Advances in Computational Sciences and Technology
ISSN 0973-6107 Volume 10, Number 5 (2017) pp. 1517-1529

http://www.ripublication.com

Efficient Index Based Query Keyword Search in the Spatial Database

S. Aquter Babu

Assistant Professor, Department of Computer Science,
Dravidian University, Kuppam, Chittoor District,
Andhra Pradesh, India

Abstract

In many real time applications there is a need to represent many real time objects (universities, temples, airports, cities, buildings, lakes and companies etc) in the spatial database. Usually, spatial database objects are mapped with keywords for their reference in business or service applications. Normally, spacial objects are queried with the requirement of minimum inter object distance. Nowadays many business decisions are directly related to the keyword rating of objects. Existing method considers inter object distance only. Many latest methods require not only the inter object distance but also object ratings for increasing their revenues. More generalized version of spatial keyword search requires both inter object distance as well as keyword ratings.

Keywords: spatial database objects, keyword, keyword cover, object rating, inter object distance

1. INTRODUCTION

Many real life applications such as mobile computing, finding a service center, location based services, finding vehicle location, finding the location of the company, and finding address of a person or animal or mobile or vehicle require spatial object keywords search. Driven by mobile computing, location-based services and wide availability of extensive digital maps and satellite imagery the spatial keywords search problem has attracted much attention recently [1].

In a spatial database, each tuple represents a spatial object which is associated with keyword(s) to indicate the information such as its businesses/services/features [1]. Spatial database consists of spatial database objects. In general, tuples are represented as spatial database objects. The real life objects are represented as tuples and these tuples are stored as spatial database objects in the spatial database. All the objects stored in the spatial database mapped with keywords for their easy reference in business as well as service transactions. For a given spatial database object query, first keywords given in the query are identified and then spatial objects are searched and returned corresponding to the given query keywords. Existing methods must require satisfying inter object distance that is minimum. for example, if a student wants to get admission in an IIT with requirements such as minimum distance from IIT to city and city to company office, and IIT to his native address. Here many requirements in minimum inter object distance.

Nowadays many business transactions are taking place based on not only internal object distance but also rating of objects. Already existing algorithm is called m closest keyword search algorithm and it is based solely on inter object distance. New algorithm is called best keyword cover algorithm and it is based on both inter object distance and object rating.

Present study considers both inter object distance as well as object ratings. This new idea is most famous and it is used by many organizations. More than 90% of the business decisions either directly or indirectly depend on rating of objects. A frame work that takes care of both inter object distance as well as rating of objects is needed. Mathematical functions consisting of two variables, one for inter object distance and another one for object ratings is required and this mathematical function must be optimized to for specific purpose.

In the literature of spatial database keyword search, already existing algorithm is called closest keyword query search. But this algorithm is not scalable for very large training data sets. Hence, what is needed is scalable, efficient and effective closet keyword query search algorithm. New algorithm must satisfy many of the desired features in addition to the already stated two features. New algorithm must use state-of-the-art indexing data structure for efficient and scalable operations. Presently R^*-tree, R^+- tree, R-tree are popular spatial database indexes. A variation of the R-tree like and powerful indexing data structure is required.

Spatial objects are stored in the spatial database. Some techniques use only single tree indexing data structure whereas other techniques consider multiple indexing tree data structures for efficient management of spatial database. For a given spatial query objects are retrieved from the spatial database. Each retrieved object is associated with keywords relevant to the query keywords and is close to the query location [2]. Spatial database indexing tree data structure is most important in organizing spatial database. The approaches proposed by Cong et al. [3] and Li et al. [4] employ a hybrid index that augments nodes in non-leaf nodes of an R/R*-tree with inverted indexes.

The work [5] aims to find a number of individual objects, each of which is close to a query location and the associated keywords (or called document) are very relevant to a set of query keywords (or called query document).

2. LITERATURE SURVEY

Spatial database keyword query search is becoming increasingly important in many applications consisting of spatially related data such as oceans, lakes, continents, countries, buildings, airports, states, cities, hospitals, universities, temples and so on. Spatial database consists of all the details of various objects that are spatially located across earth. Whenever spatial database query is posed, the features of query are: query location, and a group of keywords. Each keyword given in the query is associated with spatial database objects. Actually there may exist many spatial database objects for a given one query keyword but the query consists of a set of few keywords. Whenever a spatial database query is posed, it will be executed with respect to the query location attribute. Spatial database objects details are retrieved with respect to the query keywords and query location only.

The linear interpolation function [6] is used to obtain the score of the set O such that the score is a linear interpolation of the individually normalized diameter and the minimum keyword rating of O. Since it is likely none of individual objects is associated with all query keywords, this motivates the studies [7] to retrieve multiple objects, called keyword cover, which together cover (i.e., associated with) all query keywords and are close to each other. The baseline algorithm is inspired by the mCK query processing methods [8]. Since it is likely no individual object is associated with all query keywords, some other works aim to retrieve multiple objects which together cover all query keywords, [8].

With respect to the query location there exist one-to-one correspondence between the query keyword and corresponding spatial database objects. Different queries may require different combination of spatial database objects. All the retried objects against the spatial database query must satisfy some desirable properties. Some of the most desirable properties are - minimum inter object distance, maximum score and spatial database objects must be retrieved to the nearest location of the query, and retrieved objects must cover all the query keywords.

Already existing m closet keywords (mCK) spatial database query tries to find spatial database objects which cover all query keywords and have the smallest inter object distance property. The disadvantages of mCK query search problem are its limited scalability. Hence, actually what is needed is more scalable spatial database query keywords search algorithm, that is also based on object ratings.

Exiting m closet keywords spatial keyword search problem is mainly based on the spatial indexing tree data structure called bR^*-tree that uses bit map index. Here, bit 1 indicates presence of objects and bit 0 indicates absent of keyword. Tree nodes are combined in order to generate potential candidate keyword cover. For fast searching Depth First Search (DFS) traversal is used. Unfortunately mCK must handle very

large number of query keywords and it may not be suitable for many real time applications.

A new indexing technique is proposed as an extension to the bR*-tree indexing. This new indexing technique is called virtual bR*-indexing which can handle very large sizes of keywords. An R*-tree like structure is used in this new technique. New technique uses multi-way spatial join state-of-the-art method that uses 'n' number of query keywords. For obtaining desired results these trees are combined systematically.

3. PROBLEM DEFINITION

Spatial database is a collection of spatial database objects. Each object is clearly and definitely associated with one keyword. A spatial database object is represented by a 5-tuple,

(object_id, x, y, keyword, rating)

Assume that there are n-spatial database objects and

$O = \{o_1, o_2, o_3, \ldots\ldots\ldots\ldots, o_n\}$

Diameter(O) = maximum distance between any two objects within the set, O of spatial database objects.

Newly proposed state-of-the-art spatial keyword cover search uses two parameters - inter object distance as well as keyword rating. A mathematical function defines score of the set O as

Score of O = score of (A, B)

$$\rho\left(1 - \frac{A}{maximum\ distance}\right) + (1-\rho)\frac{B}{maximum\ rating}$$

Where A = Diameter of the set of objects, O and B = minimum object rating among all the spatial database objects and ρ is the special parameter and its value may be any real number in the range $0 <= \rho <= 1$. That is its range is the interval [0, 1]. ρ is the one of the most important decision parameter in modeling the spatial database problem. When the special parameter is zero the problem is completely based on object rating only and when the special parameter is 1, the problem is completely based on inter object distance only. Special parameter value is decided based on application domain. When inter object distance and object ratings are given equal priority then $\rho = \frac{1}{2}$.

Here, each node details are represented using three dimensional where rating is denoted as the third dimensional attribute. In other words, the new indexing tree data structure is really an extension to the original R*-tree indexing structure. For uniform processing purpose object rating values are normalized in the range [0, 1].

One solution for representing spatial database objects is to store all the keywords in only one R*-tree like indexing tree data structure. Other alternative is to store 'n' number of distinct keywords in 'n' number of distinct R*-tree like three dimensional structures. Also assume that there exist, one-to-many relationship from keyword to objects. For example, some of the possible potential spatial database objects are universities, airports, hotels, cities, buildings, bus-stands, railway stations, districts, lakes, hospitals and so on.

X value is normalized to $\frac{x}{maximum\ x\ value}$

Y is normalized to $\frac{y}{maximum\ y\ value}$

Rating is normalized to $\frac{z}{maximum\ z\ rating\ value}$

In the present study a separate indexing tree data structure is created for each query keyword. For example, if the query contains maximum of 7-keywords, then 7 indexing tree data structures are created. Indexing trees must be updated frequently.

Sometimes it may be necessary to generate all possible keyword covers of the given input spatial database keyword query. In the literature of spatial database queries some of the techniques follow the procedure of combining individual nodes incrementally. A technique was available to reduce the size of the intermediate computations and also to prune the further extra computations which are unnecessary. If the score value of the combination is less than the score value of currently computed score then it is not required to generate the superset.

4. ALGORITHMS

Existing algorithm for spatial database management is called m closest keyword algorithm. Original mCK algorithm for spatial query keyword search accesses indexing data structure from top to bottom manner but some other data structures accesses indexing data structure from bottom to top manner also. Experiences have shown that top-down index accessing methods are more efficient than bottom-up index accessing methods. Only single indexing data structure is used for storing all the keywords of different objects. But, more advanced algorithm called best keyword cover stores many 'n' number of indexing trees for 'n' distinct spatial database query keywords. Candidate keyword covers are generated by retrieving and combining child nodes of the indexing trees.

ALGORITHM-1 Basic_Closest_Keyword(S, start)

INPUT:

S is a set of query keywords and

Start is the set of all root nodes.

OUTPUT:

Optimal keyword set

1. bestsetofkeywords ← 0
2. heap←findCandidateKeywords(S,Start, bestsetofkeywords)
3. while(heap is not null) do
4. candidate keyword set ← maximum score of set in heap
5. delete candidate keyword set from heap
6.DfsTreeSearch(heap,S,candidateKeywordset,bestkeywords)
7. for every candidate in heap do
8. if (candidate.score <= bestsetofkeywords.score) then
9. delete candidate from heap
10. end of if
11. end of for
12. end of while
13. return bestsetofkeywords

ALGORITHM-2 DfsTreeSearch(heap, S, Candidatekeywordset, bestsetofkeywords)

INPUT:

heap is the candidate set

S is the set of query keywords

candidate keyword set is a candidate

bestsetofkeywords is the present best keyword cover

OOUTPUT:

Modified present bestsetofkeywords

1. If candidatekeywordSet consists of leaf nodes then
2. O ← objects in candidateKeywordsSet

3. tempbestsetofkeywords ← select the keyword cover with the highest score found in O
4. If bestsetofkeywords.score < tempbestsetofkeywords then
5. bestsetofkeywords ← tempbestsetofkeywords
6. else
7. latestcandidates←findCandidateKeywords(S,candidate keywordset, bestSetofkeywords)
8. replace candidate keywordset by latestcandidates
9. candidatekeywordset← candidateSet with highest score
10. DfsTreeSearch(heap,S,candidatekeywordset, bestsetofkeywords)

ALGORITHM-3

FindCandidateKeywords(S, start, bestsetofkeywords)

INPUT: S is the set of query keywords,

start is the candidate cover ,

bestsetofkeywords is present best solution

OUTPUT:

collection of new candidate covers

1. latestcandidates ← 0
2. combined ← combined child nodes of start to create keyword covers
3. for each c in combined do
4. if (c.score > bestset of keywords.score) then
5. latest candidates ← c
6. end of if
7. end of for
8. return latestcandidates

Original fundamental base algorithm called mCK keyword search algorithm for query keyword cover combines large number of spatial database objects. Even the best pruning version also combines exhaustive number of keywords of objects and as a result the performance of algorithm reduces drastically whenever input number of query keywords increases in size. Hence, a more sophisticated and state-of-the-art spatial database keyword search algorithm is required. One of the latest keyword search algorithm is keyword nearest neighbor algorithm. It introduces the concept of a particular query keyword called the principal query keyword. All the objects that are directly linked to the principle query keyword are called principle objects. It is

customary to denote the set of principle query objects by the set O_k where k is the principle query keyword.

Initially a set of principle query keywords are selected and then local best keyword cover is determined separately for each principle query keyword. Finally one best keyword cover called global best keyword cover is determined after comparing all local best keyword covers.

A new method is proposed for selecting principle query keyword. In general, any keyword can be selected as the principle query keyword. But proposed algorithm introduces a new procedure that always selects the principle query keyword as the keyword that contains the minimum number of objects. Whenever selected principle query keyword contains minimum number of objects, then its performance increases drastically. Once principle query keyword is selected, then best keyword cover with its neighbors are computed incrementally for scalability and pruning purpose. For a given principle query keyword, it is used in computing local best keyword cover by using all the non-principle query keywords.

PROPOSED ALGORITHM-4

NearestKeywordCover(S, SD)

INPUT: S is the set of query keywords,

SD is the spatial database

OUTPUT:

BestKeywordCover

1.sort all the special database indexing structures with increasing order of number of query objects of each query keyword

2.bestkeywordcoverscore ← 0

3.k←select the principle query keyword with minimum number of attached objects

4.newnodes ←child nodes of the indexing tree T_k

5.for each child node in the set of newnodes do

6. compute local $bestscore_k$

7. find maximum $localbestscore_k$

8.end for

9.maxlocal ← $localbestscore_k$

10.while newnodes is not empty do

11. node ←get child node of newnodes head
12. for each $node_k$ in node do
13. find localbestscore
14. insert $node_k$ in node
15. end for
16. remove newnodes head from newnodes
17. if bestkeywordscore < $localbestscore_k$ then
18. bestkeywordscore = $localbestscore_k$
19. end if
20. end of while
21. return bestkeywordscore

5. EXPERIMENTAL RESULTS

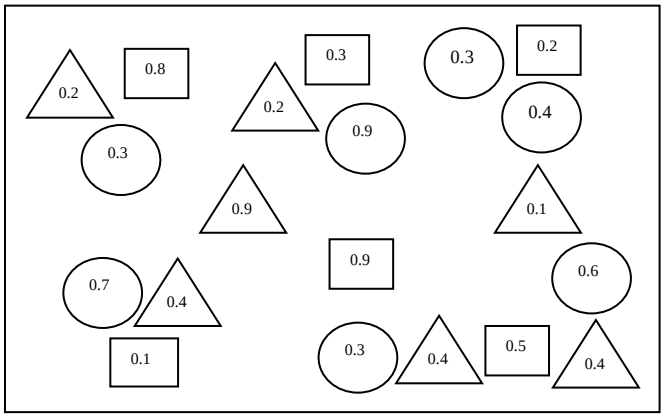

FIG-1 Sample spatial database objects

Triangles represent spatial database objects corresponding to the query keyword TEMPLE, squares represent spatial database objects corresponding to the query keyword UNIVERSTY, and circles represent spatial database objects corresponding to the query keyword HOSPITAL. Spatial database objects details are represented in 3-dimensional coordinates-x, y, and z. Here, z-axis represents object ratings. For simplicity only 3 keywords are considered and the corresponding their objects are shown in TABLES 1, 2 and 3 respectively. Same algorithm can be used for handling spatial database queries containing ‘n’ number of keywords.

TABLE-1: UNIVERSITY KEYWORD DETAILS

X1	Y1	X2	Y2	R1	R2
50	600	100	700	20	20
200	200	300	300	40	30
200	500	300	600	90	10
300	550	400	650	20	20
800	400	900	500	10	10
600	150	750	250	40	30
900	150	1000	300	60	30

TABLE-2: TEMLE KEYWORD DETAILS

X1	Y1	X2	Y2	R1	R2
100	650	200	800	80	20
150	200	250	300	10	10
400	700	500	800	30	20
400	300	550	450	90	10
800	600	900	750	20	20
700	150	900	300	50	20

TABLE-3: HOSPITAL KEYWORD DETAILS

X1	Y1	X2	Y2	R1	R2
100	500	200	600	30	20
100	300	200	400	70	10
450	500	550	600	90	10
700	600	800	700	30	20
800	550	900	650	40	20
450	100	550	250	30	20
900	200	1000	400	60	20

OUTPUT: FINAL BEST KEYWORD COVER IS

200.0, 500.0, 300.0, 600.0, 90.0, 10.0

400.0, 300.0, 550.0, 450.0, 90.0, 10.0

450.0, 500.0, 550.0, 600.0, 90.0, 10.0

BUILD SUCCESSFUL (total time: 8 seconds)

Objects belonging to each keyword are represented in a separate table. Initially 3 indexing structures are created for 3 keywords. After execution of the algorithm three spatial database objects that are selected finally for a given spatial database query are shown above. Result is based on both minimum inter object distance and maximum object ratings. The algorithm executes based on principle local best keyword selection with many neighbors.

6. ANALYSIS OF ALGORITHMS

6.1 Average Based Keyword Ratings

Sometimes the average of the keyword rating of objects is more useful in finding the scores of objects of the set of spatial database objects, O. Consider the set of keywords (Temple, University, City, hospital, and Research). Different users may assign different weights to each of the keywords. For example, Research keyword is assigned more weight than the weight of the keyword City.

6.2 Analysis of Original Fundamental Keyword Cover Algorithm

During the keyword cover search processing, for each query keyword, the child nodes of the indexing trees are retrieved and then these child nodes from different query keywords are combined to generate real candidate keyword covers. This process is repeated until the keyword cover contains only objects but not keywords. Candidate covers are pruned if their score is less than the score of present best keyword cover. All the remaining candidates are processed in the same manner.

For efficient memory management depth first search procedure is used for searching and then obtaining current best solution as soon as possible. There exists other keyword cover browsing strategy called best first search browsing. Best first search algorithm produces global optimal solution where as depth first search algorithm produces local optimal solution and it follows greedy algorithmic approach. Further processing is needed in original best keyword cover algorithm as it generates more and more candidate keyword covers than the state-of-the-art best keyword cover algorithm.

6.3 Analysis of Keyword Nearest Neighbor state-of-the-art Algorithm.

Keyword nearest neighbor approach is based on best first search strategy with minimum memory requirements. Best first search algorithm forms groups in such a way that each group is associated with one principal query keyword. Always this algorithm stores and maintains the keyword cover with highest score in main

memory. Approximately for a given ‘n’ number of principle objects, the total number of principle nodes generated is O(n*logn). Here the memory requirement for storing all candidate keyword covers is O(n logn). Nearly this algorithm maintains nearly linear memory requirements and also notes that optimality conditions holds good differently for different sample instances.

6.4 Parameter to be considered in advanced keyword cover algorithm

1. Time complexity of the algorithm
2. Query response time.
3. Time complexity list of different instances of keyword sets.
4. Scalability of the query keywords.
5. Maximum number of candidate keyword covers.
6. Maximum main memory requirements
7. Average number of nearest neighbor keywords that are retrieved for producing answer to the given query.
8. Performance computations of the algorithm.
9. Total number of local best keyword covers computed for answering query answer.
10. Average query response time.
11. Query processing efficiency.

Query processing efficiency is related to the number of query keywords. Also note that existing keyword cover algorithm is inferior to the proposed new nearest neighbor keyword cover algorithm and the proposed algorithm is more scalable than the existing keyword cover algorithm. New nearest neighbor algorithm is more robust, efficient, fast, accurate and scalable when compared with existing algorithm. The performance of the existing algorithm increases rapidly as the number of query keywords increases. But in the case of proposed advanced algorithm the performance increases slowly as the number of keywords increases. Nearest neighbor query keyword cover algorithm always produces robust and scalable solutions to many of the real and practical problems.

CONCLUSION

Existing spatial database keyword cover algorithm for retrieving special database objects is called mCK algorithm and the new algorithm for retrieving special database objects is called nearest neighbor keyword cover algorithm with more advanced features. The advanced algorithm is practically feasible and suitable for many real time applications and it is more robust, scalable, high performance, efficiency, minimum memory requirements, practical and reasonably very good in many real time applications. New algorithm reduces the intermediate computations drastically in computing keyword covers. Also new algorithm uses many indexing tree data

structures with one-to-one correspondence between keywords and corresponding index tree data structures.

In the future still there is a possibility to reduce many of the intermediate computation results during spatial database keyword cover search using different and more advanced indexing data structure techniques based on local best principle keyword selection. Intermediate results size will be decreased when more advanced local best keyword cover techniques are used on the dynamic fly.

REFERENCES

[1] Ke Deng, Xin Li, Jiaheng Lu, and Xiaofang Zhou, , "Best Keyword Cover Search," IEEE TRANS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 27, NO. 1, JANUARY 2015, pages 61 – 73

[2] I. D. Felipe, V. Hristidis, and N. Rishe, "Keyword search on spatial databases," in Proc. IEEE 24th Int. Conf. Data Eng., 2008, pp. 656–665.

[3] G. Cong, C. Jensen, and D. Wu, "Efficient retrieval of the top-k most relevant spatial web objects," Proc. VLDB Endowment, vol. 2, no. 1, pp. 337–348, Aug. 2009.

[4] Z. Li, K. C. Lee, B. Zheng, W.-C. Lee, D. Lee, and X. Wang, "IRTree: An efficient index for geographic document search," IEEE Trans. Knowl. Data Eng., vol. 99, no. 4, pp. 585–599, Apr. 2010.

[5] J. Rocha-Junior, O. Gkorgkas, S. Jonassen, and K. Nørva g, "Efficient processing of top-k spatial keyword queries," in Proc. 12th Int. Conf. Adv. Spatial Temporal Databases, 2011, pp. 205–222.

[6] S. B. Roy and K. Chakrabarti, "Location-aware type ahead search on spatial databases: Semantics and efficiency," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2011, pp. 361–372.

[7] D. Zhang, B. Ooi, and A. Tung, "Locating mapped resources in web 2.0," in Proc. IEEE 26th Int. Conf. Data Eng., 2010, pp. 521–532.

[8] D. Zhang, Y. Chee, A. Mondal, A. Tung, and M. Kitsuregawa, "Keyword search in spatial databases: Towards searching by document," in Proc. IEEE Int. Conf. Data Eng., 2009, pp. 688–699.