An Efficient Skyline Query Processing based Approach Combined with Improved Parameters for Nearest Location Search

Miss. Pooja Rao*, Prof. Imran Ali Khan** and Prof. Damodar Tiwari***
MTech Research Scholar,Dept. of CSE.,Bansal Institute of Research and Technology,Bhopal, Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal M.P.,India
-*Bansal Institute of Research and Technology, Bhopal

Abstract: Nearest location search requires data maintenance and heavy processing on server side for efficient client response. The users are sending their proper location as the query parameter, and get the result as the nearest position for stores, restaurants etc. The data of nearest locations involve incomplete data items and incomplete data suffer from non-transitive dominance relation. Skyline operator is used in a query and carries out a filtering of result from a collection of data that are best from other results. In this paper we proposed a technique for nearest location search based on Skyline Processing combined with improved search parameters like price discount etc. The proposed approach improves consumer response time.

Keywords: Skyline Query, Skyline Processing, Nearest Neighbor Search KNN.

Introduction

Skyline queries aim to optimize the search space of large amounts of multidimensional data items by eliminating items that are dominated by others. Existing skyline algorithms assume that all dimensions are available for all data items. The main objective of skyline processing is minimizing query execution time. The Skyline operator is used in a query and performs a filtering of results from a database so that it keeps only those objects that are not worse than any other. Definition: Given a set of points, the skyline query returns a set of points which is not dominated by any other point in the dataset. For example a dataset containing information about stores; the price for each data point is recorded. Consider a two dimensional plot of the dataset, where the distance and price are assigned to the X,Y axis of the plot. The goal of the search is to find a store whose items discount and the price are both minimum and the preference function in our example is "minimum price and maximum discount". Conventional approaches in skyline query processing include both index based and non-index based approaches. A method to extend database systems has been suggested by using Skyline operations [3]. This operation filters out a set of interesting points from a potentially large set of data points. In some cases, each picture is represented as a D-dimensional stage where each dimension corresponds to a specific user. When searching for the cases like best movie, a skyline query eliminates those movies for which all users agree that there is at least another exceptional movie. Due to the importance of skyline queries, many research efforts have been dedicated to develop effective skyline query processing techniques [2], [3], [4], [5], [6], [7]. Almost all of these calculations rely mainly on two implicit assumptions:

1. Data have been complete, i.e., all measurements are available for many data items. Such an assumption of completeness is not practical in many scenarios. By way of instance, consider the movie rating application [1] with countless users rating tens of thousands of movies. It is exceedingly unlikely that every single user will speed all pictures. Rather, a user will speed only the movies that interest her. Because of this, each picture is going to be represented as a D-dimensional point with various sterile measurements. Another example is from the hotel application where some resorts may not disclose some of their properties. These undisclosed properties are represented as pristine entries within the resort multi-dimensional point representation.
2. With the exception of [2], all skyline algorithms assume transitivity from the dominance relation, i.e., if information item p_i dominates p_j while p_j dominates p_k, then p_i dominates p_k.

In query processing techniques like Dynamic query processing location is continuously changing thereby changing skyline results frequently. The efficiency if skyline is computed in terms of accessing the data points and organizing the skyline result. Dynamic skyline queries report all points that are not dominated by any other points and it's calculated depending on the distance between data point and query point. Dynamic Skyline query in graph (DSG) that receives an attractive attention and it find out by employing some pruning techniques. In chart shortest space is used to measure distance between two

vertices as opposed to using Euclidean distance. Dist(x, y) that indicate the distance between vertices and query stage. To get a DSG [8] query that mainly includes two challenges

1. Handle with enormous search space
2. Expensive calculation to find the shortest route computation. Common Shortest route pruning technique is used to exploit unnecessary route. DSG [8] that uses refine and filter frame to handle the dynamic skyline query and through the filtering process it prune all the points which are dominated by some other points and generate pair of candidate vertices by perform BNL algorithm also it generate skyline result set.

Many service providers exist who offer powerful storage and computational infrastructures at low cost .However, these service providers are not fully trusted, and typically behave in honest-but-curious fashions. Specifically, they follow the protocol to answer the queries correctly, but they also collect the locations of the POIs and the subscribers for other purposes. Leakage of POI locations can lead to privacy breaches as well as financial losses to the data owners, for whom the POI dataset is an important source of revenue. Disclosure of user locations leads to privacy violations and may deter subscribers from using the service altogether. The skyline query is one very important query for users' decision making [1], [2]. In [4] an algorithm SFS has been developed, based on pre-sorting that is general, for use with any skyline query, efficient, and well behaved in a relational setting. The skyline, or Pareto, operator selects those tuples that are not dominated by any others. Extending relational systems with the skyline operator would offer a basis for handling preference queries. In [5] an algorithm to compute skyline has been developed which unlike most existing algorithms that compute the Skyline in a batch, returns the first results immediately, produces more and more results are continuously, and allows the user to give preference during the running time of the algorithm so that the user can control what kind of results are produced next.

Problem Identification

Currently there are some major problems that arise while sharing data across network which include:

- Entities specialized in various areas of interest (e.g., certain niche directions in arts, entertainment, and travel) gather large amounts of geo-tagged data that appeal to subscribed users.
- Leakage of Points of Interest (POI) locations can lead to privacy breaches as well as financial losses to the data owners, for whom the POI dataset is an important source of revenue.

The aim of the current work is to reduce the use of computational resources for processing the location based data generated during the searching of nearby locations by integrating Skyline query processing with Nearest Neighbour search.

Methodology

The large dataset obtained as search result consumes a lot of server resources and time which effects the client response time generated by server. Dynamic skyline query processing is added so as to efficiently reduce the resource consumption and memory requirements of server. In case of Dynamic queries the query location changes continuously so that skyline results also change frequently. The efficiency of skyline is computed in terms of accessing the data points and organizing the skyline result. In the improved approach utilize the Euclidean distance and treat closer objects as candidate skyline objects. It is assumed that the dominance relationship can be defined by the user as an additional category attribute as per the requirement like price, pin code, discount etc. For efficient query processing, we employ the concept of nearest neighbor queries and evaluate progressively whether each object belongs to the skyline. If an object is not dominated by other objects in terms of the distance, price and discount and all the category attributes, it belongs to the skyline. The proposed approach uses Dynamic query which means the query location is continuously changing so that skyline results are change frequently. The efficiency of skyline is computed in terms of accessing the data points and organizing the skyline result. A dynamic skyline query reports all points not dominated by other points and its calculation is based on the distance between data point and query point. The current work uses filter and refine framework during the filtering process it prune all the points that are dominated by some other points and generate set of candidate vertices by using BNL algorithm. The algorithm is used to join two relations in a relational database. This algorithm is a variation on the simple nested loop join used to join two relations R and S.

Basic Approach

The client has a query point Q and wishes to find the point's nearest neighbors. The query is encrypted using AES algorithm and sent to the server. The server receives the encrypted query and performs decryption using AES algorithm. The query is executed and K nearest neighbors is generated. The optimization is performed by integration of skyline query parameters for filtering out the points which are not dominated by any other point. The system model comprises of two distinct entities:

- The data owner with the location data.
- The client requesting for the nearest locations.

K nearest neighbors algorithm stores all available cases and classifies on the basis of similarity measure (e.g., distance functions). KNN can be used for both classification and regression predictive problems. However, it is more widely used in classifications problems in the industry. To evaluate any techniques we generally look at 3 important aspects:

- Ease to interpret output
- Calculation time
- Predictive Power

Nearest Neighbor Algorithm

1. Retrieve all the dataset of a particular location using Google search API.
2. Filter out the nearest location x by finding the location(x_i, y_i)that is nearest to latitude longitude of the searched location according to Euclidean distance:

   $$\|x - x_i\| = \sqrt{\sum_j (x_j - x_{ij})^2} \ldots (1)$$
3. Sort the dataset according to distance.

Experimental Results

A comprehensive evaluation on the proposed Skyline query is conducted. Experimental results demonstrate that the proposed algorithm is much faster than using a traditional Nearest Neighbor Search. Skyline also removes unwanted data from memory. So the memory space is been efficiently used.

Figure 1: CPU usage for processing data only with Nearest Neighbor Search

The results were recorded on the basis of average time taken (figure 1) to perform transactions during Nearest Neighbor Search excluding skyline query processing. The graph shows average transaction response times for three virtual locations. The transactions took 8,500 seconds to perform (on average) when using the Durg to Bhopal virtual location.

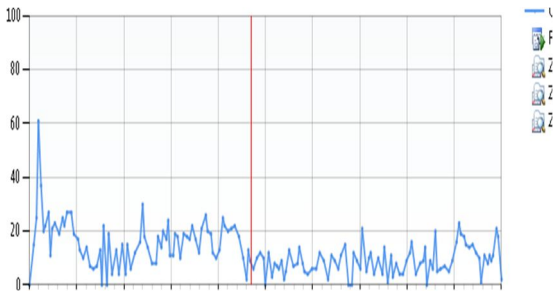

Figure 2: CPU usage for processing data Nearest Neighbor Search with Skyline processing

The figure 2 shows that the approach is much better in terms of CPU usage which enables the usage of server resources in more efficient manner. The results were recorded on the basis of average time taken to perform transactions during Nearest Neighbor Search including skyline query processing. The graph shows average transaction response times for three virtual locations. The transactions took 8,500 seconds to perform (on average) when using the Durg to Bhopal virtual location, the longest time for any of the three virtual locations.

Again the results were recorded (figure 3) on the basis of average time taken to perform transactions during Nearest Neighbor Search excluding skyline query processing. The graph shows average transaction response times for three virtual locations. The transactions took on an average 10.12 seconds to perform.

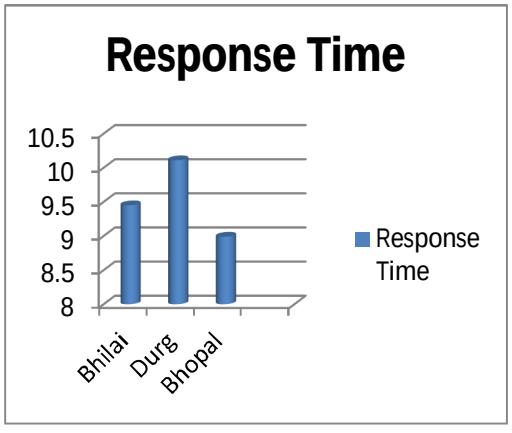

Figure 3: Response time excluding skyline query processing

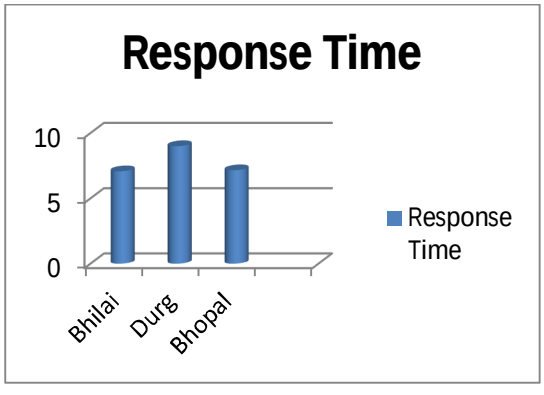

Figure 4: Response time including skyline query processing

The above readings (figure 4) clearly show that the response time for skyline query processing performs is efficient and better. The graph shows average transaction response times for three virtual locations. The transactions took on an average 9,000 seconds to perform.

Conclusion and Future Work

Recently skyline query processing that receives an interesting attention in data mining field. Skyline queries regain the non-dominated points out of a large database system based on the user preference therefore that it may be utilised in preference based applications. It successfully eliminates all of the dominated points by utilizing some effective technique. The recent addresses some major issues of closest place search like slow response time and heavy memory use on servers integrating the dynamic skyline processing technique. Experimental results reveal that the approach outperforms the conventional Nearest Neighbour search in terms of resource usage and reaction time. Future functions include research in areas of pre-processing of reaction before sending to client. Additionally a comparative study is required to determine which indexing structure is useful when various parameters of the computation like database cardinality, number of querying dimensions, communication cost, frequency of the replicate access of the data documents etc. are placed on the test. For cellular environments devices have inherent limitations of restricted processing memory. Therefore to minimize the communication and query execution costs, the participating datasets must be partitioned in this manner that no duplicity of processing attempts occur when data have a tendency to be duplicate.

References

[1] P.Rumao&P.BhorSecurekNN Query Processing in Untrusted Cloud Environments ,IJESC,Vol 6 Issue No. 7 ,2016.
[2] R. R. Meshram,Skyline Query Processing In Location Based Application , IJPRET, 2016; Volume 4 (9): 861-866

[3] S. Borzsonyi, D. Kossmann, K.Stocker, The Skyline Operator, in: Proc. Int'l Conf. Data Eng, pp. 421-430, 2001.
[4] J. Chomicki, Godfrey,J. Gryz , D. Liand ,Skyline with Presorting, in: Proceedings of ICDE, pp. 717–816,2003.
[5] D. Papadias, Y. Tao, G. Fu, B. Seeger,Progressive skyline computation in database systems, in: ACM TODS 30(1), 41–82, 2005.
[6] C. Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang, "Finding k-Dominant Skylines in High Dimensional Space," in Proceedings of the ACM International Conference on Management of Data, SIGMOD, 2006.
[7] L.Zou,L.chen,M.T.Ozsu,D.Zhao ,Dynamic Skyline Queries in Large Graphs, 15th International Conference, DASFAA 2010, Tsukuba, Japan, April 1-4, 2010, Proceedings, Part II Volume 5982, 2010, pp 62-78.
[8] D. Kossmann, F. Ramsak, and S. Rost, "Shooting Stars in the Sky: An Online Algorithm for Skyline Queries," in Proceedings of the International Conference on Very Large Data Bases, VLDB, 2002.
[9] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "Progressive skyline computation in database systems," ACM Transactions on Database Systems, TODS, vol. 30, no. 1, pp. 41–82, 2005. [6] K.-L. Tan, P.-K.Eng, and B. C. Ooi, "Efficient Progressive Skyline Computation," in Proceedings of the International Conference on Very Large Data Bases, VLDB, 2001.
[10] Y. Yuan, X. Lin, Q. Liu, W. Wang, J. X. Yu, and Q. Zhang, "Efficient Computation of the Skyline Cube," in Proceedings of the International Conference on Very Large Data Bases, VLDB, 2005.