SKYLINE QUERY PROCESSING IN GRAPH DATABASES

Dina Amr and Neamat El-Tazi

Faculty of Computers and Information, Cairo University, Giza, Egypt

ABSTRACT

Skyline queries are mostly used in decision making processes and search space reduction. They received much attention during the past years due to their importance in discarding the unneeded data and providing the users with data that best match their interest. The same attention has been directed to graph databases to handle highly connected data due to the increase in volume and connectedness of today's data. The proposed work aims to augment graph databases with skyline queries. Two skyline query processing algorithms have been proposed; nested loops and divide-and-conquer. They are used to facilitate retrieving skyline results with multiple dimensions over graph databases. Performance evaluation for both algorithms over different sized graph databases and queries with different complexity levels were presented. The conducted experiments proved that divide-and-conquer outperforms nested loops in different cases.

KEYWORDS

Skyline Queries, Graph Database, Graph Querying, Neo4j, Cypher

1. INTRODUCTION

Nowadays most of the real-world applications like social networks can use graph databases to store their data, due to their countless advantages over relational models. One of the main advantages of graph database is that it has flexible schema and new information can be added on the fly. Unlike relational database which requires joins to retrieve a relationship between tables, data in graph database is connected through bidirectional relationships between nodes which make it easy to retrieve any linkage. Another advantage in graph models over relational, is that nodes are created using information supplied by the user which means that the database does not store null values. This helps in saving storage space.

An interesting type of existing queries in relational databases are skyline queries. Skyline queries retrieve a set of interesting points from a large dataset [1]. Skyline results contain the data that is not dominated by any other data [1]. Domination occurs based on some conditions. The conditions are determined according to user's preferences.

Most of decision making processes and data pruning techniques need skyline queries to return the best results based on user's requirements. Using already existing algorithms, we propose how to implement skyline queries over graph databases. We consider skyline queries that have multiple dimensions. The dimensions reflect node properties. A node dominates another node, if it is better in all dimensions. This is based on the selected criteria in the query input. The criteria can be maximum or minimum for the selected dimensions. One of the properties of skyline queries, is

Natarajan Meghanathan et al. (Eds) : SAI, ICAITA, CSITA, ISPR, Signal - 2018
pp. 49–57, 2018. © CS & IT-CSCP 2018 DOI : 10.5121/csit.2018.81005

that there are no weights for the dimensions used by the user. Thus, the domination is done based on properties' values only.

Table 1. Hotels Database.

HotelName	PricePerNight	DistancetoBeach	Stars
A	250	20	4
B	300	50	4
C	500	10	5
D	100	80	3
E	380	30	5

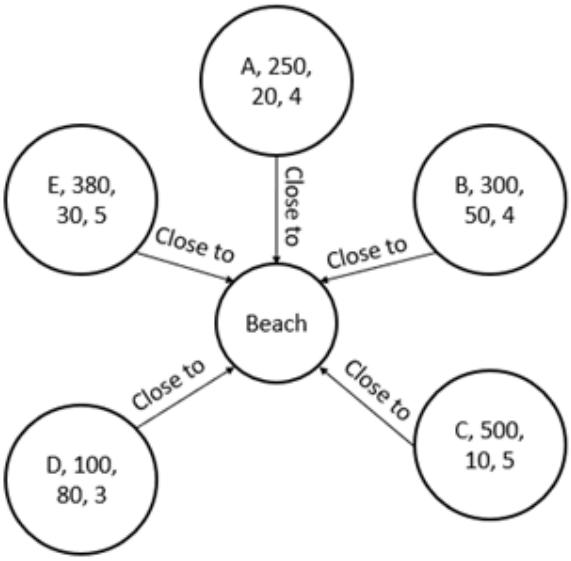

Figure 1. Hotels Database Nodes

A very famous example to illustrate skyline queries is a sample hotel database similar to that in Table 1 and Figure 1 which stores hotel names, price of room per night, distance to the beach and star rating, from this database the user needs to retrieve the hotel which is close to the beach and has low price per night. It is obvious that the two preferences may be conflicting, because hotels close to the beach will most probably be more expensive. This query has two dimensions: the first dimension is the PricePerNight and the second is DistancetoBeach. Thus, when a skyline query is applied to the dataset, it will return hotels which are better than other hotels in both preferences/dimensions. These hotels are called skyline. In this example, the hotels: "A", "C" and "D" are the skyline results as none of them is better than the others across all dimensions. At this point it is the user's decision to select from the three hotels instead of returning the whole list of hotels to select from. The importance of skyline queries appears more with larger datasets with thousands of hotels. In this paper, we propose a way to implement skyline queries over a graph model.

In addition to the properties of graph that makes us motivated to apply skyline, also the wide spread of using graph databases in real-world applications like social recommendations, authorization and access control and geospatial and logistics which makes it more valuable to have skyline queries on graph databases. We argue that introducing skyline operators to graph databases is an important research point.

The paper is organized as follows. Section 2 illustrates the background related to graph model properties. In Section 3, we review the related work. Section 4 proposes how to process skyline queries inside graph databases using nested loops algorithm. Skyline queries is re-introduced using divide- and- conquer algorithm in Section 5. Experiments, in Section 6, were conducted to show different query performance using different sized datasets and variable number of dimensions. Conclusion and future work are presented in Section 7.

2. PROPERTY GRAPH MODEL

In this paper, property graph model is used which stores data in the form of a graph with nodes and directed edges between those nodes. A graph can be traversed in bidirectional way, which means that there is no need to add duplicate relationship in both directions of an edge [11]. Edges represent relationships that connect nodes. Each two nodes can have more than one relationship. Nodes have properties which describe them. Same applies to the edge properties, which describe the relationship between nodes. Nodes have labels which represent the entity they belong to or the node type. Each node may have multiple labels. Node labels group nodes together to indicate their function in the dataset. Graph size can be determined based on the number of nodes it contains in the database. Indexes on graph databases can occur on the relationships level which helps in fast data retrieval. They also support much more flexibility in updating or extending data and its structure. Thus, more properties can be added to a node and relationships. They can be easily adapted to new business needs.

3. RELATED WORK

Skyline queries help getting the best solutions based on user preferences. The preferences are represented in graph databases in the form of node/edge properties. The properties are com-pared based on some criteria: maximization or minimization. All dimensions are compared at the same time. Nodes that are less interesting to the user are dominated and excluded from the skyline result. This type of queries helps in reducing search space and saving time. Only the most interesting data is included in the result. Thus, skyline queries guarantee that the returned nodes are the ones that most satisfy user requirements.

The Skyline operator was introduced in [1], their objective was to extend relational database systems by Skyline queries. The authors extended the SQL's select statement by proposing SKYLINE OF clause. The authors showed how Skyline operator can interact with other query operators. In [2, 3] the authors focused on dynamic Skyline. The skyline result is determined based on shortest path distance that differs according to each algorithm. Aggregate skyline queries were introduced in [4]. The query combines skyline and group by operators. The experimental results showed that the query execution time is better than using the operators directly in SQL query.

In [5] the authors focused on RDF data stored using vertically partitioned schema model. They introduced an approach for optimizing skyline queries for this type of data. They focused on pruning non-skyline tuples before reaching the complete skyline processing phase. This is done using the header point concept which keeps a summary of the already visited data space regions. The authors proposed two algorithms RSJPH and RSJPH+. The algorithms are considered near complete and they help achieve the trade-off between complete skyline queries and fast response time.

The authors focused on getting skyline for road networks in [6]. Road networks consists of nodes and edges between them. The main goal was to get skyline results based on many points of

interest, with two important factors: Size which represent the distance between nodes and relevance which focus on what the user exactly requested.

New parallel algorithm named SKY-MR+ was introduced in [7]. The algorithm uses MapReduce to process skyline queries. Experiments were conducted to prove the scalability and effectiveness of the algorithm.

In [8], the authors focused on the data items that have incomplete data. The dimensions to be compared in the skyline query are not presented in some data items. The authors developed an algorithm called ISkyline, which handles the missing data issue. The experiments conducted showed the efficiency and scalability of ISkyline.

Skyline results may be affected by outliers. This challenge was addressed in [9]. The authors implemented an algorithm which focuses on the degree of membership of a result to the skyline and the typicality degree. The main goal was to exclude outlier data from skyline result. In [10], the authors summarized the basic properties of skyline queries. They also discussed how they can be extended and generalized. We argue that introducing skyline queries to graph databases is an important research. This paper uses two algorithms for processing skyline queries over graph databases.

4. SKYLINE NESTED LOOPS ALGORITHM

Processing skyline queries in graph databases can be implemented using nested loops algorithm. It is applied on the whole set of nodes to be compared. It compares every single node with all the other nodes having the same label. All dimensions of each node are compared with their relative dimensions of the other nodes. The comparison is done based on the user's query. The query can be maximum or minimum of both dimensions. The node's dimensions are compared, if the query asks for the maximum of all dimensions, then a node that have all dimension's values less than any other node, will be dominated.

Algorithm 1 Skyline using Nested Loops

```
1: inputs
2:     G (N, E) (Graph with nodes and edges between them), P (Node property to be returned), D
       (List of dimensions or edge properties to be compared)
3: outputs
4:     N (Final skyline nodes)
5: Read nodes and edges properties
6: Add all nodes of same label into collection N
7: for i in N do
8:     for j in N do
9:         for c in D do
10:            if all c of i > all c of j then
11:                remove i from N
12:            end if
13:        end for
14:    end for
15: end for
16: Return N
```

Algorithm 1 represents minimization for all dimensions and it can be applied on maximization. For illustration, we applied the nested loop algorithm on Neo4j [11] graph database. We implement the skyline query within cypher [12] query language using the proposed adapted algorithm and execute it on a Neo4j graph engine. We use cypher query language to represent the different skyline queries that can be generated using nested loops algorithm.

The main advantage of using nested loops algorithm is its high applicability as it can be used in any graph database being extensible to a large number of dimensions. On the other hand, it has some cons; it cannot get early skyline results, the whole dataset should be scanned before returning any skyline point which leads to query time complexity of $O(n^2)$, where n is the number of nodes to be compared inside the database. In addition, it completely relies on main memory, which may lead to many iterations given small memory capacity and large graph size. To enhance the performance of the skyline operator, divide-and-conquer skyline algorithm was also adapted to work on graph databases in the next section.

5. SKYLINE DIVIDE-AND-CONQUER ALGORITHM

This section proposes the extensions made to the divide-and-conquer algorithm to make it work efficiently with graph stores to get skyline. The divide-and-conquer algorithm is considered the best-known algorithm for the worst-case scenario [1] where no nodes are dominated. Thus, no node is better than the other in all dimensions. The whole data set is divided into sub-groups. Each group of nodes are compared together, and the final skyline result is the collection of skyline of each subgroup. It avoids re-comparing nodes that are already visited.

Algorithm 2 Skyline using Divide-and-Conquer

1: **inputs**
2: G (N, E)(Graph with nodes and edges between them), P(Node property to be returned), D (List of dimensions or edge properties to be compared)
3: **outputs**
4: N (List of final skyline nodes)
5: Calculate median of all dimensions and edges values in D
6: Divide nodes into blocks based on medians values
7: Call Nested loops algorithm for each block of nodes
8: Call Nested loops algorithm for each two blocks of nodes together
9: Merge partial skyline results
10: Return N

Divide-and-conquer algorithm solved the problem of comparing all individual nodes with all other nodes. It supports early domination by getting partial skyline from each block. Since this algorithm supports multiple dimensions d, the data can be partitioned into 2^d blocks. The complexity of the algorithm is $O(n \log^{d-1} n)$ where n is the number of nodes to be compared in the whole dataset and d is the number of dimensions. Implementing skyline queries using divide-and-conquer algorithm improves query performance according to the experimental results. The average query execution time and performance comparison for the two algorithms are presented in the next section.

6. PERFORMANCE EVALUATION

In this section, we test the performance of the two proposed skyline algorithms. By changing the environmental settings, the efficiency of the two algorithms vary. The following subsections present the experiments setup and evaluation discussions.

6.1. Experiments Setup

We used Neo4j graph database [11] to conduct the performance experiments as being the most popular graph model used in fraud detection, social networks, recommendation engines and graph-based search. The experiments are conducted on a laptop running on windows 10. The processor has a Core(TM) i5 1.70 GHz CPU and 8GB of memory. If the specifications of the machine running the experiments are changed, it will affect the results. As an example, if the memory increases, the query processing time will decrease, as both skyline algorithms rely on main memory. The two algorithms were implemented using Cypher query language. We used two different datasets for conducting the evaluation; MovieLens database [13] which consists of 10,000 nodes and a synthetic database with 1,000,000 nodes.

We transformed the tuples of the MovieLens database into nodes with properties. The transformation process is done through Cypher query language, where the query reads the database in the form of tuples stored in a CSV file, and then generates the nodes with properties. While the synthetic dataset represents hotel database. We used [14] to generate the data in the form of CSV file, and in the same way it is transformed into nodes through Cypher query language.

The MovieLens database consists of nodes with label “Movie”, each node has properties; MovieID, Name, Rating, ReleaseYear and OscarWins. In the synthetic database, two labels exist; “Hotel” and “Beach”. The relation between “Hotel” and “Beach” is “Close to” which represents the closeness of the hotel to the beach. This relationship has one property called DistancetoBeach which stores the distance between the hotel and the beach. The nodes with label “Hotel” have the properties; HotelID, HotelName, Price, PoolSize, RestaurantQuality, StarsRating, ServiceQuality and NumberOfRooms.

6.2. Varying Number of Dimensions

The number of dimensions to be compared for skyline query can greatly affect the performance. Experiments were conducted on different number of dimensions with fixed database size and average execution time was recorded. For the MovieLens database we executed the algorithm on 10,000 nodes and for the synthetic database the algorithm was executed on 1,000,000 nodes. The results are shown in Figure 2 and Figure 3 respectively.

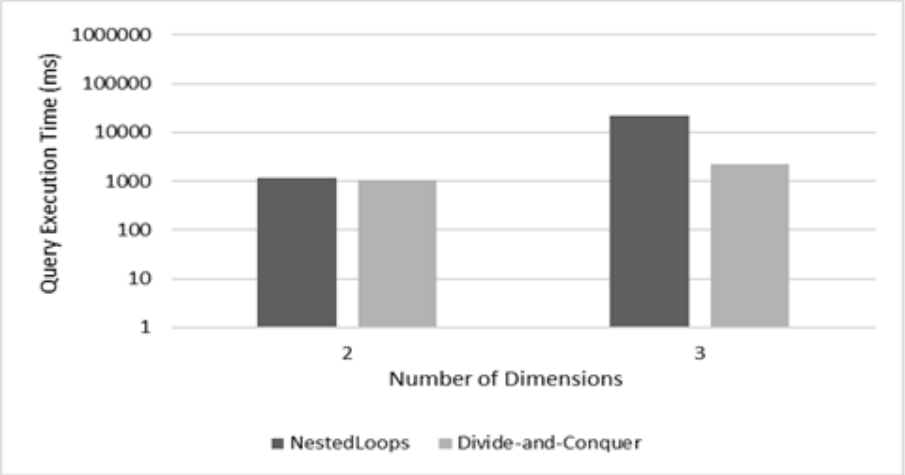

Figure 2. Comparing performance versus different number of dimensions for both skyline algorithms on MovieLens database [13]

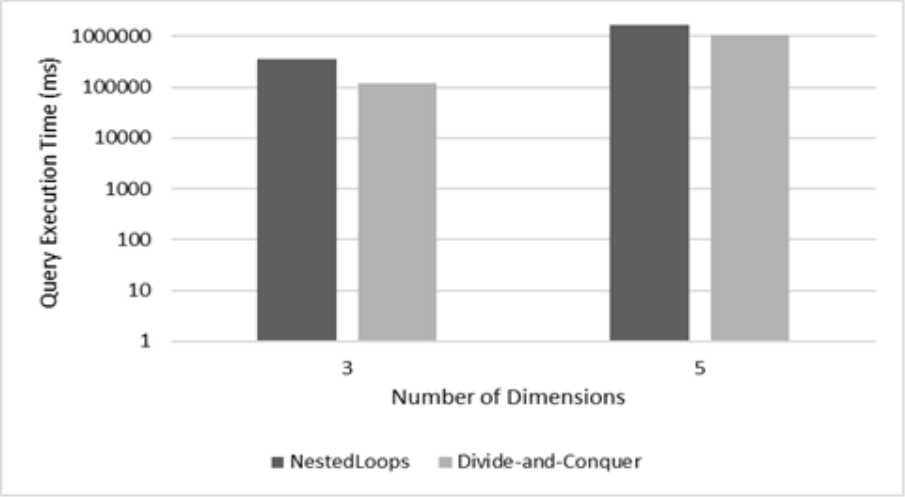

Figure 3. Comparing performance versus different number of dimensions for both skyline algorithms on the synthetic database

Query average execution time is highly affected by dimensionality, while the number of dimensions to be compared increases, the execution time increases and the difference between the two algorithms become more obvious.

6.3. Varying Dataset Size

In this experiment, the number of dimensions was fixed to 3 dimensions and we used multiple data set sizes. The results of each algorithm are shown in Figure 4 and Figure 5.

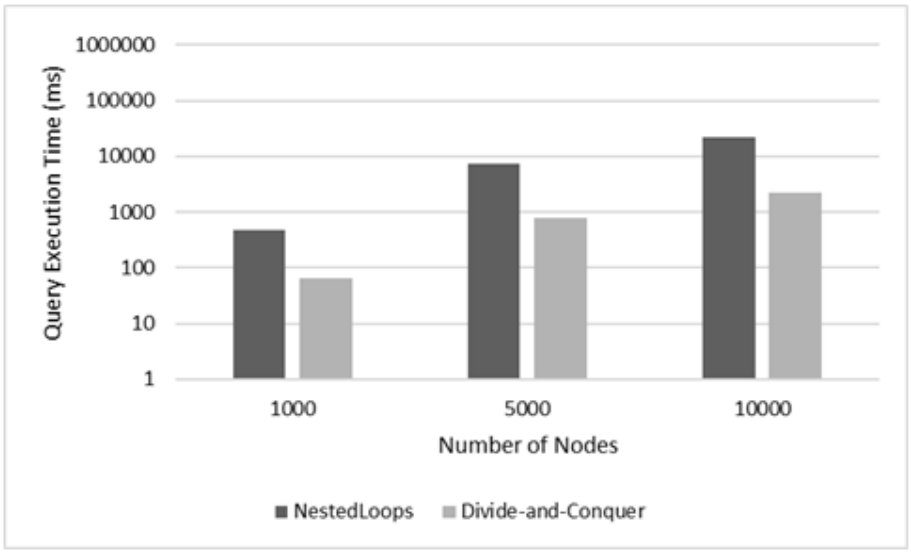

Figure 4. Comparing performance versus different dataset size for both skyline algorithms on MovieLens database [13]

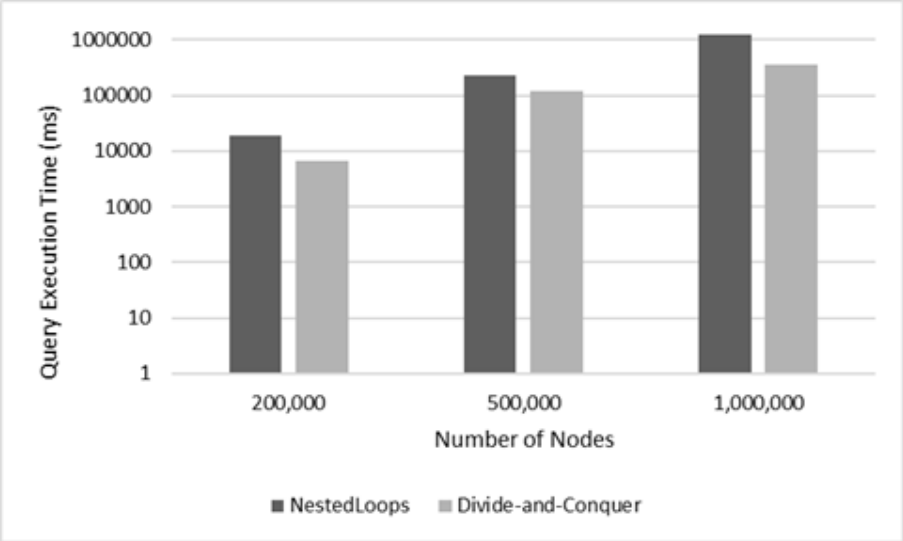

Figure 5. Comparing performance versus different dataset size for both skyline algorithms on the synthetic database

From the result of the experiments, we conclude that divide-and-conquer algorithm performs better than nested loops in all cases with varying the data set size and number of dimensions. However, with small number of dimensions the execution time for both algorithms appears to be very close, the superiority of divide-and-conquer algorithm appears more with large number of dimensions.

7. CONCLUSION AND FUTURE WORK

In this paper, different algorithms were used to process skyline queries over graph databases This type of queries is mostly important in decision making processes, data pruning and visualization. The most well-known algorithms in processing skyline queries over relational databases are nested loops algorithm and divide-and-conquer algorithm. We adapted the two algorithms to work with the graph model. The nested loop algorithm has a time complexity of O (n^2) where the divide-and-conquer has O ($n \log^{d-1} n$). Experiments for comparing the two algorithms over graph databases were conducted. An evaluation over different databases sizes and queries was also implemented and results was presented in the paper where it shows that the divide-and-conquer algorithm showed better performance than nested loops algorithm over graph databases. As a future work, we are planning to augment graph databases with skyline operator to facilitate the operation of getting skyline results on graph databases.

REFERENCES

[1] Borzsony, S., Kossmann, D., Stocker, K. 2001. The Skyline operator. Proc. 17th Int. Conf. Data Eng. 1–20.

[2] Zou, L., Chen, L., Özsu, M.T., Zhao, D. 2010. Dynamic skyline queries in large graphs. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 5982 LNCS, 62–78.

[3] Angel C Bency. 2014. A Study on Dynamic Skyline Queries. Published in International Journal for Research in Applied Science and Engineering Technology (IJRASET).

[4] Matteo Magnani, Ira Assent. 2013. From Stars to Galaxies: skyline queries on aggregate data. Presented at the proceedings of the 16th international conference on extending database technology.

[5] Chen, L., Gao, S., Anyanwu, K. 2011. Efficiently evaluating skyline queries on RDF databases. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part II. LNCS, vol. 6644, pp. 123–138. Springer, Heidelberg.

[6] Pande, S., Ranu, S., Bhattacharya, A. 2017. SkyGraph: Retrieving Regions of Interest using Skyline Subgraph Queries. Presented at the proceedings of the VLDB Endowment Volume 10 Issue 11.

[7] Yoonjae Park, Jun-Ki Min, Kyuseok Shim. 2017. Efficient Processing of Skyline Queries Using MapReduce. IEEE Transactions on Knowledge and Data Engineering.

[8] Khalefa, M.E., Mokbel, M.F., Levandoski, J.J. 2008. Skyline query processing for incomplete data. Presented at the IEEE 24th International Conference on Data EngineeringPages 556-565.

[9] Hélène Jaudoin, Pierre Nerzic, Olivier Pivert, Daniel Rocacher. 2016. On Making Skyline Queries Resistant to Outliers. In Advances in Knowledge Discovery and Management Volume 665 of the series Studies in Computational Intelligence pp 19-38.

[10] Jan Chomicki, Paolo Ciaccia, Niccolo' Meneghetti. 2013. Skyline Queries, Front and Back. ACM SIGMOD Record Volume 42 Issue 3.

[11] Neo4j. https://neo4j.com/

[12] Cypher. http://neo4j.com/docs/developer-manual/current/cypher/

[13] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872 [2] Gizem, Aksahya & Ayese, Ozcan (2009) Coomunications & Networks, Network Books, ABC Publishers.

[14] Database test data. http://www.databasetestdata.com/