Query Interface Search on Spatial Databases

Chandra Asha . N* and C. Sudarsana Reddy

 # Student of M.Tech, KMM Institute Of Technology and Sciences, Tirupati, Andhra Pradesh, India

 # Department of CSE, KMM Institute Of Technology and Sciences, Tirupati, Andhra Pradesh, India

Abstract— **A common Technology To administer a huge collections of comparatively simple and related objects are spatial databases. In various related applications a user or surfer requires to find various related objects closest to a particular position or location that includes a set of keywords. For example, List the various names of all book sales related stores with n miles of Minneapolis or A catalog of all customers who stay in Chennai and its adjoining Towns. For the given query we require return value in which the user attains a list of various information's related to businesses and its closest objects with a detailed description contains the required keywords, and other information related to their distance from the specified Information or the detailed information of all towns and their adjacent information providing the town details, distance and other related or closest information. In Search information related to nearest neighbor we face a problem on spatial data and various words search on text or file data which is a extensively studied in other fashion or separately. In this paper we provide and present an efficient method to the problem providing a answer to top-k spatial keyword queries using indexing Structure also referred as Information Retrieval R Tree. The proposed algorithms are provided to compare the Existing methods and are exposed to have better presentation of performance.**

Keywords— **Data Mining; Spatial Database, Non Spatial Database, Information Retrieval.**

I. INTRODUCTION

The large growing of various applications uses requires the well organized running and execution of closest and nearest neighbor keywords and queries constrained by the properties of the closest or spatial objects. In Internet search of keywords and provide nearest and closest should contain, in the detailed description or other elements. An example Sales and purchases of houses in real estate domain site provides the surfer to search for various related keywords and its descriptions with rank and details. A easy and simple but yet very popular, which is used in our better example, is the first distance spatial query keyword search, where all the related keywords and objects are ranked by distance and keywords. In the given Figure 1, which is our example, displays a various dataset a set of descriptive attributes.

The Information Release Structure is an R-Tree where a various cross activities are added to each node of the Information Retrieval Structure Tree to indicate the data and textual content of all closest and nearest related objects in the rooted at various objects.

This work has the following contributions:

The problem of keyword search in top-k spatial related data keyword search is varied in the definition and defined. The Information Tree is planned and future as an efficient indexed organizing structure to store spatial and data information for a various data sets of objects. Efficient algorithms are also presented to maintain the IR2-Tree, that is, insert and delete objects.

An efficient incremental algorithm is presented to answer top-k spatial keyword queries using the IR2-Tree. Its performance is evaluated and compared to

current approaches. Real datasets are used in our experiments that show the significant improvement in execution times.

The following sample datasets of hotel objects are taken as multi dimensional objects for our running example to perform the closest nearest search.

	Name	Latitude	Longitude	Amenities
H_1	Hotel A	25.4	-80.1	tennis court, gift shop, spa, Internet
H_2	Hotel B	47.3	-122.2	wireless Internet, pool, golf course
H_3	Hotel C	35.5	139.4	spa, continental suites, pool
H_4	Hotel D	39.5	116.2	sauna, pool, conference rooms
H_5	Hotel E	51.3	-0.5	dry cleaning, free lunch, pets
H_6	Hotel F	40.4	-73.5	safe box, concierge, Internet, pets
H_7	Hotel G	-33.2	-70.4	Internet, airport transportation, pool
H_8	Hotel H	-41.1	174.4	wake up service, no pets, pool

Figure 1: Sample dataset of hotel objects.

This research paper is well arranged and organized into various sections. In which we get top n spatial keyword information, top n keyword specific problem and its detailed information and IR 2 tree and various maintenance algorithms with various experimental results showing the output of the baseline algorithm with various discusses work and conclusion for our future work showing the advanced implementations for further extension.

In this our proposed work, a spatial or Nearest object n is defined as a pair (n.p,n.t), where n.p is a location descriptor in the multidimensional space, and n.t is a text document. Let x be the universe of all objects in a database. In the above Figure 1, n.p is the point composed of "latitude" and "longitude", while n.t is the concatenation of the "name" and "amenities" attributes.

A top-k spatial query Qs searches through the multidimensional space to find the k nearest objects to the specified query point p. The spatial objects are ranked by distance such that an object closer to p has a higher rank. In particular, score(T) = distance(T.p, p). For example, in Figure 1, object H4 is ranked first, given p=[30.5, 100.0].

II. LITERATURE SURVEY

Nearest Neighbor Queries

Answering k-nearest neighbor queries on a spatial database is a classical database problem. Most methods use indices built on the data to assist the k-NN search. Perhaps the most widely used algorithm is the branch-and-bound algorithm which traverses an R-tree while maintaining a list of k potential nearest neighbors in a priority queue. There have also been attempts to use range queries to solve the k-NN search problem, such as the one proposed by Korn et al.. The basic idea is to use a range query to retrieve the potential k-NNs. This algorithm is further extended by improving the region estimation, and by a better search technique of the k-NN in the region.

Top-k queries

Top-k query handle the aggregation of attribute values of objects in the case where the attribute values lie in different sources. For example consider the problem of ordering a set of restaurants by distance and price. They present an optimal sequence of random or sequential accesses on the sources (e.g., Zagat for price and Mapquest for distance) in order to compute the topk restaurants. They view the sources as black boxes in contrast to our work where we assume full access which allows us to build an IR2-Tree.

Figure 2 shows an example of an R-Tree using the hotel dataset of Figure 1. An MBR is represented by its southwest and its northeast points. An R-Tree is typically stored on disk and each R-Tree node takes a whole disk block; hence access to a node requires one disk I/O. The number of children each node can reference is called node capacity.

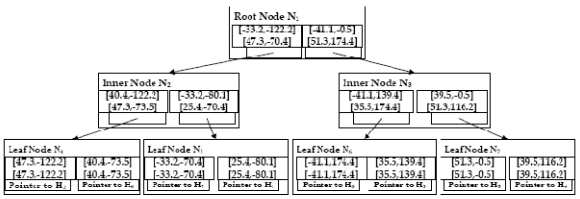


Figure 2: R-Tree for dataset of Figure 1.

The Incremental Nearest Neighbor algorithm presented by Hjaltason and Samet [HS99] uses the structure of an R-Tree to access a minimal number of R-Tree nodes and objects to retrieve the objects nearest to a given point or area in an incremental fashion. Figure 3 shows the Incremental Nearest Neighbor algorithm for two-dimensional objects. The input parameters are a point p, which is the query point (an area could be used instead), and a priority queue U which is initialized with the root of the R-Tree R. Line 2 returns the queue element which has the smallest distance from the query point.

```
NearestNeighbor (p,U)
    /* priority queue U initially contains root node of R with distance 0 */
1   while not U.IsEmpty()
2     E ← U.Dequeue()
3     if E is a non-Leaf Node
4       for each (NodePtr,MBR) in E
5         U.Enqueue(LoadNode(NodePtr),Dist(p,MBR))
6     else if E is a Leaf Node
7       for each (ObjPtr,MBR) in E
8         U.Enqueue(ObjPtr,Dist(p,MBR))
9     else /* E is an object pointer */
10      return E as next nearest object pointer to p
```

Figure 3: Incremental Nearest Neighbor algorithm.

III. IR TREE

The IR2-Tree is a combination of an R-Tree and signature files. In particular, each node of an IR2-Tree contains both spatial and keyword information; the former in the form of a minimum bounding area and the latter in the form of a signature. An IR2-Tree facilitates both top-k spatial queries and top-k spatial keyword queries as we explain below. More formally, an IR2-Tree R is a height-balanced tree data structure, where each leaf node has entries of the form (ObjPtr, A, S). ObjPtr and A are defined as in the R-Tree while S is the signature of the object referred by ObjPtr. A non-leaf node has entries of the form (NodePtr, A, S). NodePtr and A are defined as in the R-Tree while S is the signature of the node. The signature of a node is the superimposition (OR-ing) of all the signatures of its entries.

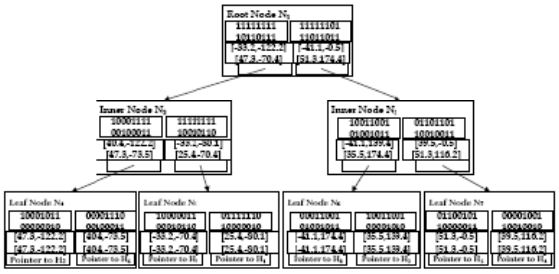


Figure 4: IR2-Tree for dataset of Figure 1.

The IR2-Tree is maintained through insert and delete operations, which are modifications of the corresponding RTree operations. The data sets show the Insert and Delete algorithms respectively. The Insert algorithm uses a standard R-Tree implementation of ChooseLeaf, which can be found in. We use the standard Quadratic Split technique for node splitting. We modify the standard AdjustTree method to also maintain the signatures of the modified nodes. That is, if a new bit is set to 1 in a node N, then it must be also set to 1 for N's ancestors. Finally, we assume that all tree related algorithms have implicit access to the root node of the IR2-Tree R. The input of the Insert algorithm is a pointer to an object T, its MBR, and its signature. Line 1 retrieves a leaf node N which is best suited according to the MBR of T. Then T's pointer, MBR, and signature are stored in N. If N has reached its maximum node capacity then it will split. If N is split into nodes O and P, on Line 4, and it is the root node, a new node M will be created. M becomes the parent O and P and stores their pointer, MBR, and signature. Finally, M is declared the new root node. If N is not the root then its parent node has to be updated as is the case on line 14 or 18. Finally, since we assume that the IR2-Tree is disk resident, the StoreNode function stores the

node to the corresponding disk block(s). Standard implementation of FindLeaf is used in the implementation of Delete. However, CondenseTree is modified to maintain the signatures of updated nodes, similarly to AdjustTree above. In Line 1 of Figure 8, a search for a leaf node N containing an unwanted object T is performed. If such N exists, T is removed from N, otherwise the algorithm stops. If T is removed, the tree is condensed and proper tree maintenance takes place.

```
Insert(ObjPtr,MBR,S)
1   N ← ChooseLeaf(MBR)
2   N.Add(ObjPtr,MBR,S)
3   if N needs to be split
4     {O,P} ← N.Split() /*nodes O and P are returned*/
5     if N.IsRoot()
6       initialize a new node M
7       M.Add(O.Ptr,O.MBR,O.S)
8       M.Add(P.Ptr,P.MBR,P.S)
9       StoreNode(M)
10      StoreNode(O)
11      StoreNode(P)
12      R.RootNode ← M
13    else
14      AdjustTree(N.ParentNode,O,P)
15  else
16    StoreNode(N)
17    if not N.IsRoot()
18      AdjustTree(N.ParentNode,N,null)
```

Figure 5: Insert method for IR^2-Tree.

Clearly, the complexity of the Insert and Delete algorithms is the same as in an R-Tree, since the only additional operation is the maintenance of the signatures of the updated nodes and their ancestors. Note that the updating of the signatures throughout a node and its ancestor is being done at the same time the tree would normally update the MBR of a node and its ancestors.

Multilevel IR2-Tree

A drawback of the IR2-Tree described above is that the same signature length is used for all levels which leads to more false positives in the higher levels, which have more 1's (since they are the superimpositions of the lower levels). To address this problem, we use varying signature lengths for different levels. This is achieved using multi-level superimposed coding [CS89,DR83,LKP95], which reduces the number of false positives, particularly in non-leaf nodes. In this case, we use the optimal signature length for each level (we use the optimal signature length formula from [MC94]), and superimpose the signatures of all objects in the subtree of each node, instead of the signatures of the children nodes as before. A drawback of this variant, called Multi-level IR2-Tree (MIR2-Tree), is that it significantly increases the complexity of the tree maintenance operations (Insert and Delete) since for each object inserted or deleted, we have to recompute the signatures of all ancestor nodes by accessing all underlying objects and not just by superimposing the children's signatures as before. We compare the performances of IR2- Tree and MIR2-Tree in Section 6.

IV. ALGORITHMS TO ANSWER TOP-K SPATIAL KEYWORD QUERIES

We consider two baseline algorithms, in Section 5.1, which are named based on the underlying data structures they use: the R-Tree, and the Inverted Index Only (IIO). In Section 5.2 we present the distance-first IR2 algorithm which uses the IR2- Tree structure to answer distance-first top-k spatial keyword queries. Then in Section 5.3 we present the general IR2 algorithm which uses the IR2-Tree structure to answer general top-k spatial keyword queries. Note that these last two algorithms can also operate on MIR2-Trees with no modification

Current Baseline Algorithms

For simplicity we describe the R-Tree and the IIO baseline algorithms for the simpler distance-first top-k spatial keyword queries, which are also used in the experiments (Section 6). Both algorithms can be extended to answer general top-k spatial keyword queries.

R-Tree Algorithm

The first baseline algorithm, R-Tree, makes use of only an R-Tree data structure. Given a distance-first top-k spatial keyword query, the algorithm first finds the top-1 nearest neighbor object to the query point Q.p. Then it retrieves that object (since the R-tree only contains object pointers) and compares that object's textual description with the keywords of the query. If the comparison fails then that object is discarded, and the next nearest object is retrieved. The incremental NN algorithm in Figure 3 ([HS99]) is used. This process continues until an object is found whose textual description contains the query keywords. Once a satisfying object is found it is returned and the process repeats until k objects have been returned.

The drawback of this algorithm is that it has to retrieve every object returned by the NN algorithm until the top-k result objects are found. This potentially can lead to the retrieval of many "useless" objects. In the worst case (when none of the objects satisfies the query's keywords) the entire tree has to be traversed and every object has to be inspected.

IIO Algorithm

The IIO baseline algorithm makes use of an inverted index. It first finds all the objects (object ids) whose text document contains the query keywords by intersecting the lists returned by the inverted index. Let V be the set of objects in this intersection. Then the objects in V are retrieved and the distance between the query point Q.p and each of the objects in V is computed. These objects are sorted and the top-k objects are returned.

V. EXPERIMENTS

To measure the performance of the IR2-Tree, MIR2-Tree, and baseline algorithms, we have implemented all algorithms and underlying data structures in Java. All index structures (R-Tree, IR2-Tree, MIR2-Tree and inverted index) are diskresident.

We focus on the distance-first version of the top-k spatial keyword query, since its results are easier to comprehend and analyze. The spatial objects are stored in a plain text file and the leaf nodes of the tree data structures store pointers to the object locations in the file. We make comparisons based on the disk accessed required to satisfy a query and the execution time. An Athlon 64 3400+ (NewCastle) with 2GB of RAM and 74GB 10000RPM drive was used for the experiments. We present the results of two datasets provided by the High Performance Database Research Center (http://hpdrc.fiu.edu/).

Both datasets are plain text files (tab delimited) where each spatial object occupies a row. The first dataset contains objects that represent hotels and will be referred as the Hotels dataset. The second dataset will be referred as the Restaurants dataset and contains restaurant data. Table 1 shows more details of the two datasets.

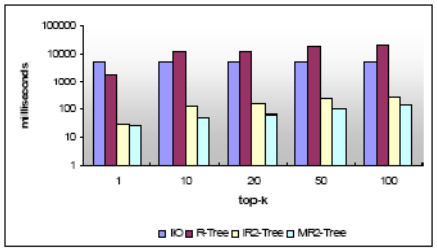
In all experiments the disk block size is 4,096 KB. Also, the number of children of a node of the R-Tree is computed given the fact the each node is a disk block. This translates to 113 children per node in our implementation. We use this same number of children for the IR^2- and MIR^2-Trees, which typically requires two disk blocks per node. As the experiments show, the extra disk block overhead adds to the size of the IR^2- and MIR^2-Trees but has little effect on the execution time. We compare the performance of the IR^2-Tree and MIR^2-Tree algorithm with that of the R-Tree and IIO algorithms. Three sets of experiments were carried out.

The first measures the performance of the algorithms for varying values of requested results k (top-k). The second set measures the effect of the number of query keywords. Finally, the third set of experiments shows the effect of the signature length r.

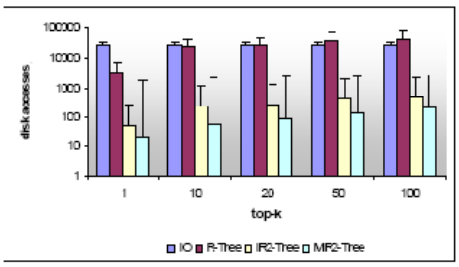
TABLE 1: DATASET DETAILS

Dataset	Size (MB)	Total # of objects	Average # unique words per object	Total # unique words in dataset	Average # disk blocks per object
Hotels	55.2	129,319	349	53906	2
Restaurants	61.3	456,288	14	73855	1

The graph shows that IR2-Tree and MIR2-Tree perform better than R-Tree for all values of k. This is expected since the R-Tree approach will have to access more objects and potentially more tree nodes as well. In contrast, the IR2-Tree and MIR2-Tree use the signatures to prune whole subtrees. In particular, the MIR2-Tree does a better job filtering inner nodes since the optimal signature length is used for each tree level, as described in Section 4. Figures 9b and 12b show the disk block accesses for the algorithms. The thick bars illustrate the number of random disk block accesses while the thin lines on top of the thick bars show the numbers of sequential disk block accesses.



(a) Execution Time



(b) Object Accesses

Vary number of keywords

In this experiment we fix the number of requested objects k to 10 and the signature lengths as above. Refer to Figures 10 and 13 for the results of this experiment. By increasing the number of keywords we reduce the number of objects that contain all of them (since distance-first top-k spatial keyword queries are conjunctive). We note that the IIO algorithm performs better as the number of keywords increases, since the intersection of the inverted lists then becomes shorter and hence the object accesses fewer.

Vary signature length

In this experiment we fix k to 10 and the number of keywords to 2. Refer to Figures 11 and 14 for the results of this experiment. First note, that the signatures chosen for the Hotels dataset are different than those for the Restaurants dataset. This is because a Hotel object contains more unique words than a Restaurant object, as shown in Table 1. Note that the displayed signature lengths are used for the leaf nodes of MIR2-Tree. Longer signatures are used for the top nodes.

There is a trade-off in increasing the signature lengths for IR2-Tree and MIR2-Tree. Increasing the signature length decreases the false positives but increases the occupied space of the tree structures, which can lead to more disk accesses. Hence, there is no clear trend for varying the signature lengths

VI. CONCLUSIONS

In this research paper we provide introduction and solution for the problem of spatial keyword search and provide detailed performance and execution limitations of existing approach. We have enhanced and proposed a dramatic solution which is faster than existing approaches and is based on a amalgamation of R-Trees and signature files methods. In our proposed IR2-Tree we showed how it is maintained in the presence of data updates. An effective and efficient algorithm was presented that uses the IR2-Tree to answer spatial keyword queries. We show our experimentally evaluated our technique, which proved its superior performance.

.

REFERENCES

[1] W. W. Chang, Hans-Jörg Schek: A Signature Access Method for the Starburst Database System. VLDB 1989: 145-153

[2] Yen-Yu Chen, Torsten Suel, Alexander Markowetz. Efficient Query Processing in Geographic Web Search Engines. SIGMOD 2006

[3] U. Deppisch. S-Tree: A dynamic balanced signature index for office retrieval. In Proc. of the ACM Conf. on Research and Development in Information Retrieval, Pisa, 1986.

[4] Ron Sacks-Davis, Kotagiri Ramamohanarao: A two level superimposed coding scheme for partial match retrieval. Inf. Syst. 8(4): 273-289 (1983)

[5] Ronald Fagin, Amnon Lotem, Moni Naor: Optimal Aggregation Algorithms for Middleware. In PODS 2001

[6] Christos Faloutsos: Signature files: Design and Performance Comparison of Some Signature Extraction Methods. In SIGMOD Conference 1985

[7] Christos Faloutsos, Stavros Christodoulakis: Signature Files: An Access Method for Documents and Its Analytical Performance Evaluation. In ACM Trans. Inf. Syst. 2(4): 267-288(1984)

[8] Christos Faloutsos, Stavros Christodoulakis: Design of a Signature File Method that Accounts for Non-Uniform Occurrence and Query Frequencies. In VLDB 1985: 165-170

[9] C. Faloutsos, D. W. Oard. A survey of information retrieval and filtering methods. Technical Report. UMI Order Number: CSTR-3514., University of Maryland at College Park, 1995

[10] A. Guttman. R-Trees: a dynamic index structure for spatial searching. In SIGMOD Conference, 1984.

AUTHOR BIOGRAPHY

N.Chandra Asha, Student of M.Tech, KMM Institute Of Technology and Sciences,Tirupati, Andhra Pradesh, India. Her main areas Data mining.

Email: n.c.asha1206@gmail.com

C.Sudarsana Reddy, Assistant Professor Department of CSE,KMM Institute Of Technology and Sciences,Tirupati, Andhra Pradesh, India. His main areas Data mining.