International Journal of Engineering Research and Development
e-ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com
Volume 10, Issue 7 (July 2014), PP.68-75

Ranking Spatial Data by Quality Preferences

Satyanarayana Maddala [1] , Avala. Atchyuta Rao[2]
[1]*II M.Tech, Gokul Institute of Technology and Sciences, Bobbili, Vizianagaram, India.*
[2]*Asst.Prof, Department of Computer Science and Engineering,*
GOKUL INSTITUTE OF TECHNOLOGY AND SCIENCES, Bobbili, Vizianagaram, India

Abstract:- The objects in real world can be ranked based on the features in their spatial neighborhood using a preference based top-k special query. In this paper, a two purpose query structure for satisfying the user requirements is implemented. For example, a user who wishes to find a hotel with 3 star categories that serves sea food which provides the nearest airport facility. This concept can be obtained by developing a system that takes a particular query as the input and displays a ranked set of top k best objects that satisfy user requirements. For that, an indexing technique R-tree and a search method BB algorithm for efficiently processing top-k spatial preference query is used. R-tree (Real-tree), a data structure is the first index specifically designed to handle multidimensional extended objects and branch and bound (BB) algorithm that makes searching easier, faster and accurate. The key idea is to compute upper bound scores for non-leaf entries in the object tree, and prunes those that cannot lead to better results. The advantage of using this algorithm is that it can reduce the number of steps to be examined.

Index Terms:- Query processing, spatial databases, R-tree**.**

I. INTRODUCTION

The management of large collection of geographical entities is possible with the help of spatial database systems. Apart from spatial attributes Spatial database systems also contain non-spatial values like size, type, price etc. In this paper, a study of an interesting type of preference queries are made, which selects the best spatial object with respect to the quality of features in its spatial neighborhood. Given a set D of interesting objects, a top-k spatial preference query[1] retrieves the k objects in D with the highest scores. The score of an object is defined by the quality of features (facilities or services) in its spatial neighborhood. As an example: The user (e.g., tourist) wishes to find a food facility that may be hotel or restaurant also with different types of transport facility can input these purposes as spatial query. For each hotel 'P' will defined in terms of (i) the maximum quality for each feature in the neighborhood region of the particular position 'P' and (ii) the aggregation of those user requirements.

This paper proposes four types of ranking; (i) spatial ranking: the objects are ranked based on their distance from a reference point, (ii) non-spatial ranking: the objects are ranked by aggregating all the non-spatial values (size, price, type etc.), (iii) neighbor retrieval: In spatial database, ranking is often associated to nearest neighbor (NN) retrieval [2]. Given a query location, we are interested in retrieving the set of nearest objects to it that qualify a condition (example: restaurants). Assuming that the set of interesting objects is indexed by an R-tree, apply distance bounds and traverse the index in a branch and bound fashion to obtain the answer, (iv) spatial query evaluation on R-tree, which is the most popular spatial access method, which indexes Minimum Bounding Rectangles (MBR'S) of objects. R-tree can efficiently process main spatial query types, including spatial range queries, nearest neighbor queries [3], and spatial joins. The top-k spatial preference query integrates these four types of ranking in an intuitive way.

II. LITERATURE SURVEY

Two query processing algorithms are proposed to answer queries in the existing system. One of which is a threshold based method and the other one is based on the hybrid index structure. They are MFA and ATRA, where MFA is a threshold-based algorithm and ATRA that is a AIR-tree based algorithm. Now in this paper, two search methods are proposed, R tree and enhanced branch and bound algorithm with the help of a "Preference based Top-k spatial keyword queries" founded in 2011 by the authors Jinzeng Zhang, dongqi Liu, Xiaofengs Meng[4].

MFA (Multiple Feature Algorithms):

A PT*k*SK query processing method called multiple feature algorithm denoted as MFA [2] is proposed, which is based on threshold algorithm denoted as TA .TA algorithm is a typical method to addressing top-k

query that returns k-tuples with the highest scores according to a monotone function. The PT*k*SK query is partitioned into three features that is query location represents spatial features sf, fuzzy constrains and query keywords respectively corresponds to user preference feature pf, and text feature tf. Therefore the submitted query Q is transformed into a set of three features, and can do adjustment in some extend [5].

ALGORITHM: MFA
Input: *Q*: a **PT*k*SK** query;
k: a positive number of returned results;
Variables: *GT*: a global threshold;
BaScore: best aScore that aggregates the score of the current best objects.
Output: *R*: The top-k objects satisfying *Q*;
1: *Qf* ←**Tranform**(*Q*);
2: *GT* ←0;
3: *BaScore*←∞;
4: *R* ← Null;
5: **for each** each feature *qi* in *Qf* **do**
6: *ti* ← 0;
7: **for** *i* from 1 to *k* **do**
8: **while**(*GT* < *BaScore*) **do**9: **for** *i* from 1 to *3* **do**
10: select query feature *qi*;
11: get the match *o j* of *qi*;
12: *ti* ← socre of *o j* on feature *qi*;
13: update *GT*;
14: **if** *GT* ≥ *BaScore* **then**
15: *break*;
16: compute *aScore*(*o j, Qf*)
17: **if** *aScore*(*o j, Qf*) < *BaScore* **then**
18: cur-bestresult← *o j*;
19: *BaScore*←*aScore(o j, Qf)*;
20: *R* ← *R* ∪ {*o j*};
21: **return *R***

ATRA (ATR Algorithm):

To reduce computation overhead, ATRA algorithm that is based on an effective hybrid indexing structure called AIR-tree (Attribute Inverted File R-tree) is used for query processing [2]. MFA algorithm may incur multiple accesses to the same nodes and retrieve the same data point through different queries. To overcome this drawback , AIR-tree is used to retrieve those objects only containing some query keywords and satisfying user preference, which can avoid checking the irrelevant objects to query (Fig 1). AIR-tree clusters spatially close objects together, and carries textual information and attribute vectors in one node. The attribute vectors are used in users preference similarity computing. The AIR-Tree therefore can improve searching efficiency for PT*k*SK queries [2].

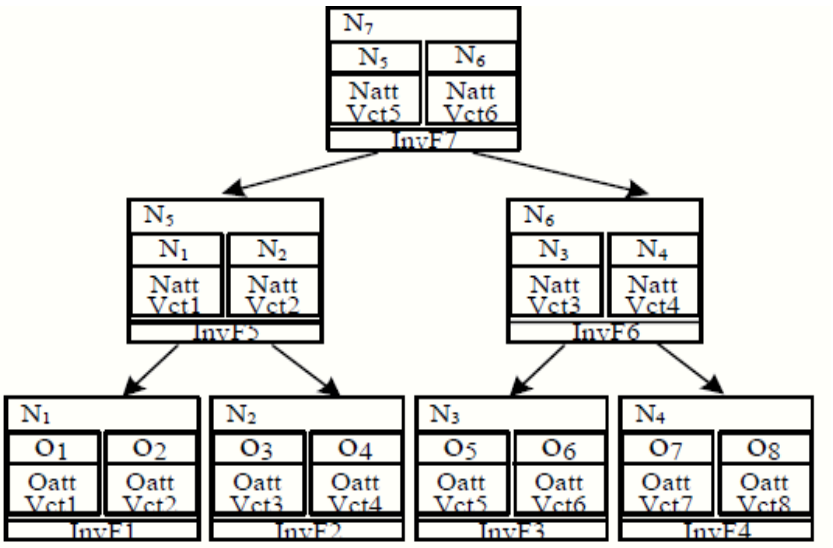


Fig 1: AIR-tree

ALGORITHM: ATRA
Input: *Q*: a **PT*k*SK** query;
T: an AIR-tree;
k: a positive number of returned results.
Output: *R*: the top-k objects satisfying *Q*;
1: *Qf* ← **Quant**(*Q*);
2: *U* ← new min-priority queue;
3: *U.Enqueue*(*T*.root, 0);
4: **while** *U* is **not** empty **do**
5: *E* ← *U.Dequeue()*;
6: **if** *E* is an object **then**
7: *R* ←*R* ∪ *E*;
8: **if** |*R*| = k **then**
9: *goto* 16;
10: **else if** *E* is an intermediate node **then**
11: **for** each entry *e* in *E* **do**12: *U.Enqueue(e, MINaScore(e, Q f))*;
13: **else if** *E* is an leaf node **then**
14: **for** each object *o* in *E* **do**
15: *U.Enqueue(o, aScore(o, Q f))*;
16: **return** *R*;

Limitations of this system:

1) The number of objects to be examined is more using this algorithm.
2) Computing upper bound scores for non-leaf entries in the object tree are not accurate.
3) Takes more time to form the AIR- tree structure.
4) More difficult to implement the Algorithm using the tree structure.

III. RESEARCH ELABORATION

A preference based top-k spatial keyword queries is proposed, that return a ranked set of k best data objects based on the scores of feature objects in their spatial neighborhood, satisfying user's requirements and needs. In order to answer PT*k*SK queries efficiently, an index tree structure called R-tree (Real tree) is proposed, which combines location proximity with preference similarity and textual relevance. Also presents a search algorithm called Enhanced BB (branch and bound). The spatial objects can be searched by use of this search algorithm [6]. In this, the data partitioning method such as R-Tree index is used.

A. R-TREE

R-trees are tree data structures used for spatial access method, i.e.; for indexing multi-dimensional information such as geographical coordinates, rectangles. The R-tree was proposed by Antonin Guttmann in 1984, and has found significant use in both theoretical and applied contexts [7]. A common real world usage for an R-tree might be to store spatial objects such as restaurant location, buildings, etc and then find answers quickly to queries such as “find all museums within 2 km of my current location”, “retrieve all road segments within 2 km of my location” or “Find the nearest gas station” etc. It essentially modifies the ideas of the B-tree to accommodate extended spatial objects. The key idea of R-tree is to group nearby objects and represent them with their minimum bounding rectangle (MBR) in the next higher level of the tree.

B. Time Complexity of R-TREE

i)If MBRs do not overlap on q, the complexity is O (log mN).
(ii)If MBRs overlap on q, it may not be logarithmic, in the worst case when all MBRs overlap on q, it is O (N).

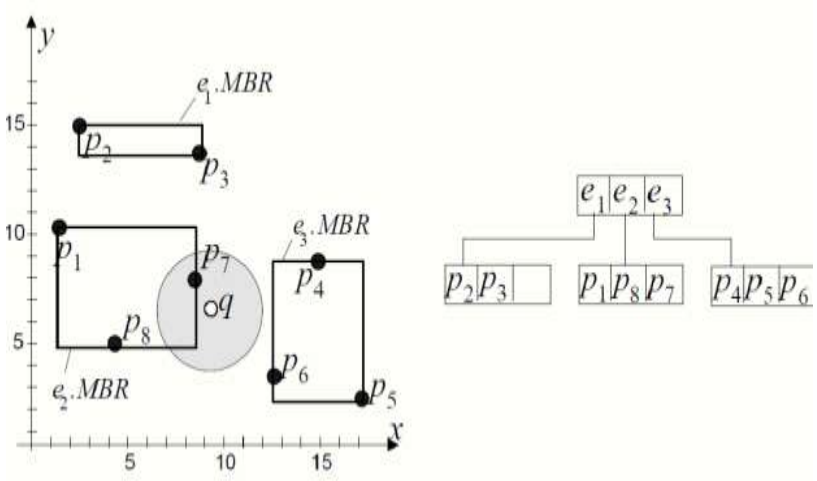


Fig 2: Spatial queries on R-Tree

For example: If we wish to get the ranked nearest object from the point 'q', in R-tree, it first forms Minimum bounding Rectangles (MBR) with the k-best element collection p1 to p8. After ranking we understand that 'p7' is the member who has the user- specified features and shortest distance from the position 'q' as shown in Fig 2.

C. Search Algorithms

The spatial objects can be searched by use of search algorithm. In this algorithm, the data partitioning method such as R-Tree index is used. The basic search algorithm on R-trees, similar to search operations on B-trees, traverses the tree from the root to its leaf nodes [8].

Branch and Bound algorithm:

The key idea is to compute component score, for non-leaf entries E in the object tree D, an upper bound T(E) of the score T(p) for any point p in the sub tree of E. The algorithm uses two global variables: 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).The pseudo-code of branch and bound algorithm (BB) is called with N being the root node of D. If N is a non-leaf node, in this algorithm, the scores T (E) for non-leaf entries E can be computed concurrently. Recall that T (E) is an upper bound score for any point in the sub tree of E. The techniques for computing T (E) will be discussed shortly. The component score Tc (E) is the range score, take maximum quality of points. With the component scores Tc (E) known so far, we can derive T+ (E), an upper bound of T (E). If T+ (E) = ?, then the sub tree of E cannot contain better results than those in Wk and it is removed from 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, then 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, the value of ? is updated during recursive call of BB. To improve the performance of Branch and bound algorithm, Enhanced branch and bound algorithm is developed as follows.

Enhanced Branch and Bound algorithm

Algorithm: Enhanced Branch and Bound
Wk: = new min-heap of size k (initially empty);
?: =0;
// k-th score in Wk
1: Call search algorithm
// Take input as search result E from search algorithm
2: V: {E| E e N}; //V denotes set in which points are to be stored
3: If N is non-leaf then
4: for c: =1 to m do
5: compute T (E) for all E e V concurrently;
6: remove entries E in V such that T+ (E) <= ?;
7: for each entry E e v such that T (E) > ? do
8: read the child node N pointed by E;
9: continue step 2;
10: else
11: for c: =1 to m do

12: compute T (E) for all E e V concurrently;
13: remove entries e in V such that T+ (E) <=V;
14: Sort entries E e V in descending order of T (E);
15: Update Wk (and?) by entries in v;
In branch and bound algorithm, changes have been made in getting input values and also about sorting the entries, resulted with enhanced branch and bound algorithm. The input values of enhanced BB are the output of searching algorithm. Instead of performing sorting individually on each node among its child nodes, entire tree node have been sorted after this process is over. This will reduce the time effectively and improve the performance.

D. Model View Controller

Model-View-controller shown in Fig.3 is a classical design pattern used in applications for who needs a clean separation between their business logic and views that represents data. MVC design pattern isolates the application logic from the user interface and permit the individual development, testing and maintenance for each component. This design pattern is divided into three parts called model, view and controller. Model - This component manages the information and notify the observers when the information changes. It represents the data when on which the application operates. The model provides the persistent storage of data, which is manipulated by the controller. In other words, Model represents an object carrying data. It can also have logic to update controller if its data changes. In my project, Java classes are used to implement this control.

View- The view displays the data, and also takes input from user. View represents the visualization of the data that model contains. It renders the model data into a form to display to the user. There can be several view associated with a single model. It is actually a representation of model data. This control is implemented in my project using java server pages (jsp).

Controller- The controller handles all requests coming from the view or user interface. The data flow to whole application is controlled by controller. It forwarded the request to the appropriate handler. Only the controller is responsible for accessing model and rendering it into various UIs. Controller acts on both Model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps View and Model separate. This control is implemented using java server pages and is maintained in a package named process.

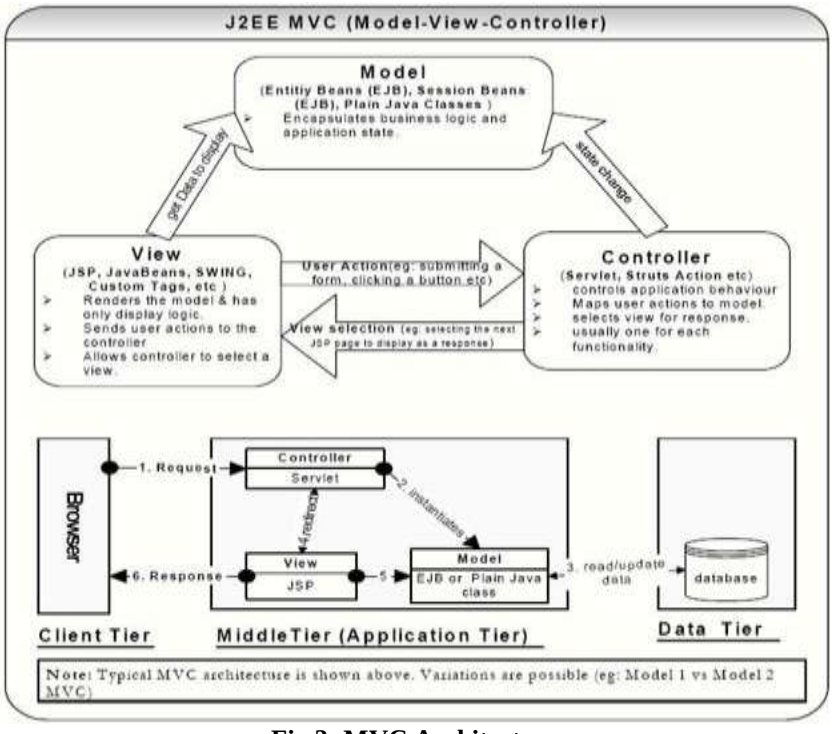


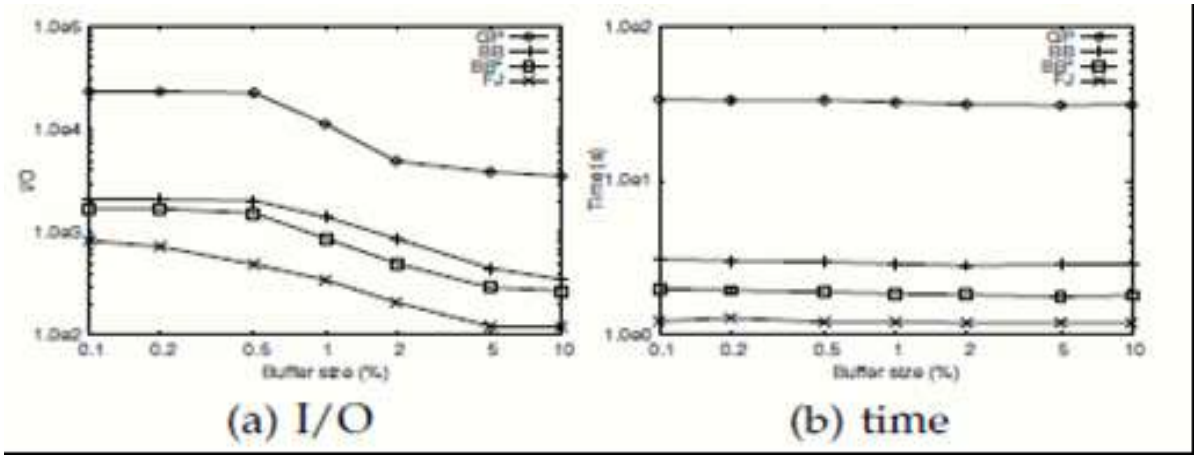
Fig 3: MVC Architecture

IV. SYSTEM ANALYSIS

System Analysis is a process by which we attribute process or goals to a human activity, determine how well those purpose are being achieved and specify the requirements of the various tools and techniques that are to be used within the system if the system performances are to be achieved.

A. EXPERIMENTAL EVALUATION

The efficiency of the proposed algorithms is compared using real and synthetic datasets. Each dataset is indexed by an R-tree with 4K bytes page size. An LRU memory buffer is used whose default size is set to 0.5% of the sum of tree sizes (for the object and feature trees used). The algorithms were implemented in C++ and experiments were run on a Pentium D 2.8GHz PC with 1GB of RAM. In all experiments, both the I/O cost (in number of page faults) and the total execution time (in seconds) of the algorithms are measured.



B. *EXPERIMENTAL SETTINGS*

Both real and synthetic data are used for the experiments. For each synthetic dataset, the coordinates of points are random values uniformly and independently generated for different dimensions. For a feature dataset Fc, qualities for its points are generated such that they simulate a real world scenario: facilities close to (far from) a town center often have high (low) quality. For this, a single anchor point s* is selected such that its neighborhood region contains high number of points. Let distmin and distmax be the minimum and maximum distance of a point in Fc from the anchor s*. Then, the quality of a feature point s is generated as:

$$\omega(s) = \left(\frac{dist_{max} - dist(s, s_\star)}{dist_{max} - dist_{min}}\right)^{\vartheta}$$

Range of parameter values

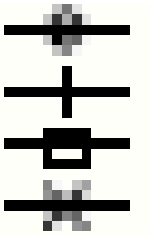
Range of parameter values

Parameter	Values
Aggregate function	**SUM**, MIN, MAX
Buffer size (%)	0.1, 0.2, **0.5**, 1, 2, 5, 10
Object data size, $\|\mathcal{D}\|$ ($\times$1000)	100, **200**, 400, 800, 1600
Feature data size, $\|\mathcal{F}\|$ ($\times$1000)	50, **100**, 200, 400, 800
Number of results, k	**1**, 2, 4, 8, 16, 32, 64
Number of features, m	1, **2**, 3, 4, 5
Query range, ϵ	10, 20, **50**, 100, 200

C. Performance on Queries with Range Scores

This section studies the performance of the algorithms for top-k spatial preference queries on range scores[9]. However, the cost of the other methods is mainly influenced by the effectiveness of pruning. BB employs an effective technique to prune unqualified non-leaf entries in the object tree so it outperforms GP. The optimized score computation method enables BB* to save on average 20% I/O and 30% time of BB.

The four symbols used in the graphical representation are as follows:



These four symbols are stands for GP (Group Probing Algorithm), BB (Branch and Bound Algorithm), BB* (Enhanced Branch and Bound Algorithm), FJ(Feature Join Algorithm) from top to bottom respectively.

Diagram 1:

Diagram 1 plots the cost of the algorithms as a function of the buffer size. As the buffer size increases, the I/O of all algorithms drops. FJ remains the best method, BB* the second, and BB the third; all of them outperform GP by a wide margin. Since the buffer size does not affect the pruning effectiveness of the algorithms, it has a small impact on the execution time.

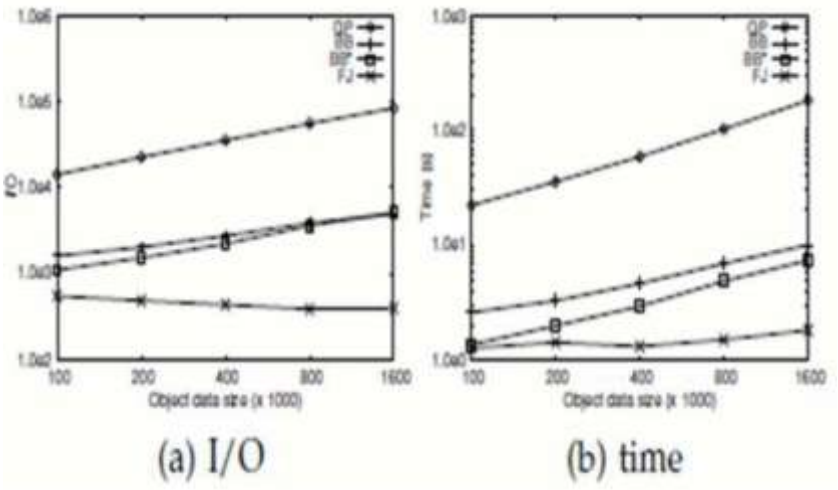


Diagram 2:

Diagram 2 compares the cost of the algorithms with respect to the object data size |D|. Since the cost of FJ is dominated by the cost of joining feature datasets, it is insensitive to |D|. On the other hand, the cost of the other methods (GP, BB, BB*) increases with |D|.

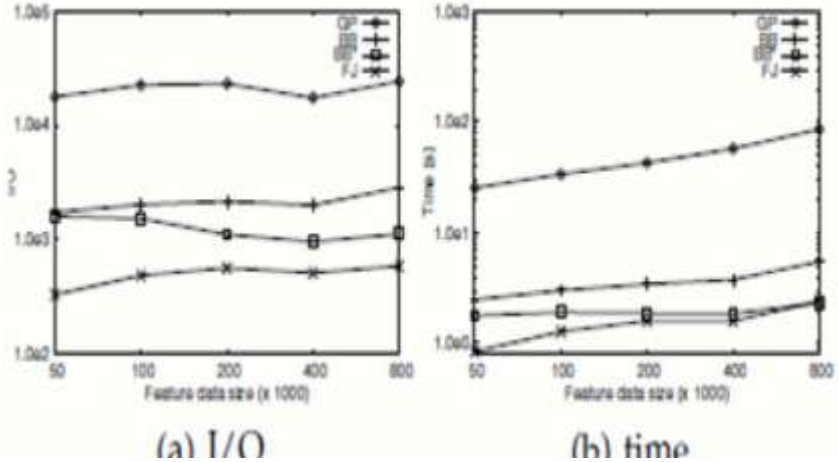


Diagram 3:

Diagram3 plots the I/O cost of the algorithms with respect to the feature data size. As size of dataset increases, the cost of GP, BB, and FJ increases. In contrast, BB* experiences a slight cost reduction as its optimized score computation method (for objects and non-leaf entries) is able to perform pruning early at a large dataset value.

V. FUTURE SCOPE

As a future scope, a study can be made on top-k spatial preference query on a road network, in which the distance between two points is defined by their shortest path distance rather than their Euclidean distance. The challenge is to develop alternative methods for computing the upper bound scores for a group of points on a road network.

The other future developments for additional improvements are as follows:
1) User friendly interfaces can be improved.
2) Security features can be improved: By using additional authentication mechanisms for authenticating users and registered objects.

V. CONCLUSION

The paper presents a comprehensive study of top-k spatial preference queries, which provides a novel type of ranking for spatial objects based on qualities of features in their neighborhood. The neighborhood of an object p is captured by the scoring function (i) the range score restricts the neighborhood to a crisp region centered at p, whereas (ii) the influence score relaxes the neighborhood to the whole space and assigns higher weights to locations closer to p. An index tree structure called R-tree and an Enhanced branch and bound algorithm for processing top-k spatial preference queries is used, that easily ranks the spatial data depends upon qualities. The proposed system is helpful for tourism development and travelling management. By using the site, a user can search the hotels, restaurants and transport facilities, in a city which satisfy his requirements and he can choose a hotel or a restaurant from a ranked list. This ranking method is effective and efficient in various applications.

REFERENCES

[1]. M.L.Yiu, X.Dai, N.Mamoulis, and M.Vaitis, “Top-k Spatial Preference Queries,” in ICDE, 2007.
[2]. K.S.Beyer, J.Goldstein, R.Ramakrishnan, and U.Shaft, “When is “nearest neighbor” meaningful?” in ICDT, 1999.
[3]. Y.Chen and J.M.Patel, “Efficient Evaluation of All-Nearest- Neighbor Queries,” in ICDE, 2007.
[4]. Jinzeng Zhang, Dongqi Liu, Xiaofeng Meng, “Preference Based Top-k Spatial Keyword Queries,” in ACM, 2011.
[5]. Y-Y.Chen, T.Suel, and A.Markowetz, “Efficient Query Processing in Geographic Web Search Engines,” in SIGMOD, 2006.
[6]. E.Dellis, B.Seeger, and A.Vlachou, “Nearest Neighbour Search on Vertically Partitioned High-Dimensional Data,” in DaWaK, 2005.
[7]. A.Guttman, “R-Trees: A Dynamic Index Structure for Spatial Searching,” in SIGMOD, 1984.
[8]. S.Hong, B.Moon, and S.Lee, “Efficient Execution of Range Top-k Queries in Aggregate R-Trees,” IEICE Transactions, 2005.
[9]. I.F.Ilyas, W.G.Aref, and A. Elmagarmid, “Supporting Top-k Join Queries in Relational Databases,” in VLDB, 2003.