2006 IEEE International Conference on Systems, Man, and Cybernetics
October 8-11, 2006, Taipei, Taiwan

An Algorithm for Multi-way Distance Join Query

Yin Liang and Hong Zhang

Abstract—**This paper presents K-DJQ algorithms for solving multi-way distance join query, which finds the K n-tuples from n spatial datasets that have the smallest distance value according to query graph. R-tree is used as index structure for each dataset. K-DJQ algorithm is recursive non-incremental approach following depth-first search strategy and synchronously traverses all R-trees, which returns the K n-tuples of the result all together at the end of the algorithm without producing any intermediate result. In addition, distance-based plane-sweep technique is used as optimization techniques for K-DJQ to reduce the total query processing time. Finally, performance and accuracy of K-DJQ algorithm are evaluated in terms of different K value and the number of datasets through experimentation.**

I. Introduction

The spatial database is a database management system with the ability to handle spatial data. In a computer system spatial data are represented by points, line segments and regions, which are referred to as spatial objects. The spatial database can not only store and represent spatial objects, but also manipulate them to process various kinds of spatial queries. Therefore, spatial databases are widely applied to specialized applications such as geographic information systems (GIS), computer aided design (CAD), multimedia information systems (MMIS), data warehouse (DW), satellite image database, location based service (LBS), transportation planning, and resource management etc. Some typical spatial queries used in the real applications are the following: point query, range query, pair wise (multi-way) spatial join query, nearest neighbor query, and K closest pair query.

In recent years, with application of spatial database popularizing, users frequently address the query problem of finding the K (K 1) n-tuples among n spatial datasets that have the smallest distance according to the edges of the query graph (QG). For example, suppose we are given three spatial datasets consisting of the locations of flats, kindergartens, supermarkets represented R_1, R_2, R_3, respectively, connected as in figure 1. User wants to find K different 3-tuples (flat, kindergarten, supermarket)($\{(s_{R1}, s_{R2}, s_{R3})| s_{R1}\in R_1, s_{R2}\in R_2, s_{R3}\in R_3\}$) with the minimum distance between a flat and a kindergarten, this kindergarten and a supermarket and this supermarket and this flat. Many other similar queries can be found in the real life. The query type is called K multi-way distance join query (simply K-MWDJQ).

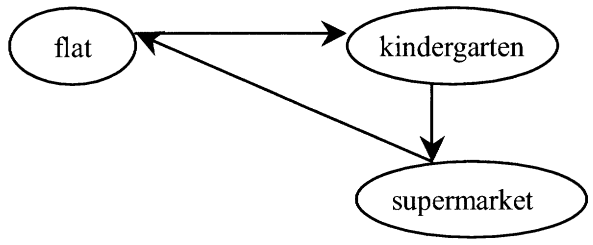

Fig. 1. Connected graph for flat, kindergarten and supermarket

Distance join queries have been recently developed. In References [1] and [2], a recursive and iterative branch and bound algorithms for K-CPQ (K-closest-pair queries) following a non-incremental approach are presented for finding the K closest pairs between two spatial datasets. References [3] and [4] used incremental approach for solving this kind of distance-based query, which returns part of the final result before algorithm is finished. In addition, References [5] and [6] present similarity join and iceberg distance join respectively, which involve two spatial datasets and a given distance threshold. By far algorithms have been presented only consider two spatial datasets, and can't apply to K-MWDJQ.

To process problem of the K-MWDJQ, in this paper, based on distance function and plane-sweep technique, we are going to propose a recursive non-incremental algorithm called K-DJQ for solving multi-way distance join query.

II. Multi-way distance join query using R-trees

In this section, R-trees are described and K Multi-way distance join query (K-MWDJQ) is defined.

A. R-trees

R-trees [7] are hierarchical, height balanced multidimensional data structures, and it is a generalization of B-tree for multidimensional data space. R-tree is consisted of root node, internal node, and leaf node. All of leaves reside on the same level; the root contains at least two entries. Each leaf node contains entries of the form (MBR, Oid), such that Oid is pointer; MBR is the minimum bounding rectangle that encloses the real space object directed by the Oid. Space object may be space data represented by points, line segments, polygons, and volumes. Internal nodes contain entries of the form (MBR, Addr), such that Addr is the pointer directed child node and MBR is minimum bounding rectangle that encloses

This work was not supported by any organization.

Yin Liang and Hong Zhang are with the Department of Computer Science and Technology, China University of Mining and Technology, XuZhou 221008, liangyinq86@xznu.edu.cn

MBRs of all entries in that child node. Fig. 2 is an R-tree example of 2-D and 3-level, which some rectangles on the left

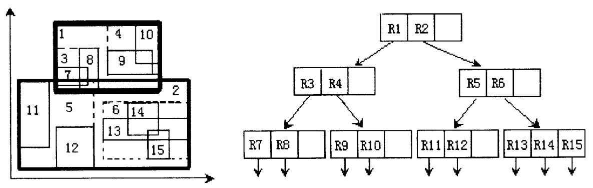

Fig. 2. An example of an R-tree

and the corresponding R-tree on the right.

An R-tree partitions multidimensional space by using hierarchical grouping objects. Each R-tree node occupied subspace is always contained in the subspace of its parent node. An MBR of an R-tree internal node always encloses the MBRs of its descendent R-tree nodes. This property called MBR enclosure property is commonly used spatial join algorithms as well as distance-based query algorithms. Other characteristic of R-tree can be found in [7]. R-tree is one of dynamic space index structures used widely in space database system. Therefore, K-DJQ algorithm use R-tree as space index structure of datasets.

B. *Multi-way Spatial Join Queries*

Multi-way distance join query is one of special multi-way spatial join queries. First of all, we describe multi-way spatial join queries as follows.

Given n spatial datasets R_1, R_2, ⋯, R_n and a query Q, the multi-way spatial join query finds all n-tuples satisfied spatial predicate. Formally, it can be expressed as follows [8]:

$Q(R_1, R_2,\cdots, R_n)=\{(s_{R1}, s_{R2},\cdots, s_{Rn})|\ \forall i,j(s_{Ri}\in R_i \quad s_{Rj}\in R_j$ $s_{Ri}\ Q_{ij}\ s_{Rj}\ \}$, where Q_{ij} represents the spatial predicate that should hold between R_i and Rj. Various spatial conditions (intersect, meet, include, distance within, set containment, etc) can be applied to spatial join predicate. The intersect is default join predicate.

A multi-way spatial join can be modeled by a query graph QG whose nodes represent datasets and edges represent spatial predicates. But, in order to process the space queries such as K-MWDJQ, query graph in K-DJQ algorithms need to be extended as follows.

QG is extended to a directed graph $G_Q=(R, E)$, which consists of a finite nonempty set of nodes $R=\{R_1, R_2,\cdots, R_n\}$ and a finite set of directed edges $E=\{e_{ij}=<R_i, R_j>|\ R_i, R_j \in R$ $R_i\rightarrow R_j\}$; each directed edge e_{ij} connects an ordered pair of nodes ($R_i\rightarrow R_j$), where R_i is called start nodes and R_j is called end nodes of the directed edge.

C. *Distance Function*

In order to determine the minimum distance between n spatial objects that depend on the G_Q, a new metric called D_{object} is defined based on distance function between tow space objects.

Given n non-empty spatial object datasets R_1, R_2,⋯, R_n and G_Q ($R_1 \rightarrow R_2 \rightarrow \cdots \rightarrow R_n$), a distance function D_{object} calculates the sum of distances between s_{R1} and s_{R2}, s_{R2} and s_{R3}, ⋯, and s_{Rn-1} and s_{Rn}, which s_{Ri} is space object from R_i,. D_{object}: $R_1\times R_2\times\cdots\times R_n\rightarrow R^+$('x' is Cartesian product; R^+ is positive real number). For each $t=<s_{R1}, s_{R2}, \cdots, s_{Rn}>\in R_1\times R_2\times\cdots\times R_n$, the function D_{object} is defined as follows:

$$D_{object}(t)= \sum_{e_{ij}\in E_{G_Q}} \text{distance} < s_{Ri},\ \ s_{Rj} > \in R^+ \qquad (1)$$

Which $e_{ij}=<s_{Ri}, s_{Rj}>$ is directed edge in G_Q, distance$<s_{Ri}, s_{Rj}>$ is a distance function between space object s_{Ri} and s_{Rj}. The distance function may be represented by the Euclidean distance [2].

Given $p=(p_1, p_2, \cdots, p_d)$ and $q=(q_1, q_2, \cdots, q_d)$ in the d-dimensional data space, the Euclidean distance is defined as:

$$\text{distance}(p,q) = \sqrt{\sum_{i=1}^{d} |p_i - q_i|^2} \qquad (2)$$

D. *K Multi-way Distance Join Query (K-MWDJQ)*

Given n non-empty spatial datasets R_1, R_2,⋯, R_n which objects of R_i is in $E^{(d)}$ (d-dimensional Euclidean space) and G_Q ($R_1\rightarrow R_2\rightarrow\cdots\rightarrow R_n$), K-MWDJQ is K(1 K $|R_1|\cdot|R_2|\cdots\cdot|R_n|$, where $|R_i|$ is radix of R_i) n-tuples of smallest D_{object}-values. K-MWDJQ is processed in two steps by using multi-way spatial join query and distance function D_{object}. The first step evaluates $|R_1|\cdot|R_2|\cdots\cdot|R_n|$ n-tuples $<s_{R1}, s_{R2}, \cdots, s_{Rn}> \in R_1\times R_2\times\cdots\times R_n$ ('x' is Cartesian product) satisfied query graph G_Q by using multi-way space join. The second step finds K n-tuples of smallest D_{object}-values from candidate n-tuples using distance function D_{object} and sort them.

$K\text{-MWDJQ}(R_1, R_2, \cdots, R_n, G_Q, K)=\{ t_1, t_2, \cdots, t_K \}$

$\forall t \in R_1\times R_2\times\cdots\times R_n - \{ t_1, t_2, \cdots, t_K \}$

$D_{object}(t_1) \quad D_{object}(t_2) \quad \cdots \quad D_{object}(t_{K-1}) \quad D_{object}(t_K)$ $D_{object}(t)$

In general, we consider $\prod_{i=1}^{n} |R_i|$ n-tuples. Thus, K-MWDJQ requires high processing cost due to large volume of spatial data. To reduce the overall processing cost, we develop an algorithm for K-MWDJQ called K-DJQ.

III. Algorithm for Multi-way Distance Join Query

In order to improve performance of K-MWDJQ, distance-based plane-sweep technique is used as optimization techniques in K-DJQ algorithm. Therefore, distance function based on MBR is discussed above all.

A. *Distance Function Based on MBR*

Since R-trees are used widely in space database, we consider that all datasets are indexed by R-tree. In K-DJQ algorithm, MINMINDIST function [2] on pairs of MBRs is defined to determine the minimum distance between two MBRs.

Given two MBRs $M_1(A, B)$ and $M_2(C, D)$ in $E^{(d)}$, where $A=(a_1, a_2, \cdots, a_d)$, $B=(b_1, b_2, \cdots, b_d)$, $C=(c_1, c_2, \cdots, c_d)$, and $D=(d_1, d_2, \cdots, d_d)$, such that a_i b_i, c_i d_i, for 1 i d. A and B, C and D are the endpoints of M_1, M_2 major diagonals, respectively. MINMINDIST(M_1(A, B), M_2(C, D)) is defined as:

MINMINDIST(M_1(A, B), M_2(C, D))

$$=\sqrt{\sum_{i=1}^{d} y_i^2}, \quad y_i = \begin{cases} c_i - b_i, \text{if } c_i > b_i \\ a_i - d_i, \text{if } a_i > d_i \\ 0, \text{otherwise} \end{cases} \qquad (3)$$

In Fig. 3, two MBRs and their MINMINDIST distances are depicted. Dashed lines express the minimum distance of any two points contained in different MBRs.

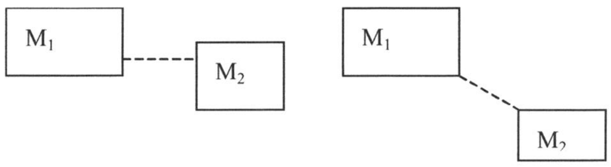

Fig. 3. MINMINDIST between two MBRs in $E^{(2)}$

In the following, we present distance function D_{MBR} between n MBRs that depends on G_Q based on function MINMINDIST,.

Given n R-tree nodes N_{R1}, N_{R2}, $\cdots$, N_{Rn} stored in internal nodes of the R-tree TR_1, TR_2, $\cdots$, TR_n, respectively, which nodes N_{Ri} encloses a set of MBRs {M_{Rij}: 1 j $|N_{Ri}|$}, an n-tuples of MBRs is consisted of n MBRs from N_{R1}, N_{R2}, $\cdots$, N_{Rn}. Let $m=(M_{R1}, M_{R2}, \cdots, M_{Rn})$, where M_{Ri} is an MBR of the node N_{Ri} in TR_i, D_{MBR} (m) is defined as:

$$D_{MBR}(m)=\sum_{e_{ij}\in E_{G_Q}} \text{MINMINDIST}(M_{Ri}, M_{Rj}) \qquad (4)$$

where $e_{ij}=(M_{Ri}, M_{Rj})$ is connected by the directed edge in G_Q.

B. Distance-based Plane-sweep Technique

In general, the plane-sweep technique is commonly used in space join query for computing intersections. The basic idea is to move a sweep-line, e.g. X-axis, from left to right. We apply this technique for restricting all possible combinations of n MBRs from n R-trees. If we do not use this technique, we may consider $\prod_{i=1}^{n} |R_i|$ n-tuples and process them.

The basic idea of distance-based plane-sweep technique is as followings:

1) Let X-axis be sweep-line from left to right.
2) MBRs stored in internal nodes of n R-tree are sorted based on lower left corner value of X-axis in creasing order.
3) Let P be initialized to the MBR with smallest X-value of lower left corner.
4) The P must be paired up with the MBRs stored in other n-1 R-tree from left to right according to sequential query of G_Q. All n-tuples m contained P that satisfies the $D_{MBR}(m)$ z are found, where z is the D_{object}-value of the K-th n-tuple of spatial objects found so far.
5) The P is updated with the MBR of the next smallest value of lower left corner.
6) Repeating 4 and 5.

Based on the previous ideas, we show process of distance-based plane-sweep technique through using an example.

In Fig. 4 illustrates three sets of MBRs stored in three (n=3) R-tree nodes M_P={M_{P1}, M_{P2}, M_{P3}, M_{P4}, M_{P5}}, M_Q={M_{Q1}, M_{Q2}, M_{Q3}, M_{Q4} }, and M_R={M_{R1}, M_{R2}, M_{R3}, M_{R4}, M_{R5}}, respectively. Sequential query of G_Q is $R_P \rightarrow R_Q \rightarrow R_R$. If we do not apply distance-based plane-sweep technique, 5*4*5=100 triplets of MBRs will be processed. Otherwise, this number of possible triplets will be extremely reduced.

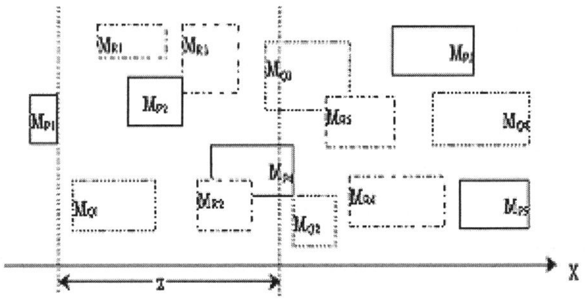

Fig.4. MBRs from three R-tree nodes

First, each MBR from three R-trees are sorted according to X-value of lower left corner of MBR in increasing order. S=(M_{P1}, M_{Q1}, M_{R1}, M_{P2}, M_{R3}, M_{R2}, M_{P4}, M_{Q3}, M_{Q2}, M_{R5}, M_{R4}, M_{P3}, M_{Q4}, M_{P5}), S_P=(M_{P1}, M_{P2}, M_{P4}, M_{P3}, M_{P5}), S_Q=(M_{Q1}, M_{Q3}, M_{Q2}, M_{Q4}), S_R=(M_{R1}, M_{R3}, M_{R2}, M_{R5}, M_{R4}) is obtained, respectively. Then, P is initialized to MBR M_{P1}. P must be paired up with MBRs in S_Q and S_R from left to right that satisfies the D_{MBR} (M_{P1}, M_{Qi}, M_{Rj}) z. Since X-value of M_{Q2}, M_{Q4}, M_{R4}, and M_{R5} are greater than or equal to z, {M_{Q2}, M_{Q4}} and {M_{R4}, M_{R5}} can be discarded. We will obtain a set of triplets {(M_{P1}, M_{Q1}, M_{R1}), (M_{P1}, M_{Q1}, M_{R3}), (M_{P1}, M_{Q1}, M_{R2}), (M_{P1}, M_{Q3}, M_{R1}), (M_{P1}, M_{Q3}, M_{R3}), (M_{P1}, M_{Q3}, M_{R2})} for candidate triplets. In this case, the number of candidate triplets is 6, not 20. 14 triplets are reduced.

D_{MBR}-values of each triplet in candidate are calculated. Since D_{MBR}(M_{P1}, M_{Q3}, M_{R1})$>$z, (M_{P1}, M_{Q3}, M_{R1}) is deleted from candidate. Moreover, since X-value of M_{R3} and M_{R2} is greater than X-value of M_{R1}, D_{MBR}-values of (M_{P1}, M_{Q3}, M_{R3}) and (M_{P1}, M_{Q3}, M_{R2}) need not to be calculated. The triplets of MBRs {(M_{P1}, M_{Q3}, M_{R3}), (M_{P1}, M_{Q3}, M_{R2})} can be directly discarded. Thus, we can further reduce process time. Finally, the number of candidate triplets is 3.

Then, P is updated with the next smallest value of lower left corner M_{Q1} and the previous process is repeated.

C. K-DJQ Algorithm

K-DJQ algorithm is recursive following depth-first search strategy and synchronously traverses all R-trees. It does not produce any intermediate result, and returns the K n-tuples of the result all together at the end of the algorithm. The distance-based plane-sweep technique is used as optimization techniques for improving performance.

First of all, we set a variable z, which is used to record maximal D_{object}-value of K n-tuples. The z is initialized to ∞. Then, we establish two data structures F and E that F stores K n-tuples of minimum D_{object}-value and E store candidate n-tuples of MBRs that satisfy $D_{MBR}(m)$ z. Their initial values are all null.

Query graph G_Q is represented by 2-dimensional array Q[ ][ ]. When an edge exits between dataset R_i and R_j, Q[i][j]=1, otherwise Q[i][j]=0.

```
Initializing: z=∞; F=Φ; E=Φ; Q[ ][ ]; K; n;
    /* n is the number of datasets; K is the number of
       n-tuples */
K-DJQ (R-treeNode N[ ], Query Q[ ][ ])
For i:=1 to n do
  { Read(N[i][ ]); /* N[i][ ] are nodes of R-tree R_i */
    Seq[i]=sort (N[i][ ].MBR); /* sort MBRs of node of
          each R-tree R_i according to lower left corner
          value of each MBR on X-axis in ascending order
          */
  }
Sequence=sort(N[ ][ ].MBR); /* sort MBRs of all nodes
     according to lower left corner value of each MBR on
     X-axis in ascending order. */
For i:=1 to |Sequence| do
  { P:= Sequence[i]; j=1; E1={P};
   while (not(P  Seq[j]) and (j  n)) do
         /* delete MBR outside sliding window z */
         { for each MBR  Seq[j] do
              if MBR.x>P.x+z then
                 Seq[j]= Seq[j]-{MBR};
           j=j+1;
         }
     For k=1 to n-1 do /* finding n-tuples that satisfy query
                           graph*/
       For each Seq do
     if Q[i][k]<>0 then E1=E1x{Seq[k]};
        /* "x" is Cartesian product */
   E=E ∪ E1;
  }
For each t  E do /* delete n-tuples that D_MBR-value greater
                     than z */
  If D_MBR(t)>z then E=E-{t};
For each t  E do /* discuss each n-tuple in E */
  If t are leaf nodes then /*qualifying tuple is at leaf level */
     FingK-n-tuple(F, K, t) /* find K n-tuples which
                                D_object (t) is smallest */
  Else /* qualifying tuple is at intermediate level */
     K-DJQ (N.ref[ ], Q); /* recursive call to lower level */
    Output(F);
FingK-n-tuple(query F[ ], int K, n-tuple t)
  If |F|<K+1 then
     { insert(F, t); /*insert t into F when then number of tuples
                        smaller than K */
       Sort(F); /* sort F in ascending order of D_object(F[i]) */
     }
  Else if D_object (F[K])<D_object (t) then return
       else { delete(F[K]); /* delete a n-tuple which
                       D_object-value is the greatest from F */
              insert(F, t);
              Sort(F);
              z=D_object (F[K]); /* update z value with
                           greatest D_object-value in F */
            }
```

IV. Experimental Results

The evaluation of K-DJQ algorithm was performed based on a variety of experimental tests on uniformly distributed rectangular datasets. R-tree index is used as a spatial access method for the datasets. The experiments were performed on a Pentium Ⅳ 2.1GHz platform on which Windows 2000 professional was running with 256 Mbytes RAM and 80Gbytes of secondary storage. The programs were created using the VC++ 6.0 compiler.

In order to evaluate K-DJQ algorithm, we have used three real spatial datasets A, B, C, representing flat, kindergarten, and supermarket, respectively. The datasets are all point objects. The characteristics of the datasets are summarized in Table 1.

We have measured the performance of our K-DJQ algorithm based on the following two performance metrics: number of disk accesses (simply DA) and response time (simply RT). We have studied two experiments.

Table.1
Cardinalities of real datasets

	A	B	C
The number of real spatial objects	490	383	184

In the first experiment, we study performance of K-DJQ algorithm with increasing K values, varying from 1 to 10000.

Fig. 5 shows performance parameters for K-DJQ over the following configuration: n=3 and G_Q =(A→B→C). In the left figure, we can notice that DA only increase about 15% when K value is increased from 1 to 10000. But, the right figure shows RT is faster for small K values (K 1000), and becomes slower for large K value (K=10000). RT is 4 times from K=10000 to K=1.

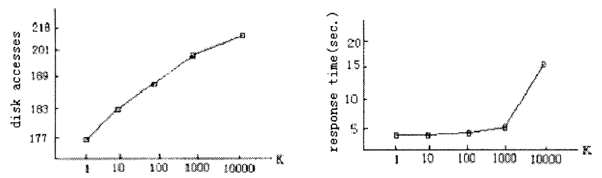

Fig. 5. performance comparison for K-DJQ algorithm varying K

In the second experiment, we study performance and accuracy of K-DJQ algorithm over the following two configuration: n=2, G_Q =(A→B), and K=100; n=3, G_Q =(A→B→C), and K=100. Table 2 shows performance parameters.

Table.2

Comparison of performance for K-DJQ algorithm varying n

	Disk accesses	Response time
n=2	12	0.12
n=3	187	6

Experiment results indicate K-DJQ algorithm can effectively resolve multi-way distance join query. It can be viewed as an extension of K-CPQ [2].

V. Conclusions

Some typical spatial queries for the window query, spatial join query, and nearest neighbor query were already developed [9]–[12]. But the distance spatial join query is immature. For the distance join query, algorithms only for processing K-closest-pair have been developed [1]–[2]. However, algorithm for multi-way distance spatial join query has not been developed yet. In this paper, we present K-DJQ algorithm for multi-way distance join query based on function D_{MBR} between n MBRs in the multidimensional Euclidean space. K-DJQ algorithm is recursive non-incremental approach following depth-first search strategy and synchronously traverses all R-trees, which returns the K n-tuples of the result all together at the end of the algorithm without producing any intermediate result. In addition, distance-based plane-sweep technique is used as optimization techniques for K-DJQ by filtering candidate n-tuples to reduce CPU cost and I/O activity. Experiment results show K-DJQ algorithm can effectively solve distance spatial join query between not only two spatial datasets, but also n (n>2) spatial datasets. It can be viewed as extension of K-closest-pair queries.

References

[1] A. Corral, Y. Manolopoulos, Y. Theodoridis, and M. Vassilakopoulos, "Closest pair queries in spatial databases", in Proceedings of ACM SIGMOD Conference, 2000, pp.189-200.

[2] A. Corral, Y. Manolopoulos, Y. Theodoridis, and M. Vassilakopoulos, "Algorithms for processing K-closet-pair queries in spatial databases", Data & Knowledge Engineering, Vol. 49(1), 2004, pp. 67-104.

[3] G. R. Hjaitason and H. Samet, "Incremrntal distance join algorithms for spatial databases", in Proceedings of ACM SIGMOD Conference, 1998, pp. 237-248.

[4] H. Shin, B. Moon, and S. Lee, "adaptive multi-stage distance join processing", in Proceedings of ACM SIGMOD Conference, 2000, pp. 343-354.

[5] N. Koudas and K.C. Sevcik, "High dimensional similarity joins: Algorithms and performance evaluation", in Proceedings of 14th International Conference on Data Engineering (ICDE'98), 1998, pp. 466-475.

[6] Y. Shou, N. Mamoulis, H. Cao, D. Papadias, and D. W. Cheng, "Evaluation of iceberg distance joins", Proceedings of 8th Symposium on Spatial and Temporal Databases (SSTD'03), 2003, pp. 270-288.

[7] A. Guttman, "R-trees: a dynamic index structure for spatial searching", in Proceedings ACM SIGMOD Conference, 1984, pp. 47– 57.

[8] D. Papadias, N. Mamoulis, Y. Theodoridis, Processing and optimization of multiway spatial join using R-tree, Proc. ACM PODS, 1999, pp. 44–55.

[9] N. Mamoulis, D. Papadias, Multiway spatial joins, ACM TODS 26 (4), 2001, pp. 424–475.

[10] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, A.Y. Wu, An optimal algorithm for approximate nearest neighbor searching .xed dimensions, Journal of the ACM 45(6), 1998, pp. 891– 923.

[11] K.L. Cheung, A.W. Fu, Enhanced nearest neighbour search on the R-tree, ACM SIGMOD Record 27(3), 1998, pp. 16– 21.

[12] M.L. Lo, C.V. Ravishankar, "Spatial hash-joins", in Proceedings ACM SIGMOD Conference, 1996, pp. 247– 258.

Table.1

Cardinalities of real datasets

	A	B	C
The number of real spatial objects	490	383	184

Table.2

Comparison of performance for K-DJQ algorithm varying n.

	Disk accesses	Response time
n=2	12	0.12
n=3	187	6

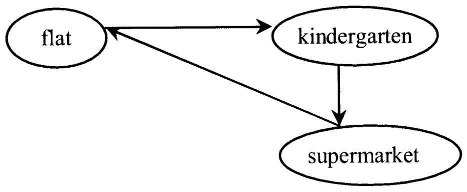

Fig. 1. Connected graph for flat, kindergarten and supermarket

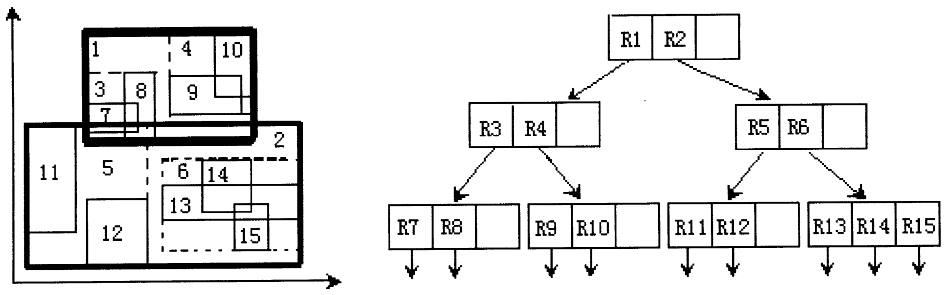

Fig. 2. An example of an R-tree

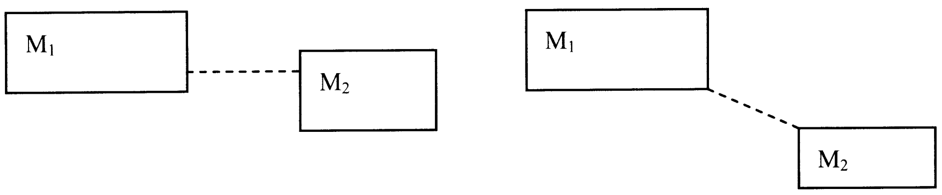

Fig. 3. MINMINDIST between two MBRs in $E^{(2)}$

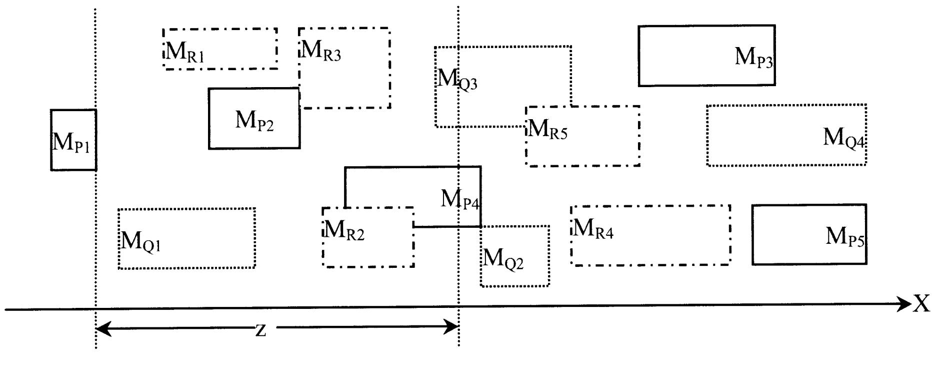

Fig.4. MBRs from three R-tree nodes

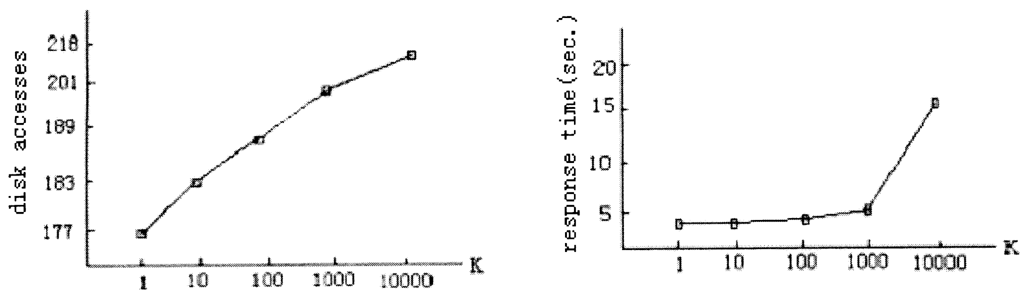

Fig. 5. Performance comparison for K-DJQ algorithm varying K