Optimal Visible Nearest Neighbor Query Processing

PD Mahendhiran
Assistant Professor, Department of Information Technology, Karpagam College of Engineering, Coimbatore
maha301@gmail.com

R.Ranjani
Assistant Professor, Department of Computer Science and Engineering, Karpagam University, Coimbatore
ranjanircse@gmail.com

Abstract — In general, encountered type of query is to find the k nearest neighbor objects to a given query point. In many applications interested only in the objects that are directly visible to query points. In this paper, we formulate invisible objects which are not directly visible to query points. We can retrieve the invisible objects with the help of visible objects which are nearer to the query point. We also propose an algorithm efficiently to process the Aggregate visible k nearest neighbor query to maintain the query updation. An aggregate k nearest neighbor query finds k objects with the smallest aggregate distance to a query point. An aggregate NN queries constitute a generalized form of nearest neighbor search, where there are multiple query points. Our algorithm dramatically reduces the time complexity during the search process.

Keywords- *query processing, spatial databases, aggregate nearest neighbor quries, query point, objects.*

1. Introduction

In the spatial applications we are very much familiar with the objects which are directly visible to the user's views[1]. If a person is met with an accident immediately what we have to do is that person should be admitted in the specialized hospitals. At that moment we don't know which hospital is specialized for which type of injuries. So we are going to give the queries to the collected databases which are having the information about all the hospitals. In this paper Nearest Neighbor queries the author has introduced the mechanisms about k-Nearest Neighbor queries.

For that before they have used efficient branch-and-bound, and R-tree traversal algorithm to find the nearest object to the query point. The search process will be focused on only the potential neighbors. In this method the proposed solution is a top-down recursive algorithm. It backtracks and exploring the sub trees which is having potentially neighbor until it has no sub tree to be visited. For this problem we are going for the another concept that is called as aggregate visible k nearest neighbor queries.(AVkNN). Some objects are directly visible to the query point[3]. But they are not concentrated on the visibility of the object, which is not visible to the query point. They might not be directly visible to the query point but they are visible to the query point visibility objects of objects. From the many number of objects used to find which the object is very much nearer to the query point.

To find the invisible object which is hidden by the visible objects from the query point has found using two algorithms. First one is pre-pruning algorithm and second one is post-pruning algorithm.

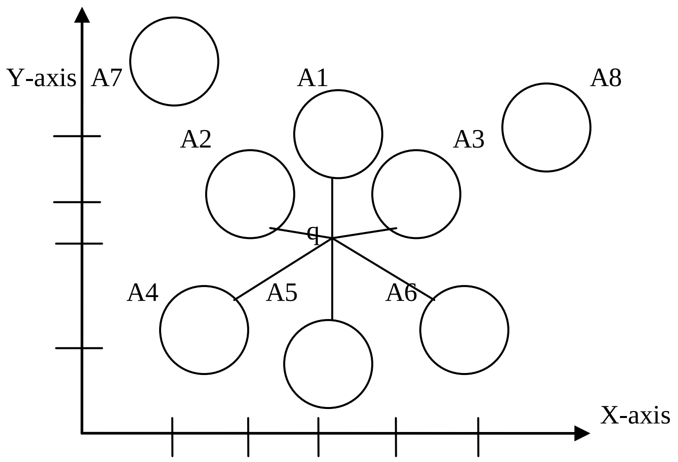


Fig1: Finding nearest neighbor

In Fig.1 the objects A1, A2, A3, A4, A5, A6, are directly visible to the query point. The objects A7, A8 are not directly visible to the query point. We need to consider about the invisible objects with the help of visible objects which is nearer to the query point. In this we are going to use distance metrics. Using the distance metrics we are going to prune the objects which is very much closer to the query point.

The rest of the paper contains the follows: section 2 discusses the related works on aggregate visible nearest neighbor queries. Section 3 provides preliminaries on the distance metrics and optimal solution for aggregate visible k nearest neighbor queries. Section 4 concludes the future research directions.

2. Related work

Our algorithm for Aggregate nearest neighbor query is based upon R-tree [2,4,5]. R-tree is a simplicity, popularity, and also . Each and every R-tree node is associated with a Minimum Bounding Rectangle (MBR). There are two important distances from a minimum bounding rectangle to a query point. The distances are MINMAXDIST and MINDIST [5].

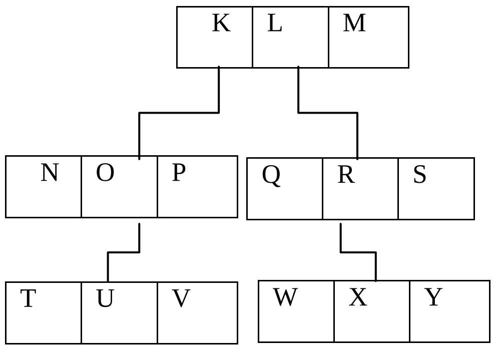


Fig2: R-tree

R-tree is represented by binary tree formation which is contains one root tree and many number of sub tree and child. That is root node can have more number of sub tree and child node.

2.1 Minimum And Maximum Distance:

Commonly used distance estimators, such as MAXDIST, MINMAXDIST, and MINDIST. MINMAXDIST distance is introduced to compute the minimum value of the maximum distances between the query point and objects on the each of the n axes respectively [6]. MINDIST is used to estimates the distance from the query point to any enclosed Minimum Bounded Rectangle (MBR) or and also it estimates the data object as the minimum distance from the point to the minimum bounded rectangle itself [7]. That is The MINDIST between the query point q and an MBR X is the smallest Euclidean distance between q and X.

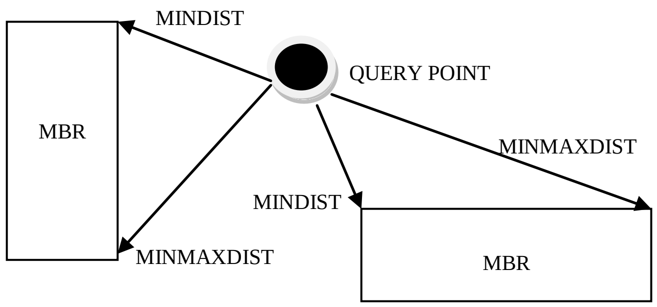


Fig3: MINDIST and MINMAXDIST

We consider Euclidean distance and 2D point datasets indexed by R-tree, but the proposed techniques are applicable to higher dimensions and alternative data partition access methods. There is one important differentiation in both MINDIST and MINMAXDIST [9]. MINDIST is the most optimistic choice and MINMAXDIST produces the most pessimistic ordering. And also MAXDIST is a pessimistic. Because the MAXDIST and MINIMAXDIST of an MBR are greater than or equal to the distance of the nearest object in that MBR.

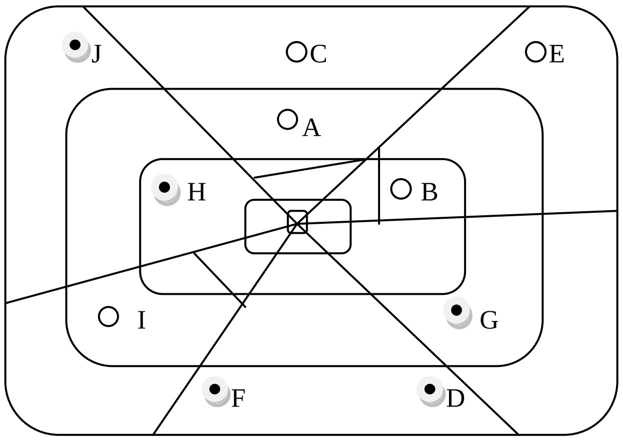
3. Aggregate Query Processing

In this section we present a data behavioral algorithm to maintain and manage the query updation. We first discuss the metrics of using query processing then we are present pre pruning and post pruning in order and prune the search tree

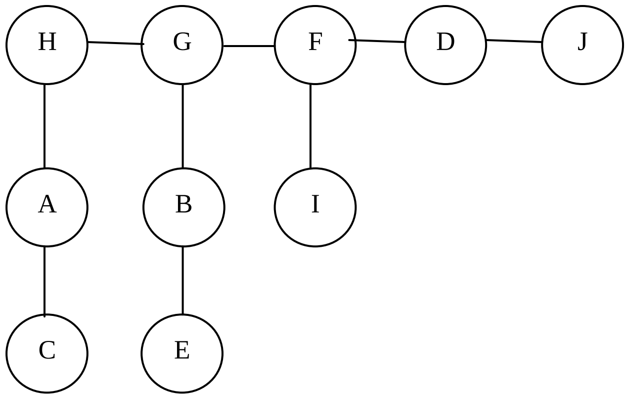
3.1. Generalization: Finding the invisible objects:

Some objects are directly visible to the query point. But they are not concentrated on the visibility of the object, which is not visible to the query point. They might not be directly visible to the query point but they are visible to the query point visibility objects of objects. From the many number of objects used to find which the object is very much nearer to the query point.

To find the invisible object which is hidden by the visible objects from the query point has found using two algorithms. First one is pre-pruning algorithm and second one is post-pruning algorithm.



fig(4) : Objects with obstacles



fig(5) : Finding invisible objects

In fig(4) we are going to find the invisible nodes from the visible nodes using pre pruning and post pruning algorithms.

Some objects are directly visible to the query point. Some set of objects might not be directly visible to the query point. But they are not concentrated on the visibility of the object, which is not visible to the query point. We need to consider about the invisible objects with the help of visible objects which is nearer to the query point. From the many number of objects used to find which the object is very much nearer to the query point. In fig (4) there are few number of objects such as A,B,C,D,E,F,G,H,I,J. Here the objects H, G, F, D, J are the visible objects and the objects A, C, B, E, I are invisible objects. In fig (5) We are going to find the invisible objects with the help of visible objects which is nearer to the query point. We can find out the objects A, C(invisible) with the help of H(visible) object. Because the object H is nearer to query point. We find the A, C (invisible) objects through J (visible) object. But one of our aims is to find k objects with smallest aggregate distance to query points. And also we can find out B, E (invisible) objects with the help of G (visible) object and also we can find the I (invisible) objects with the help of F (visible) object.

To find the invisible object which is hidden by the visible objects from the query point has found using two algorithms. First one is pre-pruning algorithm and second one is post-pruning algorithm.

The concept of the safe region provides a more effective way to achieve continuous answers to location-based spatial queries. In a safe-region-based method, an answer is returned with a safe region. As long as the query point stays in the safe region, the answer remains the same. When the query point moves out of the safe region, another answer with its associated region is returned. Therefore, a safe-region based method always (that is, continuously) provides accurate answers without the need for sampling. This approach also requires much less frequent communication between the mobile client and the server.

3.2. Aggregate query processing for NN:

In gereral, An aggregate k nearest neighbor query finds k objects with the smallest level aggregate distance to query points. There are three types of aggregate functions available. These are MAX, MIN, SUM. And also there are three types of Aggregate search regions (AGGSRs), MAXSR, MINSR, SUMSR. For example, Panjabi's that wants to open a restaurant in order to attract the maximum level number of customers[8]. A weighted *sum* ANN query results the location that minimizes the sum of distances. Then assume that a military unit wants to identify a collection point for soldiers. A *max* ANN query outputs the point that leads to the earliest pick-up time. A *min* ANN reports the cities under the highest potential danger based on their proximity to any phenomenon. Aggregate query processing works with two typical works namely Single Retrieval Front (SRF) and Multiple Retrieval Front (MRF). In MRF approach, MRF issues a VkNN query at each query point to retrieve objects, whereas SRF issues just one aggregate query to retrieve objects from the database[7]. Both approaches use a separate priority queue to rerank the retrieve objects according to the aggregate visible distance metric. In existing there is no updation for data retrieval query processing. If any moving objects changes to one place to another place inside the memory, they won't be upgrade the current status of the memory. So the result won't be correct. That is it shows there is no object in memory if it is in memory. To avoid this problem I propose to efficient data behavioral algorithms for maintain the query updation. This will improve the execution speed and also time and space complexity will be reduced. In addition I propose to find the invisible object which is not visible to the query point.

4. CONCLUSIONS

In this paper, we introduced new type of concept, retrieving invisible objects from database. Furthermore, we presented the update actions to be stored into the database for retrieving visible and invisible objects. PREPRUNING and POSTPRUNING algorithms build up the visible nearest objects and invisible objects are retrieved. PREPRUNING uses MINVIDIST and also POSTPRUNING uses MINVIDIST for result ranking and MINDIST for branch ordering. In PrePruning there are two variations, PrePruning-MinDist and PrePruning-MinVisiDist. These two variations put more effort on the visibility pruning in order to reduce the data-retrieval cost. This could be beneficial in settings with disk-based or network-based storage where the data-retrieval costs are more critical than the CPU costs. In future work, we will update each and every prune the search process of visible and invisible object. In addition we are interested in efficient data behavioral algorithms for maintain the query updation. This will improve the execution speed and also time and space complexity will be reduced.

REFERENCES

[1] Sarana Nutanong, Engemen Tanin, and Ruhi Zhang, "Incremental Evaluation of Visible Nearest Neighbor Queries" IEEE Transaactions on Knowledge and Data Engineering vol 22, no 5, May 2010.

[2] T. Asano, T. Asano, L.J. Guibas, J. Hershberger, and H. Imai, "Visibility of Disjoint Polygons," Algorithmica, vol. 1, no. 1, pp. 49-63,1986.

[3] T. Asano, T. Asano, L.J. Guibas, J. Hershberger, and H. Imai, "Visibility-Polygon Search and Euclidean Shortest Paths," Proc. 26th Ann. Symp. Foundations of Computer Science (FOCS), pp. 155-164, 1985.

[4] K.C.K. Lee, W.C. Lee, and H.V. Leong, "Nearest Surrounder Queries," Proc. IEEE Int'l Conf. Data Eng. (ICDE), pp. 85-94, 2006.

[5] S. Nutanong, R. Zhang, E. Tanin, and L. Kulik, "The V_-Diagram:
A Query Dependent Approach to Moving kNN Queries," Proc. 29th Int'l Conf. Very Large Data Bases (VLDB), pp. 1095-1106, 2008.

[6] N. Backmann, H.-P. Kriegel, R. Schneider, and B. Seegar. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. In *Proceedings of the ACM SIGMOD Conf., Atlantic City, NJ, May 23-25*, pages 322–331, 1990.

[7] N. Roussopoulos, S. Kelly, and F. Vincent. Nearest Neighbor Queries. In *Proceeding of the 1995 ACM SIGMOD Conf., San Jose, CA, May 22-25*, pages 71–79, 1995.

[8] Nick Roussopouls Stephen Kelley Frederic Vincent, Nearest Neighbor Queries, Department of Computer Science, University of Maryland, College Park, MD 20742

[9] Sarana Nutanong, Egemen Tanin, and Zhang. Incremental Evaluatioon of Visible Nearest Neighbor Queries.