CSE6488: Mobile Computing Systems

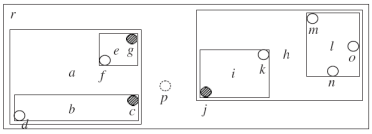
Effective Protocols for kNN search on broadcast multi-dimensional index trees

Sungwon Jung

Dept. of Computer Science and Engineering
Sogang University
Seoul, Korea
Email : jungsung@sogang.ac.kr

*k*NN Search by w-disk method

a



b

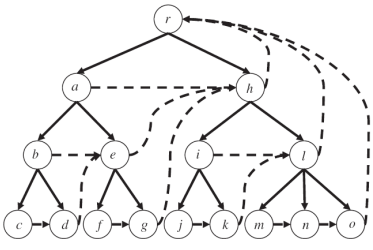


Fig. 1. The rectangles in (a) representing the MBRs of the index nodes in the 16-node *R*-tree in (b). (a) The 3NN (shaded nodes) of the query point *p* (dashed circle) among all the nodes (circles). (b) A 16-node *R*-tree and the dashed lines indicating the *next*-entries (defined in Section 4) of the nodes.

- *mindist(u)*:
  - the minimum distance from the query point ***p*** to node ***u***
- *maxdist(v)*:
  - the maximum distance from the query point ***p*** to node ***v***
- C-List, R-List :
  - The nodes in both lists are maintained in non-decreasing order by ***maxdist*** values
- Structure of index node

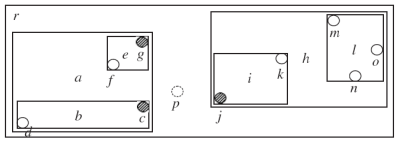
Node ID	Child's ID	Child's MBR	Child's *l*-entry	Next-entry
h	*l*	MBR	2	*r*
	i	MBR	3	

- Structure of leaf node
  - Only the node's ID and the data content

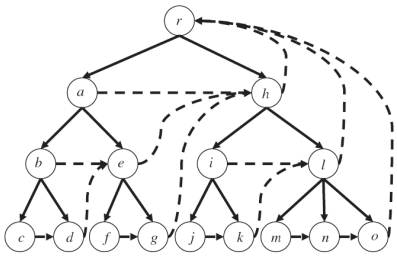
2

Broadcast Index Trees

a



b



- *BFS vs DFS*
  - BFS can achieve a better tuning time for kNN search than DFS, but they result in a large memory usage at the mobile clients.
- *p*DFS
  - Organize organizes the broadcast simply by the depth-first order
- *w*DFS
  - rearrange an R-tree by the subtree sizes in a non-increasing order and then place the nodes in the broadcast

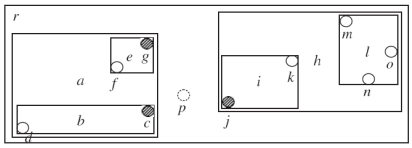
*p*DFS | r | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |

*w*DFS | r | h | l | m | n | o | i | j | k | a | b | c | d | e | f | g |

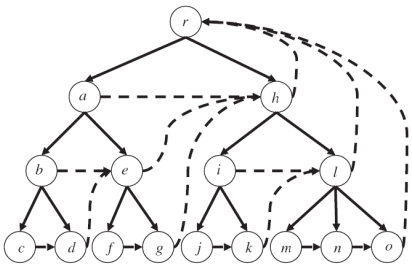
3

Mobile Client algorithm: w-disk

a



b



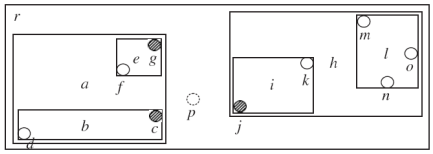
Algorithm k**NN-Explore**(v)
/* u is the Pnode used in the previous explored node */
(1) **if** node v is a leaf node **then**
  (1.1) INSERT v into *R-List*
  else /* node v is an index node */
  (1.2) **for** each child v' of v **do**
    if $mindist(v') \leq maxdist(u)$ **then**
      INSERT v' into *C-List*
  (1.3) u'=FINDPNODE();
  (1.4) DELETE the nodes of which
    $mindist > maxdist(u')$ from the both lists
(2) let w be the node closest to the currently explored node in *C-List*
(3) kNN-Explore(w)
End Algorithm k**NN-Explore**

Algorithm FINDPNODE()
(1) Scan the nodes in *C-List* and *R-List* by maxdist in a nondecreasing order using the way like the merge sort until the first node u whose $S_u \geq k$;
(2) **return** u
End Algorithm FINDPNODE

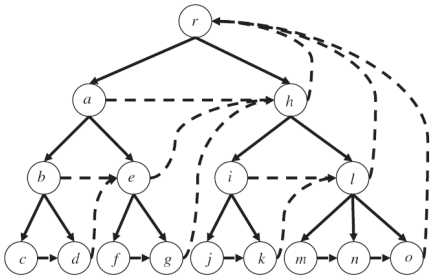
4

Example of w-disk algorithm: Finding 3NN from query point P

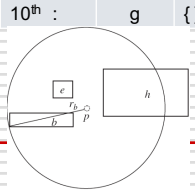
a

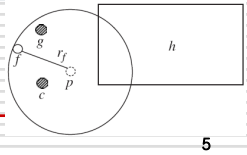


b



Currently explored node w	Pnode	C-List	R-List	u'
1st : r	Pseudo-node	{a, h}	{ }	a
2nd : a	a	{e, b, h}	{ }	b
3rd : b	b	{c, e, d, h}	{ }	e
4th : c	e	{ e, h}	{c}	e
5th : e	e	{g, f, h}	{c}	f
6th : f	f	{g, h}	{c, f}	f
7th : g	f	{h}	{c, g, f}	f
8th : h	f	{ i }	{c, g, f}	f
9th : i	f	{ j }	{c, g, f}	f
10th : j	f	{ }	{c, j, g, f}	g
10th :	g	{ }	{c, j, g}	





pDFS | r | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |

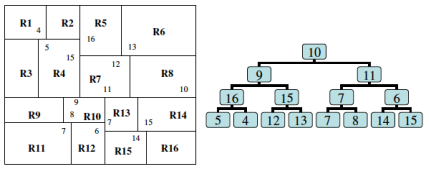
5

Shortest Path Computation on Air Indexes

6

ELLIPTIC BOUNDARY(EB) Method

- Index of EB has two components
  - The first component defines partitions and provides a mapping of nodes into regions
    - Regular Grid
    - Kd-tree



    - <10, 9, 11, 16, 15, 7, 6, 5, 4, 12, 13, 7, 8, 14, 15>
  - The second component specifies min/max distance between regions Ri and Rj: n x n Array where n is total num. of partitions
    - The min and max distance from any border node in Ri and to any border node in Rj

7

ELLIPTIC BOUNDARY(EB) Method

- The network information is followed by EB Index
  - The adjacency list of nodes belong to same regions are placed continuously in the broadcast cycle
    - Region ordering abides by the assigned region numbers
    - The index is appended with pointers to the data of each region
- (1,m) index allocation scheme is used
  - The index copies between regions as opposed to fixed
    - The adjacent data for the same regions are not cut in by index packet

8

ELLIPTIC BOUNDARY(EB) Method

- Example of EB method
  - Use Regular Grid

R1	R2	R3
R4	R5	R6

	R1		R2		R3		R4		R5		R6	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
R1			1	5	6	8	1	4	3	7	8	9
R2	1	5			1	3	2	5	1	2	2	4
R3	4	6	1	3			5	8	2	4	1	4
R4	1	3	2	4	5	8			2	3	4	7
R5	3	6	1	3	2	4	1	3			1	2
R6	7	9	2	4	1	2	5	6	1	3		

  - Use Array Index component to derive an UB for the shortest distance from the source node *s* to the destination node *t*
    - *mindist(R_s, R) + mindist(R, R_t) <= UB*
    - *The node s in R1 and the node t in R5*
      - *UB = 7*
      - *R is selected from R2, R3, R4, and R6*

9

NEXT REGION(NR) Method

- Basic Idea of NR method
  - The source in R1 and the dest in R16

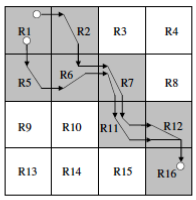


Figure 4: Shortest paths from R_1 to R_{16}

10

NEXT REGION(NR) Method

- Example of NR method

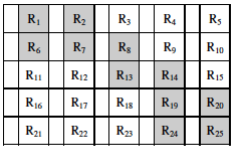
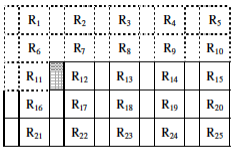


Figure 5: Needed regions



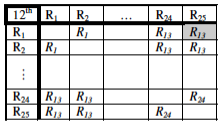
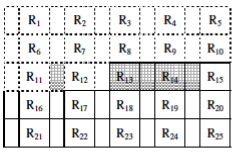


Figure 6: NR algorithm; receiving A^{12}



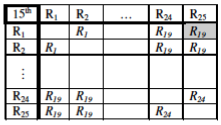


Figure 7: NR algorithm; receiving A^{15}