-Tazi, "Skyline Query Processing in Graph Databases," Academy and Industry Research Collaboration Center (AIRCC), Jul. 2018, pp. 49–57. doi: 10.5121/csit.2018.81005.

[7] L. Zou, L. Chen, M. Tamer¨ozsu, T. Tamer¨ozsu, and D. Zhao, "Dynamic Skyline Queries in Large Graphs."

[8] K. Abbaci, A. Hadjali, L. Liétard, and D. Rocacher, "A similarity skyline approach for handling graph queries - A preliminary report," in *Proceedings - International Conference on Data Engineering*, 2011, pp. 112–117. doi: 10.1109/ICDEW.2011.5767617.

[9] W. Zheng, L. Zou, X. Lian, L. Hong, and D. Zhao, "Efficient subgraph skyline search over large graphs," in *CIKM 2014 - Proceedings of the 2014 ACM International Conference on Information and Knowledge Management*, Association for Computing Machinery, Nov. 2014, pp. 1529–1538. doi: 10.1145/2661829.2662037.

[10] A. Alwan, H. Ibrahim, N. Udzir, and F. Sidi, "Missing values estimation for skylines in incomplete database," *International Arab Journal of Information Technology*, vol. 15, no. 1, pp. 66–75, 2018.

[11] H. Wang *et al.*, "Efficient Computation of Skyline Queries on Incomplete Dynamic Data," *IEEE Access*, vol. 6, pp. 52741–52753, Sep. 2018, doi: 10.1109/ACCESS.2018.2869819.

[12] D. Ouyang, L. Yuan, F. Zhang, L. Qin, and X. Lin, "Towards efficient path skyline computation in bicriteria networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, 2018, pp. 239–254. doi: 10.1007/978-3-319-91452-7_16.

[13] Y. Gulzar, A. A. Alwan, and S. Turaev, "Optimizing Skyline Query Processing in Incomplete Data," *IEEE Access*, vol. 7, pp. 178121–178138, 2019, doi: 10.1109/ACCESS.2019.2958202.

[14] 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, no. 5, pp. 1636–1657, May 2019, doi: 10.1007/s10489-018-1356-2.

[15] W. Liu, D. Wen, H. Wang, F. Zhang, and X. Wang, "Skyline nearest neighbor search on multi-layer graphs," in *Proceedings - 2019 IEEE 35th International Conference on Data Engineering Workshops, ICDEW 2019*, Institute of Electrical and Electronics Engineers Inc., Apr. 2019, pp. 259–265. doi: 10.1109/ICDEW.2019.000-3.

[16] S. Banerjee, B. Pal, and M. Jenamani, "DySky: Dynamic Skyline Queries on Uncertain Graphs," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Science and Business Media Deutschland GmbH, 2020, pp. 242–254. doi: 10.1007/978-3-030-62005-9_18.

[17] Y. Gulzar *et al.*, "IDSA: An Efficient Algorithm for Skyline Queries Computation on Dynamic and Incomplete Data with Changing States," *IEEE Access*, vol. 9, pp. 57291–57310, 2021, doi: 10.1109/ACCESS.2021.3072775.

[18] C. M. Liu, D. Pak, and A. E. Ortiz Castellanos, "Priority-Based Skyline Query Processing for Incomplete Data," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Jul. 2021, pp. 204–211. doi: 10.1145/3472163.3472272.

[19] X. Miao, Y. Gao, S. Guo, L. Chen, J. Yin, and Q. Li, "Answering Skyline Queries over Incomplete Data with Crowdsourcing," *IEEE Trans Knowl Data Eng*, vol. 33, no. 4, pp. 1360–1374, Apr. 2021, doi: 10.1109/TKDE.2019.2946798.

[20] J. He and X. Han, "Efficient Skyline Computation on Massive Incomplete Data," *Data Sci Eng*, vol. 7, no. 2, pp. 102–119, Jun. 2022, doi: 10.1007/s41019-022-00183-7.

[21] L. Ding, G. Zhang, J. Ma, and M. Li, "An Efficient Index-Based Method for Skyline Path Query over Temporal Graphs with Labels," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Science and Business Media Deutschland GmbH, 2023, pp. 217–233. doi: 10.1007/978-3-031-30675-4_15.

[22] A.-T. Kuo, H. Chen, L. Tang, W.-S. Ku, and X. Qin, "ProbSky: Efficient Computation of Probabilistic Skyline Queries Over Distributed Data," *IEEE Trans Knowl Data Eng*, vol. 35, no. 5, pp. 5173–5186, 2023, doi: 10.1109/TKDE.2022.3151740.

[23] D. Yuan, L. Zhang, S. Li, and G. Sun, "skyline query under multidimensional incomplete data based on classiication tree skyline query under multidimensional incomplete data based on classification tree," 2024, doi: 10.21203/rs.3.rs-3915982/v1.