

Improved Processing of Top-k Spatial Data by Quality Preferences

G.Harikrishnaveni¹, S.Shenbapriya², K.Tamilselvi³

¹Assistant Professor, Department of Computer Science and Engineering, P.A. College of Engineering and Technology, Pollachi

² Assistant Professor, Department of Computer Science and Engineering, P.A. College of Engineering and Technology, Pollachi

³Assistant Professor, Department of Computer Science and Engineering, P.A. College of Engineering and Technology, Pollachi

Abstract— Ranking a Spatial Preference query ranks the objects based on the qualities of features in their spatial neighborhood. Neighborhood concepts can be specified by the user through the different function. In Spatial neighborhood the distance should be calculated between each flats. The weights are assigned based on the quality of features. Each flat contain multiple features, the features are combined and then flats are ranked based upon the weight values. This should be implemented using Top-k spatial preference Queries. In Top-k spatial preference queries Indexing techniques and Search algorithms can be used. A customer may want to rank the flats with respect to the location, defined after sum upping of the qualities of other features within a distance range. Top-k spatial preference queries returns a ranked set of the k best data objects based on the scores of feature objects in their spatial neighborhood. The proposed technique is based on a new index structure called the R+ Tree to store segmented rectangular records.

Keywords— Spatial databases, Query processing, Nearest Neighborhood.

I. INTRODUCTION

Spatial database systems manage large collections of geographic entities, apart from spatial features contain non spatial information (e.g., type, name, price, size, etc.). Here, we study an interesting type of preference queries, which select the top spatial location with respect to the quality of facilities in its spatial neighborhood. The score of an object is defined by the quality of features in its spatial locality. For example, a real estate agency office that holds a database with available flats for lease. Here "feature" refers to a class of objects in a spatial map such as specific facilities A customer may want to rank the contents of this database with respect to the quality of their locations, quantified by aggregating non spatial characteristic of other features in the spatial neighborhood of the flat the user wishes to find a hotel that is close to a high-quality restaurant and a high quality hotels.

1)The maximum quality of each feature in the neighborhood region and the aggregation of those qualities. A simple score instance, called the range score, binds the neighborhood place to a circular

region, and the aggregate function to SUM. Traditionally, there are two basic ways for ranking flats.

1) Spatial ranking, which orders the objects according to their distance from a reference point

2) Non Spatial ranking, which orders the flats by an collective function on their non spatial values.

Our Top-k spatial preference query joins these two types of ranking in an perceptive way. A brute-force approach for evaluating it to compute the scores of all objects in D and select the top-k ones. In this paper, we propose different techniques that aim at minimizing the I/O accesses to the object and feature data sets, while being also computationally efficient. Our techniques apply on spatial partitioning access methods and compute upper score bounds for the objects indexed by them, which are used to effectively prune the search space. Specifically, we contribute the branch-and-bound (BB) algorithm and the feature join (FJ) algorithm for efficiently processing the top-k spatial preference query. This paper studies three types of extensions for score calculation .the first one is optimized version of BB that exploits a more efficient technique for computing the scores of the objects. The second extension is adaptations of the proposed algorithms for aggregate functions other than SUM, e.g., the functions MIN and MAX. The third extension develops solutions for the top-k spatial preference query based on the influence score. The most popular spatial access method is the R-tree, which indexes minimum bounding rectangles (MBRs) of objects. R-trees can efficiently process main spatial query types, including spatial range queries, nearest neighbor queries, and spatial joins. Given a spatial region , Object ranking is a popular retrieval task in various applications. In relational databases, we rank the tuples based on aggregate score function on their attribute values. For example, a real estate agency maintains a dataset that contains information of flats available for rent. A potential customer wishes to view the top-5 flats with the largest sizes and lowest prices. The score of each flat is expressed by the sum of two qualities: size and price, after normalization to the domain. In spatial databases, ranking is associated to nearest neighbor (NN) retrieval. Given a query location, we are interested in retrieving the set of nearest objects to it that qualify a condition (e.g., restaurants). Assuming that the set of interesting objects is indexed by an R+-Tree, we can apply distance bounds and traverse

the index in a branch-and-bound fashion to obtain the answer. Nevertheless, it is not always possible to use multidimensional indexes for top-k retrieval. First, such indexes break-down in high dimensional spaces. Second, top-k queries may involve an arbitrary set of user-specified attributes (e.g., size and price) from possible ones (e.g., size, price, distance to the beach, number of bedrooms, floor, etc.) and indexes may not be available for all possible attribute combinations (i.e., they are too expensive to create and maintain). Third, information for different rankings to be combined (i.e., for different attributes) could appear in different databases (in a distributed database scenario) and unified indexes may not exist for them. Solutions for top-k queries focus on the efficient merging of object rankings that may arrive from different (distributed) sources. Their motivation is to minimize the number of accesses to the input rankings until the objects with the top-k aggregate scores have been identified. To achieve this, upper and lower bounds for the objects seen so far are maintained while scanning the sorted lists.

II. RELATED WORK

There is no existing efficient solution for processing the top-k spatial preference query. A brute-force approach for evaluating it is to compute the scores of all objects in D and select the top-k ones. This method, is expected to be very expensive for large input datasets.

A. Spatial Ranking

Spatial ranking, which orders the objects according to their distance from a reference point.

B. Non Spatial Ranking

Non-spatial ranking, which orders the objects by an aggregate function on their non-spatial values. Our Top-k spatial preference query integrates these two types of ranking in an perceptive way. As mentioned in our examples, this new query has a wide range of applications in service recommendation and decision support systems. To this knowledge, there is no existing efficient solution for processing the Top-k spatial preference query. To this knowledge, there is no existing efficient solution for processing the top-k spatial preference query. Object ranking is a popular retrieval task in different applications. In relational databases, we rank tuples based on aggregate score function on their attribute values. For example, a real estate agency maintains a dataset that contains information of flats available for rent. A potential customer wishes to view the top-5 flats with the largest sizes and lowest prices. In this case, the score of each flat is expressed by the sum of two qualities: price and size, after normalization to the domain. In spatial databases, ranking is often associated to nearest neighbor (NN) retrieval. Based on given aquery location, we are interested in retrieving the set of nearest objects to it that qualify a condition assuming that the set of interesting objects is indexed by an R-tree, we can apply distance bounds and traverse the index in a branch-and-bound fashion to obtain the answer.

H.Samet [4] In this paper we describe an R-Tree which represent data objects by intervals in several dimensions. An R-

Tree is a height-balanced tree similar to a B-Tree with index records in its leaf nodes containing pointers to data objects. Exhaustive algorithm is used to find the minimum area node split is to generate all possible groupings and choose the best splitting. A Quadratic cost algorithm is used to find a small area split, but is not generated to find one with the smallest area possible. The linear node split algorithm proved to be as good as more expensive techniques. It was fast

Weber[5] Survey of the conventional approach to supporting similarity searches in HDVS(s) is to use a multidimensional index structure. space partitioning methods like grid file[27],K-D-B-Tree[28].Data Partitioning index trees such as R-Tree[21],R+ Tree[30]. We study the performance of both space and data partitioning methods at high dimensionality from a theoretical and practical point of view. In cost-model based analysis in the area of HDVSSs.Early work in this area did not address the specific difficulties of high dimensionality. The HDVSS ultimately become linear at high dimensionality.I.F.

Ilyas [8] Survey of rank join algorithm that makes use of the individual orders of its inputs to produce join results ordered on a user specified scoring function. Users often specify multiple features to evaluate the similarity between the query media and the stored media. A closely related problem is supporting Top-k selection queries. In Top-k selection queries, the goal is to apply a scoring function on multiple attributes of the same relation to select tuples ranked on their combined score.Top-k selection queries over relation databases can be mapped in to range queries using high dimensions histograms[1].In [13], Top-k selection queries are evaluated in relation query processors by introducing a new pipelined join operator termed NRA-RJ.NRA-RJ Modifies the NRA algorithm to work on range of score instead of requiring the input to have exact score.This experiments prove the concept and show a significant performance enhancement,especially for low values of join selectivity.C.

Boehm[21] Survey of analyze different nearest neighborhood algorithms, a large structures have been proposed for nearest neighborhood search algorithms for nearest neighborhood search may be divided in to two major groups partitioning algorithms. Partitioning algorithm partition the data space recursively and store information above the partitions in the nodes. Graph based algorithms precalculate some nearest neighborhood of points, store the distance in a graph information for a more efficient search. Our cost model is accurate even in high dimensions, where other models completely fail, because our model considers boundary effects. As a further advantage, our model uses the Euclidean metric which is relevant to many database applications.

III. SPATIAL PREFERENCE QUERIES

In this project, we look forward top-k spatial preference queries, which provides a novel type of ranking for spatial objects based on qualities of features in their neighborhood. So, we use five algorithms for processing top-k spatial preference queries. The baseline algorithm SP computes the scores of every object by querying on feature datasets. The algorithm GP is a variant of SP that reduces I/O cost by computing scores of objects in the same leaf node concurrently. The algorithm BB derives upper bound scores for non-leaf entries in the object tree, and prunes those that cannot lead to better results. The algorithm BB* is a variant of BB that utilizes an optimized method for computing the scores of objects. The algorithm FJ performs a multi-way join on feature trees to obtain qualified combinations of feature points and then search for their relevant objects in the object tree. And also in this project we are using three relevant extensions that have not been investigated in our preliminary work. The first extension is an optimized version of BB that exploits a more efficient technique for computing the scores of the objects. The second extension is that adaptations of the proposed algorithms for aggregate functions other than SUM, e.g., the functions MIN and MAX. The third extension develops solutions for the top-k spatial preference query based on the influence score.

A. R-Tree

R-tree is a hierarchical data structure based on B+-trees. They are used for the dynamic organization of a set of d-dimensional geometric objects representing them by the minimum bounding d-dimensional rectangles. Each node of the R-tree corresponds to the MBR that bounds its children. The leaves of the tree contain pointers to the database objects instead of pointers to children nodes. The nodes are implemented as disk pages. It must be noted that the MBRs that surround different nodes may overlap each other. Besides, an MBR can be included in many nodes, but it can be associated to only one of them. This means that a spatial search may visit many nodes before confirming the existence of a given MBR. Also, it is easy to see that the representation of geometric objects through their MBRs may result in false alarms. To resolve false alarms, the candidate objects must be examined.

B. R+ Tree

R+-Tree is a tree data structure, a variant of the R tree, used for indexing spatial information. R+-Tree is a variant of R-trees used for indexing spatial information. R+-Trees support point the spatial data at the same time with a slightly higher cost than other R-Trees. There is little overlap in this tree, resulting in good query performance.

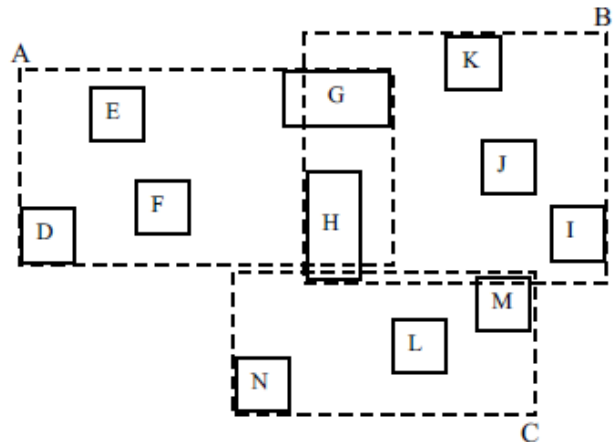


Fig. 1 R Tree data MBR and its MBR

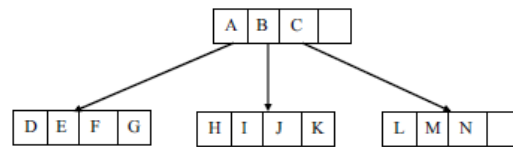


Fig. 2 Corresponding R-tree

C. Index structure of special preference queries

Formally defines the top-k spatial preference query problem and describes the index structures for the datasets. In this paper, we assume that the object dataset D is indexed by an R-tree and each feature dataset F_c is indexed by an MAX R-tree, where each non-leaf entry augments the maximum quality (of features) in its subtree. Nevertheless, our solutions are directly applicable to datasets that are indexed by other hierarchical spatial indexes (e.g., point quad-trees). The rationale of indexing different feature datasets by separate R-trees is that: (i) a user queries for only few features out of all possible features and (ii) different users may consider different subsets of features. We proceed to elaborate the aggregate function and the component score function. Typical examples of the aggregate function AGG are: SUM, MIN, Max well first focus on the case where AGG is SUM.

D. Group probing algorithm

Due to separate score computations for different objects, SP is inefficient for large object datasets. In view of this, we propose the group probing algorithm (GP), a variant of SP, that reduces I/O cost by computing scores of objects in the same leaf node of the R-tree concurrently. In GP, when a leaf node is visited, its points are first stored in a set V and then their component scores are computed concurrently at a single traversal of the tree. We now introduce some distance notations for MBRs. shows the procedure for computing the c-th component score for a group of points.

E.Branch and Bound Algorithm

GP is still expensive as it examines all objects in D and computes their component scores. We now propose an algorithm that can significantly reduce the number of objects to be examined. Is a pseudo-code of our branch and bound algorithm (BB), based on this idea. BB is called with N being the root node of D. In order to obtain points with high scores early, we sort the entries in descending order of T before invoking the above procedure recursively on the child nodes pointed by the entries in V. If N is a leaf node, we compute the scores for all points of N concurrently and then update the set Wk of the top-k results.

F.Upper Bound Score Computation

It remains to clarify how the upper bound scores of non-leaf entries within the same node N can be computed concurrently. Our goal is to compute these upper bound scores such that the bounds are computed with low I/O cost, and the bounds are reasonably tight, in order to facilitate effective pruning. we utilize only level-1 entries in Fc for deriving upper bound scores because: (i) there are much fewer level-1 entries than leaf entries(ii) high level entries in Fc cannot provide tight bounds. In our experimental study, we will also verify the effectiveness and the cost of using level-1 entries for upper bound score computation.

G.Optimized Branch and Bound Algorithm

This section develops a more efficient score computation technique to reduce the cost of the BB algorithm. BB algorithm are used to compute the scores of object points.

H.Optimized computation of scores

Based on our observation, we propose a tighter derivation for the upper bound score. we access the feature trees in a round-robin fashion, and traverse the entries in each feature tree in descending order of quality values. Round-robin is a popular and effective strategy used for efficient merging of rankings. Alternative strategies include the selectivitybased strategy and the fractal-dimension strategy. These strategies are designed specifically for coping with high dimensional data, however in our problem setting they have insignificant performance gain over round robin

I.The BB* Algorithm

We extend BB Algorithm to an optimized BB* algorithm for computing the exact scores for object points in the set V and for deriving the upper bound scores for non-leaf entries in V.

J.Feature Join Algorithm

An alternative method for evaluating a top-k spatial preference query is to perform a multi-way spatial join on the feature trees to obtain combinations of feature points which can be in the neighborhood of some object from D. Spatial regions which correspond to combinations of high scores are then examined, in order to find data objects in D having the corresponding feature combination in their neighborhood. In this section, we first introduce the concept of a combination, then discuss the conditions for a combination to be pruned, and finally elaborate the algorithm used to progressively identify the combinations that correspond to query results. more expensive than BB* in both I/O and time.

Table1: Effect of the Aggregate Function,Range Score

SUM function	SP	GP	BB	BB*	FJ
I/O	350927	22594	2033	1535	489
Time (s)	635.0	32.7	3.0	2.0	1.3

MIN function	SP	GP	BB	BB*	FJ
I/O	235602	16254	611	615	47
Time (s)	426.8	22.7	0.9	0.8	0.2

MAX function	SP	GP	BB	BB*	FJ
I/O	402704	26128	228	186	8
Time (s)	742.8	38.2	0.3	0.2	0.1

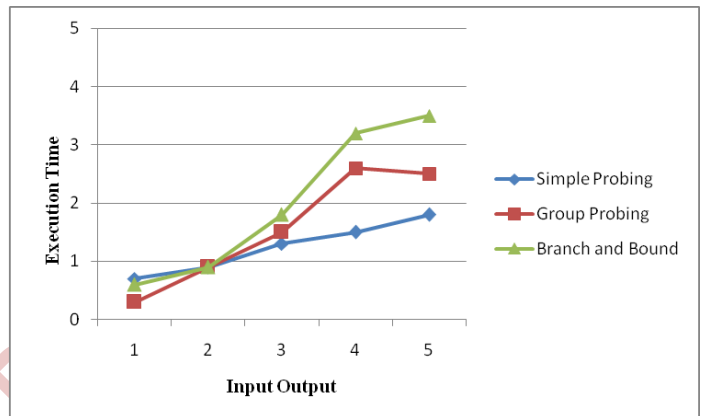


Fig 3:Algorithm Efficiency

IV. RESULT

Ranking systems have made significant progress in recent years and many techniques have been proposed to improve the ranking quality. However, in most cases, new techniques are designed to improve the accuracy of ranking, ranking spatial data according to the predicted rating values provides good predictive accuracy. Therefore, in this paper, we proposed a number of ranking techniques that can provide significant improvements in ranking values, the score value of R-Tree contains all the nearest points value. so it will take more memory, while reducing memory size and overlapping we go to another technique called R+ Tree Ranking.

Table 2: R-Tree Ranking

PID	PNAME	SCORE
1009	Hillrock	45
1006	Saisruthi	42
1010	Lakeview	39
1003	Kings	37

Table 3: R+ Tree Ranking

PID	PNAME	SCORE
1009	Hillrock	6
1006	Saisruthi	5
1010	Lakeview	5
1003	Kings	4

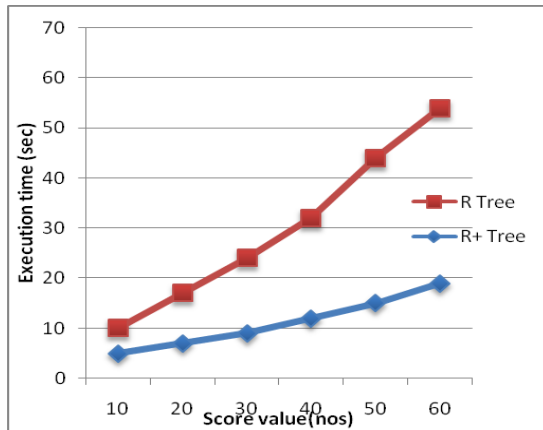


Fig 4: Execution Time Vs Score value

V. CONCLUSION

Ranking systems have made significant progress in recent years and many techniques have been proposed to improve the ranking quality. However, in most cases, new techniques are designed to improve the accuracy of ranking, In particular R-Tree and R+ Tree, while ranking according to the quality of feature values provides good predictive accuracy, it tends to perform poorly with respect to same features. Proposed many number of ranking techniques and algorithms that can provide significant improvements in ranking with only a small amount of accuracy loss. In addition, these ranking techniques offer flexibility to system designers studied top-k spatial preference queries, which provide a novel type of spatial objects based on qualities of features in their neighborhood. They are also based on features based join algorithms, are extremely efficient provide a comprehensive empirical evaluation of the proposed techniques and obtain consistent and robust aggregate diversity improvements across multiple real-world data sets and using different ranking techniques.

REFERENCES

[1] M.L. Yiu, X. Dai, N. Mamoulis, and M. Vaitis, "Top-k Spatial Preference Queries", Proc. IEEE Int'l Conf. Data Eng. (ICDE),2007.
 [2] N. Bruno, L. Gravano, and A. Marian, "Evaluating Top-k Queries over Web-Accessible Databases", Proc. IEEE Int'l Conf. Data Eng.(ICDE), 2002.
 [3] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching", Proc. ACM SIGMOD, 1984.

[4] G.R. Hjaltason and H. Samet, "Distance Browsing in Spatial Databases", ACM Trans.Database Systems, vol. 24, no. 2, pp. 265-318, 1999.
 [5] R. Weber, H.-J. Schek, and S. Blott, "A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces", Proc. Int'l Conf. Very Large Data Bases (VLDB), 1998.
 [6] K.S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is 'Nearest Neighbor' Meaningful?", Proc. Seventh Int'l Conf. Database Theory (ICDT), 1999.
 [7] R. Fagin, A. Lotem, and M. Naor, "Optimal Aggregation Algorithms for Middleware", Proc. Int'l Symp. Principles of Database Systems (PODS), 2001.
 [8] I.F. Ilyas, W.G. Aref, and A. Elmagarmid, "Supporting Top-k Join Queries in Relational Databases", Proc. 29th Int'l Conf. Very Large Data Bases (VLDB), 2003.
 [9] N. Mamoulis, M.L. Yiu, K.H. Cheng, and D.W. Cheung, "Efficient Top-k Aggregation of Ranked Inputs", ACM Trans. Database Systems, vol. 32, no. 3, p. 19, 2007.
 [10] D. Papadias, P. Kalnis, J. Zhang, and Y. Tao, "Efficient OLAP Operations in Spatial Data Warehouses", Proc. Int'l Symp. Spatial and Temporal Databases(SSTD),2001.
 [11] S. Hong, B. Moon, and S. Lee, "Efficient Execution of Range Top-k Queries in Aggregate R-Trees", IEICE Trans. Information and Systems, vol. 88-D, no. 11, pp. 2544-2554, 2005.
 [12] T. Xia, D. Zhang, E. Kanoulas, and Y. Du, "On Computing Top-t Most Influential Spatial Sites", Proc.31st Int'l Conf. Very Large Data Bases (VLDB), 2005.
 [13] Y. Du, D. Zhang, and T. Xia, "The Optimal-Location Query", Proc. Int'l Symp. Spatial and Temporal Databases (SSTD), 2005.
 [14] D. Zhang, Y. Du, T. Xia, and Y. Tao, "Progressive Computation of The Min-Dist Optimal-Location Query", Proc. 32nd Int'l Conf. Very Large Data Bases (VLDB), 2006.
 [15] Y. Chen and J.M. Patel, "Efficient Evaluation of All-Nearest-Neighbor Queries", Proc. IEEE Int'l Conf. Data Eng. (ICDE), 2007.
 [16] P.G.Y. Kumar and R. Janardan, "Efficient Algorithms for Reverse Proximity Query Problems," Proc. 16th ACM Int'l Conf. Advances in Geographic Information Systems (GIS), 2008.
 [17] M.L. Yiu, P. Karras, and N. Mamoulis, "Ring-Constrained Join: Deriving Fair Middleman Locations from Pointsets via a Geometric Constraint", Proc. 11th Int'l Conf. Extending Database Technology (EDBT), 2008.
 [18] M.L. Yiu, N. Mamoulis, and P. Karras, "Common Influence Join:A Natural Join Operation for Spatial Pointsets", Proc. IEEE Int'l Conf. Data Eng. (ICDE), 2008.
 [19] T. Suel, and A. Markowetz, "Efficient Query Processing in Geographic Web Search Engines", Proc.ACM SIGMOD, 2006.
 [20] V.S. Sengar, T. Joshi, J. Joy, S. Prakash, and K. Toyama "Robust Location Search from Text Queries", Proc. 15th Ann. ACM Int'l Symp. Advances in Geographic, Information Systems (GIS), 2007.