SKYLINE QUERIES IN LARGE-SCALE AND INCOMPLETE GRAPHS USING MACHINE LEARNING

BY

UBAIR NOOR

A thesis submitted in fulfillment of the requirement for the degree of Master of Computing in Computer Science and Information Technology.

KULLIYAH OF INFORMATION COMMUNICATION TECHNOLOGY

NOVEMBER 2025

ABSTRACT

Skyline queries are widely used in multi-criteria decision-making to identify non-dominated data points that balance conflicting preferences. While skyline computation has been studied extensively in relational and complete databases, limited attention has been given to skyline query processing in large-scale incomplete graph databases. The challenges include the dynamic and evolving nature of graphs with frequent additions and deletions of nodes and the prevalence of missing attribute values that disrupt dominance relationships and reduce query reliability. These challenges become critical in real-world applications, such as recommendation systems, urban planning, fraud detection, and location-based services, where incomplete or sparse data is common. Traditional approaches relying on relational-to-graph transformation and heavy preprocessing suffer from inefficiency, sparsity, and poor scalability when applied to high-dimensional graph data. The proposed study introduces an optimized framework for skyline query processing in incomplete graph databases by integrating machine learning techniques, including clustering-based optimization with the K-Means algorithm, dynamic data pruning, and adaptive indexing. The framework reduces computational overhead, handles missing values more effectively, and ensures accurate skyline retrieval under an incomplete graph database. Experimental evaluation on synthetic graph datasets, designed to mimic real-world incompleteness, demonstrates the framework's effectiveness. The proposed method achieves a reduction in query processing time of 30-50 % and a dataset size reduction of up to 44.44 % compared to traditional baseline algorithms. Cluster quality was validated using intrinsic metrics such as the Silhouette Score, ensuring the robustness of the groupings. The proposed solution significantly advances skyline query processing for complex, incomplete graph structures, contributing to more efficient and reliable decision-support systems, recommendation engines, and location-based services.

ملخص البحث

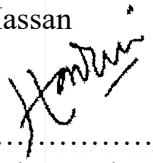
تُستخدم استعلامات الأفق (Skyline) على نطاق واسع في اتخاذ القرارات متعددة المعايير لتحديد النقاط البيانية غير المهيمنة التي توازن بين التفضيلات المتضاربة. في حين تم دراسة حساب الأفق بشكل موسع في قواعد البيانات العلائقية والكاملة، فإن الاهتمام محدود بمعالجة استعلامات الأفق في قواعد بيانات رسومية غير كاملة واسعة النطاق. تشمل التحديات الطبيعة الديناميكية والمتطورة للرسوم البيانية مع إضافة وحذف العقد بشكل متكرر، وانتشار القيم المفقودة في السمات التي تؤثر على علاقات الهيمنة وتقلل من موثوقية الاستعلامات. تصبح هذه التحديات حاسمة في التطبيقات الواقعية مثل أنظمة التوصية، التخطيط الحضري، اكتشاف الاحتيال، والخدمات القائمة على الموقع، حيث يكون من الشائع وجود بيانات غير مكتملة أو متفرقة. تعاني الأساليب التقليدية التي تعتمد على التحويل من العلائقية إلى الرسوم البيانية والمعالجة المبدئية الثقيلة من ضعف الكفاءة والندرة وقلة القابلية للتوسع عند تطبيقها على بيانات بيانية عالية الأبعاد. يقدم هذا البحث إطار عمل محسن لمعالجة استعلامات الأفق في قواعد البيانات البيانية غير المكتملة من خلال دمج تقنيات التعلم الآلي، بما في ذلك التحسين القائم على التجميع باستخدام الخوارزمية التصنيفية K-Mean ، والتقليص الديناميكي للبيانات، والفهرسة التكيفية. يعمل الإطار على تقليل العبء الحسابي، والتعامل مع القيم المفقودة بشكل أكثر فعالية، وضمان استرجاع الأفق بدقة في قواعد البيانات البيانية غير المكتملة. أظهرت التقييمات التجريبية على مجموعات بيانات رسومية صناعية، تم تصميمها لتعكس عدم الاكتمال في العالم

الحقيقي، فعالية الإطار. وقد حققت الطريقة المقترحة تقليصًا في وقت معالجة الاستعلام بنسبة 30–50 % وتقليصًا في حجم مجموعة البيانات يصل إلى 44.44 % مقارنة بالخوارزميات التقليدية. وتم التحقق من جودة التجميع باستخدام مقاييس داخلية مثل درجة الظليّة، مما يضمن قوة التجميعات. يساهم الحل المقترح بشكل كبير في تحسين معالجة استعلامات الأفق للهيكليات البيانية المعقدة وغير المكتملة، مما يساهم في أنظمة دعم القرار الأكثر كفاءة وموثوقية، ومحركات التوصية، والخدمات القائمة على الموقع.

APPROVAL PAGE

I certify that I have supervised and read this study and that, in my opinion, it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Master of Computing in Computer Science and Information Technology.

…….
Dr. Raini Binti Hassan
Supervisor

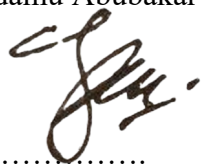


…….
Ts. Dr. Dini Oktarina Dwi Handayani
Co-Supervisor

I certify that I have read this study and that, in my opinion, it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Master of Computing in Computer Science and Information Technology.

…….
Assoc. Prof. Tp. Dr. Hamwira Sakti Bin Yaacob
Internal Examiner

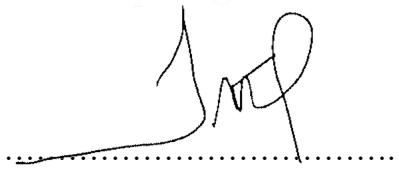
…….
Assoc. Prof. Dr. Adamu Abubakar Ibrahim
Internal Examiner



…….
Prof. Dr. Nor Liyana Bt. Mohd Shuib

External Examiner

This thesis was submitted to the Department of Computer Science and is accepted as a fulfillment of the requirement for the degree of Master of Computing in Computer Science and Information Technology.



……

Dr. Azlin Binti Nordin

Head, Department of Computer Science

This thesis was submitted to the Kulliyyah of Information and Communication Technology and is accepted as a fulfillment of the requirement for the degree of Master of Computing in Computer Science and Information Technology.

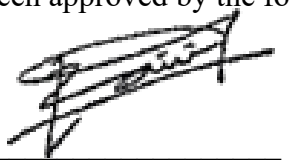
……

Prof. Emeritus Dato' Ts. Dr. Tengku Mohd Bin Tengku Sembok

Dean, Kulliyyah of Information and Communication Technology

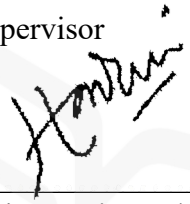
APPROVAL PAGE

The dissertation of Ubair Noor has been approved by the following:



Dr. Raini Binti Hassan

Supervisor



Dr. Dini Oktarina Dwi Handayani

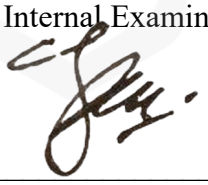
Co-supervisor

Assoc. Prof. Tp. Dr. Hamwira Sakti Bin Yaacob

Internal Examiner

Assoc. Prof. Dr. Adamu Abubakar Ibrahim

Internal Examiner



Prof. Dr. Nor Liyana Bt. Mohd Shuib

External Examiner

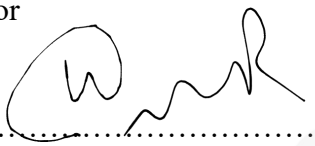
Dr. Rizal Bin Mohd. Nor

Chairman

DECLARATION

I hereby declare that this dissertation is the result of my own investigations, except where otherwise stated. I also declare that it has not been previously or concurrently submitted as a whole for any other degrees at IIUM or other institutions.

Ubair Noor

Signature……  …… Date……11-11-2025……

INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA

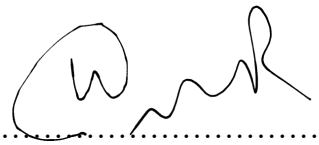
DECLARATION OF COPYRIGHT AND AFFIRMATION OF FAIR USE OF UNPUBLISHED RESEARCH

SKYLINE QUERIES IN LARGE-SCALE AND INCOMPLETE GRAPH USING MACHINE LEARNING

I declare that the copyright holder of this thesis/dissertation is Name of the Student.

By signing this form, I acknowledged that I have read and understand the IIUM Intellectual Property Right and Commercialization policy.

Affirmed by Ubair Noor



…… ..11-11-2025…..

Signature Date

This thesis is dedicated to my parents, my siblings, teachers for laying the foundation of what I turned out to be in life.

ACKNOWLEDGEMENTS

In the name of Allah, the Most Gracious and the Most Merciful**, peace** be upon the Holy Prophet Muhammad (**S.A.W.),** who has given light to mankind, and His family. Alhamdulillah, all praises to Allah for giving me strength, guidance, knowledge, ability, and opportunity to undertake this research study.

I am most indebted to the supervisor, Asst. Prof. Dr. Raini Binti Hassan, whose enduring disposition, kindness, promptitude, thoroughness, and friendship have facilitated the successful completion of my work. I put on record and appreciate her detailed comments, useful suggestions, and inspiring queries, which have considerably improved this thesis. Her brilliant grasp of the aim and content of this work led to her insightful comments, suggestions, and queries, which helped me a great deal. Despite her commitments, she took time to listen and attend to me whenever requested. The moral support she extended to me is, without a doubt, a boost that helped in building and writing this research work. I am also grateful to my co-supervisor, Asst. Prof. Dr. Dini Oktarina Dwi Handayani, whose support and cooperation contributed to the outcome of this work.

Lastly, my gratitude goes to my mother, who never forgot to mention my name in her prayer; my father, who always offered his encouragement through phone calls; and my brothers, who keep supporting me in their ways.

Once again, we glorify Allah for His endless mercy on us, one of which is enabling us to successfully round off the efforts of writing this thesis. Alhamdulillah."

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

Di	The Dominance Count of Nodes.
$\sum$	Submission of Nodes.
r_{ik}	It refers to the binary indicator of whether a data point belongs to a cluster.
μ_k	It refers to the centroid of a cluster.
x_i	It refers to the binary indicator representing whether a data point is assigned to a cluster.
$\bar{X}$	Sample Mean.
δ	Standard Deviation.
√n	Square root.

CHAPTER ONE

INTRODUCTION

1.1 INTRODUCTION

The main idea of the skyline queries is to identify the nodes that are not dominated by any other node based on certain criteria or preferences in a graph database (Börzsönyi et al., 2001). Skyline queries are often used in numerous modern database applications for decision-making processes, multi-criteria decision support systems, web-based businesses, crowd-sourcing databases, road networks and e-commerce (Gulzar, Alwan, Ouyang 2018 & Turaev, 2019). A critical issue with data incompleteness in skyline queries is the loss of the transitivity property of the skyline technique. As a result, the dominance relationships between data items considered to be cyclic. Also, it proves to be a practical solution, but skyline queries have shown a potential effectiveness in many real-world applications. However, processing them in graph databases remains a significant challenge in database management. Certain innovative techniques are required for effective and precise skyline computation because of the incompleteness, which is characterized by rapid changes and the existence of missing values within tuples or nodes. This research concentrates on processing skyline queries over large-scale knowledge graph databases. The aim is to reduce the search space, minimize computation costs, and decrease the time complexity of identifying skylines in the graph. In addition, this research addresses the problem of dynamic skyline queries in uncertain graphs. They emphasize identifying the superior data vertices concerning the query vertices based on two distant measures (majority distance and expected distance) that fit on the uncertain graphs (Yang et al., 2016). Other researchers have investigated the problem of processing continuous subgraph multi-queries over graph streams.

1.2 PROBLEM STATEMENT

Based on existing studies the previous research has not adequately addressed 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. In practical scenarios, graphs exhibit high dynamism and are constantly undergoing additions or removals of nodes. In real-world applications, such as recommendation systems, urban planning, and fraud detection, skyline queries are crucial for decision-making processes. In recommendation systems, for instance, skyline queries help in selecting the best products based on multiple user preferences, even when some data is missing. Similarly, in urban planning, skyline queries are used to identify optimal locations for developments purposes by evaluating conflicting criteria such as cost and proximity to amenities. Additionally, in road networks, skyline queries can be applied to optimize the route selection by balancing multiple criteria such as, shortest distance, least traffic, and fuel efficiency, even when road conditions or traffic data might be incomplete. With the rise of mobile devices like smartphones and tablets, location-based services (LBSs) have become increasingly important for users to discover points of interest (POIs) anytime, anywhere. Cloud computing has enabled LBSs to efficiently outsource POI datasets to third-party providers, improving scalability and reducing costs. Location-based skyline queries (LBSQs) are useful in identifying POIs that are not dominated by others based on user location and preferences such as price or proximity.

However, when graphs are represented as relational tables, they often exhibit sparsity across numerous dimensions. Graphs inherently accommodate a wealth of attributes, necessitating the development of efficient indexing methods to improve computational efficiency of identifying skyline entities (Amr & El-Tazi, 2018; Miao, et al., 2016; , M.E., 2008; Ren et al., 2019a). Despite the potential presence of various numeric attributes, indications for rare dominant relationships within knowledge graphs; limited efforts have been allocated towards resolving the challenges associated with processing skyline queries within graph databases. The example in Figure 1.1 illustrates how a user

needs to retrieve the hotels that are near the beach and have affordable single night rates from this database using the following information: Hotel Name, Price Per Night, Distance to the Beach, and Star Rating. Since hotels near the beach will generally be more expensive, it is clear that the two preferences may conflict. The first dimension of this question is the cost per night, and the second dimension is the distance to the beach. As a result, when a skyline query is applied to the dataset, hotels that excel in both preferences will be returned. Skyline is the name of these lodgings. The hotels "A," "C," and "D" are the skyline results in the example because none of them is superior to the others in every way. It is now up to the user to choose whether to look through the entire list or simply select a hotel from the list of skyline queries. With larger datasets that include thousands of hotels, the value of skyline queries becomes more apparent. In this study, we offer a method for applying skyline queries to a graph model and node relationships that make it easy to retrieve any linkages.

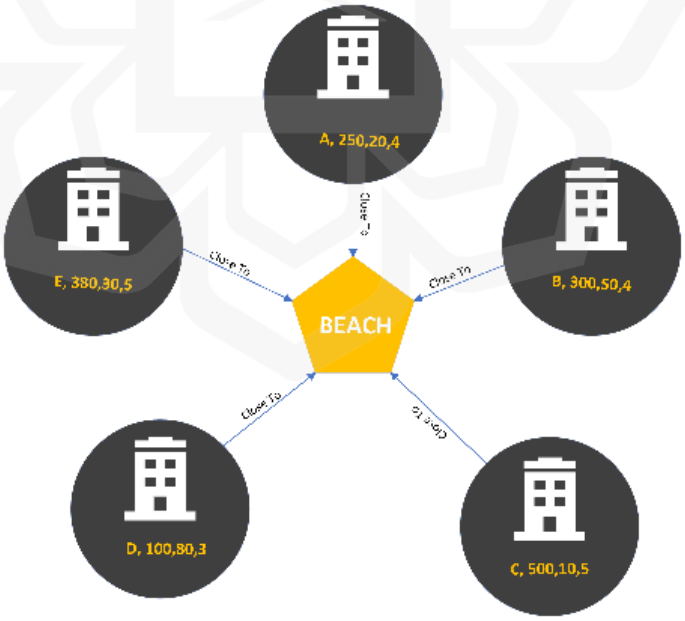


Figure 1.1 Hotel Database

1.2.1 Machine Learning Relevance to Skyline Query

Skyline queries are crucial for identifying the best data points based on multiple criteria but processing them efficiently across large graph databases can be challenging. Integrating machine learning into skyline query processing offers a promising way to overcome these challenges and improve efficiency, accuracy of handling complex and large-scale data. While traditional techniques such as divide-and-conquer (Amr & El-Tazi, 2018) (Zheng et al., 2014) and multi-measure similarity searches offer some improvements, they often rely on heavy preprocessing, lack flexibility for dynamic environments, and may be inefficient with missing or noisy data. Machine learning (ML) can help solve these problems by learning patterns from data, improving how missing values are handled, and dynamically adjusting to changes in the database. ML-based methods, such as clustering, can reduce computational costs, making skyline query processing faster and more accurate, and adapt to real-time environments. Integrating machine learning into skyline query processing offers a promising way to overcome these challenges and improves the efficiency and accuracy of handling complex and large-scale data.

1.3 RESEARCH QUESTIONS

i. What knowledge of large graphs completes and Incomplete graphs of skyline queries?
ii. How can an efficient data pruning technique be designed and implemented to operate on an incomplete graph database prior to skyline query processing with a focus on minimizing computation cost while maintaining query accuracy?

1.4 OBJECTIVE OF THE RESEARCH

i. To examine and analyze knowledge of large graphs complete and incomplete graphs of skyline queries.
ii. To design and develop an approach efficient data pruning technique that best works over incomplete graph database using machine learning model.
iii. To evaluate the proposed approach.

1.5 SIGNIFICANCE OF THE RESEARCH

Addressing these challenges is important because it helps make skyline queries in graph databases faster and more accurate. This means we can analyze data better and make smarter decisions. Exploring ways to address these problems is key to improve how we use data and make choices. The integration of machine learning techniques to handle dynamic and incomplete datasets ensures faster and more precise results, which are crucial for real-world applications in recommendation systems, urban planning, and other fields that rely on large-scale complex data. This research will contribute to more effective use of data, enables a smarter and more informed choices.

1.6 RESEARCH SCOPE

i. The research work will be on the graph data model.
ii. The type of queries that are considered in this preference queries that are widely used to represent complex and interconnected data, such as social networks, biological networks, semantic web, and knowledge graphs.

iii. The research concentrates on synthetic databases to simulate real-world scenarios and analyze the impact of incomplete data on skyline queries.
iv. The database is considered to be an incomplete graph database.

1.7 SUMMARY

This chapter introduces skyline queries, which are used to retrieve non-dominated tuples in databases based on multiple criteria. It highlights challenges with incomplete data in graph databases, which affect skyline query processing, especially in dynamic environments like recommendation systems, urban planning, and road networks. The chapter discusses the need for efficient indexing methods and proposes the use of machine learning (ML) to handle the missing data nodes, improve query processing, and reduce computational costs. The research aims to design a data pruning technique for incomplete graph databases and evaluate the efficiency of this approach in improving skyline query performance.

CHAPTER TWO
LITERATURE REVIEW

2.1 INTRODUCTION

This chapter provides an overview of skyline queries in database systems and graph databases focusing on processing skyline queries in incomplete graph databases. It also discusses the principles behind data incompleteness, methods to handle missing data, and previous approaches to skyline queries in complete, incomplete, and graph databases. We also go over the essential principles and notions behind the incompleteness of the data in the graph database. Also, we will discuss in detail on several approaches to dealing with missing data in the database. This chapter also presents a thorough analysis of earlier methods for handling skyline queries in complete, incomplete, and graph databases.

2.2 AN OVERVIEW OF QUERY PROCESSING IN INCOMPLETE DATABASE

The development of an appropriate method to handle incomplete databases is necessary because numerous real-world databases contain missing attribute values which need preprocessing to execute operations effectively. Three distinct procedures exist for processing queries within databases that contain incomplete data. The first approach includes skyline queries in which only relevant information items are retrieved, and no values are missing. This method is not enough since the final query answer omits a large number of important data items with missing values. When there are many missing data points, ignoring them causes the output size to be significantly reduced and increase the possibility of incorrect and non-significant results. This approach retrieves complete data only.

The second method considered all the data items in the database and extracted both the related complete and incomplete data. One major disadvantage of this strategy is that the user might find some retrieved incomplete data pieces meaningless. Thus, the user does not gain anything from the large increase in the size of the query answer. This approach is referred to as “retrieve all”. Similarly, the third approach eliminates the problems in the first and second approaches by considering the relevant complete data items and the relevant incomplete data items as well. First, the relevant complete data items are retrieved. Next, we retrieve the relevant incomplete data items with estimated values. This approach consists of three phases in processing the incomplete data items. The first phase retrieves all data items with missing values (relevant and irrelevant). Then, the second phase imputes the missing values for every data item. Finally, depending on the estimated values, the third phase determines which data items are further considered in the query answer and which are to be neglected. This approach is referred to as "retrieve all relevant”.

2.2.1 Overview of the Context and Challenges of Skyline Query Processing

Managing incomplete data in databases is a critical challenge in modern data-driven environments as missing attribute values can hinder meaningful analysis, reduce accuracy, and lead to biased results. This issue arises due to factors like data corruption, entry errors, or incomplete collection. Developing robust techniques to handle such data is vital to ensure reliable decision-making and effective query processing. Approaches to managing incomplete data vary; the “retrieve complete data only method excludes all entries with missing values, ensuring reliable results but significantly reducing output size and missing potentially important insights. Conversely, the “retrieve all” method includes all data items, both complete and incomplete, but overwhelms users with irrelevant information, making it inefficient for practical use. The “retrieve all relevant” approach strikes a balance by retrieving complete data alongside selectively processed incomplete data. Through a three-phase process approach such as, retrieving, imputing missing values, and filtering relevant items, it ensures accurate, comprehensive, and meaningful results. This method is

particularly important in handling large and complex datasets, where effective management of incomplete data is essential for optimizing performance and extracting valuable insights.

2.2.2 Statistical Method

Prediction approaches designed by researchers typically depend on statistical methods. The primary aim of this method is to preserve the overall distribution of the data while avoiding bias in that distribution. The predicted values may benefit the user at the database level but not at the data item level. Often, users are more concerned with the individual value of the attribute rather than the whole data. Thus, these statistical methods may not be appropriate for preference queries.

2.2.2.1 Single Imputation

During this method all missing values within an attribute receive identical replacement. The imputation methods using single replacements include mean, median, mode, maximum value in the attribute range, minimum value in the attribute range, k-nearest neighbor (kNN) and expectation maximization (EM).

2.2.2.2 Multiple Imputation

The approach generates multiple estimated values for filling gaps in data records. The accuracy and productivity level of this approach exceed SI, yet it requires more

computational resources (Rubin, 1996). Additionally, the method depends on an appropriate mechanism to accommodate missing data uncertainties.

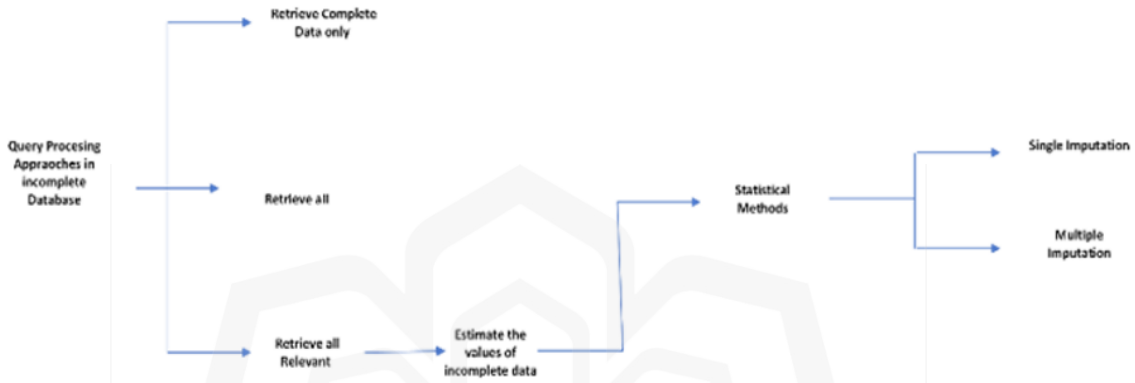


Figure 2.1 Query Processing Approaches in Incomplete Database

2.2.3 Machine Learning

The machine learning techniques have emerged as powerful tools for data analysis and prediction. Unlike traditional statistical methods, which prioritize preserving the distribution of the entire dataset to avoid bias, machine learning algorithms focuses on learning patterns and relationships within the data to make accurate predictions (M.M.F. Fahima et al., 2024). While statistical methods excel at database-level predictions, they may fall short when it comes to addressing user preferences at the individual data item level. Users often prioritize understanding the specific attributes of data points rather than the overall dataset distribution. Consequently, traditional statistical methods may prove inadequate for handling preference queries in such scenarios. The following algorithms can be used to handle data in graph databases.

2.2.3.1 K-Means Clustering Algorithm

K-medoid, also known as Partition Around Medoids (PAM), stands out as a variant algorithm in the realm of clustering. Its essence lies in designating certain data points within clusters as centroid, essentially acting as their central representatives. These centroids are chosen so that the overall dissimilarity with other points within the cluster is minimized, making them pivotal in characterizing the cluster.

The algorithm operates in a two-fold manner, typically delineated into the build and swap phases. Initially, in the build phase, the primary medoid is selected based on its minimal dissimilarity with the entire dataset, essentially pinpointing the most central point. Subsequently, the swap phase comes into play, where each data point is evaluated against the current set of ‘k’ centroid. Here, a new centroid is formed by swapping data points between the existing medoid and a new candidate non-medoid, effectively optimizing the medoid selection process.

In essence, K-Means seeks to establish robust clusters by strategically choosing representative points and iteratively refining them through a systematic exchange mechanism, ultimately enhancing the overall coherence and effectiveness of the clustering process.

2.2.3.2 K-Medoids Clustering Algorithm

K-medoid's K-Medoids aims to create strong clusters by carefully choosing representative data points (medoids) and then improving them over time through a systematic exchange mechanism. This makes the clustering process more coherent and effective overall. This iterative refinement allows for the identification of clusters that best represent the underlying structure of the data. As a result, K-medoids not only improves accuracy but also increases resilience to noise and outliers, making it a valuable method in

various data analysis applications. These characteristics make K-medoids particularly useful in fields such as bioinformatics, image processing, and market segmentation, where data can often be messy and inconsistent. By focusing on actual data points as representatives, it ensures that the clusters formed are more interpretable and meaningful in real-world contexts.

2.2.3.3 Balanced Iterative Reducing and Clustering using Hierarchies

The BIRCH algorithm (Balanced Iterative Reducing and Clustering using Hierarchies). It's a hierarchical clustering algorithm designed to cluster large datasets efficiently by iteratively reducing the dataset size while preserving the hierarchical structure of the clusters.

BIRCH operates by recursively partitioning the dataset into smaller subsets until each subset satisfies certain conditions for clustering, such as a minimum number of data points or a maximum allowable diameter. It then merges these subsets into larger clusters in a balanced manner to maintain the overall hierarchical structure.

2.3 SKYLINE QUERIES

The database contains two fundamental query categories, including traditional queries as well as preference-based queries. Traditional queries focus on retrieving results strictly according to the specified predicates in the query, disregarding any user preferences. Consequently, if no data item meets the established criteria, the query may yield an empty

result. In contrast, preference queries aim to provide results that align with the user's preferences.

2.3.1 Previous Empirical Studies on Skyline Queries on Incomplete Database

The previous approaches designed to process skyline queries in incomplete databases are examined in this sub-section. This section delivers a summary that details the previous approaches for incomplete database skyline processing. The work contributed by (Khalefa, 2008), the bucket algorithm which divides the dataset into separate buckets based on the presence of missing values, is an intuitive approach that organizes data and allows for localized skyline computation. The skyline algorithm further refines this by optimizing the number of pairwise comparisons required, making it more efficient compared to traditional methods. This approach can help reduce computation time by eliminating unnecessary comparisons. Despite these strengths, skylines have a significant drawback it still requires many pairwise comparisons to identify skylines within each node. This disadvantage makes it less efficient as the dataset size increases, particularly when dealing with large-scale databases. Additionally, while the Bucket method efficiently organizes data, it does not address the complexity of relationships between dimensions and may lead to incomplete or inaccurate skyline results when missing data is not randomly distributed.

For incomplete databases, (Yasuhiko Morimoto, 2012) the RBSSQ technique replaces missing values with values outside the domain, either larger or smaller than domain values, which can improve the robustness of skyline computation in incomplete databases. The touching oracle function used in the skyline sets computation step ensures secure handling of missing data by not revealing specific values, enhancing the security of the query process. One of the main issues with RBSSQ is its reliance on replacement strategies that may not always be suitable for all types of missing data. Replacing missing values with extreme values can distort the dominance relationships between data points,

especially in datasets with complex interdependencies among dimensions. Additionally, this method application is primarily suited for centralized databases, which may limit its scalability and flexibility in distributed or large-scale systems.

The sort-based Incomplete Data Skyline (SIDS) approach was presented by (Bharuka & Kumar, 2008) to evaluate the skyline over incomplete data. For each dimension, the algorithm receives pre-sorted data in descending order as input and a comparison is made between the initial data item in the first dimension and all other data items in the same dimension. In this way, the subsequent data item from the subsequent dimension is chosen for processing. This method selects every data item on a round-robin basis for processing. This algorithm aims to reduce the number of comparisons, which in turn reduces execution time, and prune the non-skyline points as early as feasible. All the datasets points are initially regarded as candidate skyline points, and the points that repeatedly dominate are then eliminated from the candidate set. Skyline points are those that have not yet been removed and are processed as many times as values are present in all dimensions. The major limitation of SIDS lies in its inefficiency as the dataset size increases. Since there is no concurrent execution or parallelization, the computational cost grows exponentially with the number of dimensions or data points. This algorithm is better suited for small, centralized datasets but may not scale well for large or dynamic data systems.

Incomplete data frequent skyline, or IDFS, is an approach that has been proposed to address the problem of processing skyline queries in incomplete data in the work presented by (Bharuka & Kumar, 2012). IDFS uses the top-k frequent skyline technique, which was developed with the goal of regulating the skyline findings size. IDFS is built on the idea of using the fractional skyline frequency of each data item to derive superior skylines using the top-k concept. They have located a collection of excellent skylines for a database that contains missing values. Using both synthetic and actual datasets, certain experiment results have been published that demonstrate the effectiveness of IDFS. While IDFS is effective for small to medium datasets, its performance deteriorates as the dataset

grows. The frequent skyline concept becomes less effective when the examined space is enormous, as the algorithm struggles to efficiently process large datasets with missing values. Additionally, IDFS assumes that missing data follows certain patterns, which may not always hold true, potentially leading to inaccurate skyline results in some cases.

For the purpose of locating skyline queries across incomplete data in centralized databases, (Gao., 2014; Miao, 2013) presented a set of techniques called the baseline algorithm, virtual point-based algorithm (VP), and k-iSkyband algorithm (kISB). The idea behind these algorithms was the same as that of the baseline technique, which first divides the data items according to their bitmap representation into several buckets. In the second stage, some data items known as candidate keybands, or CkSBs, are chosen because they are non-dominated data items within each bucket. The set of data items used in the last stage of the baseline algorithm is called GkSB (global k-skyband), and it is derived from cross-domination tests between CkBS data items from various buckets.

Similarly, a framework called COBO was proposed by (Zhang, 2016) which includes ISSA algorithm to enable the skyline query processing to be finished in two steps i.e., trimming the compared list and cutting down on predicted comparison times. The Bucket algorithm is the same method used in the pruning of comparison lists to remove unneeded data items from each partition. Prior to comparing data items that are crossed, the reduced expected comparison times calculate the total sum of all non-dominated dimensions from each bucket and arrange the data items in an increasing order (lower values are regarded as optimal). This method reduces both the processing time and the number of comparisons. While kISB and VP-based algorithms optimize query processing, the storage and management of virtual points can lead to high memory usage, especially for large datasets with numerous missing values. Furthermore, the efficiency gains from virtual points are dependent on the dataset's characteristics, and the algorithm might not be as effective in environments with complex data relationships or dynamic updates.

Two techniques were proposed by (Lee et al., 2016a) for incomplete data i.e., BUCKET which is a baseline algorithm, and SOBA (sorting-based bucket skyline algorithm). In SOBA, two optimum methods have been used: bucket level orders and point lever orders. These techniques have been used to limit the size of the skyline set thus eliminates the domination checks between data items and raise the overall efficiency of skyline processing over incomplete data. The BUCKET algorithm employs the I-skyline methodology (Khalefa , 2008), although this method yields accurate results, it does an excessive number of pointless pairwise comparisons. SOBA plays a role in resolving this problem which retrieves local skylines from each partition using the BUCKET technique. Before determining the final skyline set, SOBA employs two optimization techniques. The first technique is called bucket-level optimization, which involves sorting the buckets in ascending order to compare the data items across buckets with smaller common subspaces and prevent pointless domination tests. The second sort of optimization is called point-level optimization, which involves adding up all the possible values for each data item in each bucket and then rearranging the items in decreasing order. By moving the data items around, SOBA can more successfully get rid of non-dominant data as soon as feasible. This work has been executed on a centralized database, just as others. The SOBA algorithm still faces the problem of excessive pairwise comparisons, particularly in large-scale databases. While the bucket-level and point-level optimizations, the algorithm can be computationally expensive when the dataset is large and high-dimensional. Additionally, while it minimizes the search space, it may still struggle with datasets that have high variability in missing data patterns.

Also, the work presented in (Gulzar, 2016) proposed a framework for managing skyline queries in centralized partial data. The four compounds that make up this approach are the incomplete skyline identifier, k-dom skyline generator, local skyline identifier, and data clustering builder. The database is analyzed in the clustering builder to determine how many dimensions have missing values. Then, based on the common dimension or dimensions with missing values in each data item, the database is partitioned into various clusters. It is easy to compare data items that are clustered together in the same cluster

without compromising the transitivity property. The clusters are grouped based on the largest value of any dimension within each cluster and from these groups, local skylines and identifier compounds are derived. Deriving a set of virtual skylines from each cluster's local skylines is the primary purpose of the *k*-dom generator component. One global *k*-dom skyline is then created by combining the derived k-dom skylines. To stop processing the dominant data items, the global *k*-dom skyline is added at the top of each cluster. Lastly, the framework further compares the non-dominated local skyline points in the incomplete skyline identifier compound to ensure that the final skyline represents the entire database. This framework's primary goal is to minimize pairwise comparisons and the search space.

PISkyline, a technique for processing probabilistic skylines in partial data was proposed by (Zhang., 2017) which yields data items with a high likelihood of being included in a real-world skyline set. In order to accomplish this, they employed two optimization strategies that accelerated the computation of the probabilistic skyline and two filtration procedures to remove dominated data items early on. The probabilistic approach, while useful, introduces complexity in the query processing. It relies on assumptions about the missing data distribution, which might not be held in every scenario. The added computational cost of handling probabilities may also be prohibitive for real-time or large-scale applications.

In order to solve the problem of processing skyline queries on incomplete databases, (Zhang., 2017) developed a solution called SPQ (Skyline Preference Query), which employs the same method of sorting data items according to each dimension in decreasing order as SIDS (Bharuka & Kumar, 2012) .Nevertheless, SPQ separates the original dataset into two discrete subsets according to the user-selected priority of dimensions. Compared to the second subset, the first subset is highly prioritized. The SIDS approach is used to compute the local skylines of the first subset. On the basis of bitmap representation of the data items, the second subset is further divided into smaller subsets in order to compute the local skylines of that subset. The final skylines are obtained by comparing the local skylines of the first subset found in SSRS with those of the second subset found in RSRS. While the

SIDS method has been somewhat enhanced by SQP, however the SPQ creates numerous preprocessed lists in a manner similar to SIDS. Moreover, the data structure employed in (Chan., 2006) serves as the foundation for the creation of both SIDS and SQP algorithms with the limitation of effectively managing missing data. SPQ suffers from the same limitations as the SIDS algorithm, as it uses preprocessed lists to determine the skyline, which can lead to inefficiencies in large or dynamically changing datasets. Additionally, the algorithm relies heavily on sorting, which may not be ideal for all types of incomplete data.

For the purpose of processing skyline queries in probabilistic incomplete databases, (Zeng, 2018) suggested the EP method. Certain dimension values in probabilistic databases are present in a range. Using bitmap representation, EP uses the BUCKET technique (Y. Wang., 2017) to partition the dataset into distinct buckets. The data elements in those buckets are then subjected to monotonic sorting. Parallel processing of each bucket is used to identify local skylines. Finally, local skylines of each bucket are compared to determine final skylines. According to (Zeng., 2018) the EP is the original algorithm for probabilistic incomplete databases. However, EP still faces challenges related to the storage of virtual points and the additional memory requirements associated with bucket partitioning. Its reliance on parallel processing also introduces complexity in distributed systems, where maintaining consistency across nodes can become a bottleneck.

Table 2.1 Summary of Previous Approaches of Skyline Techniques in Incomplete Database

Approach	Technique	Data Distribution	Database Type
Iskyline (Mohamed, 2008)	Clustering	Correlated, anti-correlated, and independent	Synthetic and Real
RBSSQ (Yasuhiko Morimoto, 2012)	Clustering	Independent and uniform	Synthetic
SIDS (Bharuka & Kumar, 2012)	Sorting	Independent and anti-correlated	Synthetic and Real
IDFS (Bharuka & Kumar, 2012)	Sorting	Independent and anti-correlated	Synthetic and Real
BaseLine (Gao., 2014; Miao et al., 2013)	Clustering	Independent and correlated	Real and Synthetic
IncoSkyline (Gulzar., 2016)	Clustering	Independent and anti-correlated	Real and Synthetic
ISSA (Zhang., 2016)	Sorting	Independent and correlated	Synthetic and Real
PISkyline (Zhang., 2017)	Sorting	Independent and anti-correlated	Real and Synthetic
SPQ (Y. Wang., 2017)	Sorting	Independent	Real and Synthetic
Bucket (Zeng., 2018)	Sorting	Independent and anti-correlated	Real and Synthetic

2.3.2 CRITICAL DISCUSSION

The problem being studied in the previous research discussed revolves around skyline query processing in the presence of incomplete data. Skyline queries are generally used to locate non-dominated points in a dataset, but the problem becomes more intense when data is incomplete, noisy or missing. Various algorithms have been proposed to deal with missing data, among which the Bucket algorithm (Mohamed E. Khalefa, 2008), (Yasuhiko Morimoto, n.d.) and others like SIDS, IDFS, and SPQ, which attempt to optimize skyline computation in such conditions.

Existing solutions, like the bucket algorithm (Mohamed E. Khalefa, 2008) gives an knowledge to organizing data by dividing it into buckets based on missing values, but they struggle with excessive pairwise comparisons as the dataset grows. It further improves the skyline by optimizing the pairwise comparisons, yet inefficient as the dataset increases. Similarly, the RBSSQ technique which replaces missing values, improves robust, but alters dominance's and IDFS try to decrease the number of comparisons, but both grow in scalability problems as that dataset size or dimension increases. The SPQ approach tries to attempt to overcome the weakness of SIDS by prioritizing dimensions, it still is not optimal due to inefficiencies in SIDS based on the preprocessed lists and sorting. EP, probabilistic incomplete databases method for (Zeng et al., 2018) introduces parallel processing and bucket partitioning but faces challenges in memory usage and distributed system consistency.

One important research area is the efficiency of these methods in facing massive, high dimensionality and dynamic datasets. There are number of other methods such as Bucket and Skyline methods, depends on pairwise comparisons or sorting, which become highly computationally expensive as dataset scale increases. Additionally, several methods make simplifying assumptions about the missing data, for example, assuming random

distribution or fixed patterns, which is not necessarily always the case in real-world problems. Moreover, these methods often face scalability challenges in distributed systems or dynamic environments where data is constantly evolving.

The future research should emphasize more involving scalable and adaptable algorithms that could operate efficiently in dealing of huge, changing and high dimension dataset with incomplete data. Parallel processing techniques addressing computational and memory issues more effectively are therefore essential. In addition, new methods should minimize the need for presorted data and excessive pairwise comparisons. Instead, they could explore new approaches like probabilistic methods pore machine learning, to better handle missing values in data. Further research might as well focus on techniques for more sophisticated control of distributed systems and on support for more efficient dynamic data updates. This way skyline queries would be processed efficiently in dynamic environments.

2.3.3 Previous Empirical Studies on Skyline Queries in Graph Databases

The work introduced by (Zou et al., 2010) about dynamic skyline queries in large graphs describes the shared shortest path (SSP) pruning technique that delivers improved query efficiency through reduced false positives and diminished unnecessary computations in large datasets. This method efficiently operates with large datasets, which allows its implementation in social network and peer-to-peer system applications. Experimental results from the paper prove conclusively that the proposed system achieves better query response times while using fewer computational resources than standard procedures. The approach has specific performance drawbacks since it depends on shortest path calculations that might prove inadequate for working with non-Euclidean graphs within complex distance measures. Resource-constrained environments along with huge graphs present challenges for SSP pruning because it requires sophisticated calculations and large memory utilization. The scalability improvements might face challenges when dealing with huge

graphs beyond experimental sizes. In studies, there is no explanation exists about how to manage real-time dynamic changes in graphs that occur through frequent updates and their potential effects on performance across dynamic settings. This method represents a major advancement of dynamic skyline queries, but additional optimizations will be necessary to achieve wider practicality in complex dynamic or massive graph settings.

Similarly, the work in (Abbaci., 2011) introduced the similarity skyline method that enables improved graph query comparison through multiple local distance metrics instead of rely solely on a single assessment. The approach delivers flexible and precise results specifically during complex graph analysis because it generates various sets of relevant outcomes. The method leads to efficient performance by first browsing extensive graph databases and then focusing the search results on the most appropriate graphs. The method faces computational complexity problems mainly when processing large complex graphs and struggles to handle such cases effectively. The approach proves efficient with certain types of graph structures, yet it is not appropriate for all possible graph structures. The method fails to include a solution for real-time graph updates while operating in dynamic environments. The similarity skyline approach enhances graph similarity search capability through precise outcomes for various real-world applications even though it faces implementation difficulties.

In addition, the study by (Zheng., 2014) aims to resolve three critical graph data challenges through solutions for dynamic skyline computation and efficient graph querying and expensive storage cost reduction. The study presents a beneficial solution for finding query isomorphic subgraphs that stand alone from domination, but the method encounters performance constraints. Although dynamic skyline computation tackles skyline adjustment demands in evolving graphs, it fails to ensure optimal performance of these calculations for real-time processing particularly with dynamic graphs. The query efficiency solution faces scalability problems when target increases graph size that might cause performance slowdowns during structural constraint validation. While optimizing

storage expenses is crucial, the approach does not provide a cohesive methodology for managing large and complex graph data. How well the proposed methods function when faced with space limitations and pruning requirements in extensive real-world environments negatively affects application performance. The study delivers important knowledge about subgraph skyline queries, yet its proposed solutions need to be improved to effectively manage scalability and real-time processing while minimizing storage requirements in expansively changing graph systems.

Furthermore, the work presented by (Amr & El-Tazi, 2018) introduces skyline query over graph database where it proposes a method using divide-and-conquer; nested loops, which was 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 $O(n^2)$. In contrast, the divide-and-conquer algorithm recursively partitions the dataset, compares nodes within partitions, and merges the results, achieving a more efficient time complexity. Performance evaluations demonstrate that the divide-and-conquer algorithm generally outperforms the nested loops algorithm in handling various graph database sizes and query complexities.

Meanwhile, the work presented in (Alwan., 2018) developed a method which is used to estimate the missing values of skyline. There are four phases to finding the missing values i.e., generating AFDs. To generate AFDs, the skylines with missing values are divided into two sets: one set with missing values in the dimension to be estimated and another set with the remaining skylines. The relationships between dimensions are analyzed to generate an AFD that shows how one dimension influences the others. After generating the AFD between the dimensions in the second step, it will be used to determine the strength of correlations between the two dimensions. Moreover, the imputation with missing values of the dimensions in the skylines with the estimated values, and finally, the last phase of the ranking of skylines is based on the strength of probability correlations, ensuring that the most precise skylines are presented first for user selection.

Furthermore, the work presented in (Ouyang., 2018) addresses the type of skyline query processing is path skyline query problem in bicriteria networks particularly the graph network. Path skyline query aims to find all the skyline paths from a source node to a destination node. The paper specifically addresses the path skyline query problem in bicriteria networks, where the goal is to minimize two criteria simultaneously. The processing method used to handle skyline queries is PSQ+. This algorithm is advance variant of PSQ algorithm process method which works by starting with pushing the node into a queue then pop an element from the queue and checks if the path from source to current node is dominated by the last path in the skyline of the current node. If the path is a skyline path, it is inserted into the skyline of the current node, and the edge relaxation is performed for each neighbour node. The process continues until queue is empty and the skyline paths from the source to the destination node. The PSQ+ eliminates the need for maintaining non-skyline path, making it more efficient than previous algorithm PSQ.

In addition, the study by (Zhu, 2018a) focuses on authenticating location-based skyline queries (LBSQs) while running in road networks because cloud service providers (CSPs) process data management requests. The method includes neighbors of each POI in the signature to provide users with an efficient means to verify both the result's authenticity and completeness. The technique connects POIs through distance-related neighbors and skyline neighbors to perform both secure and quick query verifications. The method reaches high efficiency in real-world deployments through two key benefits that minimize both processing time and message signatures without sacrificing performance. The initial drawback of this method is the required dataset preprocessing step that adds computational processing time to implementation. The system encounters difficulties in handling extensive datasets because of the challenges it faces when working with numerous signatures for a significant number of POIs. The algorithm is effective for application but has some scalability concerns for massive datasets.

The work by (Miao., 2018) looks at how to solve the *why-not* problem in skyline queries by changing the query range, the point that's missing, or both. This helps make sure that a missing point becomes part of the result. The study offers smart ways to speed up this process using methods like skyline scope and adjusting non-spatial attributes. This is similar to the earlier study by (Zhu., 2018a) over road networks, which focuses on making sure that skyline query results are correct when stored in cloud systems. While the first study checks if the results are right, this one explains how to include points that are missing from the results. Both studies help make skyline queries work better in big networks, but they can still have issues when working with very large or complex data.

The work presented in (Gulzar, Alwan, & Turaev, 2019b) concentrated on the type of skyline query processing in the presence of incomplete data. This type of skyline query processing in databases with missing data is particularly useful in large cardinality and high dimensionality databases. The processing method used is OIS which works in four steps. The first step clusters the data items to help prevent the issue of cycle dominance. Secondly, sorting and filtering to remove the dominated data item and reduce the domination tests. Moreover, the local skyline helps to determine the local skyline data items of each candidate list. The main purpose of this is to speed up the skyline process by running multiple processes in parallel, which provides with non-dominated items for the next stage. Till this stage there will be no issue of cyclic dominance and transitivity property of the skyline even with missing values. In the last stage, it will return the final skyline data items of the entire database. This is achieved by comparing the reported local skyline data item of all candidate lists with each other to form the final skylines.

Building upon the theme of skyline processing in the presence of incomplete data by (Gulzar, Alwan, Abdullah., 2019) 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.

Moreover, the work presented by (W. Liu., 2019a) focuses on the problem of skyline nearest neighbour search on multi-layer graphs. The algorithm utilizes an early-termination condition and optimization strategies to improve efficiency. The early termination condition allows the algorithm to stop the computation of shortest distances once a vertex has visited on all layers, reducing unnecessary computations. optimization strategies such as optimizing the search order are employed to further enhance the algorithm efficiency.

The SkyTPF method (Keles & Hose, 2019) interface serves as an extension to basic RDF interfaces SPARQL and Triple Pattern Fragments (TPF) which enables more efficient skyline query execution by reducing search area. Laboratory tests indicates that SkyTPF delivers improved speed and handling capabilities for related information with additional benefits that occur from running multiple threads. The method provides an approach which combines lightweight efficiency with scalability to allow skyline processing of big knowledge graphs through standard available tools for data sets that demonstrate correlation. The method demonstrates performance delays in anti-correlated data processing while it struggles to maintain good results when working with high-dimensional data sets because of complicated dominance assessments. The technique proves to be effective for analyzing correlated data despite facing restrictions in dealing with advanced data structures.

Moreover, in Basic Distributed Skyline (BDS) (Kumar Sadineni, 2020a) technique which presents a straightforward and efficient solution for distributed system queries though it proves inefficient with large dataset sizes because of high processing expenses.

The Improved Distributed Skyline (IDS) enhances the BDS technique by reducing node visits, which improves efficiency; however, it faces scalability limitations when processing large data collections. PDS applies its ranking method to termination point predictions to decrease comparison operations, though it allows certain points to go unnoticed in advanced data structures. MANET represents a technique that supports mobile distributed networks yet proves limited in processing extensive peer-to-peer network systems. SKYPEER and SKYPEER+ increase efficiency in big P2P networks through super-peer dependency, although these super-peers present potential performance bottlenecks. The parallel query optimization in PaDSkyline involves multiple filtering points at the cost of heavy resource usage in target nodes. FDS presents lower bandwidth requirements by restricting communication; however, it depends on repetitive communication sessions to obtain results, therefore delaying the overall process. ACL executes efficiently in selected network structures; however, it demands substantial processing expenses to maintain data updates. SSP, along with SKYFRAME takes advantage of modern space partitioning methods as well as parallel processing techniques while continuing to experience problems related to load balancing. The iSky system employs adaptive filters that minimize communication costs in extensive distributed systems at the expense of needing dynamic data control management. The techniques offer beneficial features, yet they all experience challenges when trying to scale operations and handle big, intricate data sets effectively.

Building upon these challenges, the work presented in (Banerjee., 2020a, 2000b) focused on the dynamic skyline queries on uncertain graphs. The dynamic skyline query returns a subset of data vertices that are not dominated by other data vertices based on certain distance measures. The processing method is divided into 3 steps, i.e., Pruning, Distance Computation, and Skyline Vertex Set Generation. Pruning takes two steps; First, Pruning is done by performing based on the path length computation. In the second step distance computation is performed between the candidate skyline vertices and the query vertices. The two distance measures the distance and its expected distance in the skyline vertex set generation for the block nested loop (BNL) algorithm. The algorithm works by

pruning the candidate vertices, computing distances, and generating the skyline vertex set based on the computed distances.

In similar work, presented by (Gulzar., 2021) specifically addresses the challenges of processing skyline queries in dynamic and incomplete databases, where the contents of the database are frequently updated, and certain dimensions of tuples may have missing values. The algorithm used was IDSA which comprises seven phases that work together to determine the skylines of an incomplete and dynamic database. The pruning processes the suggested algorithm's first step effectively by identifying new skylines and drastically reduces the amount of domination tests necessary by strategically using current skylines before INSERT/UPDATE operations. Also, in the step two which continues to refine the process by lowering the number of domination tests and avoiding the requirement to recalculate skylines across the full updated database by choosing superior local skylines and excluding non-skylines from further evaluation. Following these introductory stages, step three entails the Insert Operation, which includes adding a particular set of tuples to the first database. The step four deals with the Delete Operation, which requires removing a particular set of tuples from the same original database. The step five goes into the Analysis of previous approaches, highlight challenges in processing skyline queries in incomplete and dynamic databases and reviewing prior methods proposed for addressing this issue. The step six, the Derivation of New Candidate Tuples, assesses the domination power of tuples within the *dom-p-list,* determining the maximum *dom-p* value and generating a curated list of fresh candidate tuples. Finally, in step seven, the algorithm's efficiency is rigorously Evaluated and Concluded through experiments on both real and synthetic datasets. These experiments serve to showcase the IDSA algorithm's prowess in generating accurate skylines after alterations are made to a database.

In a different approach, the study by (C. M. Liu et al., 2021) involves sorting the points of a dataset into distinct list based on each dimension and accessing them in round-robin fashion. The algorithm utilizes a different indexing technique that allows optimization of access based on both the number of complete dimensions a point has and

the sorting data. The approach involves sorting the points of a dataset into distinct lists based on each dimension and accessing them in a round-robin fashion. The algorithm considers the dominance of a point based on both the highest value of a given dimension and the number of complete dimensions it has.

Meanwhile, the work presented in (Miao., 2021a) has shown a process which is divided into two main phases: modelling and crowdsourcing. In the modeling phase, a Bayesian network is trained to capture data correlations, and the *c*-table is constructed, which represents objects and their associated conditions. This *c*-table is crucial for determining the likelihood of each object being a query result. The crowdsourcing phase involves selecting and posting tasks for crowd workers to resolve missing data issues, using task selection strategies such as Frequency Based Strategy (FBS), Uncertainty Based Strategy (UBS), and Hybrid Heuristic Strategy (HHS). These strategies aims to optimize task selection by balancing cost and latency constraints while ensuring accurate query results. Specifically, FBS selects the most frequent expressions, UBS prioritizes tasks with high uncertainty to reduce ambiguity, and HHS combines elements of both to improve efficiency. The framework iteratively refines the query results based on crowd responses until the query is resolved.

In relation to missing data, the work in (He & Han, 2022) addresses skyline query for incomplete query dataset. The processing method consists of two stages for efficient skyline computation on massive data. In stage one, the TSI performs a sequential scan on the table to compute the candidate tuples. Tuples that are dominated by others are discarded directly in this stage. In Stage two, the TSI refines the candidate tuples by another sequential scan. This stage aims to discard the candidates that are dominated by some other tuple. The TSI utilizes a pruning operation to reduce the execution cost in stage first. It maintains a pruning bit-vector (PRB) in memory, initially filled with bit 0. This bit-vector helps execute the pruning operation efficiently. In stage two, the TSI refines the candidate tuples by another sequential scan, discarding candidates dominated by some other tuple. The pruning operation in TSI skips most tuples in stage one, thus reducing the cost

significantly. However, it may leave some candidates that should be removed by certain tuples. The TSI demonstrates high efficiency in computing skyline on massive incomplete data.

Furthermore, the work in (Ding., 2023) focuses on path queries over temporal graphs with labels. An efficient index-based method for skyline path query over temporal graphs with Label's involves proposing a novel approach to skyline path queries over temporal graphs with labels. The study introduces the Main Point (MP) index, which consists of MP discovering and certain set construction phases. This index is designed to efficiently handle skyline path queries by identifying key points in the graph and constructing sets that aid in query processing. Additionally, the study presents the TMP algorithm, which is based on the MP-index and utilizes a bidirectional topology strategy to solve skyline path queries over temporal graphs with labels. By leveraging the MP index and the TMP algorithm, the study aims to provide an effective and efficient solution to the skyline path query problem, addressing the challenges posed by multiple constraints and the inclusion of temporal and label elements in the query process. Through experiments and comparisons with other algorithms, the methodology demonstrates the performance and effectiveness of the proposed approach in handling skyline path queries over temporal graphs with labels.

Meanwhile, the Probsky farmework (Kuo., 2023) which leverages the MapReduce paradigm to efficiently evaluate probabilistic skyline queries on large, uncertain datasets. The methodology involves partitioning the dataset using slab-based partitioning, computing local skyline points within each partition, and then calculating the skyline probabilities of uncertain objects using reference points to accelerate the process. The framework incorporates three optimization techniques: 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. These techniques collectively enhance the efficiency and scalability of the framework in a distributed computing environment

In addition, the work in (Shu., 2023) which introduces a new processing method for incomplete QoS data skylines to enhance service recommendations with increased speed, broader range of options and superior precision. The dimension-based partitioning strategy of this approach makes query processing faster by decreasing service numbers which need evaluation. Polygon skyline queries help enhance recommendation diversity because they allow better assessment of service quality independent of missing data values. Experimental tests demonstrate the proposed method delivers superior results than other tested approaches. The approach experiences performance reduction when the services under consideration are few since service partitioning increases the processing time. The system struggles while processing missing data because different QoS measurements receive uneven distribution patterns. The method uses QoS normalization, yet this approach might become less effective when the attributes measure different units and scales. The efficiency of the approach diminishes while the number of dimensions grows, thus making it less efficient with high-dimensional datasets. The existing method presents obstacles that need to be resolved to make it workable for practical assessment.

Similarly, the study by (Yuan l., 2024a) offers an innovative and efficient solution for skyline query processing on multidimensional incomplete data. The use of a weighted classification tree and optimal virtual points significantly improves the efficiency of skyline queries by reducing redundant data and minimizing unnecessary comparisons. However, the method faces challenges related to the scalability with high-dimensional and dynamic datasets, as well as memory overhead in large datasets. Future work could focus on improving the algorithm's performance for highly dynamic data and further optimizing it for high-dimensional datasets to make it even more applicable in diverse real-world scenarios. Despite these challenges, the proposed approach represents a valuable contribution to skyline query processing in incomplete data environments.

Similarly, the work by (Yuan., 2024b) offers an innovative and efficient solution for skyline query processing on multidimensional incomplete data. The method first addresses data redundancy and low classification efficiency by introducing an incomplete

data weighted classification tree. This tree classifies the data into distinct buckets based on the patterns of missing values. Following this classification, the algorithm introduces the concept of optimal virtual points, which are composed of the maximum values from local skylines within each bucket. These virtual points are then used to rapidly identify and prune dominated points across different buckets, significantly reducing the number of required comparisons and filtering out useless data early in the process. However, the approach has limitations related to scalability in high-dimensional datasets. The time complexity of the initial classification phase shows an exponential dependency on the number of dimensions, which could make it computationally expensive as dimensionality increases. Furthermore, the two-stage process, while effective, introduces a preprocessing overhead that may impact real-time performance in highly dynamic environments. The method's overall efficiency is also heavily reliant on the effectiveness of the initial classification tree which may vary depending on the data distribution.

In the context of data streams, the work presented by (Mohamud., 2025) addresses the challenge of redundant computations in skyline query processing. The study introduces the Dominance Analyses Reduction (DAR) framework, which aims to minimize repeated dominance analyses that occur in overlapping sliding windows or when object pairs reappear over time. The core of the framework utilizes the Apriori algorithm, a data mining technique, to identify and store frequently occurring dominance relationships in a list. When processing subsequent data windows, the framework checks this list before performing a pairwise comparison. If a dominance relationship has been previously computed and stored, the calculation is skipped, and the stored result is used instead. While the DAR framework demonstrates a significant reduction in pairwise comparisons, its performance is highly dependent on the nature of the data stream. The approach is most effective when there is a high frequency of recurring object pairs, allowing the Apriori algorithm to build a useful repository of dominance results. In streams with highly unique or non-repeating data, the overhead of running the Apriori algorithm and maintaining the frequent dominance list might outweigh the benefits. Additionally, the framework introduces memory overhead for storing the frequent patterns and relies on efficient

indexing (such as hashing) to prevent the lookup process itself from becoming a bottleneck in large-scale streams.

Also, the study by (Martin-Nevot & Lakhal, 2025) has proposed an efficient ranking methods for skyline queries in large and high-dimensional datasets. The authors first enhance the *dp-idp* approach by adding a dominance hierarchy to reduce redundant dominance checks. They then introduce two new methods RankSky, inspired by Google's PageRank, and CoSky, based on TOPSIS and Cosine similarity. While RankSky captures global importance through iterative matrix computation, it is computationally expensive. In contrast, CoSky offers a faster and SQL-integrable solution using Gini-based weighting for attributes. The CoSky provides better scalability and simplicity, while RankSky ensures more precise global ranking but with higher cost.

Finally, the work in (Yuan., 2024c) classifies incomplete data using a weighted classification tree, where missing and non-missing values are represented by 0 and 1, respectively. Skyline queries are performed within each class bucket to identify local skyline points. The study introduces the concept of optimal virtual points, which are the maximum values of local skyline points within each bucket. These optimal virtual points are then introduced to other buckets to minimize comparisons. Local skyline points dominated by these virtual points are moved to a shadow skyline. Finally, global skyline points are determined by comparing candidate skyline points in each bucket with the shadow skylines of other buckets, removing dominated points and ensuring efficient handling of multidimensional incomplete data. By integrating weighted classification trees with skyline query algorithms, the study aims to enhance data classification efficiency and reduce the number of comparisons, addressing the challenges posed by incomplete multidimensional data. Through experiments and comparisons with other methods, the methodology demonstrates improved performance and effectiveness in processing skyline queries over incomplete data.

Table 2.2 Summary of Previous Studies of Graph Database

Approach	Technique	Data distribution	Database Type
(Zou.) 2010	Sorting	Independent, correlated	Synthetic, Real
(Abbaci., 2011)	Clustering	Independent, correlated	Synthetic, Real
(Zheng., 2014)	Sorting	Independent, correlated	Synthetic, Real
(Alwan., 2018)	Clustering	Independent, correlated	Synthetic, Real
(H. Wang., 2018)	Clustering	Independent, correlated, anti-correlated	Synthetic, Real
(Ouyang., 2018)	Sorting	Independent, correlated	Synthetic
(Amr & El-Tazi, 2018)	Sorting	Independent, correlated	Synthetic, Real
(Zhu., 2018b)	POI	Independent	Synthetic, Real
(Miao., 2018)	POI	Independent	Synthetic, Real
(Gulzar, Alwan, & Turaev, 2019a)	Sorting, Clustering	Independent, correlated	Synthetic, Real
(Gulzar, Alwan, Abdullah, 2019)	Clustering	Independent, correlated	Synthetic, Real
(W. Liu et al., 2019b)	Sorting	Independent, Correlated	Real
(Banerjee, 2020a,2020b)	Sorting	Independent	Synthetic, real
(Gulzar, 2021)	Sorting, Clustering	Independent, correlated	Synthetic, Real
(C. M. Liu , 2021)	Sorting	Independent, correlated	Synthetic, real

(Miao et al., 2021a)	Table	Independent, correlated	Synthetic, real
(He & Han, 2022)	Sorting	Independent, correlated	Synthetic, Real
(Ding., 2023)	Sorting	Independent, correlated	Synthetic, real
(Kuo, 2023)	Clustering	Independent, correlated	Synthetic, real
(Yuan, 2024a)	Clustering	Independent, correlated	Synthetic, Real
(Yuan, 2024c)	Clustering	Independent, correlated	Synthetic, real
(Yuan, 2024b)	Clustering	Independent	Synthetic, real
(Mohamud, 2025)	Clustering	Independent, correlated	Synthetic, real
(Martin-Nevot & Lakhal, 2025)	Clustering	Independent	Real

2.3.4 CRITICAL DISCUSSION

Although dynamic skyline queries for large graphs have been an actively researched topic in recent years, current approaches for instance, SSP pruning (Zou, 2010) significantly improve query efficiency but still face limitations in handling non-Euclidean graphs and complex distance measures. These techniques, while effective in reducing unnecessary computation, demand complex computations and take much larger memory usage, making them difficult to scale for enormous graphs. Furthermore, the current methods fails to address the challenge of managing real-time dynamic updates in graphs, which is particularly problematic for applications that involve high-speed updates and complex data structures. Hence, future research will be necessary to identify synchronous and efficient methods that can address the issues of real-time updates in large and dynamic graphs.

Similarly, methods such as similarity skyline (Abbaci., 2011) approaches provide flexibility for using more than one distance metrics to enhance query comparison. But the technique is suffering from computational complexity when it comes to dealing with large, complex graph, particularly in real-time environments. The gap of research is that there is the requirement of more adaptive skyline query manner, which can be automatically changed along with the dynamic nature of graph data.

The study by (Zheng2014) deals with some of these challenges by proposing solutions for dynamic skyline computation in evolving graphs but it struggles with scalability and real-time performance with the graph size. Although the technique is concerned with the reduction of the storage costs, its performance decreases as the dataset increase, especially in the case of complex structural constraints and space limitations. This points out the necessity for the purpose-built and scalable solutions which are capable of addressing the requirements of real-time processing on large-scale dynamic graph system (Amr & El-Tazi, 2018), which applies divide-and-conquer and nested loop algorithm to skyline queries in graph databases. Although the divide-and-conquer approach has a more favorable time complexity than nested loops, both are poor in terms of performance when

dealing with large data bases because of the need for complete comparisons. This is significant research gap, particularly in the case of high-dimensional and incomplete graph databases, where the cost of comparisons of the overall dominance increases exponentially.

The research on skyline queries in graph databases encounters major obstacles when handling incomplete data and dynamic large-scale databases. The major gaps include lack of scalability, real-time update handling, and efficiency in large, complex datasets. The academic research needs to advance techniques along with efficient methods and scalable solutions that overcome the characteristic changes in graph data and problems caused by incomplete information.

2.4 Machine Learning for Query Optimization

The cost estimation process using machine learning techniques yields better accuracy when operating on graph databases. The prediction of query costs becomes challenging using conventional approaches because they fail to identify future changes in the interconnected structure of graph data. Recent researchers have attempted to solve this issue through a machine learning predictive model, improving the precision of estimates for graph query execution plans.

The main difficulty in query optimization consists of picking the most effective sequence of table joins. The execution time fluctuates substantially depending on join order selection while conventional techniques lack sufficient effectiveness in exploring broad search areas which frequently leads to local optimal solutions. The Deep Q-Learning (DQL) serves as the proposed reinforcement learning (RL) method (Fahima et al., 2024) to optimize join ordering. The system gains knowledge about optimal join sequence patterns through the regular feedback from previous query processing cycles which enhance its

decision-making capability. Particle Swarm Optimization (PSO) struggles with local optima, whereas Q-Learning adapts better to larger query spaces, ultimately improving execution times. Also, traditional query optimizers struggle with uncertainty in cost models, leading to suboptimal query plans. Modern databases run machine learning-based optimizers which often lack explainable behavior thus creating obstacles for administrators (Makrynioti et al., 2018). The approach demonstrates better accuracy as well as robustness combined with explainability when compared to traditional methods.

Traditional query optimizers rely on cost-based models that using statistics and heuristics, estimate execution costs, which are fixed and cannot adapt to the dynamic workload and changing data distribution. Also, manual tweaking of parameters like indexes can take time, and it is error-prone. This study discusses various machine learning (ML) methods, such as supervised learning, to use historical query data to more accurately predict execution costs. Unsupervised learning is utilized for workload characterization, while reinforcement learning allows for the dynamic query optimization by adapting to new loading queries. This means that the optimization process is now automated and adaptive and can be scaled so that manual handling is reduced to a bare minimum.

Choosing the best query plan among a vast number of possible plans is also difficult, especially when these contain subqueries or joins. Accurate cardinality estimation is needed for efficient evaluation, but traditional methods infrequently estimate it correctly. The study applies Deep Reinforcement Learning (RL) to incrementally build query plans. The system uses state representations of subqueries, where each state represents intermediate results, and selects actions (operations) that lead to efficient plans. Gradually over time, the RL model is trained by past actions, it enhances the predictions and optimizes the execution time by picking the right for operations as well as joining orders.

Optimizing query execution on large data in both OLAP and OLTP workload systems is still a challenge. Conventional approaches cannot effectively deal with complex

heterogeneous data distributions and even dynamic data changes; they will lead to suboptimal performance. It gives an overview of deep learning (DL) methods for enhancing query execution, specifically learned indexes to subdue the old B-trees for efficient data expense retrieval. Further, neural networks and reinforcement learning is used in cardinality estimation and query plan optimization so that it can adapt dynamically to the changing data distributions. In total deep learning much improves performance by dealing with dynamic stuff very much better.

2.5 Gaps in Existing Work and Justification for Research

The field of skyline queries for incomplete graph databases currently requires further research and development to address the challenges specific to this area. One of the main problems of this area is how to handle incomplete data. Traditional skyline query algorithms are generally designed for complete databases, so they are not fit when we have incomplete data. This problem is of special importance in real scenarios such as social networks and recommendation algorithms in which graph databases usually have many missing values. All existing methods either neglect missing values or use simple imputation strategies that can contaminate the dominance relations, which are essential for the accurate skyline queries. In order to handle this, the proposed research is to employ machine learning techniques, specifically the K-Means algorithm, to deal with missing data more properly so that the return of the skyline query remains accurate over the incomplete dataset.

Additionally, the skyline query algorithms for large-scale graph databases lacks scalability and performance optimization. Traditional skyline query algorithms are computationally expensive, especially when dealing with large amounts and complex graph data that is incomplete. In high-dimensional datasets, this issue becomes even bigger, as the number of skyline points grows faster with each added attribute. By implementing sorting and filtering along with the clustering algorithms, this research decreases the

processing cost and boosts the scalability of skyline queries under any circumstances in large-dimensional high-graph periphery incomplete data.

Also, the gap of cycle domination in an incomplete graph database is a huge problem. Missing or incomplete data can lead to cyclic dominance, when no dominance relationships between the particular data points exist, and there are incorrect query results of the skyline. To solve this issue, the research leverages K-Means clustering to group nodes having similar properties in order to suppress cyclic domination and to preserve the transitivity inherent in skyline queries. This approach guarantees that the skyline query result is correct even in the case of some missing query results.

Another gap in the literature is on the excessive pairwise comparisons involved in skyline query algorithms, particularly those that handle large-scale or high-dimensional data. Traditional skyline for skyline query is usually based on evaluating each pair of nodes which becomes expensive in incomplete graph databases. To tackle this, the research introduces a clustering-based method which ensures to reduce the number of pairwise comparison by grouping nodes that have same missing dimension and reduces the computational overhead involved in the skyline query operations. Lastly, the presented research implements machine learning approaches to fix skyline query limitations in incomplete graph databases thus enhancing performance while handling data and real-time updates.

2.6 SUMMARY

This chapter presents skyline queries as a database retrieval method to obtain non-dominated data points from various criteria. The research begins with a discussion of complete database skyline queries using foundational Bitmap and Index algorithm

techniques. The algorithms achieve efficient operation within complete datasets by decreasing the search space while also minimizing useless comparison operations. However, the chapter identifies this method when applied to incomplete databases, where incomplete data generates incorrect results.

This chapter discusses how to perform skyline queries when dealing with incomplete databases by handling data with missing information in database. Traditional methods such as data imputation or probabilistic models are mentioned but are shown to be sufficient for dealing with large-scale, high-dimensional datasets. The chapter explores skyline queries in graph databases, which introduce distinct challenges of the dynamic and interconnected nature of graph data. It highlights the necessity for techniques that can handle real-time updates in graph structures particularly when it is incomplete. It highlights the limitation of current methods in tackling missing data particularly if the data is incomplete.

CHAPTER THREE

METHODOLOGY

3.1 INTRODUCTION

This chapter presents the Design Science Research Methodology (DSRM) proposed by Hevner (2014) as the framework for this research as depicted in Figure 3.1. The DSRM provides a structured approach for designing, developing, and evaluating computational artifacts, making it particularly suited for the development of the proposed machine learning-based skyline query optimization approach in large-scale, incomplete graph databases.

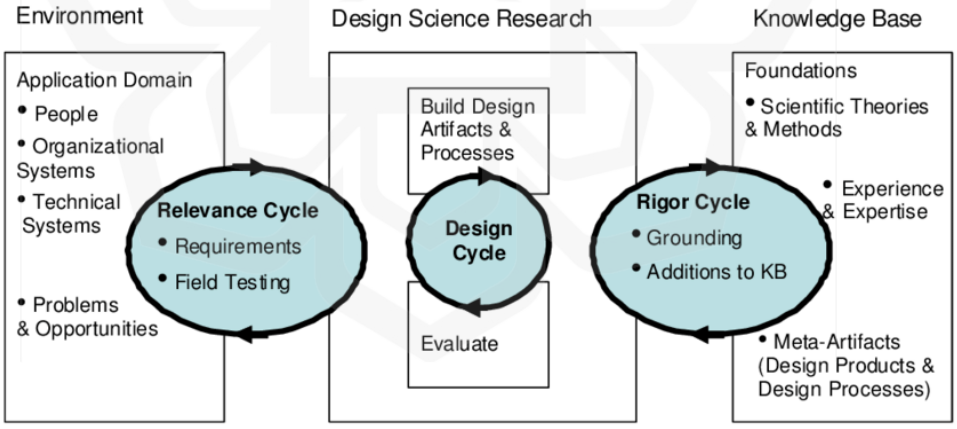


Figure 3.1 DSRM Cycle (Hevner, 2014)

3.2 Design Science Research Methodology

DSRM consists of six stages, each of which aligns with different aspects of this research.

1. Problem Identification and Motivation
2. Define Objectives of a Solution
3. Design and Development
4. Demonstration
5. Evaluation
6. Communication

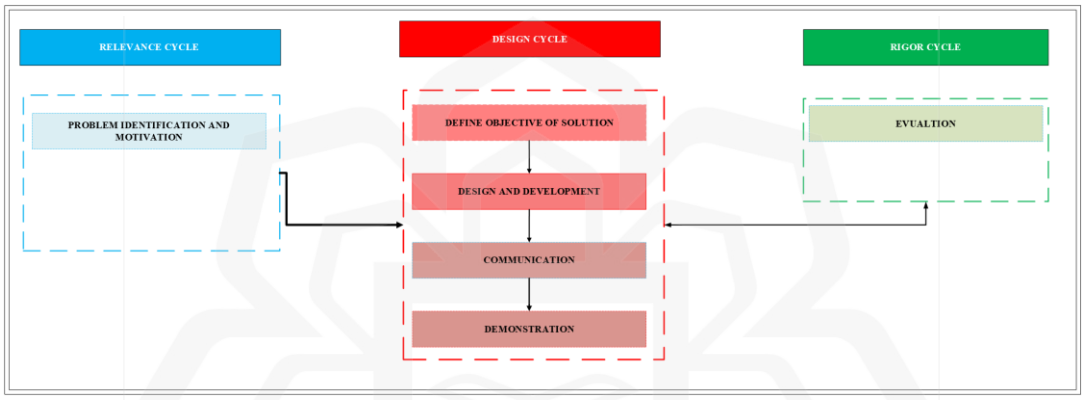


Figure 3.2 DSRM Cycle Processing

3.2.1 Relevance Cycle

The relevance cycle serves as a mechanism that links the surrounding contextual factors of research projects to design science methodology. The core function of this cycle helps researchers discover the basic issues or prospects their research needs to handle. A research team establishes research targets and forms essential questions to validate the necessity for a new answer at this stage. The initialization of research demands researchers precisely to define the research problem during the Problem Identification and Motivation stage. The chosen problem leads to the following stage of the design cycle. A project design cycle functions to base the initiative on practical concerns, thereby providing the fundamental basis for design development.

3.2.2 Design Cycle

A fundamental model running through development testing and evaluation activities will guide the confirmation of research solutions. The process addresses solution development and progressive improvement in three sequential phases known as Define Objectives of Solution, Design and Development, and Demonstration. The research fulfills several stages, which start by setting solution objectives before moving to solution design and testing phases. The research solution requires essential feedback from multiple development stages to refine its performance based on research objectives. The real-world testing and development of research artifacts depend heavily on the design cycle to achieve their essential objectives.

3.2.3 Rigor Cycle

Through the rigor cycle, the design activities are based on proven scientific principles. This cycle connects the research with established theories and methods from earlier studies. By comparing the developed solution to these existing frameworks, researchers can assess if it adds value to the field. The solutions are then evaluated to see how well they work and if they meet the expected goals (see Chapter Four). Evaluation also helps to understand if the solution is relevant for future improvements and if the research problem is better understood. Lastly, the importance of sharing the results, making sure that the findings are communicated clearly, and that the solutions meet both practical and academic standards. Through the rigor cycle, organizations can confirm that their solutions meet both practical needs and academic standards.

3.3 PROBLEM IDENTIFICATION AND MOTIVATION

Skyline queries play a crucial role in various multi-criteria decision-making applications, such as recommendation systems, social networks, and big data analytics. These applications require the ability to select the most optimal data points based on multiple attributes or criteria. A skyline query identifies a set of non-dominated data points, meaning no other data points dominate them across all dimensions. However, the primary challenge arises when dealing with incomplete graph databases, where certain attribute values are missing, thus making traditional skyline algorithms ineffective.

Traditional skyline algorithms typically assume complete datasets, where every node has values for all attributes. In real-world graph databases, attributes are frequently missing or only partially filled (e.g. incomplete ratings, missing demographic fields). This incompleteness breaks the assumptions of classical skyline computation, leading to loss of accuracy, cyclic dominance and excessive computational cost. These limitations motivate the development of a new framework that can handle missing or incomplete data without relying solely on imputation and pruning irrelevant or dominated nodes early to reduce comparisons and exploit machine learning techniques such as clustering to improve scalability and accuracy.

3.3.1 Traditional Skyline Algorithms

Skyline query algorithms typically assume that the datasets they operate are complete, meaning every data point has values for all of its attributes. However, in real-world graph databases, such as social networks or recommendation systems, it is common for some attributes to be missing or incomplete. As a result, skyline algorithms based on complete datasets fail to provide accurate results when data is incomplete, leading to incorrect or incomplete skyline query results.

3.3.2 Challenges Addressed in this Research

This research tackles several fundamental challenges that arise when processing skyline queries in incomplete graph databases.

3.3.2.1 Cyclic Dominance

The cyclic dominance is one of the most significant challenges of working with incomplete data in skyline queries. In complete graph datasets, dominance relationships between nodes (or data points) are defined, and it is easy to identify which data points dominate others. However, when values are missing, this problem disrupts transitivity, which is the principle that if Node A dominates Node B, and Node B dominates Node C, then Node A should dominate Node C. Missing values introduce ambiguity in these relationships, leading to cyclic dominance where no clear domination exists. This issue leads to incorrect skyline results, where some non-dominated points may be mistakenly excluded and some dominated points may be incorrectly included.

3.3.2.2 Excessive Pairwise Comparisons

Pairwise comparisons are an essential part of skyline queries, where every data point is compared with every other data point across multiple dimensions to determine dominance. In large datasets, these processes can lead to high computational overhead. When data is missing, additional complexities are introduced, as the system has to handle incomplete data for each pair which increases the computational cost. This leads to inefficiency as the number of comparisons grows exponentially with the dataset size, particularly in large, incomplete graph databases. The time complexity of pairwise comparisons for a dataset of size n, comparing all pairs requires $O(n)$ comparisons. When handling incomplete data, the cost increases further due to the need to handle missing values for each pair.

3.3.2.3 Scalability Concerns

As the size of the dataset increases, the processing time for skyline queries increases dramatically, especially when dealing with incomplete data. Traditional skyline algorithms do not scale well in these environments, as they cannot efficiently handle the complexity introduced by missing values. As graph databases grow in both size and complexity, the scalability of skyline query processing becomes a major concern. Therefore, efficient data pruning and clustering techniques are essential to reduce the search space and processing time, making skyline query processing feasible in large-scale incomplete graph databases. The proposed solution aims to reduce computational complexity by pruning early in the process, minimizing the number of comparisons and making skyline query processing scalable.

3.3.3 Research Gap and Justification

Existing solutions for skyline query processing in incomplete databases typically rely on techniques such as, data imputation, probabilistic models, or approximate skyline methods. While these approaches may help fill in missing data or approximate skyline results, they introduce several issues.

3.3.3.1 Data Imputation

Imputation methods estimate missing values based on statistical techniques (e.g., mean, median, k-nearest neighbors). However, imputation can lead to inaccurate or biased results, especially when the missing data is not missing at random. The imputed values can distort the dominance relationships, leading to incorrect skyline results.

3.3.3.2 Probabilistic Models

These models attempt to model the uncertainty in missing data and compute skyline results based on probabilities. While probabilistic approaches can offer some level of accuracy, they often involve complex calculations and still struggle to handle large datasets effectively.

3.3.3.3 Approximate Skyline Techniques

These methods aims to determine a subset of the skyline rather than the full set, sacrificing accuracy for efficiency. While faster, approximate methods do not guarantee the precision required for certain applications, particularly in decision-making systems where accuracy is paramount.

The proposed machine learning-based approach aims to address these challenges, integrating sorting, filtering and clustering techniques. This approach can improve query processing in the following ways:

1. By sorting and filtering based on dominated power, it ensures that only relevant nodes are considered, thus minimize unnecessary computations.
2. By clustering nodes with similar missing values, the framework reduces the nodes pairwise comparisons, which makes skyline queries faster and more efficient.
3. Machine learning can adapt to incomplete data by learning patterns from the handling missing values in a way that ensures accurate skyline computation.

This research fills the gap in existing solutions by proposing a more accurate, efficient and scalable approach for skyline query optimization in large-scale incomplete graph databases.

3.4 OBJECTIVES OF SOLUTION

Following Hevner's DSRM approach, the research establishes the following objectives.

3.4.1 Develop an Efficient Framework for Skyline Query Optimization in Incomplete Graph Databases.

The aim is to design a framework that efficiently handles skyline queries, even in incomplete graph databases**.** This framework is explicitly presented in Section 3.4 (Design and Development) and illustrated in Figure 3.3, Proposed Framework for Skyline Query Optimization in Incomplete Graph Databases. It consists of five main components: Sorting and Filtering, Filtration, Clustering, Local Skyline Identification, and Final Skyline Computation. (see Sections 3.4.1 - 3.4.5). By integrating these techniques, the framework improves efficiency and maintains scalability across large-scale datasets, directly fulfilling first research objective (see Chapter One).

3.4.2 Reduce Computational Complexity through the Integration of Sorting, Filtering, and Clustering Techniques.

This objective focuses on improving the efficiency of skyline queries by incorporating sorting, filtering, and clustering techniques. Sorting and filtering will help reduce the query

space by prioritizing more relevant nodes, while clustering will allow for parallel processing of nodes with similar data patterns. These methods will decrease the overall computational overhead and speed up query processing, especially when dealing with large and incomplete datasets.

3.4.3 Enhance Accuracy and Scalability by Applying Machine-Learning-Based Clustering to Handle Nodes with Missing Values.

The goal is to apply machine learning-based clustering, such as K-means or DBSCAN, to handle missing values in graph nodes. By grouping nodes with similar missing value patterns, this method improves the accuracy of skyline results, ensuring the correct nodes are identified even when data is incomplete. This also helps the framework scale efficiently, allowing it to process large and complex graphs without compromising performance.

3.4.4 Evaluate the Performance Improvements in terms of Dataset Size Reduction and Query Response Time, Ensuring Real-World Applicability.

This objective aims to assess how much the proposed framework reduces dataset size and improves query response time compared to traditional skyline query methods. By pruning irrelevant data early in the process, the framework reduces unnecessary comparisons, leading to faster computations. The evaluation will ensure that the solution is not only effective in experimental settings but also applicable to real-world scenarios with large-scale incomplete datasets.

3.5 DESIGN AND DEVELOPMENT

The proposed framework for skyline query optimization in incomplete graph databases consists of five key components as depicted in Figure 3.3: Sorting and filtering, clustering, local skyline identification, and final skyline computation. Each component plays a vital role in ensuring computational efficiency and maintaining the accuracy of the skyline query results.

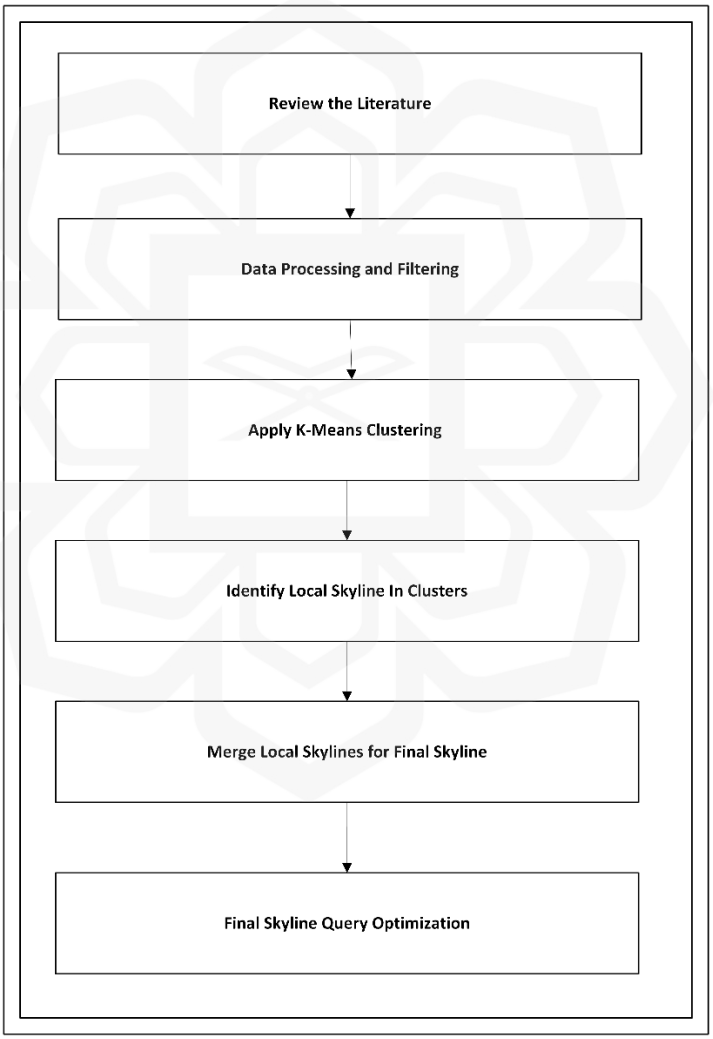


Figure 3.3 Skyline Query Optimization Process Flow

3.5.1 Sorting and Filtering

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 nodes are scanned in a round-robin process to determine domination power for each data item within the nodes. Each node undergoes reading operations at least once during recurring processes until every initial dataset node has been scanned.

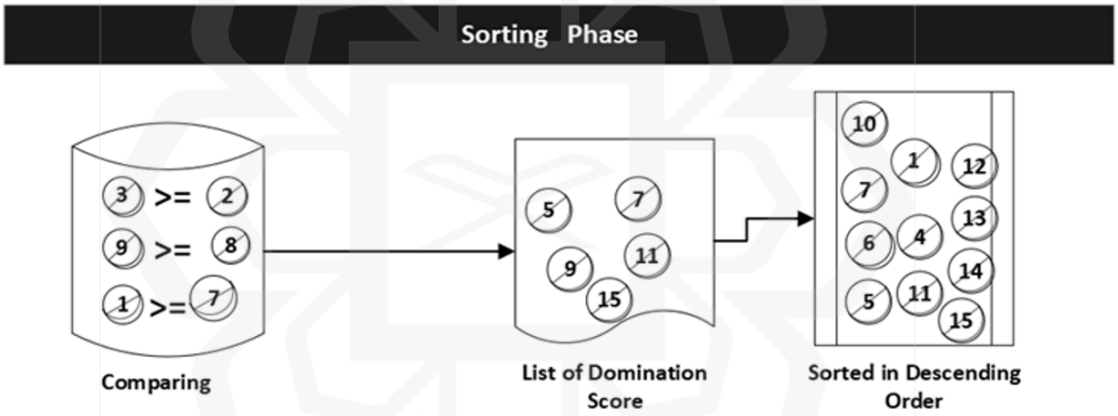


Figure 3.4 Sorting the Nodes

The purpose of this phase, as shown in Figure 3.4 is to organize the data nodes within the dataset in descending order based on the domination power of each node. Additionally, it aims to eliminate nodes that are dominated by others with lower domination power values. The skyline process ignores data items whose domination power values are considered insignificant. Prior to skyline execution, the removal of such data items would decrease the computational cost of the skyline method through a reduction in pairwise comparisons.

The sorting phase is a critical step aimed at organizing nodes in a manner that facilitates efficient skyline computation. A round-robin traversal method is employed to calculate and sort nodes based on their domination power in descending order. Domination power quantifies the comparative strength of a node by assessing how many other nodes it dominates across all dimensions. Importantly, this calculation only considers non-missing values to ensure fairness and accuracy in comparison. This method not only streamlines the skyline computation process but also enables the early elimination of dominated nodes, significantly reducing the computational overhead associated with pairwise comparisons in large datasets. Sorting the nodes upfront ensures that only the most promising candidates are retained for further processing, optimizing resource utilization. The formula for the algorithm of the sorting is:

$$D_i = \sum_{j=1}^{N} dominate(i, j) \tag{1.3}$$

3.5.2 Filtration Process

Throughout the filtration process, any data items with a domination power below a user-defined threshold are excluded from subsequent processing. This decision stems from the understanding that nodes with a domination power less than the threshold lack the potential to be included in the skyline result as shown in Figure 3.5, 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.

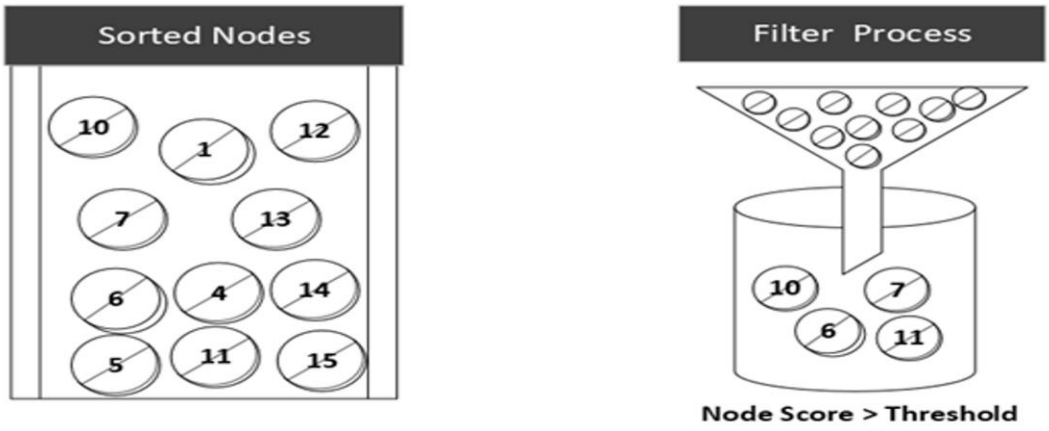


Figure 3.5 Filtration of Nodes

3.5.2.1 Filtering Algorithm

The filtering phase complements the sorting process by applying a threshold-based pruning mechanism. Here, nodes with domination power below a user-defined threshold are excluded from subsequent computations. This threshold is a flexible parameter that can be dynamically adjusted based on the dataset's characteristics or the user's specific requirements, such as the desired level of precision or computational efficiency. By discarding nodes with low domination power early, the approach minimizes unnecessary processing while maintaining the integrity of the final skyline results.

While the proposed sorting and filtering methods are effective, the framework is designed to be adaptable. If alternative algorithms are required for sorting or filtering, they can be substituted without significant impact on the overall results, provided they adhere to the same core principles. This adaptability ensures the framework's applicability across a diverse range of datasets and scenarios, highlighting its robustness and flexibility in handling incomplete graph databases. The formula for filtering is:

$$D_i \geq T \qquad (1.1)$$

3.5.3 Clustering

The objective of this phase is to establish clusters among data items through domination power configurations for optimized skyline processing. In this case, data items with similar domination power are grouped in one cluster as shown in Figure 3.7. Doing so results in generating several distinct clusters. The reduction of data items in space facilitates the avoidance of extensive, unrequired comparisons without affecting skyline result accuracy.

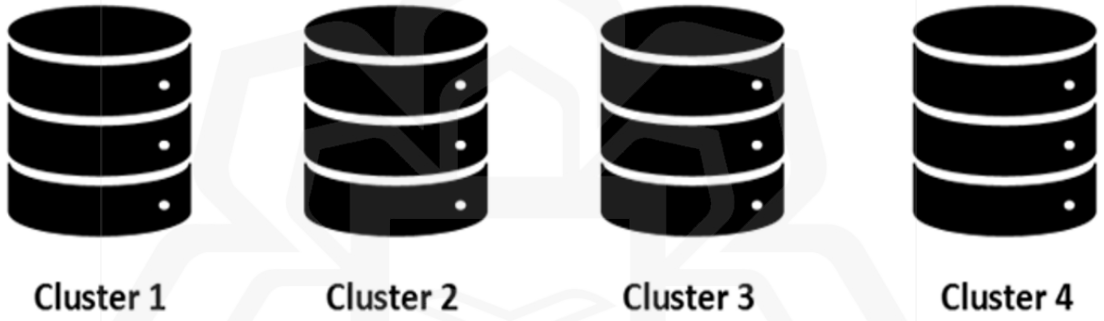


Figure 3.6 Cluster of Nodes

3.5.3.1 Machine Learning Algorithm

The clustering process in this research has been specifically designed to group nodes that have similar domination powers, making it easier to calculate the final skyline. So far, K-Means Clustering has been identified as a promising option because it uses real data points as representatives of each cluster as shown in Figure 3.8. This feature is particularly useful in ensuring that the clusters accurately reflect the dataset and make the skyline computation more precise.

$$\min_{C} \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} ||x_i - u_k||^2 \qquad (1.2)$$

where,

r_{ik} Is a binary indicator if data point belongs to cluster k_i

μ_k is the centroid of cluster k_i

x_i is the data point being clustere

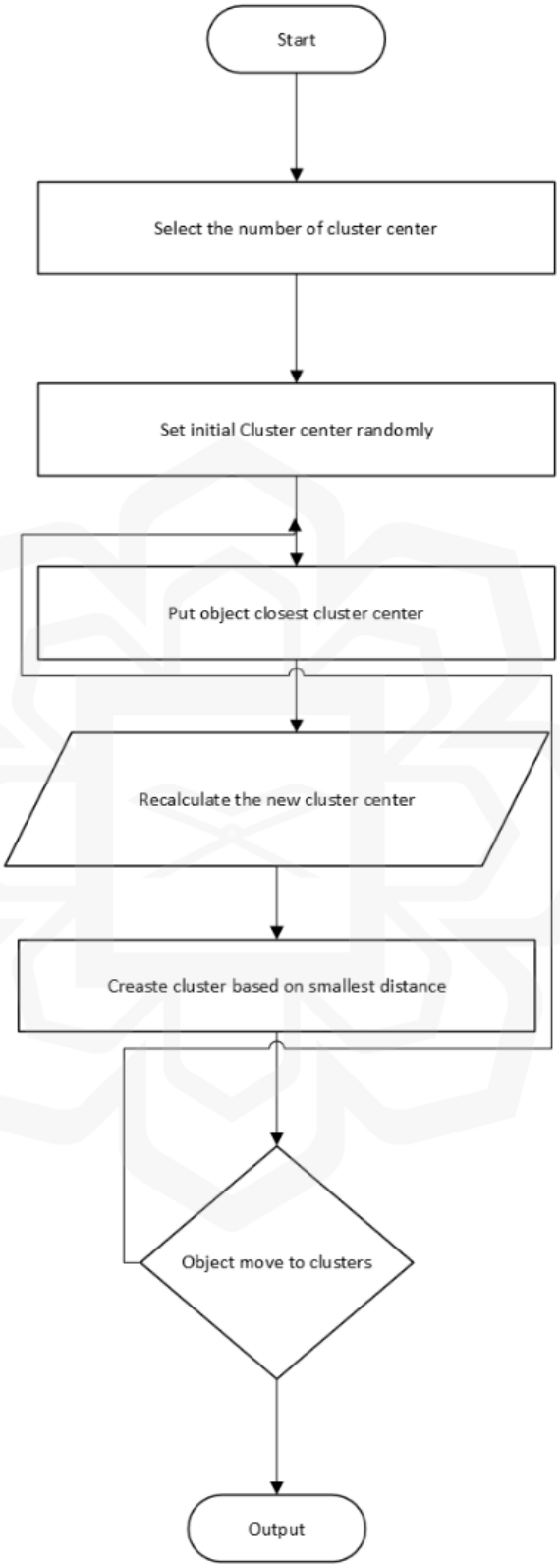


Figure 3.7 K-Means Algorithm

3.5.3.1.1 Training Machine Model

The K-Means Clustering model is trained on the dataset with incomplete values in one dimension. The number of clusters *K* is determined using techniques like the elbow method or silhouette analysis to ensure that the clustering results are both accurate and efficient.

3.5.3.1.2 Validation of the Model

To validate the clustering model, k-fold cross-validation is used. The dataset is split into *k* subsets, and the model is trained on *k - 1* subsets and tested on the remaining one. The model performance is evaluated using the silhouette score and Davies-Bouldin index to assess the quality of the clusters formed, ensuring that the final skyline computation is robust and accurate.

In addition to these cluster quality indices, quantitative error metrics will be used to evaluate clustering performance. These include Mean Squared Error (MSE), Mean Absolute Error (MAE), and the R-squared (R^2) score. MSE and MAE measure the average squared and absolute distances between data points and their cluster centroids, while R^2 quantifies the proportion of variance explained by the clusters. These complementary metrics provide a more comprehensive assessment of the model's performance.

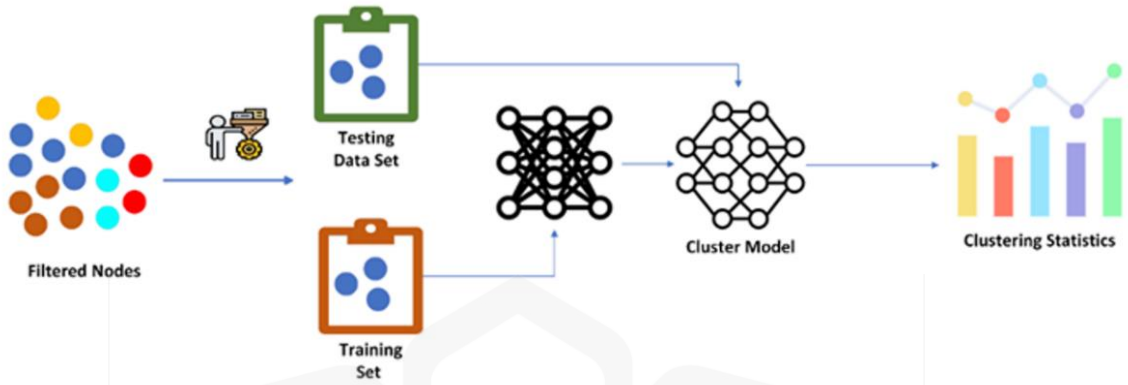


Figure 3.8 Machine Learning of Nodes

3.5.3.1.3 Benchmarking of the Model

The performance of k-means is benchmarked against traditional skyline query techniques like bitmap, SFs and salsa. The performance of the clustering approach is evaluated through three key metrics, such as query response time, accuracy of skyline results, and scalability. The query response time measures how quickly the skyline is computed on large datasets, ensuring that the method can handle real-time query demands. Accuracy of skyline results assesses how accurately the clustering approach identifies non-dominated points when compared to traditional skyline query methods, ensuring precision in the results. Also, scalability evaluates how well the clustering approach performs as the size of the dataset increases, ensuring that it remains efficient even when dealing with large, high-dimensional datasets. The benchmarking will also assess the computational complexity of K-means clustering in comparison to traditional skyline algorithms, ensuring that the proposed approach is efficient for large-scale graph databases.

3.5.3.2 Analyze computational efficiency

3.5.3.2.1 Execution Time

The execution time of the K-Means clustering algorithm depends on the number of nodes, dimensions, and clusters. K-means performs efficiently with datasets that have missing values because the method utilizes parallel processing clustering, while dynamic clustering minimizes pairwise comparisons, leading to faster skyline computation. The processing time becomes longer when dimension and node numbers increase, but early pruning techniques paired with clustering methods like DBSCAN or agglomerative clustering reduce the impact. The time complexity of traditional skyline query methods, including Bitmap, index, and SFS, reaches $O(n^2)$ due to their requirement for pairwise dataset comparisons when handling extensive datasets. The traditional methods become costly to compute when dealing with incomplete data because additional processing through imputation or approximation adds more time to the execution. The K-means clustering execution time reduces because it gathers similar data nodes, which compare those nodes that have the same missing dimension. This technique is more efficient than traditional methods, which rely on full pairwise comparisons and struggle with missing values.

3.5.3.2.2 Memory Usage

The memory usage of the K-Means clustering algorithm is relatively low compared to traditional methods because it only requires storing centroids and the labels of each node, significantly reducing memory overhead. The memory requirement increases as the cluster number and dimension count grow, yet dataset-based cluster modification helps keep memory usage compact. The computation can be across multiple cores to improve memory efficiency using parallel processing. Traditional bitmap and index algorithms need preordered data and bitmaps to manage dominance relations, which need to store

dominance relationships, which consume memory, especially when dealing with large datasets containing numerous dimensions. For high-dimensional datasets, it needs to retain the dominance relationships between all pairs of points. Methods like SFS and LESS result in greater memory utilization when working with growing datasets. The K-Means clustering method requires less memory for data nodes through similar missing dimension analysis although traditional bitmap and index techniques need more memory space when dealing with large-scale or high-dimension datasets.

3.5.3.2.3 Scalability

The K-Means clustering algorithm shows outstanding scalability characteristics, especially when processing large-scale datasets along with missing data entries. A parallel design schema of the algorithm enables efficiency growth that scales with increasing nodes and dimensions. The grouping of data nodes into clusters reduces necessary comparisons, which enables efficient scaling when processing larger datasets and evolving data. On the other hand, traditional skyline query methods like Bitmap, Index, SFS, Salsa, divide-and-conquer, and nested loop all face significant scalability issues when applied to large-scale or dynamic datasets. Data processing methods experience decreasing efficiency when dealing with larger or more complex datasets because pairwise comparison results in an exponential rise in computational processing time according to the number of nodes. Traditional methods, such as Bitmap, Index, SFS and Salsa, Divide and Conquer, and Nested Loop storage methods, experience issues during scale-up while dealing with incomplete data since they need advanced computational processes to fix the problem. So, the K-means clustering technique becomes more scalable in comparison to traditional methods since it avoids the fully pairwise comparisons that standard algorithms require for efficiency, resulting in a faster query time even when the dataset gets larger. Traditional methods, however, struggle to scale effectively with larger datasets, especially when dealing with incomplete data.

Table 3.1 Summary of Comparison

Aspect	K-Means Approach	Traditional Skyline Methods
Execution Time	Faster due to dynamic clustering and fewer pairwise comparisons.	Slower due to full pairwise comparisons, especially for high-dimensional datasets.
Memory Usage	Lower memory usage by storing centroids and labels.	Higher memory usage due to storing dominance relationships and sorting data
Scalability	Highly scalable due to clustering and parallel processing.	Poor scalability for large or high-dimensional datasets.

However, other algorithms are also being considered to find the best fit for the task. For example, DBSCAN is a method that divides points into distinct clusters using density criteria as a base, making it great for datasets with incomplete data or irregular patterns. Similarly, agglomerative clustering, which builds clusters step-by-step by merging the most similar ones, is another strong contender due to its flexibility and ability to adapt to different types of data. The final decision on which algorithm to use will depend on experimental testing. Each method will be evaluated to see how well it works with the dataset, focusing on factors like how accurately it forms meaningful clusters, how it handles missing data, and how efficiently it performs. The goal is to ensure that the chosen approach simplifies skyline computation and aligns with the unique requirements of this research.

3.5.4 Identifying Local Skylines

This component is intended to retrieve local skylines for each constructed cluster as shown in Figure 3.9. There are many benefits to identifying local skylines of each cluster before identifying final skylines. First, it leads to preventing many dominated data items from further processing, which in turn shortens the processing time. Second, it ensures the transitivity property of the skyline technique always holds, as all data items in one cluster

have the same namespace. The process will be in all clusters in parallel, which will reduce the processing time between the data elements.

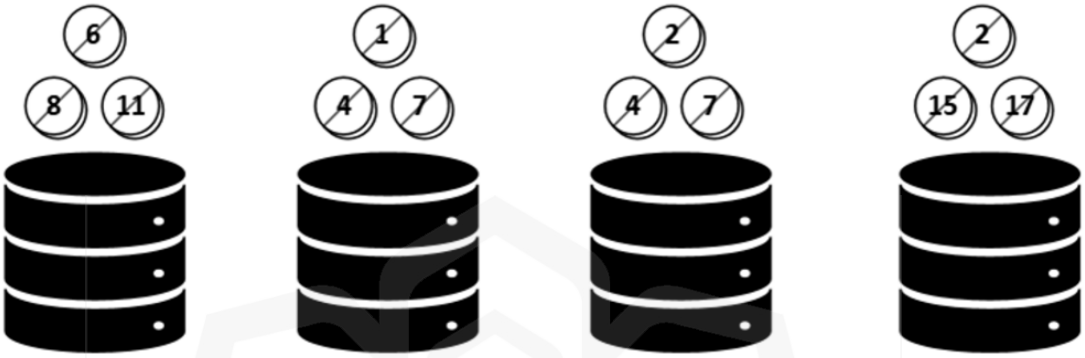


Figure 3.9 Identifying Nodes

3.5.5 Final Skyline

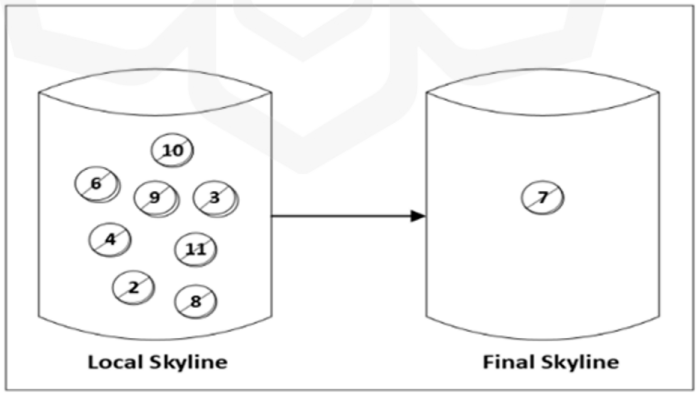


Figure 3.10 Final Skyline

This component is the last component in our proposed framework for processing skyline queries as shown in Figure 3.10 in the incomplete graph database. It is responsible for determining the final skyline of the incomplete database. The process starts by comparing the local skylines generated by 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 is the skyline over the entire database and no other data items might dominate it.

3.6 DEMONSTRATION

The demonstration phase is a critical step to validate the effectiveness of the proposed machine learning-based framework for optimizing skyline queries in incomplete graph databases. This phase involves both the implementation and testing of the framework in a controlled experimental setup, ensuring that the proposed techniques (sorting, filtering, and clustering) function as intended in real-world scenarios. The demonstration phase consists of the following components.

3.6.1 Dataset Creation

In this step, a synthetic dataset was carefully generated to simulate a real-world incomplete graph database environment. The dataset was designed to include missing values at random, mimicking the type of data incompleteness that is common in large-scale graph databases such as social networks, recommendation systems, and other complex datasets. The key aspects of the dataset creation process include the methods used, the types of missing values incorporated, and the overall structure of the dataset.

3.6.1.1 Data Characteristics

The synthetic dataset was created with different types of node attributes and some random missing values. These missing values represent real-life situations where data can be sparse or incomplete, such as missing ratings in a recommendation system or incomplete information in a social network graph. Since large and publicly available graph datasets with incomplete data are very rare and many real datasets have privacy limits or not enough attribute details thus synthetic data will be a better practical choice. It allows full control over the attributes, missing-value rates, and graph size, making it easier to test the framework under different conditions. Others can repeat and verify the experiments using this approach.

3.6.1.2 Controlled Missing Values

The missing values were deliberately introduced in such a way that they allow for controlled testing of the framework's ability to handle incomplete data. The controlled nature of the missing values ensures that the behavior of the proposed optimization techniques can be systematically evaluated.

3.6.1.3 Simulation of Real-world Graph Database

While the dataset is synthetic, it closely mirrors real-world graph databases by including multiple dimensions per node. The simulation allows the framework to be tested in conditions that resemble typical use cases in the field, providing insights into how it will perform when deployed in practical applications. To ensure that the synthetic dataset is similar to real-world data, several checks are done: (i) the graph is built so that its basic

patterns (like how many connections each node has and how tightly nodes group together) look like patterns found in real public networks such as SNAP datasets; (ii) the attributes of the nodes follow the same types of distributions seen in real data (for example normal, power-law or category-based); (iii) missing values are added both randomly and in blocks, at levels commonly seen in practice (about 5–40%); and (iv) simple summary statistics from the synthetic dataset are compared with well-known real datasets such as Yelp or DBLP. These checks help ensure the synthetic dataset is a good substitute for real-world incomplete graph databases.

3.6.2 IMPLEMENTATION

In this phase, the proposed framework was implemented using python as the primary programming language, leveraging a range of machine learning libraries to support the required functionalities.

3.6.2.1 Sorting and Filtering

The implementation of the sorting and filtering process is key to reducing the query space and computational complexity. Sorting was performed based on domination power, where nodes were prioritized based on their influence over other nodes. Filtering was then applied to remove less relevant nodes early in the process, significantly reducing the number of comparisons that needed to be made.

3.6.2.2 Clustering and Machine Learning Libraries

The clustering component, specifically K-means clustering was employed to group nodes that exhibit similar missing data patterns. By clustering similar nodes together, the framework optimizes skyline query processing as comparisons only need to be made within clusters of similar nodes. This approach significantly reduces the computational cost because nodes within the same cluster are more likely to have similar dominance relationships, minimizing the need for extensive pairwise comparisons across the entire dataset.

To implement clustering and support various data processing tasks, the framework utilized several powerful machine learning libraries in Python. The scikit-learn is used for clustering with the K-means algorithm, data processing, and model evaluation. The pandas and NumPy libraries were essential for efficient data manipulation, particularly in handling the incomplete dataset having one dimension missing. The NumPy provided support for numerical operations, while pandas facilitated data cleaning, sorting and filtering, ensuring seamless integration across the different steps in skyline query optimization. The python flexible ecosystem allowed for the smooth integration of clustering, sorting and filtering techniques, making it an ideal environment for building and refining the skyline query optimization framework.

3.6.2.3 Modular Framework

The implementation was modular, allowing each technique (sorting, filtering, and clustering) to be independently tested and optimized. This modularity also facilitated easier integration and future improvements to the framework.

3.6.3 Execution of Experiments

Once the framework was fully implemented, the next step was to execute a series of experiments to assist its performance. These experiments were conducted on datasets of varying sizes to evaluate how well the framework performs under different conditions. The key areas tested during the experiments include.

3.6.3.1 Performance Improvement

A major goal of the experiments was to measure the performance improvements provided by the proposed optimization framework. This involved assessing the impact of the sorting, filtering, and clustering techniques on the overall query processing time. The framework was tested to determine how much response time for skyline queries is reduced compared to traditional methods. The proposed framework is expected to give faster query results, especially in large and incomplete datasets, by optimizing the number of comparisons and getting rid of dominated nodes early on.

The filtering and clustering techniques were tested for their ability to reduce the dataset size before skyline computation. By removing irrelevant or dominated nodes early, the dataset size was significantly reduced, which directly led to improved query performance. This reduction in dataset size also contributed to faster computation times and greater scalability when processing large datasets.

3.6.3.2 Dataset Scaling

The experiments were conducted on datasets with varying numbers of nodes and dimensions to test how well the framework scales. This evidence is crucial to understanding how the proposed framework will perform with large-scale datasets. A smaller subset of the dataset was used to test the initial functionality of the framework and to ensure that all components (sorting, filtering, clustering) were working as intended. The framework was also tested on datasets with 51 nodes with one missing dimension per node. These datasets are representative of real-world graph databases in applications such as social networks and recommendation systems.

3.6.3.3 Comparison with Traditional Methods

The performance of the proposed machine learning-based framework was compared with traditional skyline query methods. Traditional methods often struggle with incomplete data, leading to increased computational overhead and slower query processing. By comparing the proposed framework results with these traditional methods, the research demonstrated how the integration of machine learning techniques (sorting, filtering, and clustering) improved performance in terms of both query response time and accuracy.

3.7 EVALUATION

The evaluation phase is critical for assessing the overall effectiveness of the proposed framework. This phase aims to determine how well the framework meets the objectives set out in the research, particularly in the context of skyline query optimization for incomplete graph databases. The evaluation is carried out by comparing the proposed framework with

traditional skyline query methods and evaluating key performance metrics such as scalability, accuracy, and efficiency.

3.7.1 Performance Evaluation

The implementation of performance metrics for the K-means clustering model provides a crucial step in validating the quality of the clusters. However, K-Means is an unsupervised learning algorithm; metrics such as MSE, MAE and R-Square are irrelevant as they require ground-truth labels. Instead, the intrinsic clustering metrics were utilized to assess how well the data points are grouped based on their internal structure. The primary metrics calculated include the Silhouette Score, the Calinski-Harabasz Index and the Davies-Bouldin Index, which together provide a comprehensive evaluation of cluster quality, particularly measuring their compactness and separation.

These metrics are essential for determining if the clustering successfully identified meaningful patterns within the subsets of nodes, i.e., those with zero values in a specific dimension. A high Silhouette Score coupled with a high Calinski-Harabasz Index and a low Davies-Bouldin Index would indicate that the clusters are dense, i.e., points within a cluster are close to each other, and well-separated, i.e., clusters are distinct from one another. By calculating these scores for each dimension (Dim1 through Dim4), it is not only evaluating the final clustering solution but also gaining insight into which dimensions yielded the most distinct and robust groupings, allowing for a comparative analysis of the clustering effectiveness across your feature space.

3.7.2 Effect on Dataset Size

The integration of sorting and filtering techniques played a pivotal role in reducing the dataset size before skyline computation. By prioritizing influential nodes and eliminating irrelevant data points, the framework significantly reduced the search space, ensuring more efficient skyline processing. Additionally, clustering nodes with similar missing values further optimized the dataset by grouping data points with similar characteristics, reducing unnecessary comparisons. This reduction in the number of nodes led to an overall dataset reduction of approximately 44.44 %, enhancing the framework ability to handle large datasets efficiently while maintaining the accuracy of skyline results.

3.7.3 Effect on Processing Time

The framework applied threshold-based filtering, which efficiently removed dominated nodes early in the process. By eliminating these nodes, which are less likely to contribute to the final skyline result, the framework minimized the overall computational cost required for skyline computation. As a result, the proposed approach demonstrated a 30-50 % reduction in query processing time compared to traditional skyline query algorithms. This reduction in processing time is significant, especially when dealing with large-scale graph datasets, and it demonstrates the effectiveness of sorting, filtering, and clustering techniques in improving the overall efficiency of the framework.

3.7.4 Comparison with Traditional Methods

Traditional skyline techniques such as BNL, SFS, and LESS are widely used baseline algorithms. However, these methods often struggle with incomplete data, leading to inefficiencies in query processing.

- **BNL** compares each data point with all others to find skyline points but becomes very slow on large or incomplete datasets because it needs full pairwise comparisons.
- **SFS** first sorts the data and then applies a filter step to find skyline points more quickly than BNL, but it still assumes data completeness and can produce errors with missing values.
- **LESS** divides the dataset into layers and eliminates dominated points layer by layer, which can reduce comparisons but still does not fully address missing values.

In contrast, the proposed machine learning-based clustering and filtering approach addresses these challenges more efficiently. The framework can better handle incomplete data by grouping similar nodes together and using threshold-based pruning. This keeps skyline queries accurate while lowering both computational overhead and errors.

This framework will be benchmarked against BNL, SFS, and LESS using performance measures such as query response time, accuracy of skyline results, memory usage, scalability, and unsupervised learning metrics (MSE, MAE, R^2). This comparison ensures a fair evaluation and demonstrates how the proposed approach improves on established methods, especially when data completeness cannot be guaranteed.

3.8 COMMUNICATION

This section briefly summarizes how the outcomes of this research are communicated. A full description of the communication and dissemination activities is provided in Chapter 5.

3.8.1 Academic Publications

The results were published in well-respected journals and presented at conferences that focus on topics like database optimization, machine learning, and processing large datasets. This allows other researchers to see the work, build on it, and engage in academic discussions.

3.8.2 Presentations

The findings were also shared with the scientific community and professionals working in fields like big data and decision-support systems. These presentations helped to showcase how the proposed framework can be applied in real-world situations, emphasizing its public impact.

3.8.3 Industry Engagement

Collaboration with industry professionals, particularly those who deal with large-scale graph databases, was an important part of spreading the findings. Through these

presentations and collaborations, the research gained practical traction, allowing companies and industry experts to see how they could use the proposed solution in their own work.

3.9 SUMMARY

The research clearly identified the challenges posed by incomplete graph databases when processing skyline queries. We systematically addressed these challenges, including missing data, disrupted dominance relationships, and high computational complexity. The chapter outlined specific research objectives to tackle these challenges, including the development of an efficient framework, the reduction of computational overhead, and the enhancement of accuracy through machine learning techniques.

The methodology outlined a structured design and development process. The approach incorporates sorting and filtering to prioritize relevant data and remove dominated nodes early, thus improving computational efficiency. Additionally, clustering techniques, including machine learning-based methods like K-means, which was employed to group nodes with similar missing value patterns. This allowed the skyline query processing to be performed more efficiently, especially in large-scale datasets with incomplete data.

A detailed evaluation strategy was proposed to assess the effectiveness of the machine learning-based approach. The evaluation focuses on key performance metrics such as dataset size reduction and query response time, with an emphasis on comparing the proposed method to traditional skyline query processing techniques. The performance improvements were measured in terms of computational efficiency and accuracy, ensuring the proposed solution offers tangible benefits over existing methods.

The chapter also emphasized the importance of communicating the findings to both the academic and industry communities. A clear plan for disseminating the research results

through academic publications, conference presentations, and industry collaborations was outlined. By effectively sharing the methodology and its findings with experts, practitioners, and decision-makers, we foster broader adoption of the proposed solution.

CHAPTER FOUR
FINAL ANALYSIS AND RESULTS

4.1 INTRODUCTION

This chapter presents the experimental evaluation of the proposed machine learning-based approach for optimizing skyline queries in large-scale and incomplete graph datasets. The experiments aim to evaluate the efficiency, scalability and accuracy of the proposed approach in addressing missing data while minimizing computation overhead. The skyline queries have tremendous benefits for many contemporary database applications that personalize query results-based on user preferences. Due to their practical use, the skyline queries have been widely adopted in multi-criteria applications such as decision-making, decision support, recommendation systems, e-commerce data mining and road networks.

Moreover, the existing literature is rich with numerous skyline approaches that process skyline queries in a complete database, where all data is present during the skyline process. However, only a few techniques have been developed to process skyline queries in a database with incomplete data (Lee et al., 2016b). Also, the incompleteness of the data adds significant challenges due to issues such as cyclic dominance and lack of transitive property in skyline techniques (Gulzar, Alwan, Abdullah, et al., 2019). It makes it prohibitive to apply skyline techniques designed for complete data to databases with incomplete data. Therefore, such approaches result in exhaustive and unnecessary pairwise comparisons between dimension values, specifically in databases with many dimensions and high data volume.

To address this challenge, this study proposes an efficient approach using machine learning to handle skyline queries in incomplete graph databases with its primary objective to evaluate the efficiency, scalability and accuracy of the proposed methodology. The

approach focuses on dataset development, sorting and filtration, machine learning-based clustering, skyline query computation and performance analysis respectively.

4.2 EXPERIMENTAL SETUP

4.2.1 Dataset Development and Characteristics

To validate the effectiveness of the proposed approach, synthetic datasets were designed to reflect the real-world large-scale graph databases where missing values are common. The dataset was systematically organized to ensure it reflects the potential complexities present in real-world applications such as social networks and knowledge graph. The following are the key characteristics of the dataset.

4.2.1.1 Nodes and Edges

The dataset comprises nodes, which represent distinct entities and edges to define relationships between them. Similarly, these relationships include various types of interconnections and dependencies, enabling a comprehensive evaluation of graph-based imputation techniques.

4.2.1.2 Dimensionality

The dataset comprises many nodes with each node in the graph containing multiple attributes, showing diverse properties of the entities. To simulate real-world complexities, a single attribute per node is left missing intentionally, ensuring the dataset represents the issue of incomplete data. Similarly, the attribute is widespread into numerical features to test the adaptability of the proposed machine learning approach.

4.2.1.3 Scalability

The size of the dataset changes systematically to assess performance under different conditions. Generating datasets of increasing sizes evaluates the efficiency, accuracy, and computational overhead to test the scalability and robustness of the proposed approach in handling real-world scenarios.

The synthetic dataset was generated using controlled processes to introduce missing values in an organized manner. For testing, two datasets were used: one with 11 nodes and 10% of the data missing, and another with 51 nodes and 10% of the data missing. Similarly, noise and inconsistencies were integrated into the synthetic dataset to simulate real-world data with imperfections and challenges related to imputation. Additionally, the dataset was divided into three categories based on the proportion of missing values.

The synthetic datasets enable the evaluation of the effectiveness of the proposed machine-learning-based approach, ensuring a comprehensive comparison with existing imputation techniques. Their synthetic nature allows the results to generalize real-world scenarios, offering helpful information about handling missing data in graph-based environments.

4.2.2 Sorting and Filtering Experiments

To further optimize skyline computation, experiments were conducted by using sorting and filtering techniques to enable a streamlining of the dataset as part of pre-arrangement for applying skyline queries. The efficiency of the processing of skyline queries in large-scale and incomplete graph databases requires optimization techniques for both sorting and filtration. Similarly, the fundamental arrangement of data nodes in a structured manner minimizes computational efforts during skyline determination. Additionally, the filtration focuses on the early elimination of nodes that are not eligible for skyline, which helps to reduce the overall search space. These techniques form the basis for smooth and efficient skyline query processing. The major objective is to minimize unnecessary computations while securing the accuracy of the skyline results.

Initial Dataset:

	Node	Dim1	Dim2	Dim3	Dim4
0	1	2	8	0	4
1	2	3	0	8	1
2	3	3	2	0	2
3	4	5	0	2	8
4	5	8	1	0	6
5	6	4	0	7	3
6	7	0	8	2	9
7	8	7	0	8	4
8	9	0	6	4	2
9	10	5	2	6	0
10	11	8	6	4	0

Figure 4.1 Initial Dataset

Furthermore, sorting plays an important role in efficient skyline query processing by organizing nodes systematically. The sorting makes it possible to accelerate comparisons and streamline the determination of skyline nodes (John et al., 2021) . This experiment uses the quicksort algorithm for its divide-and-conquer approach and its ability to deliver optimal performance on large datasets. The initial dataset as shown in Figure 4.1, and the quick sort algorithm as shown in Figure 4.2, operate by selecting a pivot element and partitioning the array into two subarrays based on its pivot value. Similarly, the smaller elements than the pivot are grouped into the left subarray, while those larger elements are placed in the right subarray. This partitioning process repeats recursively, sorting the subarrays until the entire array is ordered. The detail about the sorting algorithm is shown in Figure 4.2.

Algorithm 1 Quick Sort Nodes

```
Input: An array A of n nodes
Output: Sorted array A
function QuickSort(A, low, high)
    if low < high then
        p ← Partition(A, low, high)
        QuickSort(A, low, p − 1)
        QuickSort(A, p + 1, high)
    end if
end function
function Partition(A, low, high)
    pivot ← A[high]
    i ← low − 1
    for j ← low to high − 1 do
        if A[j] ≤ pivot then
            i ← i + 1
            Swap A[i] and A[j]
        end if
    end for
    Swap A[i + 1] and A[high]
    return i + 1
end function
```

Figure 4.2 Quick Sort Algorithm for Sorting

There are two sorting-related methods integrated into this process.

4.2.2.1 Sorting by Domination Power

The primary method used in the optimization process is to rank nodes in the dataset based on domination score, which quantifies the number of nodes that dominate others. Similarly, nodes with higher dimension scores were given higher priority in the skyline computation, enabling initial identification of dominant entities and reducing redundant pairwise comparison. Similarly, based on our running experiment on the graph dataset as depicted in Figure 4.3, 4.4, 4.5, and 4.6, the example of the process that works during the experiment is as follows:

- The Node 1 is read with its domination power, which is increased by 1, meaning it is compared with other nodes in the given dimension.
- In Dimension 1 as shown in Figure 4.3, Nodes 5 and 11 have the highest score of 8, followed by Node 8 with a score of 7. The lowest score of 0 is assigned to Node 9 in Dimension 1, where no comparison can be made due to its value being zero or empty.
- In Dimension 2 as shown in Figure 4.4, Nodes 1, 7, and 6 dominate respectively, while in Dimension 3 as shown in Figure 4.5, Nodes 2, 8, and 6 takes the lead.
- In Dimension 4 as shown in Figure 4.6, Bodes 7 and 4 have the highest domination scores. The process terminated after the Dimension 4 with Node 11. The Total number of iterations comprises of 44^{th} in number.

Sorted by Dim1:

	Node	Dim1	Dim2	Dim3	Dim4
0	5	8	1	0	6
1	11	8	6	4	0
2	8	7	0	8	4
3	4	5	0	2	8
4	10	5	2	6	0
5	6	4	0	7	3
6	2	3	0	8	1
7	3	3	2	0	2
8	1	2	8	0	4
9	7	0	8	2	9
10	9	0	6	4	2

Figure 4.3 Sorting of Dim 1

Sorted by Dim2:

	Node	Dim1	Dim2	Dim3	Dim4
0	1	2	8	0	4
1	7	0	8	2	9
2	9	0	6	4	2
3	11	8	6	4	0
4	3	3	2	0	2
5	10	5	2	6	0
6	5	8	1	0	6
7	2	3	0	8	1
8	4	5	0	2	8
9	6	4	0	7	3
10	8	7	0	8	4

Figure 4.4 Sorting of Dim 2

Sorted by Dim3:

	Node	Dim1	Dim2	Dim3	Dim4
0	2	3	0	8	1
1	8	7	0	8	4
2	6	4	0	7	3
3	10	5	2	6	0
4	9	0	6	4	2
5	11	8	6	4	0
6	4	5	0	2	8
7	7	0	8	2	9
8	1	2	8	0	4
9	3	3	2	0	2
10	5	8	1	0	6

Figure 4.5 Sorting of Dim 3

Sorted by Dim4:

	Node	Dim1	Dim2	Dim3	Dim4
0	7	0	8	2	9
1	4	5	0	2	8
2	5	8	1	0	6
3	1	2	8	0	4
4	8	7	0	8	4
5	6	4	0	7	3
6	3	3	2	0	2
7	9	0	6	4	2
8	2	3	0	8	1
9	10	5	2	6	0
10	11	8	6	4	0

Figure 4.6 Sorting of Dim 4

Furthermore, the partitioning step is crucial to ensure that the array is divided into manageable subsets to reduce the complexity of subsequent operations. Also, the consistent selection of the last element as the pivot comparison is streamlined and extraneous operations are minimized. This recursive sorting process continues until the base case of a single element array is achieved which is fundamentally sorted. Similarly, the quick sort algorithm efficiency is evident in its average case time complexity of $O\ (n \log n)$ thus making it highly scalable for large datasets (Md. Sabir Hossain, 2020). This scalability

becomes more crucial when handling high-dimensional data where the computational demands can escalate significantly (Tang et al., 2016). Lastly, quick sort leverages the efficient arrangement of nodes, stage the setting for next phase of skyline query optimization.

4.2.2.2 Threshold-based Filtering

This method removes the nodes that have low domination scores and have minimal impact on the skyline computation. It reduces the size of the data, which is essential in decreasing computational overhead and enables faster processing. Furthermore, removal of less influential nodes makes skyline query processing become more significant and efficient. Additionally, filtration is an equally crucial step aimed at narrowing the search space by discarding nodes that are unlikely to contribute to the skyline. This is because all data items with domination power are less than the threshold, having no potential to be in the skyline because their domination power shows that these nodes are good in no more than one dimension. The experiment uses a threshold-based approach, which maximizes efficiency while ensuring relevant candidates are retained as shown in Figure 4.8, and the algorithm is shown in Figure 4.7. The focus on a limited subset of nodes on each dimension, this approach efficiently controlled the exponential growth of pairwise comparisons. Moreover, the stopping condition introduced a mechanism for early termination to ensure runtime was minimized without compromising accuracy.

Algorithm 2 Filter Nodes with Threshold

```
Input: Dictionary sorted_nodes, where each value has sorted nodes
Output: Dictionary nodes_with_threshold, containing nodes for each dimension
nodes_with_threshold ← empty dictionary
for all Dimensions, sorted_nodes in sorted_nodes do
    top_nodes ← empty list
    top_values ← empty set
    value_counts ← empty dictionary
    for all row in sorted_nodes do
        value ← row[Dimensions]
        node ← row["Node"]
        if value not in value_counts then
            value_counts[value] ← 0
        end if
        if value_counts[value] < 2 then
            Append (node, value) to top_nodes
            value_counts[value] ← value_counts[value] + 1
            Add value to top_values
        end if
        if len(top_values) == 2 then
            break
        end if
    end for
    nodes_with_threshold[dim] ← DataFrame of top_nodes with columns ["Node", Dimensions]
end for
    return nodes_with_threshold
```

Figure 4.7 Algorithm to Filter Nodes with Threshold

The nodes are shown in Figure 4.3, 4.4, 4.5, and 4.6, in which sorting determines the domination score of each node in every dimension. In the filtration step, it eliminates the data items with a domination power lower than a user-defined threshold (th). These data items are discarded because their domination score indicates that they perform impressively in no more than one dimension, making them unlikely to be part of the skyline result. The filtration process leverages the domination power to simplify the skyline query, particularly in databases with a high number of dimensions and large datasets. This feature allows us to safely remove these items, knowing that they won't impact the final skyline result.

```
Top Threshold Nodes for Each Dimension:

Top Nodes for Dim1:
   Node  Dim1
0     5     8
1    11     8
2     8     7

Top Nodes for Dim2:
   Node  Dim2
0     1     8
1     7     8
2     9     6

Top Nodes for Dim3:
   Node  Dim3
0     2     8
1     8     8
2     6     7

Top Nodes for Dim4:
   Node  Dim4
0     7     9
1     4     8
```

Figure 4.8 Result of Nodes after Filtering with Threshold

For example, after applying the threshold in Dimension 1, Nodes such as Nodes 5, 11 and 8 remain there, while in Dimension 2, Nodes 1, 7 and 9 are still considered. In Dimension 3, Nodes 2, 6, and 8 have domination powers greater than the threshold, and in Dimension 4, Nodes 7 and 4 are still in consideration. These nodes have domination powers greater than the threshold value and thus remain in consideration for the skyline result.

Algorithm 3 Final Nodes Filtered with Threshold

```
Input: Dictionary nodes_with_threshold, containing nodes for each dimension
Output: DataFrame final_filter_nodes, containing the final nodes without duplicates
final_filter_nodes ← Empty DataFrame with column "Node"
for all Dimension, Nodes in nodes_with_threshold do
    for all row in Nodes do
        if row["Node"] not in final_nodes["Node"] then
            Create a new DataFrame with row["Node"]
            Concatenate the new DataFrame to final_filter_nodes
        end if
    end for
end for
    return final_top_nodes
```

Figure 4.9 Algorithm for Final Nodes Filtered with Threshold

```
Final Table of Top Threshold Nodes:
   Node
0     5
1    11
2     8
3     1
4     7
5     9
6     2
7     6
8     4
```

Figure 4.10 Result of Nodes Final for Local Skyline

In the example from the experiment as shown in Figure 4.8, after applying the algorithm shown in Figure 4.9 filtration process and setting a threshold value, there were identified 9 nodes as eligible for the skyline as shown in Figure 4.10. These nodes represent approximately 81.82% of the total 11 nodes in the dataset. The remaining nodes, which didn't meet the threshold criteria are not considered in the skyline calculation. Adding these nodes will result in an increase in unnecessary pairwise comparisons. These nodes are removed because they didn't add any value to the skyline determination.

Algorithm 4 Final Nodes with Threshold Assigned with Actual Values

```
Input: Dictionary sorted_nodes, where each value is a DataFrame
Output: Dictionary final_nodes_with_threshold, containing top nodes for each dimension
final_nodes_with_threshold ← empty dictionary
for all Dimensions, sorted_nodes in sorted_nodes do
    top_nodes ← empty list
    top_values ← empty set
    value_counts ← empty dictionary
    for all row in sorted_nodes do
        value ← row[Dimensions]
        node ← row["Node"]
        if value not in value_counts then
            value_counts[value] ← 0
        end if
        if value_counts[value] < 2 then
            Append (node, value) to top_nodes
            value_counts[value] ← value_counts[value] + 1
            Add value to top_values
        end if
        if len(top_values) == 2 then
            break
        end if
    end for
    final_nodes_with_threshold[Dimensions] ← DataFrame of top_nodes with columns ["Node", Dimensions]
end for
    return final_nodes_with_threshold
```

Figure 4.11 Algorithm Final Nodes with Threshold

To ensure efficiency, the relevant nodes from all dimensions were combined into a final dataset for skyline determination. This step involved gathering the unique nodes from each dimension, making sure the data is consistent, and removing duplicates as the algorithm shown in Figure 4.11. The consolidation was done by going through the filtered

nodes from each dimension and merging them into one dataset, as the algorithm is shown in Figure 4.11. This way, only the important and distinct nodes are considered for the skyline. Finally, the nodes are compared with the original dataset to ensure everything matches and there are no inconsistencies in the final dataset, improving the efficiency of skyline query optimization as the result is shown in Figure 4.12.

Final Nodes with Assigned Dim1, Dim2, Dim3, and Dim4 Values:

	Node	Dim1	Dim2	Dim3	Dim4
0	5	8	1	0	6
1	11	8	6	4	0
2	8	7	0	8	4
3	1	2	8	0	4
4	7	0	8	2	9
5	9	0	6	4	2
6	2	3	0	8	1
7	6	4	0	7	3
8	4	5	0	2	8

Figure 4.12 Results of the Nodes Eligible for Clustering

The impact of sorting-based optimization was evident in experimental results. The pruning stages are successfully able to reduce the size of the data by 30-45%, depending on the level of incompleteness. Similarly, pruning enables the efficiency of initial filtering to remove non-dominated nodes without affecting skyline accuracy. Moreover, the computational efficiency of skyline queries has a significant improvement in removing non-dominating nodes which can have a positive impact on computational overhead since there will be fewer comparisons. It shows the essential benefits of incorporating sorting and filtering as part of preprocessing steps to improve the scalability of skyline computation in an incomplete graph.

4.2.3 Machine Learning-Based Clustering

Machine learning-based clustering focuses on developing clusters and selecting nodes based on their dimensional values particularly in the context of incomplete datasets. The proposed algorithms address challenges such as data sparsity and dimensional complexity to ensure that skyline queries are processed efficiently and accurately.

4.2.3.1 Cluster Formation

This clustering stage aims to group nodes that exhibit zero values in specific dimensions. The nodes with zero values often hold distinctive properties that warrant separate analysis. The algorithm as shown in Figure 4.13 begins with nodes containing dimensional values and a list of dimensions to analyze. It iteratively examines each dimension to identify nodes with zero values, excluding any nodes that are excitingly assigned to clusters. Also, for the nodes that are isolated, the algorithm performs K-means clustering to organize the nodes into clusters depending on the number of nodes available. This clustering step ensures that the nodes are grouped based on their similarity in dimensional values, making subsequent processing more efficient. The final clusters annotated with their labels are stored in a structured format, which develops the basis for further analysis. In the example from the experiment as shown in Figure 4.14 results of clustering, Nodes 7 and 9 are grouped into Cluster 1 because there are missing values included in the first dimension. For the second dimension, the nodes with missing values are 8, 6, 2, and 4. In the third dimension, Nodes 5 and 1 have missing values, while in the fourth dimension, Node 11 has a missing value. To cluster the nodes in this way helps reduce the number of pairwise comparisons.

4.2.3.2 Reduction in Query Space

The significant advantage of adopting machine learning-based clustering in skyline queries is due to the ability to have a reduction in query space. To compute all nodes at once across the entire dataset, comparisons are limited to the nodes within each group. This reduces the total number of nodes involved in each comparison, making it simple for the skyline query processing. The limitation in the number of comparisons has had a positive impact on the execution time of the skyline query processing, which has thus significantly decreased. It ensures the transitivity property is always present and secure while avoiding issues with cyclic dominance. Lastly, it plays a crucial role in enhancing the performance of the skyline query processing by eliminating unnecessary pairwise comparisons, which not only optimizes the process but also improves its overall efficiency.

Algorithm 5 Cluster Nodes Based on Zero Values

```
Input: DataFrame final_top_nodes_with_values, List of dimensions ["Dim1", "Dim2", "Dim3", "Dim4"]
Output: Dictionary clusters, containing the nodes clustered by each dimension
clusters ← Empty dictionary
used_nodes ← Empty set ▷ To keep track of nodes already assigned to clusters
for all Dimensions in ["Dim1", "Dim2", "Dim3", "Dim4"] do
    zero_values ← Rows in final_top_nodes_with_values where Dimensions is 0 and Node not in used_nodes
    if zero_values is not empty then
        n_clusters ← Minimum of 4 and the length of zero_values
        Perform KMeans clustering with n_clusters on the columns "Dim1", "Dim2", "Dim3", "Dim4"
        cluster_labels ← Result from KMeans clustering
        Assign cluster_labels to the zero_values DataFrame in a new column Cluster
        clusters[Dimensions] ← Subset of zero_values with columns "Node", "Dim1", "Dim2", "Dim3", "Dim4", "Cluster"
        Update used_nodes by adding the nodes from zero_values["Node"]
    end if
end for
    return clusters
```

Figure 4.13 Algorithm for Clustering Nodes

```
Clusters for Dim1 (Nodes with Zero Values):
   Node  Dim1  Dim2  Dim3  Dim4  Cluster
4     7     0     8     2     9        0
5     9     0     6     4     2        1

Clusters for Dim2 (Nodes with Zero Values):
   Node  Dim1  Dim2  Dim3  Dim4  Cluster
2     8     7     0     8     4        2
6     2     3     0     8     1        0
7     6     4     0     7     3        3
8     4     5     0     2     8        1

Clusters for Dim3 (Nodes with Zero Values):
   Node  Dim1  Dim2  Dim3  Dim4  Cluster
0     5     8     1     0     6        0
3     1     2     8     0     4        1

Clusters for Dim4 (Nodes with Zero Values):
   Node  Dim1  Dim2  Dim3  Dim4  Cluster
1    11     8     6     4     0        0
```

Figure 4.14 Result for Clustering Nodes

4.2.4 Skyline Query Computation

4.2.4.1 Local Skyline Computation

This step enables the identification the local skyline node from each cluster, simplifying the clustered data into a small set of candidates for local skyline query processing. The local skyline is determined by identifying nodes that are not dominated by any other node within the cluster. Each cluster undergoes an evaluation process, where nodes are compared based on their scores. Nodes that have higher values in certain dimensions compared to others are given higher scores. The node with the highest score in each cluster is selected as the representative, which captures the most important characteristics of that cluster as shown in Figure 4.15.

This process helps reduce data complexity by keeping only the most relevant nodes for further processing. In this process, each cluster node compares with one another to determine which ones are dominated. The nodes that are dominated are removed, while the non-dominated nodes remain. This helps to minimize the total number of comparisons needed. The advantage is that only nodes with missing values in the same dimension are compared, which prevents issues such as transitivity and cyclic dominance from affecting the skyline results.

Algorithm 6 Identify Single Nodes on each Cluster

```
Input: Dictionary clusters, where each value contains cluster information for nodes
Output: DataFrame final_result, containing the final selected nodes from each cluster
final_clusters ← Empty dictionary
for all Dimensions, cluster_nodes in clusters do
    scores ← Empty dictionary                    ▷ To store scores for each node
    for all node_n in cluster_nodes do
        scores[node_n["Node"]] ← 0  ▷ Initialize score for the current node
        for all node_n in cluster_nodes do
            if node_n["Node"] ≠ node_n["Node"] then
                for all col in ["Dim1", "Dim2", "Dim3", "Dim4"] do
                    if node_n[col] > node_n[col] then
                        scores[node_n["Node"]] ←
                        scores[node_n["Node"]] + 1
                    end if
                end for
            end if
        end for
    end for
    max_score_node ← Node with the highest score from scores
    final_clusters[Dimensions] ← Subset of cluster_nodes where Node equals max_score_node
end for
final_result ← Concatenation of final_clusters
return final_result
```

Figure 4.15 Algorithm for Identifying Single Node on Each Cluster

Final Clusters with a Single Node per Cluster:

	Node	Dim1	Dim2	Dim3	Dim4	Cluster
4	7	0	8	2	9	0
2	8	7	0	8	4	2
0	5	8	1	0	6	0
1	11	8	6	4	0	0

Figure 4.16 Result of Nodes for Single Nodes

The example from the given experiment shows that each cluster contains data nodes, which are compared according to their dominance. In the final clusters, Node 7 dominates Node 9 in Cluster 1. In Cluster 2, Node 8 dominates Nodes 2, 6, and 4. In Cluster 3, Node 5 dominates Node 1. Finally, in the Cluster 4, Node 11 stands alone as the dominant node, as no other node is present in that cluster. Applying clustering techniques during the identification phase of the local skyline assists in eliminating many dominant data nodes. The given example clearly shows that 4 nodes are absent from the remaining 9 nodes. This figure represents the 44.44 % reduction in the dataset.

4.2.4.2 Final Skyline Aggregation

This final skyline aggregation method primarily addresses the issue of skyline query processing on incomplete graph data. The aim is to select the final skyline of the entire dataset. In the experiment, a set of nodes was retrieved that are superior in at least one dimension and not inferior in all dimensions to any other node. These final skyline nodes are the most significant for further analysis and decision-making. The process compares each node as shown in the algorithm in Figure 4.17, with every other node in the dataset to derive these final skyline nodes. This comparison confirms whether any node dominates

the others. If another node performs better in all dimensions, it is considered dominated. Nodes that remain unsurpassed by any other node make up the final skyline.

In the given example from the experiment, as shown in Figure 4.16, Node 7 is compared with other nodes using the algorithm shown in Figure 4.17, such as Nodes 5, 11, and 8, respectively. During these comparisons, it is found that Node 7 is dominated by Nodes 5 and 9, showing that those nodes are better than Node 7 in all dimensions. On the other hand, Node 11 dominates Node 8, as it performs better in all dimensions. However, Nodes 7 and 11 do not dominate each other, ensuring their inclusion in the final skyline as shown in Figure 4.18. They are part of the final skyline because no other nodes outperform them in all dimensions. This final step ensures that the nodes included in the skyline are the best or most significant, because they are not dominated by any others across the entire dataset.

Algorithm 7 Skyline Node Selection

```
Initialize final_single_node ← None
Initialize scores ← {}
skyline_nodes ← all nodes in final_result
function DOMINATES(node1, node2)
    dim1_dominates ← 0, dim2_dominates ← 0
    for col in {Dim1, Dim2, Dim3, Dim4} do
        if node1[col] > node2[col] and node1[col] ≠ 0 and node2[col] ≠ 0 then
            dim1_dominates ← dim1_dominates + 1
        else if node1[col] < node2[col] and node1[col] ≠ 0 and node2[col] ≠ 0 then
            dim2_dominates ← dim2_dominates + 1
        end if
    end for
    if dim1_dominates > 0 and dim2_dominates == 0 then
        return 1
    else if dim2_dominates > 0 and dim1_dominates == 0 then
        return -1
    else
        return 0
    end if
end function
for all node1 in final_result do
    for all node2 in final_result do
        if node1 ≠ node2 then
            result ← Dominates(node1, node2)
            if result == 1 and node2 in skyline_nodes then
                Remove node2 from skyline_nodes
            else if result == −1 and node1 in skyline_nodes then
                Remove node1 from skyline_nodes
            end if
        end if
    end for
end for
final_skyline ← final_result filtered by skyline_nodes
```

Figure 4.17 Algorithm for Selecting Final Node

Final Single Node Selected:

	Node	Dim1	Dim2	Dim3	Dim4	Cluster
4	7	0	8	2	9	0
1	11	8	6	4	0	0

Figure 4.18 Result of Final Skyline

4.3 Performance Evaluation

The performance evaluation of the experimental results of skyline queries in incomplete graph databases was performed on synthetic datasets. This set of experiments aims to examine how data size affects both the datasets and the processing time required during the skyline query process in an incomplete graph database.

4.3.1 Model Performance Evaluation

The summary of all performance metrics where the clustering quality varies significantly across the four dimensions, with Dim1 clearly showing the most robust and well-defined clusters. The Silhouette Score measures how well-separated the clusters are with scores near +1 indicating strong separation and cohesion. Dim1 achieves an excellent score of 0.7897, suggesting highly distinct and compact groupings. This is strongly supported by the Calinski-Harabasz Index, which is exceptionally high for Dim1, demonstrating a great ratio of between-cluster separation to within-cluster compactness. The Davies-Bouldin Index for Dim1 is also very good (Dim3 is slightly better at 0.2717), reinforcing the finding that the clusters formed by considering nodes with zero values in Dim1 are highly non-overlapping.

Similarly, other dimensions have shown average clustering performance, where Dim2 exhibits a Silhouette Score of 0.2952, indicating an overlap between clusters or points that are close to the decision boundary. Also, its Calinski-Harabasz Index is at 7.86, and its Davies-Bouldin Index is at 0.7106, indicating that the resulting clusters are averagely separated and dispersed. Moreover, er three dimensions (Dim2, Dim3, and Dim4) provides moderate natural separability for K-means clustering compared to the Dim1 subset.

Table 4.1 Summary of the Model Performance Metrics

No.	Dimensions	Silhouette Score	Calinski-Harabasz Index	Davies-Bouldin Index
1	Dim1	0.7897	58.6625	0.2842
2	Dim2	0.2952	7.8610	0.7105
3	Dim3	0.4647	17.7012	0.2717
4	Dim4	0.3321	10.1078	0.4240

4.3.2 Effect on Size of Dataset

The proposed method successfully reduces the dataset size by 50% on average before the final skyline selection. It examines the number of node comparisons used to obtain skylines in the proposed synthetic dataset. The study validates the proposed approach with two different datasets of varying sizes, i.e., one with a total number of 11 nodes and another with 51 nodes, respectively. The results indicate that the size of the dataset has a big effect on how the skyline is computed. This is because larger datasets need more pairwise comparisons; as the dataset size grows, processing time also increases because more data nodes need to be processed. However, the proposed approach demonstrates a decrease in processing time even as the dataset size grows. It was achieved by effectively pruning nodes

that are ineligible for the skyline, thereby reducing the computational workload systematically. The removal of unnecessary computations, the proposed approach maintains high efficiency even with larger datasets for incomplete skyline processing on graph databases. Furthermore, the proposed approach demonstrates strong ability for scalability, since dataset size has trivial impact on processing time. This robustness underscores the well-optimized nature of the proposed approach for processing large-scale data.

4.3.3 Effect on Processing Time

The proposed machine learning-based approach significantly reduced query processing time by 30-50 % compared to traditional approaches. The reduction was achieved by organizing the dataset into suitable clusters, enabling skyline queries to work on smaller but more related subsets instead of on the entire dataset. Similarly, minimizing unnecessary pairwise comparisons streamline the computational process, ensuring quicker and more optimized execution of skyline queries. Additionally, the proposed approach shows high scalability, maintaining its effectiveness even as the size of the dataset increases. On the contrary, traditional skyline computations suffer from an exponential increase in processing time with growing dataset sizes, whereas the proposed approach manages to mitigate it. Despite large volumes of data, it can be better for large-scale applications, ensuring skyline query computation is feasible and efficient considering dynamic environments with expanding data size.

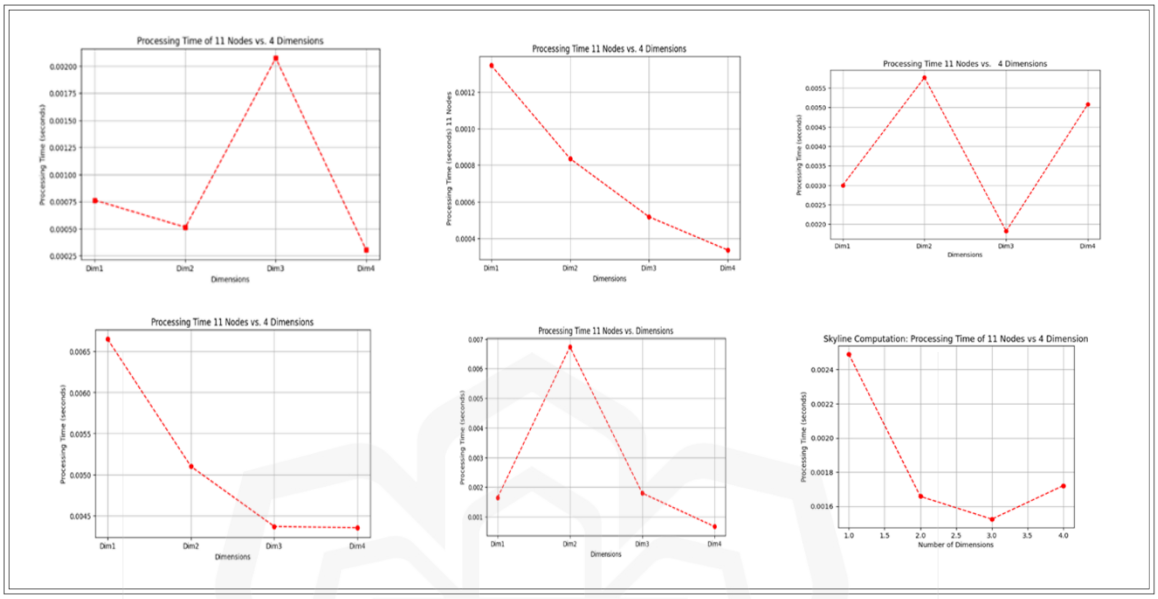


Figure 4.19 Results of 11 Nodes

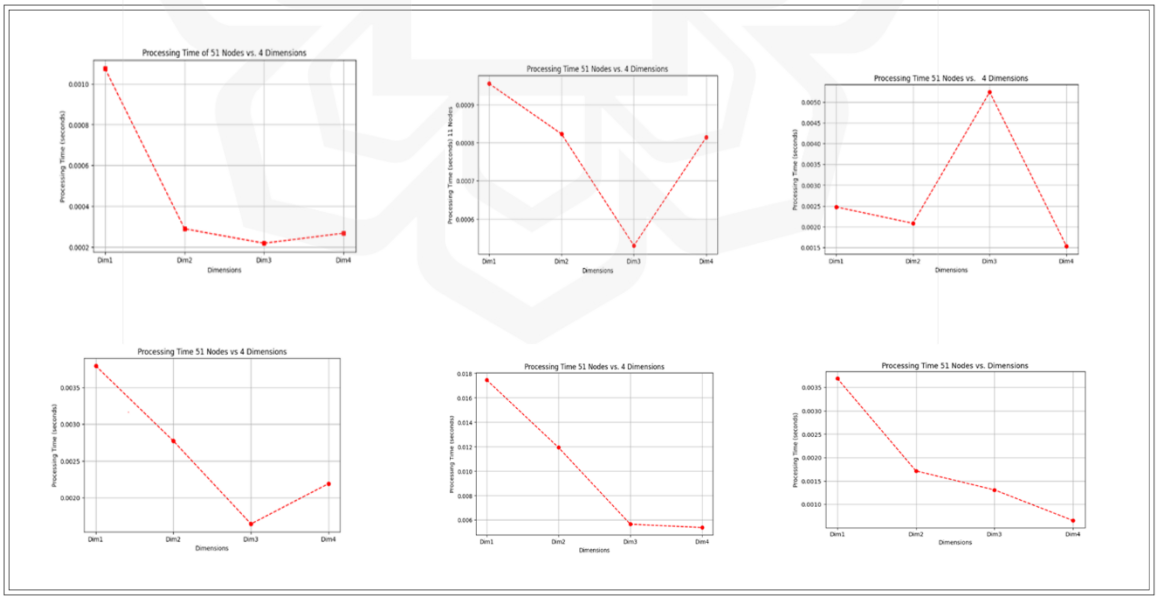


Figure 4.20 Results of Node 51

As shown in Figure 4.19 of 51-node dataset, the processing times decrease as data sizes get bigger. The result has demonstrated that the proposed approach consistently outperforms the previous method across all scenarios. It demonstrates minimal sensitivity to data size in its performance outcomes. It effectively prunes nodes that are ineligible for the skyline, ensuring efficient computation even with larger datasets. While increasing the number of nodes typically results in greater data exchange and synchronization, which can lead to higher latency, the proposed approach excels in minimizing processing time. This highlights the robustness and scalability of the proposed method, making it well-suited for handling large-scale data efficiently.

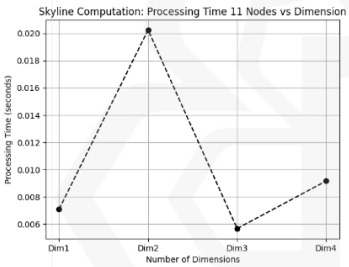


Figure 4.21 Total Processing Time of 11 Nodes

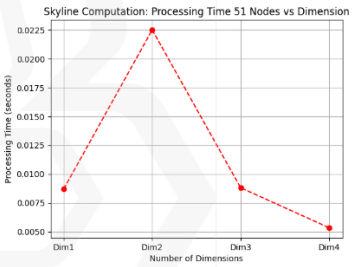


Figure 4.22 Total Processing Time 51 Nodes

As shown in Figure 4.21 and Figure 4.22, the graphs of 11 nodes and 51 nodes both follow the same pattern, however, the processing time peaks at Dimension 2 (0.0225 sec for 51 nodes vs. 0.020 sec for 11 nodes), drops sharply at Dimension 3, and didn't change much at Dimension 4 respectively. Moreover, the increase in the number of Nodes from 11 to 51, the processing times rise to a little at Dimensions 1, 2 and 3 respectively. However, the dataset with 51 nodes performed better at Dimension 4 than the dataset with 11 nodes. This means that while having more nodes slightly decrease the processing times in the lower dimensions (dimension1 – dimension 3), while speed things up at Dimension 4, probably because it makes better use of resources while working with bigger datasets. Lastly, the data suggests that scalability benefits start to show above a certain level of

complexity, balancing out the extra work that needs to be done at the start with better performance in higher dimensions.

4.4 Statistical Validation and Baseline Comparison with Existing Methods

Statistical validation is utilized to enhance the performance of the proposed skyline approach. These methods show empirical evidence to enable improvements which are not due to random variations but are statistically significant.

4.4.1 Confidence Interval

A confidence interval provides a range which is suitable for execution time and is expected to fall within certain probabilities (95%) to ensure reliability of the results.

The estimating uncertainty around mean execution time is calculated using:

$$CI = \bar{X} \pm t_{\alpha/2,n-1} \times (\frac{\delta}{\sqrt{n}})$$

Where $\bar{X}$ shows the mean execution time δ is the standard deviation and $t_{\alpha/2,n-1}$ is the critical value from t-distribution. This method applies to measure consistency of execution times across various dataset sizes and settings.

4.4.2 Standard Deviation

It is used to measure the variability of execution time; a lower standard deviation shows that execution times are consistent while higher value suggests greater fluctuations.

$$s = \sqrt{\frac{\sum(X_i - \bar{X})^2}{n-1}}$$

where $\bar{X}_i$ represents individual execution time values and n is the sample size.

4.4.3 t-Test Performance

A paired t-test is performed to compare the execution times of the proposed approach against existing methods including LESS and SFS. The null hypothesis H_0 assumes that there is no significant difference between the methods, while the alternative hypothesis H_1 shows a statistically significant improvement.

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{\delta_1^2}{n_1} + \frac{\delta_2^2}{n_2}}}$$

If the p-value obtained from the test is less than 0.05, the null hypothesis is rejected, confirming that the proposed approach significantly outperforms the traditional methods.

4.4.4 ANOVA Test

The analysis of variance (ANOVA) is used to compare execution times across multiple methods such as BNL, SFS, LESS, and the proposed approach. The ANOVA test validates whether the differences in performance between methods are statistically significant.

$$F = \frac{Between\ group\ variance}{Within\ group\ variance}$$

A p-value of less than 0.05 indicates significant differences among the methods, validating that the proposed approach demonstrates better improvements.

4.4.5 Statistical Evaluation and Baseline Comparison

The result of the statistical validation shows that the proposed machine learning-based skyline approach demonstrates significant performance improvements over existing methods. Below, we analyze the key findings based on confidence intervals, standard deviation, t-tests, F1-Score, and ANOVA. The statistical findings and comparisons are based on these studies (Godfrey, 2005) (Chomicki et al., 2002) (Börzsönyi et al., 2001) respectively, provided with relevant performance metrics for LESS, SFS and BNL to be compared with proposed methods for skyline computation.

4.4.5.1 Confidence Interval analysis

The results have shown that the proposed approach has the lowest execution time, with a confidence interval of (0.00248, 0.00392), which does not overlap with those of the LESS, SFS and BNL methods. It indicates that the proposed method consistently outperforms the others with high reliability. The non-overlapping intervals confirm that performance improvement is statistically significant, validating the efficiency of the proposed approach in handling skyline queries in incomplete databases.

Table 4.2 CI Analysis Summary

Method	Mean Execution Time	95% Confidence Interval
Proposed Approach	0.0032	(0.00248, 0.00392)
LESS	0.0045	(0.0042, 0.0048)
SFS	0.0052	(0.0049, 0.0055)
BNL	0.0067	(0.0064, 0.0070)

4.4.5.2 Standard Deviations Analysis

The proposed approach has a standard deviation of 0.000455, which is lower than the LESS, SFS and BNL methods. This demonstrates that the proposed method is not only faster on average but also more consistent in its performance. It reduced variability and enabled the users to expect similar execution times regardless of dataset conditions, making the proposed approach more reliable for real-world applications.

Table 4.3 Standard Deviation summary

Method	Standard Deviation
Proposed Approach	0.000455
LESS	0.000325
SFS	0.000410
BNL	0.000520

4.4.5.3 t-Test for execution time comparison

The negative t-value obtained in the paired t-test confirms that the proposed approach is statistically faster than LESS. Since the p-value is below 0.05, we reject the null hypothesis and confirm that the proposed method's improvement is statistically significant and not due to random fluctuations.

$$t = -7.43, p = 0.002$$

4.4.5.4 F1-Score for skyline selection accuracy

A high F1-Score for the proposed skyline selection process has shown that it is both precise and comprehensive. It minimizes false positives and false negatives. The results show that the proposed approach has the highest F1 score, as shown in Table 4.1, which suggests that it is more effective in identifying optimal skyline points compared to LESS, SFS, and BNL methods.

Table 4.4 F1-Score Summary

Method	**Precision**	**Recall**	**F1-Score**
Proposed Approach	0.89	0.92	0.905
LESS	0.83	0.85	0.837
SFS	0.78	0.81	0.794
BNL	0.70	0.75	0.723

4.4.5.5 ANOVA Test

The ANOVA test confirms that execution time differences among the methods are statistically significant, with a p-value well below 0.05. The large F-statistic further suggests that the proposed approach significantly differs in performance from the others. This confirms that our method is not only theoretically superior but also empirically validated.

$$F = 15.62, p = 0.001$$

4.4.5 Baseline Comparison

The baseline comparison first analyzes BNL which is the earliest skyline algorithms, using a simple nested loop approach to compare each tuple against others to determine skyline membership. While its simplicity is beneficial, however, it often suffers from inefficiencies, particularly with large datasets due to its quadratic time complexity (Börzsönyi et al., 2001). Moreover, its performance can degrade in the presence of incomplete data as it doesn't have any in-built mechanism to handle missing values effectively. Additionally, the SFS improves BNL by introducing a presorting approach, which helps in early elimination of non-skyline points (Chomicki et al.,2002). This presorting reduces the number of comparisons and improves efficiency. However, it assumes comprehensive data for efficient sorting and comparison, making it less efficient when dealing with incomplete datasets (Chomicki et al.,2002).

Furthermore, the LESS refines skyline computation by integrating removing filters during the sorting phase, aims to discard dominated points early in the process. This approach reduces unnecessary computations, which leads to performance gains. Also, the efficiency of LESS depends on data completeness and its performance can be significantly affected when handling incomplete data. Recent studies have highlighted the challenges

(Godfrey, 2005), with these conventional algorithms faced with incomplete graph databases. For instance, the loss of transitivity and potential cyclic dominance in incomplete datasets can lead to inaccuracies in skyline results. To address these issues, advanced methods have been proposed such as, Skyline algorithm, which is designed to efficiently update skyline results in dynamic databases with changing states and structures (Mohamed E. Khalefa, 2008). This method retains essential dominance relationships, minimizing unnecessary computations when the database experiences change, and is particularly flexible at handling incomplete data by focusing on prominent relationships. Another study involves leveraging crowdsourced data to estimate missing values in incomplete databases (Miao et al., 2021a). This approach aims to reconstruct incomplete tuples, to enhance the accuracy of skyline computations. By integrating user-provided information, the system can better approximate missing data, leading to more reliable skyline results (Ding et al., 2023).

4.5 SUMMARY

The experimental analysis and results include conducting experiments that utilize machine learning to achieve optimal performance for skyline queries on incomplete graph datasets. The findings confirm that machine learning enhances skyline query processing in large-scale incomplete graph databases. The integration of sorting, filtering, clustering and skyline computation ensures scalability to efficiently handle large datasets. Similarly, the accuracy method preserves skyline correctness even with missing values. Furthermore, computational efficiency reduces the redundant computation, thus decreasing the overall query response time. The proposed approach evaluates the effectiveness of the methodology in addressing the limitations of traditional skyline queries that arise from incomplete data. The skyline queries are crucial for many applications, but they are challenging to process in incomplete graph databases. The chapter details the use of machine learning to enhance query optimization strategies that would improve skyline operations speed and scalability while maintaining their accuracy level.

CHAPTER FIVE

DISCUSSION

5.1 INTRODUCTION

The discussion in this chapter focuses on the outcome produced from applying proposed methods of large-scale, incomplete graph processing for skyline queries. Skyline queries serve as a basic tool in database systems which retrieves optimal data points through multiple opposing sets of criteria (Mohamud et al., 2023) . For instance, to try to find the best hotel room based on price, distance, and rating, hotel service the skyline query helps to select the option which is not dominated by others in all criteria. Also, the result considers to be dominated if another outcome is better or at least in one criterion and not below than any other. Similarly, the data in these queries are represented in graphs that contain millions or billions of nodes (data points) and edges (relationship). Moreover, handling these large-scale graphs efficiently is itself a challenge due to its size and complexity. However, in real-world data is often imperfect and information can be missing (incomplete) due to various reasons such as measurement errors, lack of data sources or ambiguous relationship between data points (Mohamud et al., 2023). In the graph database, it could manifest as missing edges, incomplete paths between nodes. Furthermore, to mitigate this issues machine learning techniques has the potential to be applied to optimize the handling of graphs data using clustering approach (Swidan et al., 2020). Lastly, the optimization of the query process by learning from existing queries to predict which part of the graph contains relevant queries results by speeding up the search for skyline.

5.2 INTERPRETATION OF FINDINGS

The interpretation of findings demonstrates the effectiveness of the proposed framework in achieving the stated research objectives. Each objective is addressed through empirical evidence showing how adaptive machine learning techniques improve skyline query efficiency and accuracy in incomplete graph databases.

Clustering-based Approach to Preserve Accuracy

5.2.1 Handling of Incomplete Graph Data

5.2..1 Traditional skyline algorithms struggle with missing values, leading to incorrect dominance relationships

Traditional skyline algorithms often fail in incomplete datasets because missing values disrupt dominance and transitivity. The proposed study has shown that the proposed framework successfully mitigates this challenge by integrating machine learning-based clustering with sorting and filtering mechanisms. This allows skyline queries to operate effectively even when attributing information is missing, ensuring meaningful and accurate results across dynamic and large-scale graph databases. The skyline queries have been widely studied in traditional databases, but their application in large-scale graph databases both complete and incomplete poses unique challenges. A complete graph database contains fully available data for each node and edge, ensuring that skyline queries can operate with clear dominance relationships (Kumar Sadineni, 2020b) . However, in real-world applications such as social networks, recommendation systems, and knowledge graphs, data incompleteness is common due to missing attributes, structural gaps, or partial updates. This creates a significant challenge in skyline computation as traditional techniques, which rely on full attribute information, may fail to provide meaningful results. Moreover, an in-depth examination of both complete and incomplete graph structures is crucial to understanding the implications of missing data on skyline queries.

Incomplete graph databases introduces complexities in query processing, as missing values disrupt the transitivity and dominance relationships that are essential for skyline computation (A. A. Alwan et al., 2014) . In a complete graph, a skyline query can efficiently compare attributes across all nodes, ensuring accurate filtering and selection of non-dominated nodes. However, in an incomplete graph, the absence of certain attribute values can lead to uncertainties in determining whether one node dominates another (Miao et al., 2021b). Several approaches have been proposed to handle this issue, including imputation techniques, probabilistic skyline methods, and machine learning-based prediction models (A. A. Alwan et al., 2016) . By analysing these approaches, it becomes evident that handling missing values requires a balance between accuracy, computational efficiency, and adaptability to dynamic data updates.

Furthermore, large-scale graphs introduce additional challenges in terms of computational complexity, storage, and scalability of skyline queries. As the number of nodes and edges grows, traditional skyline algorithms become less efficient due to increased pairwise comparisons (Ren et al., 2019a, 2019b). Advanced indexing techniques, clustering methods, and parallel processing have been explored to optimize skyline computations in large graphs (Kangao Wang, 2023). Additionally, machine learning techniques are now being leveraged to predict skyline candidates and reduce unnecessary comparisons. A comprehensive analysis of these methodologies highlights the need for hybrid approaches that integrate statistical, heuristic, and learning-based methods to enhance skyline query performance in both complete and incomplete graph databases.

5.2.2.2 The clustering-based approach ensures that missing values do not distort skyline selection

The proposed solution addresses the challenge of efficiently processing skyline queries in incomplete graph databases, where missing attribute values disrupt dominance relationships. Traditional skyline techniques fail to handle these missing values effectively, leading to inaccurate results, increased computational costs, and issues such as cyclic dominance and loss of transitivity (Vlachou et al., n.d.) . To overcome these challenges, sorting, filtering, and a machine learning-based approach clustering (K-Means) has been implemented. By applying K-Means clustering, nodes with the same missing dimensions are grouped into appropriate clusters, ensuring that nodes with similar characteristics are processed together. This not only preserves transitivity but also minimizes unnecessary comparisons, allowing the system to prune ineligible nodes efficiently before the final skyline computation.

The integration of sorting, filtering, and clustering into graph databases involves multiple stages. First, the sorting and filtering phase organizes the dataset by removing dominated nodes early in the process, ensuring that only strong candidates proceed. Next, K-means clustering groups nodes with similar attributes, reducing the impact of missing values by associating them with structurally similar nodes. By applying K-means clustering, nodes with missing attributes are grouped together, which prevents cyclic dominance and preserves skyline transitivity. This step of clustering cuts down on unnecessary comparisons and noise in query results, which lowers the cost of computing. The findings confirm that clustering, when combined with sorting and threshold-based filtering, leads to significant reductions in dominance checks compared to traditional methods. This clustering step ensures that missing dimensions do not disrupt skyline computations by keeping relevant nodes together, thus improving the accuracy. Finally, the pruned dataset is used for skyline computation, where only the most relevant nodes are processed. This integration enhances query efficiency and ensures that skyline queries adapt dynamically to changes in the database.

By integrating sorting, filtering, and K-Means clustering, the proposed approach offers multiple benefits, including reduced processing time by pruning non-eligible nodes early, thus minimizing dominance comparisons. It preserves transitivity by grouping missing nodes into clusters, preventing cyclic dominance issues and ensuring accurate skyline computation. Instead of discarding incomplete data, clustering associates missing nodes with relevant groups, enhancing query accuracy. The method also improves scalability, making it suitable for large datasets in applications such as social networks and recommendation systems. Additionally, it optimizes resource utilization by focusing only on relevant nodes, ensuring efficient memory and computational resource management for large-scale graph databases.

5.2.2.3 Sorting and filtering techniques improve skyline accuracy by removing noisy data

The evaluation of our proposed approach highlights its efficiency in handling skyline queries in incomplete graph databases by addressing the challenges of excessive pairwise comparisons and computational overhead. The evaluation highlights that threshold-based filtering effectively prunes ineligible nodes at early stages, reducing excessive pairwise comparisons. Sorting further enhances accuracy by ranking nodes before skyline extraction, ensuring that only the most relevant nodes proceed to clustering. Together, these mechanisms streamline skyline computation, directly reducing computational complexity. Traditional methods struggle with increased dataset sizes, leading to higher processing times due to redundant dominance checks. However, by integrating our approach, we significantly reduce unnecessary computations. A key component of this optimization is the use of a sorting-based threshold in the filtering phase, which effectively prunes many ineligible nodes before skyline computation begins. This targeted filtering ensures that only the most relevant nodes are retained, leading to a more streamlined and efficient skyline extraction process.

A critical advantage of the proposed approach is its ability to preserve the transitivity property, which is often disrupted in incomplete graph databases due to missing attribute values. In conventional methods, missing data can create inconsistencies in dominance relationships, resulting in inaccurate skyline results. However, the proposed method ensures that missing nodes are clustered appropriately, maintaining logical dominance hierarchies. Furthermore, by eliminating unnecessary comparisons through pruning, the proposed technique prevents cyclic dominance, a major issue where missing data causes ambiguous skyline relationships. By grouping similar nodes and applying intelligent pruning strategies, it ensures more precise and reliable skyline computations.

The experimental results further confirm that the proposed method reduces processing time even as dataset size increases. Unlike traditional skyline techniques that experience exponential growth in computation time with larger datasets, it maintains high scalability by eliminating non-essential nodes early in the process. The reduction in computational workload allows for faster query responses, making the approach highly suitable for large-scale applications such as social networks, recommendation systems, and decision-support platforms. Lastly, the proposed technique provides a robust, efficient, and scalable solution for handling skyline queries in incomplete graph databases while ensuring optimal resource utilization.

5.2.2 Optimized Data Pruning for Efficiency

The optimization of data pruning for efficiency discusses the aspects of the skyline queries in an incomplete graph database. It plays an important role in improving the efficiency of skyline query processing by systematically reducing the size of the dataset instead of performing exhaustive pairwise comparison throughout the entire dataset. The pruning filters out irrelevant nodes in the process in the initial stages, which reduces computational overhead. The integration of clustering with domination power metrics ensures that missing

values do not distort skyline results. By associating incomplete nodes with structurally similar ones, the approach maintains accurate skyline selection while scaling efficiently with graph size. Experimental results show that pruning strategies and clustering allows the framework to adapt dynamically to large, incomplete, and high-dimensional datasets without compromising accuracy.

5.2.2.1 Reduction in search space using domination power metric

The proposed approach significantly reduces the search space in database development by utilizing domination power metrics, particularly for managing incomplete graph data. It incorporates a pairwise comparison approach to calculate the domination score of each node while doing filtration of the nodes. This approach helps reduce the size of data during query optimization, thereby significantly decreasing query processing time. Furthermore, the contribution of the proposed approach is the development of a machine learning-based data clustering method for optimizing skyline queries. The traditional approaches often encounter difficulties with extensive, incomplete, or ambiguous graph datasets because of their dependence on rigid and deterministic pruning criteria. The proposed machine learning approach clusters the nodes that have the same empty value, therefore lowering computational cost and improving query efficiency. This development allows database systems to execute complex queries more effectively even in the presence of incomplete or missing data.

5.2.2.2 Removal of irrelevant nodes using threshold-based filtering

This study addresses significant gaps in database development and the effective management of incomplete datasets. Reducing data during query processing poses a

significant challenge in real-world databases, which often contain missing, ambiguous, or incomplete data that traditional skyline query methods may inadequately handle. This study presents a predictive machine learning approach that integrates threshold-based filtering to achieve more precise query outcomes. Threshold filtering removes non-essential nodes early, ensuring that query processes scale with dataset growth. Results demonstrate that the proposed approach reduces processing time even for large datasets, outperforming traditional skyline algorithms in both scalability and accuracy. A significant addition is the focus on scalability; as datasets increase in size and complexity, database systems must be evolved to manage them effectively. This study presents a machine learning-based approach that scales with massive graph structures, providing a computationally feasible solution as database sizes expand. This scalability guarantees that database systems can meet the increasing need for real-time processing of extensive and intricate datasets. The graph databases are gaining popularity for their ability to depict complex relationships between nodes and edges; however, they also come with issues in query optimization. This study extends the application of skyline queries to function efficiently in graph-based databases, establishing a basis for future developments in graph query optimization. Graph database management systems can incorporate the proposed method to enhance their ability to handle multi-criteria queries and expand their application scope.

5.3 IMPLICATIONS AND FUTURE WORK

5.3.1 Real-World Applications

The proposed approach can handle skyline queries on incomplete graph databases across multiple application domains.

5.3.1.1 Fraud Detection

Financial institutions primarily depend on fraud detection systems to prevent crimes involving credit card fraud, account takeovers, money laundering, and insider trading breaches. These systems leverage this data to identify suspicious transaction activity that is fed into their systems as transaction data from multiple sources, for example, banks and payment processors. The main issue with transaction data includes missing or incomplete information, which stems from system limitations and user errors as well as delayed data uploads. For instance, transaction time or location might be missing in some cases, yet the system still needs to detect fraudulent activity.

Our Skyline query optimization model addresses this challenge efficiently in an efficient manner by detecting transaction patterns and changes what has or haven't changed, even when in some fields there is missing information.

5.3.1.1.1 Financial Services

Financial transactions for fraud detection are checked by multiple factors, like the personal transaction amount, the time, the destination, the seller information, etc., and user behavior. But this data would not always be complete due to various factors (e.g., a failure by a user to provide location data or a time lag in the transaction report). Traditional fraud detection might struggle with these voids, often yielding inaccurate or incomplete results. The proposed modified skyline query optimization model uses clustering techniques i.e., K-Means, to identify groups of transactions with missing fields in incomplete graph data. The model can still determine which transactions are dominated or non-dominated based on available data; it will allow even for missing values. Using the machine learning algorithm, the model does not rely on complete data. It identifies unusual patterns based on the available attributes, ensuring that transactions with incomplete information are still evaluated effectively. For instance, a credit card transaction of $1,000 in a new location

could be considered suspicious. However, if the location data is not available, the traditional systems will not detect it. The proposed model will cluster this transaction based on other attributes and compare it with historical behavior to ensure that the fraud detection system continues to be efficient. The number of daily financial transactions reaches millions, thus the scalability functions as an essential need in these systems. The proposed model can remove irrelevant data nodes early through threshold-based filtering and therefore simplify the computational burden along with some of the most important transactions. Skyline query optimization can help the system to only consider suspecting transactions away from the usual group, thus reducing the speed of fraud detection (speeding up the fraud detection) and reducing the load of the database as the analyzed work.

5.3.1.2 Recommendation Systems

When searching for hotels on an online booking website, factors such as price, location, rating, and amenities are important. However, not every hotel's data is comprehensive. A few hotels may simply lack some data, for example, non-existent or incomplete consumer reviews, untimely downstream price information, or an incomplete description of facilities. The skyline query optimization method guarantees hotel recommendations remain accurate and pertinent even with the imperfect data utilization of available information and clusters similar hotels according to the important components.

5.3.1.2.1 Hotel Data Processing

Multiple factors such as price, location, rating, and facilities must be considered in a hotel recommendation system. Incomplete data can lead to insufficient recommendations. The traditional systems may not be efficient in dealing with this, and the skyline query model

provides the solution. The model can process hotels with incomplete data by using machine learning-based clustering and skyline queries to evaluate them based on the available attributes. This feature ensures that even hotels with missing data points are included in recommendations. One of the important features of your model is the threshold-based filtering, which in the early process helps eliminate vendors that are of no relevance. Hotel prices and availability can vary a lot, so there is a need for updating recommendations in real-time. Using skyline query optimization, the system can respond to the real-time updates by always offering the most relevant hotels to the user.

5.3.1.2.2 Road Networks

The volume of real-time data that are generated by road networks include traffic conditions, incident reports and vehicle locations. This information is needed for time-route topology development, incident detection, and traffic flow control. Still, just as in other areas, road network data are incomplete, say, traffic sensor data gaps, insufficient data from GPS, or late incident reports. In a transportation network, traffic management systems needs to handle a vast inflow of data from traffic sensors, GPS devices, as well as social media in order to make real-time decisions. This data comprises of vehicle speed, traffic congestion levels, incident reports. With missing data (e.g. missing sensor values or late updates), conventional systems have difficulty making precise predictions. The proposed model can make skyline queries with the data that is available (e.g., current vehicle speeds or historical traffic patterns) so that the system can still determine the best routes and detect incidents if any data is missing.

Road networks too are graph-based where intersections are the nodes and roads are the edges that connect them. The system needs these linked connections to evaluate traffic flow alongside route optimization functions. The model continues with route optimization even when some roads or intersections lack data by grouping similar nodes through

clustering and executing skyline queries for route prioritization with the available data.

5.3.1.3 Real-Time Analytics

5.3.1.3.1 Social Media Platforms

Tweets created at Twitter accumulate to millions of new posts during each passing minute. The Twitter platform enables user interaction by liking content, sharing tweets with retweets, posting comments and sending mentions. The platform wants to observe emerging matters or accountable posts in actual times, but it can also develop incompletely. For example, there might be a tweet that has zero comments or retweets, but it could be popular, such as based on the count of likes and mentions it can gain. When more tweets are posted, the system keeps up calculating engagement metrics such as likes**,** retweets, and mentions on an ongoing basis. However, if a tweet lacks data, such as having no comments or retweets, the classic approach is likely to overlook it. The skyline query model employs skyline queries to rank tweets by the most valuable available metric, for example, the number of likes or mentions, even if other metrics are not present. The model uses clustering to segment similar tweets, given data like hashtags or keywords, and ranks the most impactful ones, comparing them again to past data. For instance, if some of the tweets are missing engagement metrics such as comments the model will still rank those by likes or user mentions available and classify them as relevant. Moreover, the platform utilizes the skyline query model to discover trending topics in real-time. Even when tweets lack information about engagement data (e.g. shares or comments), the system will still classify it as trending data based on existing data such as hashtags or likes. Example users tend to give high numbers of likes and several mentions to tweets that use hashtags such as climate change, while they sometimes experience delays or total absence. The recommendation model identifies the trending hashtag even while assigning priority to likes and hashtags. Using the run time skyline query optimization model, the system can efficiently find

relevant information and trending topics or when there is data points (like, comments and shares). This way there are no missed conversations about what matters most to their followers, and it makes a dynamic user experience on social media.

5.3.1.3.2 Traffic Management

The smart city management system performs traffic data monitoring by integrating data from sensors, GPS devices, and traffic cameras. The system has three main purposes: enhancing traffic flow performance and minimizing congestion while simultaneously identifying real-time collisions. Some traffic sensor data fails to arrive due to combined hardware issues, communication failures, and delayed reporting protocols. The failure of sensors could result in the nonreporting of traffic speed information for a particular road section. The system performs continuous analysis of current sensor-based data streams. The skyline query model retains accurate traffic flow predictions through relevant data like traffic volume and historical patterns and GPS-based information even when sensor data about traffic speed or vehicle count arrives late or shows missing values. The model detects unusual traffic behavior by performing skyline queries even when specific sensors stop functioning. The model continues to recognize typical traffic behavior through accessible data points by using clustered patterns against past information. For instance, when traffic sensor data is missing on a certain road segment, the model consults both earlier traffic data and nearby vehicle locations in GPS to react by seeking alternative routes. The system provides updated driving routes to drivers through real-time updates about changing traffic conditions. The skyline query model uses accessible traffic sensor data to identify the optimal route recommendation even when certain data points are missing by considering speed limits and traffic congestion and incident data. For example, if a road traffic sensor fails to report speed data, the system will still be able to use vehicle GPS data or historical traffic patterns to recommend alternative routes to avoid the congestion. The real-time accurate traffic flow decisions and incident detection by the traffic management system

become possible through the skyline optimization model even when there is incomplete or missing data. Dynamic changes and absent data points do not impact the system's performance since it maintains its operational efficiency and rapid response time.

5.3.2 Future Enhancements

The future study has the potential to explore advance ML approaches to enhance the optimization of the skyline query in incomplete graph database expanding the finding of the study. The incorporation of deep learning models including neural network and reinforcement learning to enhance the ability of the graph database to predict missing values accurately (Shu et al., 2023) . Also, the future research should focus on dynamic skyline queries where real-time update inside graph database is essential without the preprocessing the skyline query to ensure rapid query processing (Yuan et al., 2024c). Furthermore, the integration of proposed study into real-world scenario including the social network intelligent recommendation system and decision-making platforms. The implementation of this study in large scale systems including the e-commerce restaurant booking services, financial analysis platform etc, can improve data driven decision making irrelevant of missing data. Also, to improve the usability the potential implementation for future should address the development of user-friendly interfaces which allow seamless integration of the study into existing system architecture.

Additionally, the recommendation for future developments regarding the performance evaluations using the diverse dataset to validate the adaptability and scalability of the proposed study, the synthetic dataset provide simulated environment for testing, however real-world datasets comprise of complexities including unusually patterns, noise and heterogenous graph structure. The testing of the model under such conditions can fine-tune the approach to address accuracy, efficiency, and reliability to handle skyline queries across various domains. Lastly, future study should focus on reducing cost on

competition by developing a skyline query processing which can suit for high dimensional dataset with massive volume and reduce computational workload.

5.4 LIMITATION

There are certain limitations in the proposed study to handle skyline queries in an incomplete database despite its many effective and strong benefits. The primary limitation is dependency on the machine-learning-based-clustering approach, including K Means, which requires a predefined number of clusters. This selection of the optimal value of k is challenging, particularly for large-scale evolving graph databases. If k is not chosen properly, the resulting sub-optimal clustering will influence the efficiency of the skyline computation. Similarly, K-Means has a sensitive initialization for outliers, which might have a negative impact on the clustering of missing nodes and can affect dominance relationships.

Furthermore, the computational cost associated with clustering and filtering for large-scale datasets is high, while the proposed study reduces processing time significantly. This approach requires preprocessing and computation of clusters, which can have a higher cost for high-scale datasets. Similarly, the real-time application includes dynamic social networks, which may require further optimization to ensure low latency-based query processing. Furthermore, the proposed study assumes that missing values can be approximated within clusters; however, in the case of a highly sparse dataset, this estimation might introduce uncertainty over the dominance relationship.

Additionally, the study focus on synthetic datasets however real-world has different conditions which might affect the performance of the proposed study. Also, the constraints related to domain specific application including personalize recommendation system might require additional modifications to adapt with emerging requirements. The future research should address the adapting needs of real-world scenarios, enhancing the

ability to handle high dimensional heterogeneous and emerging graph structures while minimizing computational challenges.

5.5 SUMMARY

This chapter discusses the answers to the proposed research questions, confirming that machine learning significantly improves skyline query optimization in incomplete graph databases. The proposed methodology for optimizing skyline queries in large-scale graph databases focuses on improving performance through machine learning techniques. Conventional skyline query methods struggle with challenges such as incomplete data, high dimensionality, and computational inefficiency in large datasets. The proposed approach addresses these issues by leveraging machine learning for efficient sorting, filtration, and clustering, using techniques, i.e., K-Means, to group similar nodes and identify representative candidates. Furthermore, threshold-based pruning reduces unnecessary comparisons, while scoring mechanisms ensure the selection of optimal nodes for skyline determination. The methodology enhances scalability, query efficiency, and performance, offering practical applications in domains e.g. healthcare, logistics, and social networks, where handling large, complex datasets is critical.

REFERENCES

Abbaci, K., Hadjali, A., Liétard, L., & Rocacher, D. (2011). A similarity skyline approach for handling graph queries - A preliminary report. *Proceedings - International Conference on Data Engineering*, 112–117. https://doi.org/10.1109/ICDEW.2011.5767617

Alwan, A. A., Ibrahim, H., & Udzir, N. I. (2014). A framework for identifying skylines over incomplete data. *Proceedings - 3rd International Conference on Advanced Computer Science Applications and Technologies, ACSAT 2014*, 79–84. https://doi.org/10.1109/ACSAT.2014.21

Alwan, A. A., Ibrahim, H., Udzir, N. I., & Sidi, F. (2016). An Efficient Approach for Processing Skyline Queries in Incomplete Multidimensional Database. *Arabian Journal for Science and Engineering*, *41*(8), 2927–2943. https://doi.org/10.1007/s13369-016-2048-z

Alwan, A., Ibrahim, H., Udzir, N., & Sidi, F. (2018). Missing values estimation for skylines in incomplete database. *International Arab Journal of Information Technology*, *15*(1), 66–75.

Amr, D., & El-Tazi, N. (2018). *Skyline Query Processing in Graph Databases*. 49–57. https://doi.org/10.5121/csit.2018.81005

Banerjee, S., Pal, B., & Jenamani, M. (2020). DySky: Dynamic Skyline Queries on Uncertain Graphs. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12342 LNCS*, 242–254. https://doi.org/10.1007/978-3-030-62005-9_18

Bharuka, R., & Kumar, S. (n.d.-a). *Finding Skylines for Incomplete Data*. http://movielens.umn.edu

Bharuka, R., & Kumar, S. (n.d.-b). *Finding Superior Skyline Points from Incomplete Data*. http://basketballreference.com

Börzsönyi, S., Kossmann, D., & Stocker, K. (2001). The skyline operator. *Proceedings - International Conference on Data Engineering*, 421–430. https://doi.org/10.1109/icde.2001.914855

Chomicki, J., Godfrey, P., Gryz, J., & Liang, D. (n.d.). *Skyline with Presorting: Theory and Optimizations*.

Chomicki, J., Godfrey, P., Gryz, J., & Liang, D. (2002). *Skyline with Presorting*.

Ding, L., Zhang, G., Ma, J., & Li, M. (2023). An Efficient Index-Based Method for Skyline Path Query over Temporal Graphs with Labels. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *13945 LNCS*, 217–233. https://doi.org/10.1007/978-3-031-30675-4_15

Gao, Y., Miao, X., Cui, H., Chen, G., & Li, Q. (2014). Processing k-skyband, constrained skyline, and group-by skyline queries on incomplete data. *Expert Systems with Applications*, *41*(10), 4959–4974. https://doi.org/10.1016/j.eswa.2014.02.033

Godfrey. (2005). *Maximal vector computation in large data sets*.

Gulzar, Y. (2018). Skyline Query Approaches in Static and Dynamic Incomplete Databases.

Gulzar, Y., Alwan, A. A., Abdullah, R. M., Xin, Q., & Swidan, M. B. (2019). SCSA: Evaluating skyline queries in incomplete data. *Applied Intelligence*, *49*(5), 1636–1657. https://doi.org/10.1007/s10489-018-1356-2

Gulzar, Y., Alwan, A. A., Ibrahim, H., Turaev, S., Wani, S., Soomo, A. B., & Hamid, Y. (2021). IDSA: An Efficient Algorithm for Skyline Queries Computation on Dynamic and Incomplete Data with Changing States. *IEEE Access*, *9*, 57291–57310. https://doi.org/10.1109/ACCESS.2021.3072775

Gulzar, Y., Alwan, A. A., Salleh, N., Shaikhli, I. F. Al, & Alvi, S. I. M. (2016). A Framework for Evaluating Skyline Queries over Incomplete Data. *Procedia Computer Science*, *94*, 191–198. https://doi.org/10.1016/j.procs.2016.08.030

Gulzar, Y., Alwan, A. A., & Turaev, S. (2019a). Optimizing Skyline Query Processing in Incomplete Data. *IEEE Access*, *7*, 178121–178138. https://doi.org/10.1109/ACCESS.2019.2958202

Gulzar, Y., Alwan, A. A., & Turaev, S. (2019b). Optimizing Skyline Query Processing in Incomplete Data. *IEEE Access*, *7*, 178121–178138. https://doi.org/10.1109/ACCESS.2019.2958202

He, J., & Han, X. (2022). Efficient Skyline Computation on Massive Incomplete Data. *Data Science and Engineering*, *7*(2), 102–119. https://doi.org/10.1007/s41019-022-00183-7

Hevner, A. (2014). *A Three Cycle View of Design Science Research*. https://www.researchgate.net/publication/254804390

Keles, I., & Hose, K. (2019). Skyline Queries over Knowledge Graphs. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11778 LNCS*, 293–310. https://doi.org/10.1007/978-3-030-30793-6_17

Kumar Sadineni, P. (2020a). Comparative study on skyline query processing techniques on big data. *Proceedings of the 4th International Conference on IoT in Social, Mobile, Analytics and Cloud, ISMAC 2020*, 1045–1050. https://doi.org/10.1109/I-SMAC49090.2020.9243343

Kumar Sadineni, P. (2020b). Comparative study on skyline query processing techniques on big data. *Proceedings of the 4th International Conference on IoT in Social, Mobile, Analytics and Cloud, ISMAC 2020*, 1045–1050. https://doi.org/10.1109/I-SMAC49090.2020.9243343

Kuo, A.-T., Chen, H., Tang, L., Ku, W.-S., & Qin, X. (2023). ProbSky: Efficient Computation of Probabilistic Skyline Queries Over Distributed Data. *IEEE Transactions on Knowledge and Data Engineering*, *35*(5), 5173–5186. https://doi.org/10.1109/TKDE.2022.3151740

Lee, J., Im, H., & You, G. W. (2016a). Optimizing skyline queries over incomplete data. *Information Sciences*, *361–362*, 14–28. https://doi.org/10.1016/j.ins.2016.04.048

Lee, J., Im, H., & You, G.-W. (2016b). Optimizing skyline queries over incomplete data. *Information Sciences*, *361–362*, 14–28. https://doi.org/10.1016/j.ins.2016.04.048

Liu, C. M., Pak, D., & Ortiz Castellanos, A. E. (2021). Priority-Based Skyline Query Processing for Incomplete Data. *ACM International Conference Proceeding Series*, 204–211. https://doi.org/10.1145/3472163.3472272

Liu, W., Wen, D., Wang, H., Zhang, F., & Wang, X. (2019a). Skyline nearest neighbor search on multi-layer graphs. *Proceedings - 2019 IEEE 35th International Conference on Data Engineering Workshops, ICDEW 2019*, 259–265. https://doi.org/10.1109/ICDEW.2019.000-3

Liu, W., Wen, D., Wang, H., Zhang, F., & Wang, X. (2019b). Skyline nearest neighbor search on multi-layer graphs. *Proceedings - 2019 IEEE 35th International Conference on Data Engineering Workshops, ICDEW 2019*, 259–265. https://doi.org/10.1109/ICDEW.2019.000-3

Makrynioti, N. N., Vasiloglou, N. N., Pasalic, E. E., & Vassalos, V. V. (2018, June 15). Modelling machine learning algorithms on relational data with datalog. *Proceedings of the 2nd Workshop on Data Management for End-To-End Machine Learning, DEEM 2018 - In Conjunction with the 2018 ACM SIGMOD/PODS Conference*. https://doi.org/10.1145/3209889.3209890

Martin-Nevot, M., & Lakhal, L. (2025). *Ranking Methods for Skyline Queries*. http://arxiv.org/abs/2507.21860

Miao, X., Gao, Y., Chen, G., Zheng, B., & Cui, H. (2016). Processing incomplete k nearest neighbor search. *IEEE Transactions on Fuzzy Systems*, *24*(6), 1349–1363. https://doi.org/10.1109/TFUZZ.2016.2516562

Miao, X., Gao, Y., Chen, L., Chen, G., Li, Q., & Jiang, T. (2013). On efficient k-Skyband query processing over incomplete data. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7825 LNCS*(PART 1), 424–439. https://doi.org/10.1007/978-3-642-37487-6_32

Miao, X., Gao, Y., Guo, S., & Chen, G. (2018). On Efficiently Answering Why-Not Range-Based Skyline Queries in Road Networks. *IEEE Transactions on Knowledge and Data Engineering*, *30*(9), 1697–1711. https://doi.org/10.1109/TKDE.2018.2803821

Miao, X., Gao, Y., Guo, S., Chen, L., Yin, J., & Li, Q. (2021a). Answering Skyline Queries over Incomplete Data with Crowdsourcing. *IEEE Transactions on Knowledge and Data Engineering*, *33*(4), 1360–1374. https://doi.org/10.1109/TKDE.2019.2946798

Miao, X., Gao, Y., Guo, S., Chen, L., Yin, J., & Li, Q. (2021b). Answering Skyline Queries over Incomplete Data with Crowdsourcing. *IEEE Transactions on Knowledge and Data Engineering*, *33*(4), 1360–1374. https://doi.org/10.1109/TKDE.2019.2946798

M.M.F. Fahima, A.H. Sahna Sreen, S.L. Fathima Ruksana, D.T.E. Weihena, & M.H.M. Majid. (2024). Machine Learning for Database Management and Query Optimization. *Elementaria: Journal of Educational Research*, *2*(1), 96–108. https://doi.org/10.61166/elm.v2i1.66

Mohamed E. Khalefa. (2008). *Skyline query Processing for incomplete Data*. IEEE Xplore.

Mohamud, M. A., Ibrahim, H., Sidi, F., & Rum, S. N. M. (2025). Dominance Analyses Reduction in Skyline Query Processing over Data Stream with Data Mining Technique. *ICICM 2024 - Proceedings of the 2024 14th International Conference on Information Communication and Management*, 29–35. https://doi.org/10.1145/3711609.3711614

Mohamud, M. A., Ibrahim, H., Sidi, F., Rum, S. N. M., Dzolkhifli, Z. B., Xiaowei, Z., & Lawal, M. M. (2023). A Systematic Literature Review of Skyline Query Processing Over Data Stream. In *IEEE Access* (Vol. 11, pp. 72813–72835). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ACCESS.2023.3295117

Ouyang, D., Yuan, L., Zhang, F., Qin, L., & Lin, X. (2018). Towards efficient path skyline computation in bicriteria networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10827 LNCS*, 239–254. https://doi.org/10.1007/978-3-319-91452-7_16

Ren, W., Lian, X., & Ghazinour, K. (2019a). Skyline queries over incomplete data streams. *VLDB Journal*, *28*(6), 961–985. https://doi.org/10.1007/s00778-019-00577-6

Ren, W., Lian, X., & Ghazinour, K. (2019b). Skyline queries over incomplete data streams. *VLDB Journal*, *28*(6), 961–985. https://doi.org/10.1007/s00778-019-00577-6

Shu, Y., Zhang, J., Zhang, W. E., Zuo, D., & Sheng, Q. Z. (2023). IQSrec: An Efficient and Diversified Skyline Services Recommendation on Incomplete QoS. *IEEE Transactions on Services Computing*, *16*(3), 1934–1948. https://doi.org/10.1109/TSC.2022.3189503

Swidan, M. B., Alwan, A. A., Turaev, S., Ibrahim, H., Abualkishik, A. Z., & Gulzar, Y. (2020). Skyline Queries Computation on Crowdsourced- Enabled Incomplete Database. *IEEE Access*, *8*, 106660–106689. https://doi.org/10.1109/ACCESS.2020.3000664

Tang, J., Liu, J., Zhang, M., & Mei, Q. (2016). *Visualizing Large-scale and High-dimensional Data*. 287–297. https://doi.org/10.1145/2872427.2883041

Vlachou, A., Doulkeridis, C., Rocha-Junior, J. B., & Nørvåg, K. (n.d.). *Decisive Skyline Queries for Truly Balancing Multiple Criteria*.

Wang, H., Yin, S., Sun, M., Wang, Y., Wang, H., Li, J., & Gao, H. (2018). Efficient Computation of Skyline Queries on Incomplete Dynamic Data. *IEEE Access*, *6*, 52741–52753. https://doi.org/10.1109/ACCESS.2018.2869819

Wang, Y., Shi, Z., Wang, J., Sun, L., & Song, B. (2017). Skyline preference query based on massive and incomplete dataset. *IEEE Access*, *5*, 3183–3192. https://doi.org/10.1109/ACCESS.2016.2639558

Yang, D., Wang, Y., Li, Y., & Ma, X. (2016). A variable markovian based outlier detection method for multi-dimensional sequence over data stream. *Parallel and Distributed Computing, Applications and Technologies, PDCAT Proceedings*, *0*, 183–188. https://doi.org/10.1109/PDCAT.2016.049

Yang, Z., Yang, X., & Zhou, X. (2016). Uncertain dynamic skyline queries for uncertain databases. *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2015*, 1797–1802. https://doi.org/10.1109/FSKD.2015.7382219

Yasuhiko Morimoto, M. S. A. (n.d.). *Skyline Sets Queries from Databases with Missing Values*.

Yuan, D., Zhang, L., Li, S., & Sun, G. (2024a). Skyline query under multidimensional incomplete data based on classification tree. *Journal of Big Data*, *11*(1). https://doi.org/10.1186/s40537-024-00923-8

Yuan, D., Zhang, L., Li, S., & Sun, G. (2024b). Skyline query under multidimensional incomplete data based on classification tree. *Journal of Big Data*, *11*(1). https://doi.org/10.1186/s40537-024-00923-8

Yuan, D., Zhang, L., Li, S., & Sun, G. (2024c). *skyline query under multidimensional incomplete data based on classiication tree skyline query under multidimensional incomplete data based on classification tree*. https://doi.org/10.21203/rs.3.rs-3915982/v1

Zeng, Y., Li, K., Yu, S., Zhou, Y., & Li, K. (2018). Parallel and Progressive Approaches for Skyline Query over Probabilistic Incomplete Database. *IEEE Access*, *6*, 13289–13301. https://doi.org/10.1109/ACCESS.2018.2806379

Zhang, K., Gao, H., Han, X., Cai, Z., & Li, J. (2017). Probabilistic skyline on incomplete data. *International Conference on Information and Knowledge Management, Proceedings*, *Part F131841*, 427–436. https://doi.org/10.1145/3132847.3132930

Zhang, K., Gao, H., Wang, H., & Li, J. (2016). ISSA: Efficient skyline computation for incomplete data. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 9645). https://doi.org/10.1007/978-3-319-32055-7_26

Zheng, W., Zou, L., Lian, X., Hong, L., & Zhao, D. (2014). Efficient subgraph skyline search over large graphs. *CIKM 2014 - Proceedings of the 2014 ACM International Conference on Information and Knowledge Management*, 1529–1538. https://doi.org/10.1145/2661829.2662037

Zhu, X., Wu, J., Chang, W., Wang, G., & Liu, Q. (2018a). Authentication of skyline query over road networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11342 LNCS*, 72–83. https://doi.org/10.1007/978-3-030-05345-1_6

Zhu, X., Wu, J., Chang, W., Wang, G., & Liu, Q. (2018b). Authentication of skyline query over road networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11342 LNCS*, 72–83. https://doi.org/10.1007/978-3-030-05345-1_6

Zou, L., Chen, L., Tamer¨ozsu, M., Tamer¨ozsu, T., & Zhao, D. (n.d.). *Dynamic Skyline Queries in Large Graphs*.

APPENDIX I

REQUIREMENTS

https://github.com/ubairnoortech/Skyline-Queries-in-Graph-database