IOSR Journal of Computer Engineering (IOSRJCE)
ISSN: 2278-0661, ISBN: 2278-8727Volume 6, Issue 5 (Nov. - Dec. 2012), PP 30-35
www.iosrjournals.org

Ranking Preferences to Data by Using R-Trees

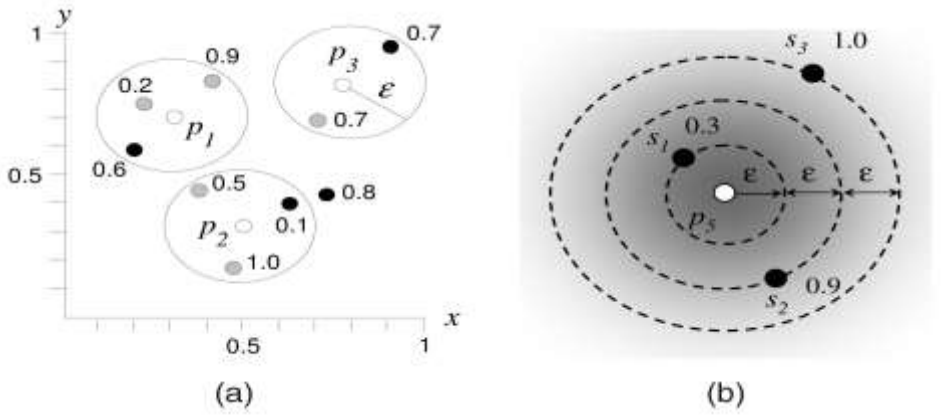
Nageswarrao.Vungarala[1], Manoj Kiran.Somidi[2], Krishnaiah.R.V.[3]
[1]*Department of CS, DRK College of Engineering& Technology, Ranga Reddy, Andhra Pradesh, India*
[2]*Department of SE, DRK Institute of Science & Technology, Ranga Reddy, Andhra Pradesh, India*
[3]*Principal Department of CSE, DRK Institute of Science & Technology, Ranga Reddy, Andhra Pradesh, India*

Abstract: *A Spatial data preference query ranks the given objects based on the quality and their features in nearest neighborhood. Here "feature" refers to a class of objects in a spatial map such as specific facilities or services. For example, a real estate agency maintains database which holds the details of flats for rent. Here, A customer may want to rank the contents with respect to the appropriateness of their locations, the ranks are given by using the top-k spatial aggregate functions with respect to quality of features and nearest neighborhood. A neighborhood concept can be specified by the user via different functions. Every customer maximum prefers the quality of the flat and more facilities in that flat. The potential customer wishes to view the top 10 flats with the largest size and lowest prices, which is nearest to Hospital, School, Bus top, Market etc. In this paper, we define top-k spatial preference queries and branch and bound algorithm.*
Keywords: *Spatial data, Top-k spatial.*

I. Introduction

A Spatial databases store data that is related to the objects in space. There are two basic ways for ranking the 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 spatial values. Top-k spatial algorithm combines these two types spatial and non spatial ranking. A Spatial database systems contains large collection of geographic entities, which apart from spatial attributes contain non spatial data. A customer want to rank the contents of this database with respect to the quality of their locations, quantified by aggregating non spatial characteristics of other features (e.g. Hospital, School, Market etc.) in the spatial neighborhood of flat. Quality may be subjective and query parametric. A neighborhood concept can be specified by the user via different functions.

A set of D best data objects based on the quality of feature objects in their spatial neighborhood. Fig 1 illustrates three locations (p1, p2, p3) of user interest and two feature sets (v and t) these are within the spatial neighborhood of the location. The feature values are labeled with the quality values. The quality values are normalized to values in [0, 1].



(a) Range score, (b) Influence score

Fig. 1. Top-k spatial preference queries.

According to above example, a customer wants to retrieve the top-k flats from a spatial database which is maintained by the real estate agency. In this case, the score of each is depends on sum of two qualities those are size and price. The customer wants to retrieve the top-k flats which are near to the high quality hospital and high quality restaurant. Fig. 1(a) illustrates the locations of the object data set D (hotel) in white and two other objects (restaurants) in gray and (cafes) in black. The score p1 of a hotel p is defined in terms of: 1) the maximum quality for each feature in the neighborhood region of p, and 2) the aggregation of those qualities.

The spatial neighborhood can be specified by the user to restrict the distance of the eligible feature objects. If the user wants to rank the flats based on the range score which is nearest neighborhood region to a circular at p with radius € and used the aggregate functions SUM, MIN and MAX. SUM function is implemented on Fig. 1 the maximum quality of gray and black points within the circle of p1 are 0.9 and 0.6 respectively. Then the score of P_1 is p1= 0.9 + 0.6 = 1.5. Similarly, the maximum quality of gray and black points within the circle of P_2 are 1.0 and 0.1, so the score of P_2 is p2 = 1.0 + 0.1 = 1.1 and p3 = 0.7 + 0.7 = 1.4. Hence, within above 3 results the p1 (hotel) has the maximum value and has quality. The SUM function attempts to balance the overall qualities of all features.

We are implemented the MIN function on Fig. 1, then the top results becomes P_3, with the score P_3 = Min {0.7, 0.7} = 0.7, it shows that the top result has reasonable high qualities in all features. The MAX function is used optimize the quality in a particular feature. For the MAX function the top result is P_2, with P_2 = Max {1.0, 0.1} = 1.0 Top-k spatial preference queries comprise a useful tool for a wide range of location based applications. But processing this query is quite complex. Because it may require computing the scores of all data objects and select the top-k spatial preference query. It is not always possible to use multidimensional indexes for top-k spatial retrieval. First such indexes break down in high dimensional spaces [10]. Second top-k spatial preference queries may involve an arbitrary set of user specified attributes (e.g., size and price) from possible ones (e.g., size, price, floor etc,). Third, information for ranking s to be combined could appear in different databases and unified indexes may not exist for them top-k spatial preference queries [2], [8] focusing on the different sources.

Specifically, we contribute the branch and bound (BB) algorithm and Feature join (FJ) algorithm for efficiently processing the top-k spatial preference query.

II. Background Knowledge

Object ranking (ordering) is a popular retrieval task in various applications. The tuples in relational databases are ordered using aggregate functions on their attribute values [2]. Consider a real estate agency database maintaining information regarding flats available for rent. A customer wishes to view top ten flats with the largest sizes and lowest prices. Here, the score of each flat is expressed by the sum of two attributes: size and price. Ranking (ordering) in spatial databases is often associated to the nearest neighbor (NN) retrieval. The NN returns set of nearest objects to a given query location that satisfies a condition (e.g., restaurants). The set of interesting object is indexed by an R-tree [3]. Then the index can be traversed in a branch and bound fashion to obtain the answer [4]. It is not always possible to use multidimensional indexes for top-k spatial preference retrieval. So such indexes break down in high dimensional spaces [5], [6]. Solutions for top-k spatial preference queries [2], [7], [8], [9] focus on the efficient merging of object rankings that may arrive from different sources.

2.1 Spatial Query Evaluation on R-Trees

Several approaches have been proposed for ranking spatial data. In order to handle spatial data efficiently, an effective indexing mechanism is required. One of the most popular spatial access method is the R-Tree [3] which indexes minimum bounding rectangles (MBRs) of spatial objects.

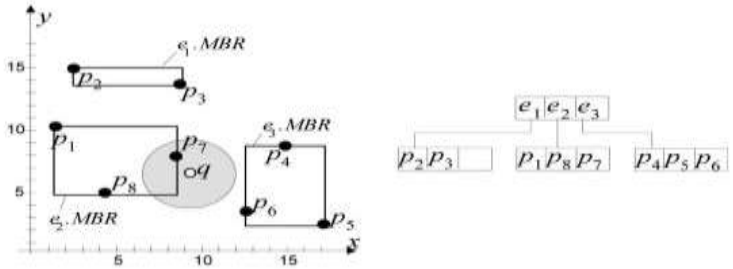


Fig. 2 Spatial queries on R-Trees

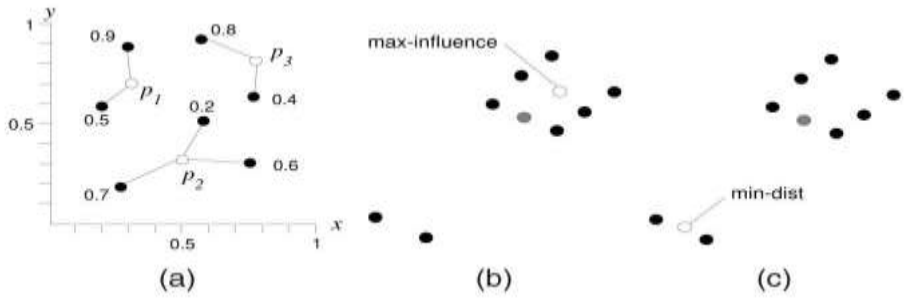
Fig 2 shows a set of interesting points D= {P_1, P_2…P_8) indexed using an R-tree. A spatial range query returns the objects in D that intersect the given query region W. For example, consider a range query that asks for all objects within the shaded area in Fig 2. Starting from the root of the tree, the query is processed by recursively following entries having MBRs that intersect the query region. For instance, e1 does not match intersect the query region. 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 variant of an R-tree is a R-tree [14]. Here each non-leaf node entry develops an aggregate [10] value (MAX) for some attribute measures in its sub tree. For instance, a MAX an R-tree can be constructed over the point set given in Fig. 2, if the entries e1, e2, e3 contain the maximum measure values of sets {P_2, P3}, {P_1, P_8, P_7}, {P_4,

P_5, P_6}, respectively. Assume that the measure values of P_4, P_5, P_6 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 data sets (e.g., restaurants), in order to accelerate the processing of top-k spatial Preference queries.

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, range score considers multiple spatial regions based on the points from the objects data set D, and attempts to find the best k regions.

2.2 Feature based spatial queries

Xia et al. [12] solved the problem of finding top-k sites (e.g., restaurants) based on their influence on feature points (e.g., residential buildings). As an example, Fig. 3a shows a set of sites (white points), a set of features (black points with weights), such that each line links a feature point to its nearest site. The influence of a site pi is defined by the sum of weights of feature points having pi as their closest site. For instance, the score of p1 is 0.9 + 0.5 = 1.4. Similarly, the scores of p2 and p3 are 1.5 and 1.2, respectively. Hence, p2 is returned as the top-1 influential site.



(a) Top-k influential (b) Max- influential (c) Min- distance.

Fig. 3 Influential sites and optimal location queries.

Related to top-k influential sites query are the optimal location queries studied in [13], [14]. The goal is to find the location in space that minimizes an objective function. In Figs. 3(b) and 3(c) features points and existing sites are shown as black and gray points, respectively. Assume that all feature points have the same quality. The maximum influence optimal location query [13] finds the location with the maximum influence (as defined in [12]) where as the minimum distance optimal location query [14] searches for the location that minimizes the average distance from each feature point to its nearest site. The techniques proposed in [12], [13], [14] are specific to the particular query types described above and cannot be extended for our top-k spatial preference queries. The novel spatial queries and joins [15], [16], [17], [18] have been proposed for various spatial decision.

III. Preliminaries

Let F_c be a feature data set, in which each feature object s € F_c is associated with a quality w(s) and a spatial point. We assume that the domain of w(s) is the interval [0, 1]. As an example, the quality w(s) of a restaurant s can be obtained from a ratings provider.

Let D be an object data set, where each object p € D is a spatial point. In other words, D is the set of interesting points (e.g., hotel locations) considered by the user.

Given an object data set D and m feature data sets F_1, F2 . . . F_m, the top-k spatial preference query retrieves the k points in D with the highest score. Here, the score of an object point p € D is defined as

$$\prod^{\square}(p) = AGG\{\prod^{\square}(p) \mid c \in [1, m]\}$$

Where AGG is an aggregate function and $\prod^{\square}(p)$ is the ($\mathbf{C^{th}}$) component score of p with respect to the neighborhood condition □ and the ($\mathbf{C^{th}}$) feature data set F_c. Typical examples of the aggregate function AGG are SUM, MIN, and MAX.

The component score function, $\prod^{\square}(p)$ taken is the range score function. The range score, $\prod_c^{rng}(p)$ is taken as the maximum quality w(s) of points s $\in F_c$ that are within a given parameter distance € from p, or 0 if no such point exists.

$$\prod_c^{rng}(p) = \max(\{w(s) \mid s \square F_c \wedge dist(p, s) \square \square\} \cup \{0\})$$

3.1 Branch and Bound Algorithm

A brute force approach for processing the top-k spatial preference query computes the score of every point p □ D in order to obtain the query results. Brute force approach is expensive as it examines all objects in D and computes their component scores. The algorithm proposed here 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) of the score for any point p in the sub tree of e. If T < □**,** then need not access the sub tree of e, thus numerous score computations can be saved.

Branch and bound algorithm is a pseudo code of the proposed 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 T (e) for non leaf entries e concurrently. T (e) is an upper bound score for any point in the sub tree of e, with the component scores T_C (e) known so far, derive $T_{+(e)}$, an upper bound of T (e). If T_+ (e) □ □, then sub tree of an e cannot contain better results than those in Wk and it is removed from the set V. In order to obtain points with high scores early, sort the entries in descending order of T (e) before invoking the above procedure recursively on the child nodes pointed by the entries in V. If N is a leaf node, compute the scores for all points of N concurrently and then update the set W_k of the top-k results. Since both W_k and □ are global variables, their values are updated during recursive call of branch and bound algorithm.

The following shows the branch and bound algorithm implementation.

Wk: = new min-heap of size k (initially empty);

□ **:** =0; Δ kth score in Wk

Algorithm BB (Node N)

1: V: = {e|e □N};
2: **If** N is non leaf then
3: **fo**r c: = 1 to m **do**
4: compute T_c (e) for all e € V concurrently;
5: remove entries e in V such that T + (e) □ □;
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 Ń pointed by e;
9: BB (Ń);
10: **els**e
11: **for** c: = 1 to m **do**
12: compute T_c (e) for all e €V concurrently;
13: remove entries e in V such that $T_{+(}$e) ≤ □;
14: update W_k (and □) by entries in V;

Upper bound scores T_c (e) of non leaf entries (within the same node N) can be computed concurrently (at line 4). Upper bound score is to be computed such that 1) the bounds are computed with low I / O cost , and 2) the bounds are reasonably tight, in order to facilitate effective prunning. To achieve this, only level-1 entries (i.e., lowest level non leaf entries) in F_c are utilized for deriving upper bound scores because: 1) there is much fewer level-1 entries than leaf entries, 2) high-level entries in F_c cannot provide tight bounds.

3. 2 Feature Join Algorithm

Feature join algorithm is an alternative method for evaluating a top-k spatial preference query to perform a multyway spatial join [23] on feature trees F_1, F_2, F_m to obatin combinations of featurepoints which can be in the neighborhood of some object from D. We first introduce the concept of a combination to be pruned and finally elaborated the algorithm used to progressively identify the combinations that correspond to query results.

The following algorithm shows the feature join algorithm,

W_k: = new min-heap of size k (initially empty);

□ := 0; Δ k^{th} score in W_k

Algorithm FJ (Tree D, Trees F1, F2 . . ., Fm)

1: H: = new max-heap (combination score as the key);
2: insert (F_1.root, F_2.root. . . Fm. Root) into H;
3: while H is not empty do
4: deheap (f_1, f_2. . . Fm) from H;
5: **if** for all c 1€ [1, m], c f_c points to a leaf node

6: for c: = 1 to m **do**
7: read the child node L_c pointed by f_c;
8: Find_ Result (D: root, L_1,. . . , L_m);
9: **else**
10: f_c : = highest level entry among f_1, f_2; . . . ; fm;
11: read the child node N_c pointed by fc;
12: **for each** entry e_c € N_c **do**
13: insert (f1, f2, . . . ,e_c . . . ,f_m) into H if its score is Greater than □and it qualifies the query;

Algorithm Find_ Result (Node N, Nodes L_1,... L_m)
1: **for each** entry e € N **do**
2: **if** N is non leaf **then**
3: compute T (e) by entries in L_1,.. ., L_m;
4: **if** T (e) > □**then**
5: read the child node N**'** pointed by e;
6: Find_ Result (N', L_1, . . ., L_m);
7: **else**
8: compute T (e) by entries in L_1,...,L_m;
9: update W_k (and ❑) by e (when necessary);

IV. Conclusion

Top-k spatial preference queries, provides a novel type of ranking for spatial objects based on qualities of features in their neighborhood. But processing these queries is quite complex. The brute force approach computes the scores of every data object by querying on feature data sets. This method is expensive for large input data sets. The alternative technique, branch and bound algorithm, 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 Feature join algorithm performs a multi way join on feature trees to obtain qualified combinations of feature points and then search for relevant objects in object tree. Feature join is the best algorithm in cases where the number m of feature data sets is low and each feature data set is small.

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. Ramakrishna, and U. Shaft, "When is' Nearest Neighbor'
[7] Meaningful?" Proc. Seventh Int'l Conf. Database Theory (ICDT), 1999.
[8] R. Fagin, A. Lotem, and M. Naor, "Optimal Aggregation Algorithms for Middleware," Proc. Int'l Symp. Principles of Database Systems (PODS), 2001.
[9] 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.
[10] 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.
[11] 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.
[12] 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.
[13] 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.
[14] Y. Du, D. Zhang, and T. Xia, "The Optimal-Location Query," Proc. Int'l Symp. Spatial and Temporal Databases (SSTD), 2005.
[15] D. Zhang, Y. Du, T. Xia, and Y. Tao, "Progessive Computation of The Min-Dist Optimal- Location Query," Proc. 32nd Int'l Conf. Very Large Data Bases (VLDB), 2006.
[16] Y. Chen and J.M. Patel, "Efficient Evaluation of All-Nearest- Neighbor Queries," Proc. IEEE Int'l Conf. Data Eng. (ICDE), 2007.
[17] 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.
[18] M.L. Yiu, P. Karras, and N. Mamoulis, "Ring-Constrained Join: Deriving Fair Middleman Locations from Point sets via a Geometric Constraint," Proc. 11th Int'l Conf.Extending Database Technology (EDBT), 2008.
[19] M.L. Yiu, N. Mamoulis, and P. Karras, "Common Influence Join: A Natural Join Operation for Spatial Point sets," Proc. IEEE Int'l Conf. Data Eng. (ICDE), 2008.
[20] Y.-Y. Chen, T. Suel, and A. Markowetz, "Efficient Query Processing in Geographic Web Search Engines," Proc. ACM SIGMOD, 2006.

[21] 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.

About Authors

Nageswarrao Vungarala is a student of DRK College of Engineering and Technology, Ranga Reddy, Andhra Pradesh, India. He has received M.C.A degree and M.Tech Degree in Computer Science. His main research interest includes Data Mining and Cloud Computing

Manoj Kiran Somidi is a student of DRK Institute of Science & Technology, Ranga Reddy, and Andhra Pradesh, India. He has received B.Tech Degree in Computer Science and Engineering and M.Tech Degree in Software Engineering. His main research interest includes Software Engineering, Data Mining.

Dr.R.V.Krishnaiah is working as Principal at DRK INSTITUTE OF SCINCE & TECHNOLOGY, Hyderabad, and Andhra Pradesh, INDIA. He has received M.Tech Degree (EIE&CSE) and Ph.D. His main research interest includes Data Mining, Software Engineering