Top-K Ranking based on Spatial Preferences

A.V.V.N.Swathi.#1, Mr.A.Rama Satish#2

#1 Student, Dvr & Dr. Hs Mic College Of Technology, Kanchikacherla,Krishna(dt)

#2 Assoc. professor, Dvr & Dr. Hs Mic College Of Technology, Kanchikacherla,Krishna(dt)

#1 swathiatluri18@gmail.com, #2 ramsatpm@gmail.com,

Abstract: The recent top-k spatial preference queries ranks objects based on the spatial neighborhood quality features. A spatial preference query ranks objects based on the qualities of features in their spatial neighborhood. Using appropriate nearest neighbors or indexing techniques, spatial preference queries were developed. For example, consider a road network; before a new road construction, we want to rank the requirements with respect to their spatial neighborhood like creation of a continuous right-of-way, overcoming geographic obstacles and having grades in the region. An intuition is to retrieve results by assigning higher weights to the features based on their proximity to the flat in a preferential search and vice versa in a normal search. In this paper, we formally define spatial preference queries and propose appropriate indexing techniques and search algorithms for them. Our technique applies 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.

I INTRODUCTION

Spatial database systems manage large collections of geographic entities, which apart from spatial attributes contain non-spatial information (e.g., size, type etc.). In this paper, we study an interesting type of preference queries, which select the best spatial location with respect to the quality of facilities in its spatial neighborhood.

Spatial objects in reality consist with multiple quality attributes in addition to their spatial locations. Traditional spatial queries and joins mainly focused on distances and manipulating only spatial locations, but they ignore the importance of quality attributes. The dominance comparison is suitable for comparing two objects with respect to multiple quality attributes. In this system, we study an interesting type of spatial queries, which select the best spatial location with respect to the quality of facilities in its spatial neighborhood. Given a set D of interesting objects (e.g., candidate road locations), a top-k spatial preference query retrieves the k objects in D with the highest scores. The score of a n object is defined by the quality of features in its spatial neighborhood.

For example, consider road networks which consists of different features like location of road, type, name if exist etc. Based on this features, we retrieve the top-k spatial preference query retrieves the k objects i.e., roads with highest scores.

Traditionally, there are two basic ways for ranking objects: 1) spatial ranking, which orders the objects according to their distance from a reference point, and 2) non spatial ranking, which orders the objects by an aggregate function on their non spatial values. The top-k spatial preference query integrates these two types of ranking in an intuitive way. Despite the usefulness of the top-k spatial preference query, to our knowledge, it has not been studied in the past. 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, however, is expected to be very expensive for large input datasets.

In this paper, we propose alternative techniques that aim at minimizing the I/O accesses to the object and feature datasets, while being also computationally efficient. The neighborhood of an object is captured by the scoring function: 1) the

range score restricts the neighborhood to a crisp region centered at p, whereas 2) the influence score relaxes the neighborhood to the whole space and assigns higher weights to locations closer to p. We presented probing algorithms for processing top-k spatial preference queries. 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.

II RELATED WORK

In spatial databases, ranking is often associated to nearest neighbor (NN) retrieval. Given a query location, we are often interested in retrieving the set of nearest objects to it that qualify a condition (e.g., roads, restaurants). Assuming that the set of interesting objects is indexed by a hierarchical spatial access method (e.g., the R-tree), we can use distance bounds while traversing the index to derive the answer in a branch-and-bound fashion. Tao et al. noted that top-k queries can be modeled as (weighted) nearest neighbor queries, in the multi-dimensional space defined by the involved attribute domains, where the query point is formed by taking the maximum value of each dimension.

Nevertheless, it is not always possible to use multidimensional indexes for top-k retrieval. First, such indexes usually break-down in high dimensional spaces. Second, top-k queries may involve an arbitrary set of attributes from a set of possible ones 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.

a) Spatial Query Evaluation on R-trees

R-trees can efficiently process main spatial query types, including spatial range queries, nearest neighbor queries, and spatial joins. Given a spatial region W, a spatial range query retrieves from R the objects that intersect W. A nearest neighbor (NN) query takes as input a query object q and returns the closest object in R to q. A popular generalization is the k-NN query, which returns the k closest objects to q, given a positive integer k. NN (and k-NN) queries can be efficiently processed using the best-first (BF) algorithm, provided that R is indexed by an R-tree. A priority queue PQ which organizes R-tree entries based on the (minimum) distance of their MBRs to q is initialized with the root entries. The aggregate R-tree (aR-tree) is a variant of the R-tree, where each non-leaf entry augments an aggregate measure for some attribute value (measure) of all points in its subtree.

b) Feature-based Spatial Queries

Recently, solved the problem of finding top-k sites (e.g., roads) based on their influence on feature points (e.g., junctions, individual roads). Related to top-k influential query are the optimal location queries. The goal is to find the location in space (not chosen from a specific set of roads) that minimizes an objective function. Here, Assume that all feature points have the same quality. The maximum influence optimal location query finds the location (to insert to the existing set of roads) with the maximum influence, whereas the minimum distance optimal location query searches for the location that minimizes the average distance from each feature point to its nearest site. The optimal locations for both queries are marked as white points. The techniques proposed are specific to the particular query types described above and cannot be extended for the class of top-k spatial preference queries and they deal with a single feature dataset whereas our queries consider multiple feature datasets.

III Algorithms for Spatial Preference Queries

By implementing top-k spatial preference query, we will retrieve the k points in an object dataset D (i.e., set of interesting points like junctions or roads) which has the highest score. We assume that each feature dataset Fc is indexed by an MAX aR-Tree and the object in the given dataset D is indexed by R-tree, where each non-leaf entry augments the maximum quality (of features) in its

subtree. Nevertheless, our solutions are generic and it can be adapted where the datasets are indexed by other hierarchical spatial indexes (e.g., point quad-trees).

The rationale of indexing different feature datasets by separate a R-trees is that: (i) out of all possible features (e.g., road networks, hospital, market, etc.), a user queries for only few features which are required (e.g., roads), and (ii) different subsets of features are considered or used by different users. Based on the indexing schemes which are specified in related work, we implement various algorithms for processing top-k spatial preference queries which are listed below:

Probing Algorithms:

Initially, we introduce a brute-force solution that computes the score of every point p which belongs to dataset D in order to obtain the query results. Then, we propose to implement a special group evaluation technique that computes the scores concurrently for multiple points.

(1) Simple Probing Algorithm:

The simple probing algorithm (SP) compute the score for every object point present in dataset then retrieve the query results based on the computed scores. To implement this algorithm, global variables are used. They are: Wk is a min-heap for managing the top-k results and represents the top-k score so far (i.e., lowest score in Wk). Initially, the algorithm is started at the root node of the object tree (i.e., N = D.root). The procedure is recursively applied on tree nodes until a leaf node is accessed. Score is calculated at each node until it receives the leaf node. When a leaf node is reached, the component score ᒣc(e) is computed by executing a range search (NN search) on the feature tree Fc for range score (NN score) queries.

(2) Group probing Algorithm

Group Probing Algorithm: Due to separate score computations for each and every object in data set, SP is inefficient when large object datasets. To overcome this drawback, we implement the group probing algorithm (GP) which is a variant of SP, that reduces overall 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 Fc tree.

(3) Branch and Bound Algorithm

In order to overcome the problem of SP, GP was introduced but it is expensive. GP examines all objects in D and computes their component scores. Now, we propose an algorithm that can significantly reduce the number of objects to be examined. The key idea is to compute, for non-leaf entries e in the object tree D, an upper bound T (e) (but not ᒣ (e)) of the score for any point in the subtree of e.

BB(Branch and Bound) is called with N being the root node of D. If N is a leaf node then we update the set Wk of the top-k results with concurrently computed scores for all points of N. If N is a non-leaf node, compute the scores T(e) concurrently for non-leaf entries. Now, recall that T (e) which is an upper bound score for any point in e. With the component scores Tc(e) known so far, we can derive a new score called T+(e), an upper bound of T (e). If T+(e) ≤ γ , then the subtree of e cannot contain better results than those in Wk and it is removed from V . We sort the entries in descending order of T(e) then we invoke the procedure recursively on child node which wasx pointed by the entries presented in V, in order to obtain points with high scores. Since both Wk and γ are global variables, the value of γ is updated during recursive call of BB.

Algorithm 1. Branch-and-Bound Algorithm
Wk =new min-heap of size k (initially empty);
Γ:=0; kth score in Wk
algorithm BB(Node N)
1. V :={e|eϵ N};
2. If N is nonleaf then
3. for c:=1 to m do

4. compute Tc(e) for all eЄV concurrently;
5. remove entries e in V such that T +(e)<=γ;
6. sort entries eЄV in descending order of T(e);
7. for each entry eЄV such that T +(e)<=γ; do
8. read the child node N′ pointed by e;
9. BB(N′);
10. else
11. for c=1 to m do
12. compute rc(e) for all eЄV concurrently;
13. remove entries e in V such that T +(e)<=γ;
14. update Wk (and γ) by entries in V ;

Our major goal is to compute these upper bound scores with low I/O cost and the bounds not to be too loose, in order for pruning to be effective. For this, we utilize only level-1 entries (i.e., lowest level non-leaf entries) in Fc for deriving upper bound scores because: (i) there are much fewer level-1 entries than leaf entries, and (ii) high level entries in Fc cannot provide tight bounds. Algorithm can be used for this purpose after changing Line 2 to check whether child nodes of N are above the leaf level.

(4) Optimized Branch-and-Bound Algorithm

To reduce the cost of the BB algorithm, we develops a more efficient score computation technique. Let p be an object point in Data set D. Suppose that we have traversed some paths of the feature trees on F1,F2, . . . ,Fm. Let γ be an upper bound of the quality value for any unvisited entry(leaf or non-leaf) of the feature tree Fc.

algorithm Optimized_Group_Range(Trees F1,F2, . . . ,Fm, Set V , Value c , Value γ)
1. for c := 1 to m do
2. Hc := new max-heap (with quality score as key);
3. insert Fc.root into Hc;
4. μc := 1;
5. for each entry pЄV do
6. rc(p):= 0;
7. α:=1; . ID of the current feature tree
8. while |V| > 0 and there exists a nonempty heap Hc do
9. deheap an entry e from Hc;
10..update threshold
11. if γ:= V ; mindist ϵ pthen
12. continue at Line 8;
13. for each p ϵ V do . Prune unqualified points
14. remove p from V ;
15. read the child node CN pointed to by e;
16. for each entry e′ of CN do
17. if CN is a non-leaf node then
18. insert e′ into Hc;

The pseudo code for computing the scores of objects efficiently from the feature trees F1,F2, . . . ,Fm. The set V contains objects whose scores need to be computed. Here, ϵ refers to the distance threshold of the range score, and γ represents the best score found so far. For each feature tree Fc, we employ a maxheap Hc to traverse the entries of Fc in descending order of their quality values. The root of Fc is first inserted into Hc. The variable maintains the upper bound quality of entries in the tree that will be visited. We then initialize each component score rc(p) of every object pЄV to 0.

IV PERFORMANCE

In NN score computation, we weight based on a maximum acceptable distance of a nearest neighbor. Thus, data objects with very close nearest neighbor of moderate quality may be preferable to objects with a far nearest neighbor, even if it has a higher quality. Our algorithms can directly be applied for such extended preference queries. A simple optimization applicable for these queries is to prune non-leaf entries at feature trees, if they are closer than ⅁c, but cannot improve , due to the distance of their MBRs to the examined objects. Another extension of our queries include preferences also on the data objects. For instance, assume that we are looking for flats that are cheap, big, and close to vegetarian restaurants.

There are several ways to process such queries. First, our algorithms can be used for this purpose if the object tree is also an aR-tree, where the non-spatial attributes (e.g., type, size) are aggregated accordingly. BB can be optimized to prioritize the examination of data in D, based on preferences on the object attributes and prune subtrees that cannot lead

to better results than the ones currently found. Another method is to search primarily on the nonspatial preferences, with the use of a top-k algorithm and probe the feature sets for the spatial preference component, as long as the current top-k results can be improved.

V CONCLUSION

Spatial database systems manage large collections of geographic entities, which apart from spatial attributes contain nonspatial information. In this paper, we studied top-k spatial preference queries, which provide a novel type of ranking for spatial objects based on qualities of features in their neighborhood. Initially, we implemented a baseline algorithm Simple probing which computes the scores of every object by performing spatial queries on feature datasets. But SP reduces the number of component score computations for the objects which is optimized by an incremental computation technique. Next, we presented a variant of SP i.e., group probing algorithm which reduces I/O cost by computing scores of objects in the same leaf node concurrently. Later based on GP, we developed algorithm Branch and Bound, which prunes non-leaf entries in the object tree that cannot lead to better results. For efficient results, we optimized branch and bound algorithm to reduce the cost of the BB algorithm. 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.

VI 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] C.S. Jensen, J. Kolar, T.B. Pedersen, and I. Timko, “Nearest Neighbor Queries in Road Networks,” Proc. ACM Int’l Workshop Geographic Information Systems, 2003.

[8] N. Jing, Y.W. Huang, and E.A. Rundensteiner, “Hierarchical Encoded Path Views for Path Query Processing: An Optimal Model and Its Performance Evaluation,” IEEE Trans. Knowledge and Data Eng., vol. 10, no. 3, pp. 409-432, 1998.

[9] I. F. Ilyas,W. G. Aref, and A. Elmagarmid. Supporting Top-k Join Queries in Relational Databases. In VLDB, 2003.

[12] N. Mamoulis, K. H. Cheng, M. L. Yiu, and D. W. Cheung. Efficient Aggregation of Ranked Inputs. In ICDE, 2006.