Efficiently Identifying Top Quality Feature Objects in Spatial Data

[1]Chakrapani Avala, [2]Yamini Santhoshi P
[1]Dept. of CSE, Gokul Institute of Technology and Sciences, JNTU Kakinada, Kakinada, AP, India

Abstract
To select the top rank Objects based on the quality of features in their spatial neighborhood. An attractive type of preference queries, which select the best spatial location with respect to the quality of facilities in its spatial area. Given a set D of interesting objects (e.g., candidate locations), a top-k spatial preference query retrieves the k objects in D with the highest scores. The featured score of a given object is derived from the quality of features (e.g., location and nearby features) in its spatial neighborhood. User preference queries are very important in spatial databases. With the help of these queries, one can found best location among points saved in database. In many situation users evaluate quality of a location with its distance from its nearest neighbor among a special set of points. In this paper R-trees can efficiently process main spatial query types, including spatial range queries, nearest neighbor queries, and spatial joins. It finds out the top most feature objects.

Keywords
Query processing, Spatial Databases

I. Introduction

In many situations for decision making, users need select one or more data from database in accordance with their interest. The selected data must meet their desired constraints. For example suppose in a database about a shoreline city, information of its hotels such as cost and distance of each hotel from beach has been saved. A user wants to select a hotel with less cost and distance to beach. User hasn't accurate asked (for example cost of hotel below 100$ and distance to beach less than 1Km is accurate asked) but wants to find a set of data that are closer to their own interests. Such constraints called soft constraints and queries about these problems called user preference queries [1].

There are two basic queries for these problems. In the first type of queries that are known to top k, each of data attribute based on their importance to user gives a weight. The score of the data is computed by multiplying its values with the corresponding weights and aggregating them by a function. This query retrieves the k data with the highest scores [2]. In the second type of queries that are known to skyline, the set of all data that no other data dominate them are retrieved. A data dominate another if and only if for all attributes is better than or equals and for at least one attribute is better than the other. Indeed, in this way all data that aren't worse than any other data in database are retrieved [3].

User preference queries are very important in spatial databases. Spatial data in addition to non spatial data can be stored in these databases. With the help of these queries, user can find best places in database according to their interest

For many application users evaluate quality of a location with its distance from its nearest neighbor among special set of points. For example suppose a user wants to find a hotel for rest that is near to a restaurant and a coffee shop. So he considers coffee shops and restaurants as two query point sets and evaluates hotels with their distances to nearest coffee shop and restaurant. Less attention has been about subject distance of a point to its nearest neighbors as preference of the user.

Related works in this field are in based on top-k queries. In top-k queries setting a weight for each attribute and a scoring function for aggregating attribute.

II. Related Works

Object ranking is a popular retrieval task in various applications. In relational databases, we rank tuples using an aggregate score function on their attribute values [2]. For example, a real estate agency maintains a database that contains information of flats available for rent. A potential customer wishes to view the top-10 flats with the largest sizes and lowest prices. In this case, the score of each flat is expressed by the sum of two qualities: size and price, after normalization to the domain [0,1] (e.g., 1 means the largest size and the lowest price). In spatial databases, ranking is often 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 [3], we can apply distance bounds and traverse the index in a branch-and-bound fashion to obtain the answer [4]. Nevertheless, it is not always possible to use multidimensional indexes for top-k retrieval. First, such indexes break-down in high dimensional spaces [5-6]. 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. may arrive from different (distributed) sources.

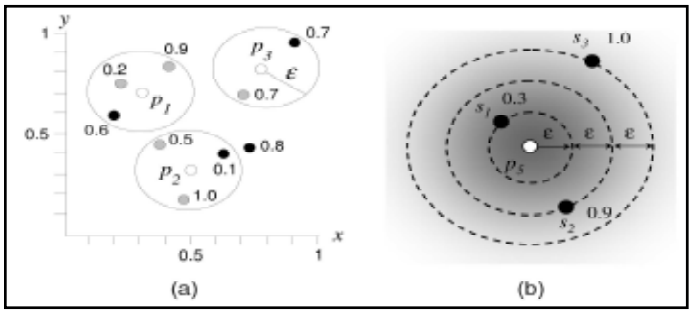

Fig. 1: Examples of Top-k Spatial Preference Queries. (a) Range Score 0:2 km. (b) Influence Score 0:2 km

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. In the following subsections, we first review the R tree, which is the most popular spatial access method and the NN search algorithm of [4]. The most popular spatial access method is the R-tree [3], which indexes minimum bounding rectangles (MBRs) of objects. Fig. 2 shows a set D = {p1,……..., p8} of spatial objects (e.g., points) and an R-tree that indexes them. 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 D the objects that intersect W. For instance, consider a range query that asks for all objects within the shaded area in fig. 2. Starting from the root of the tree, the query is processed by recursively following entries, having MBRs that intersect the query region. For instance, e1 does not intersect the query region, thus the sub tree pointed by e1 cannot contain any query result. In contrast, e2 is followed by the algorithm and the points in the corresponding node are examined recursively to find the query result p7. A nearest neighbor (NN) query takes as input a query object q and returns the closest object in D to q. For instance, the nearest neighbor of q in Figure 2 is p7. Its 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 of [4], provided that D is indexed by an R-tree. A min-heap H which organizes R-tree entries based on the (minimum) distance of their MBRs to q is initialized with the root entries. In order to find the NN of q in Figure 2, BF first inserts to H entries e1, e2, e3, and their distances to q. Then the nearest entry e2 is retrieved from H and objects p1, p7,p8 are inserted to H. The next nearest entry in H is p7, which is the nearest neighbor of q. In terms of I/O, the BF algorithm is shown to be no worse than any NN algorithm on the same R-tree [4]. The aggregate R-tree (a R-tree) [10] is a variant of the R tree where each non-leaf entry augments an aggregate measure for some attribute value (measure e) of all points in its sub tree. As an example, the tree shown in Figure 2 can be upgraded to a MAX a R-tree over the point set, if entries e1, e2, e3 contain the maximum measure values of sets {p2, p3} {p1, p8,p7} (p4, p5, p6}, respectively. Assume that the measure values of p4, p5, p6 are 0.2, 0.1, 0.4, respectively. In this case, the aggregate measure augmented in e3 would be max{0.2, 0.1, 0.4} = 0.4. In this paper, we employ MAX a R-trees for indexing the feature datasets (e.g., restaurants), in order to accelerate the processing of top-k spatial preference queries. Given a feature dataset F and a multi-dimensional region R, the range top-k query selects the tuples (from F) within the region R and returns only those with the k highest qualities. Hong et al. [7] indexed the dataset by a MAX a R-tree and developed an efficient tree traversal algorithm to answer the query. Instead of finding the best k qualities from F in a specified region, our (range score) query considers multiple spatial regions based on the points from the object dataset D, and attempts to find out the best k regions (based on scores derived from multiple feature datasets Fc).

III. Proposed System

A. Simple Probing Algorithm

We propose a group evaluation technique that computes the scores of multiple points concurrently.

- It retrieves the query results by computing the score of every object point.
- This algorithm uses two global variables. Wk□It is a min-heap for managing the top-k results and represents the top-k scores.
- Initially, the algorithm is invoked at the root node of the object tree.
- The procedure is recursively applied (at Line 4) on tree nodes until a leaf node is accessed.
- When a leaf node is reached, the component score Fc(e) (at Line 8) is computed by executing a range search on the feature tree Fc for range score queries.
- Lines 6-8 describe an incremental computation technique, for reducing unnecessary component score computations.

Algorithm 1. Simple Probing Algorithm

```
algorithm SP(Node N)
1: for each entry e ∈ N do
2:    If N is nonleaf then
3:       read the child node N' pointed by e;
4:       SP(N');
5:    else
6:       for c := 1 to m do
7:          If τ+(e) > γ then          ▷ upper bound score
8:             compute τc(e) using tree Fc; update τ+(e);
9:       If τ(e) > γ then
10:         update Wk (and γ) by e;
```

B. Group Probing Algorithm

- Due to separate score computations for different objects, SP is inefficient for large object datasets.
- 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 Fc tree.
- Here MBR → for distance notation.
- Given a point p and an MBR e, the value mindist(p, e) (maxdist(p, e)) denotes the minimum(maximum) possible distance between p and any point in e.
- Similarly, given two MBRs ea and eb, the value mindist(ea; eb) (maxdist(ea; eb)) denotes the minimum (maximum) possible distance between any point in ea and any point in eb.

Algorithm 1 ITERATE (Joint query Q, Tree root $root$, Integer k)

```
1: for each subquery q_i do
2:    V_i ← new max-priority queue;        ▷ maintain the top k objects
3:    Initialize V_i with k null objects with distance ∞;
4:    U ← new min-priority queue;
5:    U.Enqueue(root, 0);
6:    while U is not empty do
7:       e ← U.Dequeue();
8:       if e is an object then
9:          update V_i by (e, dist(q_i.λ, e.λ));
10:         if V_i has k non-null objects then
11:            break the while-loop;
12:      else                              ▷ e points to a child node
13:         read the node CN of e;
14:         read the posting lists of CN for keywords in q_i.ψ;
15:         for each entry e' in the node CN do
16:            if q_i.ψ ⊆ e'.ψ then
17:               U.Enqueue(e', mindist(q_i.λ, e'.λ));
18: return {V_i};                         ▷ top-k results of each subquery
```

Group Range Algorithm

- Initially, the procedure is called with N being the root node of Fc.
- If e is a non-leaf entry and its mindist from some point p 2 V is within the range, then the procedure is applied recursively on the child node of e, since the sub-tree of Fc rooted at e may contribute to the component score of p.

Algorithm 2. Group Range Score Algorithm

```
algorithm Group_Range(Node N, Set V, Value c,
Value ϵ)
1: for each entry e ∈ N do
2:   If N is nonleaf then
3:     If ∃p ∈ V, mindist(p,e) ≤ ϵ then
4:       read the child node N' pointed by e;
5:       Group_Range(N',V,c, ϵ);
6:   else
7:     for each p ∈ V such that dist(p,e) ≤ ϵ do
8:       τc(p) := max{τc(p),ω(e)};
```

C. 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. The key idea is to compute, for non leaf entries e in the object tree D, an upper bound T of the score for any point p in the sub tree of e. If then we need not access the sub tree of e, thus we can save numerous score computations.

Algorithm 3 is a pseudo code of our BB algorithm, based on this idea. BB is called with N being the root node of D. If N is a non leaf node, Lines 3-5 compute the scores Top for non leaf entries e concurrently. Recall that Top is an upper bound score for any point in the sub tree of e. The techniques for computing Top will be discussed shortly. Like (3), with the component scores Top known so far, we can derive Top an upper bound of Top then the sub tree of e cannot contain better results than those in Wk and it is removed from V. In order to obtain points with high scores early, we sort the entries in descending order of 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. Since both Wk and _ are global variables, their values are updated during recursive call of BB.

Algorithm 3. 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|e ∈ N};
2: If N is nonleaf then
3:   for c := 1 to m do
4:     compute Tc(e) for all e ∈ V concurrently;
5:     remove entries e in V such that T+(e) ≤ γ;
6:   sort entries e ∈ V in descending order of T(e);
7:   for each entry e ∈ V 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 τc(e) for all e ∈ V concurrently;
13:     remove entries e in V such that τ+(e) ≤ γ;
14:   update Wk (and γ) by entries in V;
```

D. Spatial Query Evaluation on R-Trees

The most popular spatial access method is the R-tree [3], which indexes minimum bounding rectangles (MBRs) of objects. Fig. 2 shows a set D {p1, . . . , p8} of spatial objects (e.g., points) and an R-tree that indexes them. 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 D the objects that intersect W. For instance, consider a range query that asks for all objects within the shaded area in Fig. 2. Starting from the root of the tree, the query is processed by recursively following entries, having MBRs that intersect the query region. For instance, e1 does not intersect the query region, thus the sub tree pointed by e1

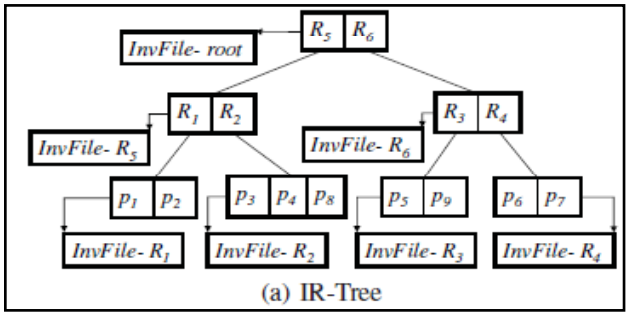

Fig. 2:

Given a feature data set F and a multidimensional region R, the range top-k query selects the tuples (from F) within the region R and returns only those with the k highest qualities. Hong et al. [11] indexed the data set by a MAX a R-tree and developed an efficient tree traversal algorithm to answer the query. Instead of finding the best k qualities from F in a specified region, our (range score) query considers multiple spatial regions based on the points from the object data set D, and attempts to find out the best k regions (based on scores derived from multiple feature data sets Fc).

IV. Results

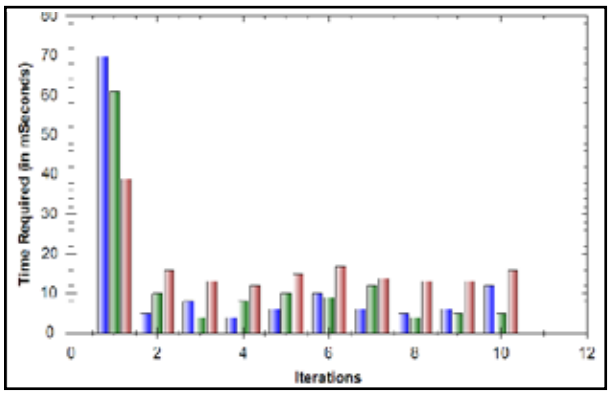

Fig. 3: Bar Chart on basis on Time Complexity

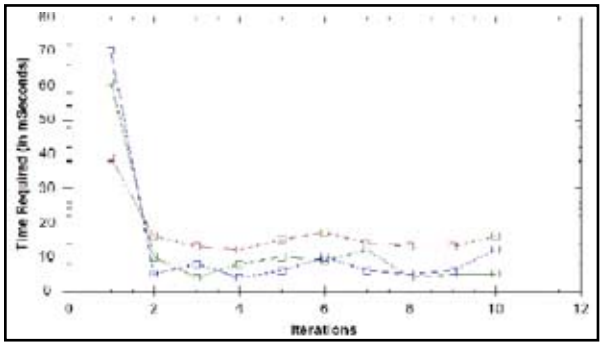

Fig. 4: Line Chart on Basis on Time Complexity

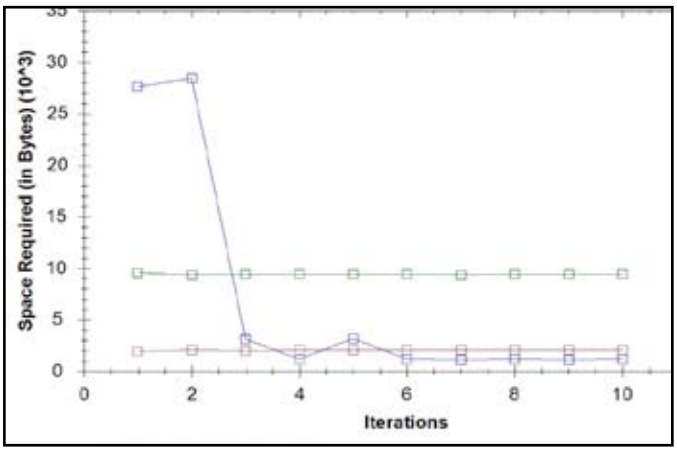

Fig. 5: Line Chart on basis on Space Complexity

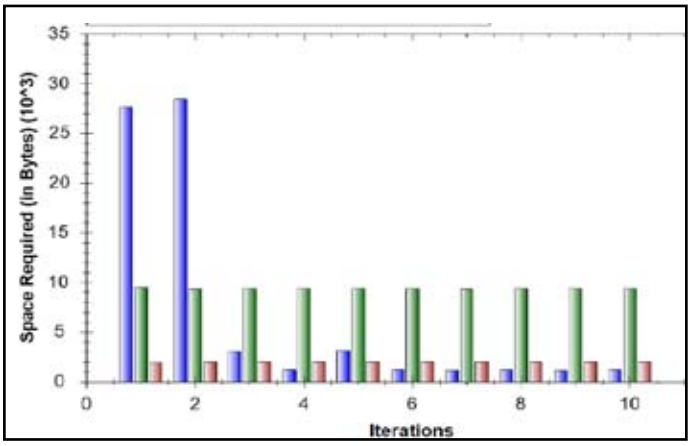

Fig. 6: Bar Chart on basis on Space Complexity

V. Conclusion

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. In this paper R-trees can efficiently process main spatial query types, including spatial range queries, nearest neighbor queries, and spatial joins. Experimental results shows that R-tree is best compared to branch and bound algorithm.

References

[1] M. L. Yiu, X. Dai, N. Mamoulis, M. Vaitis,"Top-k Spatial Preference Queries", in ICDE, 2007.
[2] N. Bruno, L. Gravano, A. Marian,"Evaluating Top-k Queries over Web-accessible Databases", in ICDE, 2002.
[3] A. Guttman,"R-Trees: A Dynamic Index Structure for Spatial Searching", in SIGMOD, 1984.
[4] G. R. Hjaltason, H. Samet,"Distance Browsing in Spatial Databases", TODS, Vol. 24(2), pp. 265–318, 1999.
[5] R.Weber, H.-J. Schek, S. Blott,"A quantitative analysis and performance study for similarity-search methods in high dimensional spaces", in VLDB, 1998.
[6] K. S. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, "When is "nearest neighbor" meaningful?" in ICDT, 1999.
[7] R. Fagin, A. Lotem, M. Naor,"Optimal Aggregation Algorithms for Middleware", in PODS, 2001.
[8] I. F. Ilyas, W. G. Aref, A. Elmagarmid,"Supporting Top-k Join Queries in Relational Databases", in VLDB, 2003.
[9] N. Mamoulis, M. L. Yiu, K. H. Cheng, D. W. Cheung,"Efficient Top-k Aggregation of Ranked Inputs", ACM TODS, Vol. 32, No. 3, p. 19, 2007.
[10] D. Papadias, P. Kalnis, J. Zhang, Y. Tao,"Efficient OLAP Operations in Spatial Data Warehouses", in SSTD, 2001.

Yamini Santhoshi P received her B.Tech degree in CSE from Pydah College of Engg & Tech, Visakhapatnam in 2009 and currently pursuing M.Tech in CSE from Gokul Institute of Technology and Sciences, Bobbili, Vizianagaram Dist., AP.

Chakrapani Avala received the M.Tech degree from GITAM UNIVERSITY, VISAJKHAPATNAM in 2011. Currently he is working as Assistant Professor in Gokul Institute of Technology and Sciences, Bobbili ,Vzm Dist. AP. He has two years experience in teaching and four years of IT experience in Software Quality Testing arena in Miracle Software Systems .His research interests include Manual testing versus Automated testing, Data Mining.