ISSN : 0976-8491 (Online) | ISSN : 2229-4333 (Print)

Grading Spatial Data By Value Preferences using N2S2 Algorithm

[1]Suchitra Reyya O, [2]M.Ramesh Babu
[1,2]Dept. of CSE, Pydah College of Engineering Boyapalem,Visakhapatnam, India

Abstract
Customer liking queries are very important in spatial databases. We propose a definition of a spatial database system as a database system that offers spatial data types in its data model and query language and supports spatial data types in its implementation, providing at least spatial indexing and spatial join methods. Spatial database systems offer the underlying database technology for geographic information systems and other applications. We survey data modeling, querying, data structures and algorithms, and system architecture for such systems. The emphasis is on describing known technology in a coherent manner rather than on listing open problems.. The featured score of a given object is derived from the quality of features (e.g., location and nearby features) in its spatial neighborhood
This neighborhood concept can be defined by different functions by the user. It can be an explicit circular region within a given distance from the flat. Another sensitive definition is to assign higher rates to the features based on their proximity to the land. In this paper, we formally define spatial preference queries and propose suitable dynamic index techniques and searching algorithms for them. We extend results with dynamic index structure in order to accommodate time - variant changes in the spatial data. In my current work is the top-k spatial preference query on road network, in which the distance between object and road is defined by their shortest path distance. By separating this query as a subset of dynamic skyline queries N2S2 algorithm is provided for computing it. This algorithm has good performance compared with the general branch and bound algorithm for skyline queries.

Keywords
User preference queries, nearest neighbor, skyline queries, spatial databases.

I. Introduction

With the popularization of geographic information, there has been an increasing number of Webs information systems specialized in providing interesting results through location-based queries. However, most of the existing systems are limited to plain spatial queries that return the objects present in a given region of the space. In this paper, we study a more sophisticated query that returns the best spatial objects based on the features (facilities) in their spatial neighborhood. Given a set of data objects of interest, a top-k spatial preference query returns a ranked set of the k best data objects. The score of a data object is defined based on the non-spatial score (quality) of feature objects in its spatial neighborhood. On the other hand, the score of a feature object does not depend on its spatial location, but on the quality of the feature object.
In many situations for decision making, users need select one or more data from database in accordance with their interest. The selected data must meet their desired constraints. A user wants to select a hotel with less cost and distance to beach. User hasn't accurate asked (for example cost of hotel below 100$ and distance to beach less than 1Km is accurate asked) but wants to find a set of data that are closer to their own interests. Such constraints called soft constraints and queries about these problems called user preference queries other features (e.g., restaurants, cafes, hospital, market, etc.) in the spatial neighborhoods of the flat (defined by a spatial range around it). Quality may be subjective and query-parametric. the locations of an object data set D (hotels) in white, and two feature data sets: the set F1 (restaurants) in gray, and the set F2 (cafes) in black. Quality points are labelled by excellence values that can be obtained from rating providers. For the ease of argument, the qualities are normalized to values in [0,1]. The score T (p) of a hotel p is defined in terms of:
(a) Range score
(b) Influence score
(c) Nearest neighbor

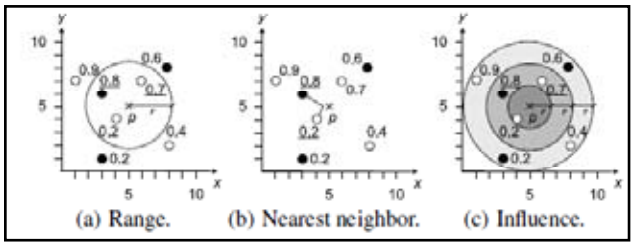


Fig. 1:

User preference queries are very important in spatial databases. Spatial data in addition to no spatial data can be stored in these databases. With the help of these queries, user can find best places in database according to their interest
Related works in this field are in based on top-k queries. In top-k queries setting a weight for each attribute and a scoring function for aggregating attributes are hard for user. Indeed user is more willing to ask for the skyline first in order to get the "big picture" and then apply a top-k query to get more specific results. So the use of skyline query with considering the distances of points to their nearest neighbors is subject that less attention has been on it.
Distance of the point to its nearest neighbor is a dynamic attribute for that point. Dynamic attributes are not directly stored in database but according to data from database can be calculated. Existence algorithms for respond skyline queries such as branch and bound algorithm don't have good performance for dynamic skyline queries and in based on type of dynamic attributes. In this paper a new type of dynamic skyline query called spatial nearest neighbor skyline query is introduced. In these queries, user has some set of query points. For each point in database, attributes are its distances to nearest neighbors from each set of query points. All points that there is a point better than them according to all attributes are deleted and the rest are returned as the answer.
This query is the subset of dynamic skyline queries. By separating it we are trying to provide a more efficient algorithm for computing it. This algorithm that called N2S2 is in based on branch and bound algorithm and such as branch and bound algorithm supposes there is R tree index on data but this solution can be extended to all kind of data partitioning methods on spatial databases.
In this algorithm by considering the density of the query points in high levels of the index instead of accurate distance to nearest neighbors, a better view with less cost is obtained about nodes

of tree may be more promising. Also with storing nodes that are used to find nearest neighbors of parent nodes and using of them for finding nearest neighbor of lower level nodes of tree, many extra fetch of nodes will be avoided.

II. Problem Identification

A Spatial partiality query ranks lands based on the qualities of features in their spatial zone. Feature refers to a class of objects in a spatial plan such as specific conveniences or services. A customer may 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 in the Spatial neighborhood of the flat. Quality may be subjective and query-parametric.

For example, using a landed property agency database of flats for Sale, a customer may want to rank the flats with respect to the appropriateness of their location, top-10 flats with the largest sizes and lowest prices, defined after aggregating the qualities of other features (e.g., restaurants, bus stop, hospital, market, school, etc.) within their spatial neighborhoods. In spatial databases, ranking is often associated to nearest neighbor (NN) recovery. Given a query location, we are interested in retrieving the set of nearest objects to it that qualify a condition (e.g., restaurants). Assuming that the set of interesting objects is dynamic indexed by an R-tree, we can apply distance score bounds and traverse the index in a branch-and-bound approach to obtain the answer.

Therefore, we propose extend results with dynamic index structure in order to accommodate time-variant changes in the spatial data using searching algorithms. Spatial ranking, which orders the objects based on their distance and score from a reference feature, and second Non-spatial ranking, which orders the objects by an combined method on their nonspatial values.

This paper, we propose alternative techniques that aim at minimizing the I/O accesses to the object and feature data sets, while being also computationally efficient. Our techniques apply on spatial-separation access functions and work out score bounds for the objects indexed by them, which are used to effectively trim the try to find space. Specifically, we contribute the Branchland-Bound (BB) algorithm and the Feature Join (FJ) algorithm for efficiently processing the top-k spatial preference query.

A type of spatial user preference queries is presented in it that ranks objects based on the qualities of features in their spatial neighborhood and retrieves the K objects in set with the highest ranks. In fact in [6] user has some sets of query points. Each query point in based on its quality has a score between 0 and 1. For each object in database scores of nearest neighbors from each set of query points are aggregated with a function such as sum and object is ranked in based on result of this function. K points that have highest rank are retrieved as query results.

So for example an object that its nearest neighbor is far from it may have high rank. This method use top-k queries so user should present an appropriate ranking function.

In which, in addition to the nearest neighbor, the scores of the higher level neighbors are considered for ranking objects with the condition that the score of i'th nearest neighbor is divided into 2t . The problem with this method is that it doesn't consider concept of distance exactly and so for example it is not important that how much the distance of an object with its nearest neighbor is or how much the distance of the first and the second nearest neighbors is.

III. KNN Query Processing

A. Nearest Neighbor Query

- Given: a point p, a set of points D and distance function d
- Find: q in D such that for any point q' in D, $d(p,q) \leq d(p, q')$

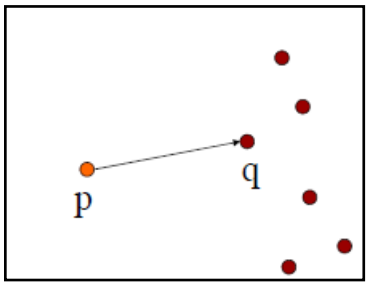


Fig. 2: Linear search – the naïve method

To compute the distance from p to every point in D

- Using spatial indexes – minimize the data accessed
- Search with expanding range
- Start with a circle centered at p, and increase the radius

B. MBR

D-dimensional rectangle, which is the minimal rectangle that fully encloses an object or a set of objects

MBR Face Property:

Every face of the MBR contains at least one point of some object in the database.

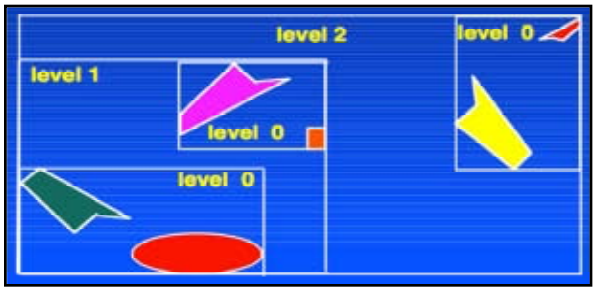


Fig. 3:

1. MinDist 1:

A lower bound distance for any point in R to p

• if p is inside R, then MINDIST=0

• if p is outside of R, MINDIST is the distance of p to the closest point of R (one point of the perimeter)

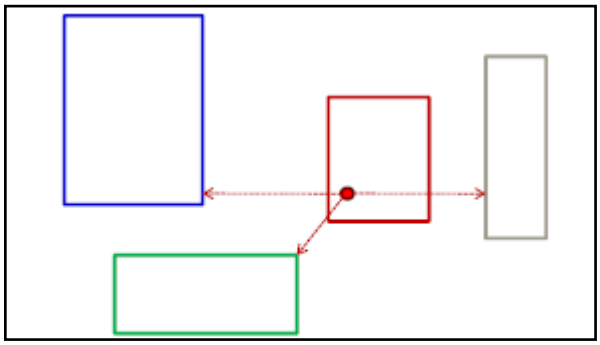


Fig. 4:

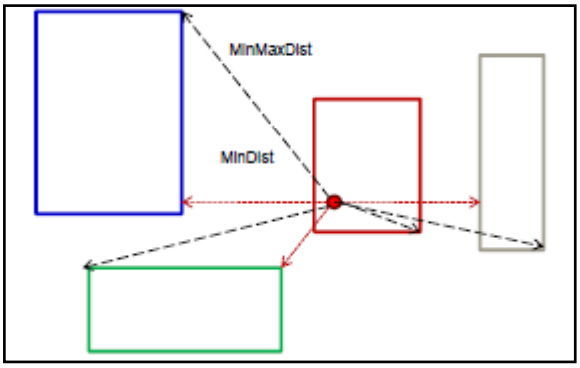
2. MinMaxDist 2

An upper bound of distance from p to R

•for each dimension, find the closest face

•compute the distance to the furthest point on this face

•take the minimum of all these (d) distances



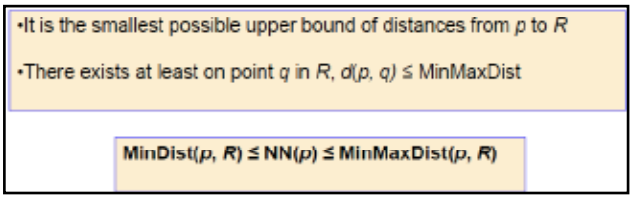


Fig. 5:

Pruning Strategies:
1. (downward pruning) An MBR R is discarded if there exists another R' such that MINDIST(p, R)>MINMAXDIST(p, R')
Example 1:
Downward pruning:
An MBR R is discarded if there exists another R'
s.t. MINDIST(p,R)>MINMAXDIST(p,R')

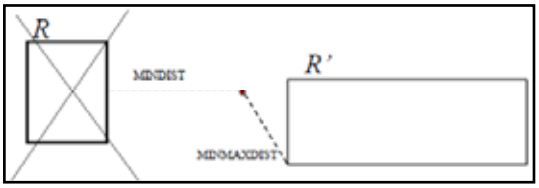


Fig. 6:

2. (downward pruning) An object o is discarded if there exists an R such that d(p, o) > MINMAXDIST(p, R)
Example 2:
Downward pruning:
An object o is discarded if there exists an R
s.t. the Actual-Dist(p,o) > MINMAXDIST(p,R)

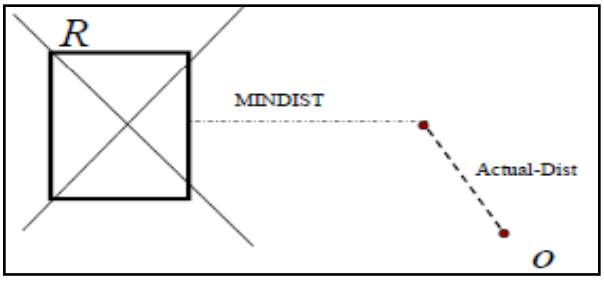


Fig. 7:

3. (upward pruning) An MBR R is discarded if a point q is found such that MINDIST(p, R)> d(p, o)
Example 3
Upward pruning:
An MBR R is discarded if an object o is found
s.t. the MINDIST(p,R) > Actual-Dist(p,o)

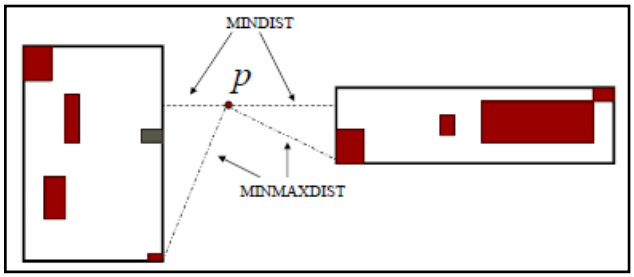


Fig. 8:

For tree traversal:
MINDIST is an optimistic distance, whereas
MINMAXDIST is a pessimistic one.

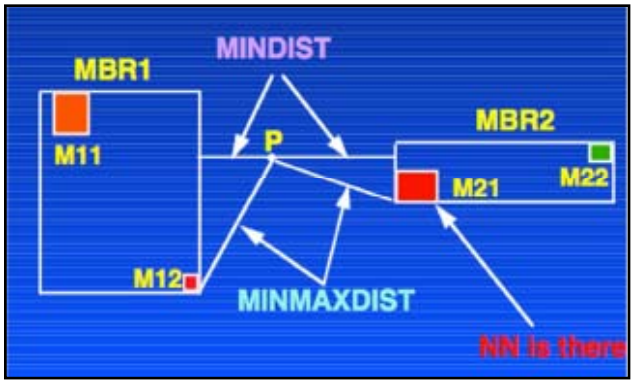


Fig. 9:

IV. (Skyline Queries)&(Branch and Bound Algorithm(BBS)):

A. Skyline Queries

- Find a hotel in gold coast that is:
- cheap and close to the beach!
- Which are the candidate interesting hotels?!

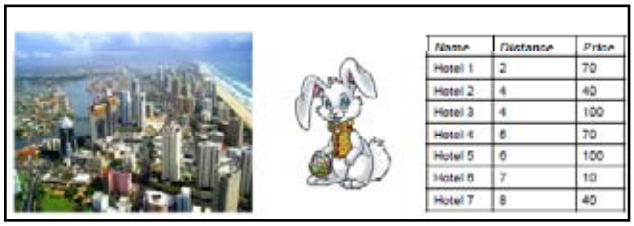


Fig. 10: Finding a Hotel… 26

B. Skyline query: retrieve all points that are not dominated Skyline Query Applications

- Find best NBA players: (#points, #rebounds), or any other subset of the 17 dimensions.
- Find best laptops: (price, screen size, weight, battery, memory size, disk size, CPU speed, warranty…)
- Any table in a RDBMS has a list of records with multiple attributes, so ……

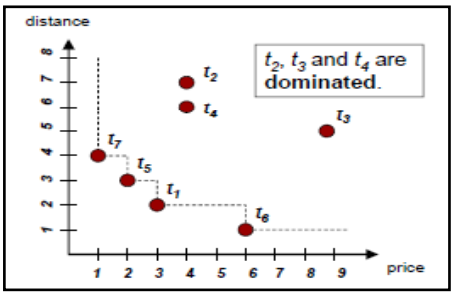


Fig. 11:

C. Skyline Query Processing

The naïve way_O(n2) complexity!

```
SELECT *
FROM Hotels h
WHERE h.city = 'gold coast'
AND NOT EXISTS(
                SELECT *
                FROM HOTELS h1
                WHERE h1.city = 'gold coast' AND
                h1.distance <= h.distance AND
                h1.price <=h.price AND
                (h1.distance < h.distance OR
                h1.price <h.price)
               );
```

Fig. 12:

Skyline Query Processing

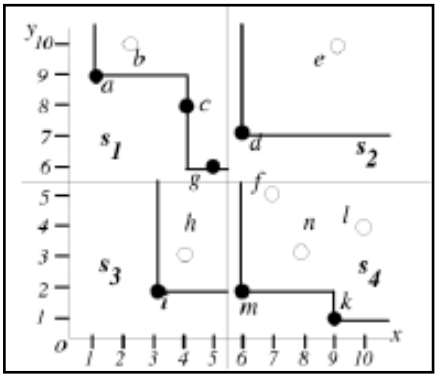


Fig. 13:

- A slightly better way: divide-and-conquer
- The space is divided into several sub-spaces
- Skyline points for each sub-space is computed
- Those skyline points in sub-spaces are merged
- Efficient if the entire dataset fits in memory!

Skyline via NN Query

1. Find nearest neighbor point and add to skyline
2. Divide the space by the nearest neighbor point
   - Prune dominance region
   - Add sub-spaces into a "to-do" list
3. Compute recursively until empty space.

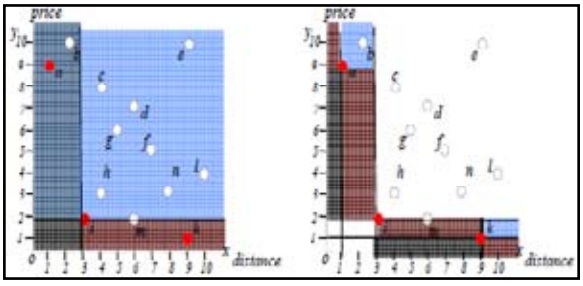


Fig. 14:

V. Branch-and-Bound Approach:

A. Branch & Bound Skyline (BBS)

uses mindist(MBR) = the L1 distance between its lower-left corner and the origin.

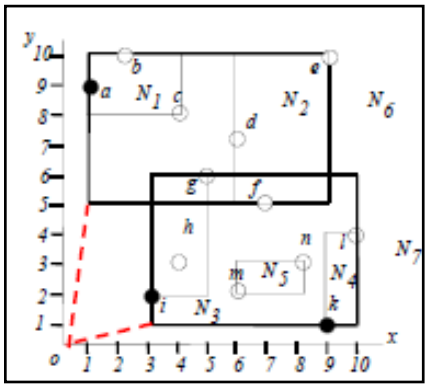


Fig. 15: Branched and Bound Skyline (BBS):

- Assume all points are indexed in an R-tree.
- Top-down Approach
- mindist = the L1 distance between its lowerleft corner and the origin.

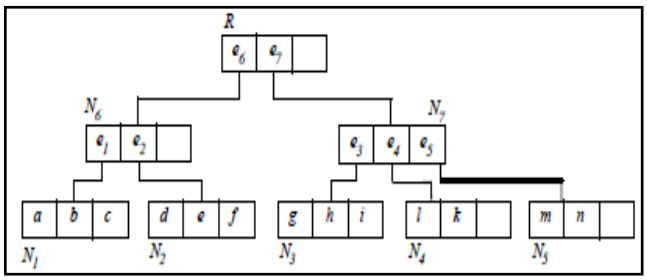


Fig. 16: F(x,y)= x+y

B. BBS Skyline Algorithm

Let heap H = Φ, and skyline result S = Φ.
Insert all entries of the root R in the heap
While heap not empty
Remove top entry e
If e is dominated by some point in S, discard e
  else
    If e is an intermediate entry
    For each child ei of e
If ei is not dominated by some point in S, insert ei into H
Else // e is a data point
Insert ei into S

VI. N^2S^2 Algorithm

A. Problem with BBS Algorithm

When a node is removed from priority queue, its nearest neighbors from each set of query points should be calculated and in based on it decided for that node. Suppose there are separate R tree indexes on each sets of query points. For finding nearest neighbor (from a set of query points) for each child of a node, search should be started from the root of the query points R-tree index, almost the same route that went for finding nearest neighbor of parent node. So fetching a lot of nodes for finding nearest neighbor are repeated. Therefore by storing the nodes of query points[10] R tree that are used to find nearest neighbor of parent node and continue the search from them, we can prevent many of the additional

fetches.

On the other hand for higher level and close to the root nodes of data points R tree, don't need the exact distance from its nearest neighbor but only need an overview of the density of query points around the MBR1 of the node to recognize the priority of it. Therefore the search for finding nearest neighbor need not continue to leaves of query points R tree but nearest MBRs at levels above the leaves can be found and the distances to these MBRs is considered. Therefore many additional fetches of nodes can be avoided.

Before presenting algorithm, appropriate metrics for calculating the distance between two MBRs are needed. There are different metrics for the distance of two MBRs. in this paper two metrics that are introduced in [15] will be used. The first is MinMindist that is the minimum distance between any points in two MBRs and the other is MaxMaxdist that is the maximum distance between any points in two MBRs. in figure 1 this two metrics are shown for two MBRs. this two metrics are used for pruning R tree in N2S2 algorithm Distance between two MBRs

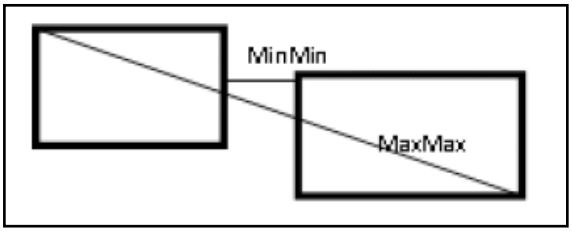


Fig. 17:

N^2S^2 Algorithm is in based on BBS Algorithm in which both problems that discussed in BBS algorithm with spatial nearest neighbor skyline queries are solved. The search is started from the root of the data points R tree. For each node of the data points R tree that is fetched, a typical data structure like the data structure that presented in [16] is created that this node is its owner. If there are m sets of query points, this data structure has m lists. These lists are used for saving nodes of query points R tree used for finding nearest neighbor of the owner. So the search for nearest neighbor doesn't need to start from the root node of the query points R tree but can continue from saving nodes in lists of parent data structure. For each list in data structure of a node minimum of MinMindist and minimum of MaxMaxdist between MBR of owner node and MBRs of the nodes in list are stored.

Procedure of algorithm is that at first a new data structure is created that the root of data points R tree is its owner. In each m lists of this data structure, the root of each query points R tree is inserted. [12], Same as BBS algorithm, a priority queue is used for storing and retrieving these data structures. After creating the root data structure, it is inserted to priority queue. The priority of each data structure determined with sum of minimums of MinMindists for all lists of it. The node that this value is lower for it is more promising. At each stage the data structure of the more promising node is removed from the priority queue and if it isn't dominated by skyline points have been found so far, it will be examined.

If the owner of this data structure is an intermediate node, a new data structure is created for each child of owner. For filling lists of child data structure, the nodes of corresponding lists of parent data structure are examined. If those nodes are leaves, themselves and if they are intermediate nodes, their children are inserted in the list of new data structure. Therefore by going down one level in data points R tree, we are going down one level in query points R trees too.

When a node is inserted in a list of data structure of child node the values of minimum of MinMindist and minimum of MaxMaxdist are updated for that list. When a list was filled with examination of all nodes in corresponding list of parent data structure, all nodes in the list are examined again and each node that isn't promising for finding nearest neighbor, is pruned. In the other words all nodes in list that minimum distance between their MBRs and owner MBR is greater than minimum of MaxMaxdist for that list, are removed from list. Because there is another node in the list that maximum distance between its MBR and owner MBR is less than minimum distance between this node and owner. So it is impossible that this node has nearest neighbor.

If the owner of new data structure is a leaf node, the exact distance of that node and its nearest neighbor from each set of query points should be calculated. So the procedure of filling lists with children of further nodes in lists and pruning none promising nodes should be continued until all nodes in lists are leaves and exact distances to nearest neighbors from each query sets are obtained.

This pseudo code has five functions. The INSERT function inserts a node into one of lists of a DS and updates minminmindist and minmaxmaxdist values corresponding to this list of DS. The PRI function calculates the priority of a DS. This function calculates the sum of all minminmindist and minmaxmaxdist values corresponding to all lists of the DS and returns this value. If the value of this function is less for a DS, it is more promising.

A. NN-search Algorithm

Initialize
The nearest distance as infinite distance
Traverse
The tree depth-first starting from the root.
At each
Index node; sort all MBRs using an ordering
(e.g., MINDIST) and put them in an Active Branch List (ABL).
Apply
Pruning : rules 1 and 2 to ABL (downward Pruning)
If Leaf node, compute actual distances, compare with the best NN so far, update if necessary.
At the return from the recursion, use
Pruning: rule 3 (upward pruning)
When the ABL is empty, the NN search returns.

1. K-NN search

Keep a sorted buffer of at most k current nearest neighbors
Pruning is done using the k-th neighbor in buffer

2. KNN in Road Networks

Distance computing in road networks is costly
Shortest path: Dijkstra algorithm and A* algorithm
Optimization focus is different – not only the number of points to be accessed, but also the network data to be accessed
Euclidean distance for approximation

```
Algorithm N2S2(R-tree R, R-tree R_Q1 ... R-tree R_Qm)

1.  S=Ø//list of DSes that their owner is skyline point
2.  ds_root = New DS(R.root,m)
3.  for each R-tree R_Qi
4.    INSERT(ds_root,R_Qi,root,i)
5.  insert ds_root in the heap with priority PRI(ds_root,m)
6.  while heap not empty
7.      remove top DS ds
8.      if DOMINATE(S_i,ds,m) for some S_i in S discard ds
9.      else // ds is not dominated
10.       if ds.owner is an intermediate entry
11.         for each child ei of ds.owner
12.           ds_ei= New DS(ei,m)
13.           for all i from 1 to m
14.             FILL(ds_ei,ds,i)
15.           if DOMINATE(S_i,ds_ei,m) for some S_i in S discard ds_ei
16.           else // ds_ei is not dominated
17.             insert ds_ei in the heap with priority PRI(ds_ei,m)
18.       else // ds is a data point
19.         remove all S_i from S that DOMINATE(ds,DS_i,m)
20.         insert ds into S
21. end while
22. return S
23. End N²S²
```

Fig. 2: N2S2 Algorithm

```
Algorithm INSERT(DS ds, Node n, Integer i)
//insert node n in i'th queue of data structure ds

1.  ds.list(i).ADD(n)
2.  temp= MinMin(ds.owner.MBR , n.MBR)
3.  if temp < ds.minminmindist(i)
4.     ds.minminmindist(i) = temp
5.  temp= MaxMax(ds.owner.MBR , n.MBR)
6.  if temp < ds.minmaxmaxdist(i)
7.     ds.minmaxmaxdist(i) = temp
8.  End INSERT
```

Figure 3. INSERT function

```
Algorithm PRI(DS ds, Integer m)
//determine priority of the ds

1.  Sum=0;
2.  for all i from 1 to m
3.     sum=sum + minminmindist(i) + minmaxmaxdist(i)
4.  return sum
5.  End PRI
```

Figure 4. PRI function

The DOMINATE function gives two DS as input and examine if first DS dominate another or not. A DS dominate another DS if for all lists in this DS the value of mindist is less than or equal and for at least one list this value is less than the corresponding value of another list.

```
Algorithm DOMINATE(DS p, DS q, Integer m)
//if S_i dominate ds return true

1.  check1=true
2.  check2=false
3.  for all i from 1 to m
4.     if p.minminimndist(i)>q.minminmindist(i)
5.        check1= false
6.     else
7.        if p.minminimndist(i)<q.minminmindist(i)
8.           check2=true
9.  return (check1 AND check2 )
10. End DOMINATE
```

Figure 5. DOMINATE function

```
Algorithm RECONFIG(DS ds, Integer i) //reconfigure queues of DS and prune

1.  l=new list()
2.  temp= ds.list(i).HEAD
3.  while temp is not null
4.     if MinMin(ds.owner.MBR , temp.MBR) < minmaxmaxdist(i)
5.        l.ADD(temp)
6.        temp=temp.NEXT
7.  ds.list(i)=temp
8.  End RECONFIG
```

Figure 6. RECONFIG function

The RECONFIG function gives a DS and examines all nodes in one of its lists and removes all nodes from the list that are not promising for finding nearest neighbor of the owner. Each node that the minimum distance between it and the owner is greater than minmax maxdist of that list is not promising and removes from the list.

VII. Conclusion

This paper a new kind of user preference queries in spatial data bases is introduced that called spatial nearest neighbor skyline queries. This query is very practical in many fields such as service recommendation systems and investment planning. With separating it as a subset of dynamic skyline query, the N2S2 algorithm is presented for solving this kind of query with better performance. The number of IO in this algorithm is much less than BBS algorithm for this kind of queries. In future works we want to extend this kind of query and consider the quality of query points in addition to their distance from data point.

References

[1] Werner Kießling,"Foundations of preferences in database systems", in 28th international conference on Very Large Data Bases, Hong Kong, 2002, pp. 311 – 322

[2] Ihab F Ilyas, George Beskales, Mohamed A Soliman,"A survey of top-k query processing techniques in relational database systems", ACM Computing Surveys , Vol. 40, no. 4, pp. 110 125, October 2008.

[3] S. Borzsony, D. Kossmann, K. Stocker,"The Skyline operator", in 17th International Conference on Data Engineering, Heidelberg, 2001, pp. 421 - 430.

[4] Yufei Tao, Greg Fu, Bernhard Seeger, Dimitris Papadias, "An optimal and progressive algorithm for skyline queries", in international conference on Management of data, New York, 2003, pp. 467-478.

[5] Jongwuk Leea, Gae-won Youa, Seung won Hwang, "Personalized top-k skyline queries in highdimensional space", Information Systems, Vol. 34, No. 1, pp. 45-61, March 2009.

[6] Man Lung Yiu, Xiangyuan Dai, N. Mamoulis, M. Vaitis, "Top-k Spatial Preference Queries", in IEEE 23rd International Conference on Data Engineering, Istanbul, 2007, pp. 1076 - 1085.

[7] M. Yiu, H. Lu, N. Mamoulis, M. Vaitis, "Ranking Spatial Data by Quality Preferences", IEEE Transactions on Knowledge and Data Engineering, Vol. pp. no. 99, pp. 1-1, July 2010.

[8] Yang Du, Donghui Zhang, Tian Xia,"The Optimal-Location Query", in 9th International Symposium Advances in Spatial and Temporal Databases, Angra dos Reis, 2005, pp. 163-180.

[9] Baihua Zheng, K.C.K. Lee, Wang-Chien Lee, "Location-Dependent Skyline Query", in 9th International Conference on Mobile Data Managemen, Beijing, 2008, pp. 148-155.

[10] Ke Deng, Xiaofang Zhou, Heng Tao Shen, "Multi-source skyline query processing in road networks", in IEEE 23rd International Conference on Data Engineering, Istanbul, 2007, pp. 796 - 805.

[11] Man Lung Yiu, Nikos Mamoulis,"Multi dimensional top-k dominating queries", The VLDB Journal, Vol. 18, No. 3, pp. 809 - 835, June 2009.

[12] D Zhang, Y Du, T Xia, Y Tao,"Progressive computation of the min-dist optimal location query", in the 32nd international conference on Very large data bases, Seoul, 2006, pp. 643 - 654.

[13] M. L. Yiu, P. Karras, N. Mamoulis,"Ring-Constrained Join:Deriving Fair Middleman Locations from Pointsets via a Geometric Constraint", in EDBT, 2008.

[14] Y.-Y. Chen, T. Suel, A. Markowetz,"Efficient Query Processing in Geographic Web Search Engines", in SIGMOD, 2006.

[15] E. Dellis, B. Seeger, A. Vlachou,"Nearest Neighbor Search on Vertically Partitioned High-Dimensional Data", in DaWaK, 2005, pp. 243–253.

Suchitra Reyya received her B.Tech degree from Gayathri vidhya parished, JNTUH, Andhra Pradesh, India, in 2008 and the M.Tech degree from pydah engineering college, JNTUK, Andhrapradesh, India, 2011. She was an assistant professor, with Department of Computer science and Technology, Vitam engineering college and pydah engineering college. She was an assistant professor, with Department of Computer science in pydah engineering college. She has 2 years and 6 months of experience in teaching to engineering students. She is heading special interest research groups in Web technology, Oops through Java, Data mining.

M. Ramesh Babu received his B.Tech degree from narasaraopeta engineering college, JNTUK, Andhra Pradesh, India and the pursuing M.Tech degree from pydah college of engineering &Technology, JNTUK, Andhra Pradesh, India.