Revisiting 2DR-tree Insertion

Marc Moreau
Department of Mathematics
and Computer Science
University of Lethbridge
4401 University Drive West
Lethbridge, Alberta T1K 3M4
marc.moreau@uleth.ca

Wendy Osborn
Department of Mathematics
and Computer Science
University of Lethbridge
4401 University Drive West
Lethbridge, Alberta T1K 3M4
wendy.osborn@uleth.ca

ABSTRACT

We take another look at the 2DR-tree. In particular, we revisit its node structure, validity rules and the insertion strategy. The 2DR-tree uses 2D nodes so that the relationships between all objects can be maintained. The existing structure has many advantages. However, two limitations include a high tree height and a low space utilization of its nodes. We propose changes to the 2DR-tree structure, validity rules and insertion strategy. Preliminary results show significant improvements in height and space utilization over the existing 2DR-tree.

Categories and Subject Descriptors

H.2.2 [**Database Management**]: Physical Design—*access methods*

General Terms

Algorithms, Design

Keywords

spatial access methods, insertion, performance

1. INTRODUCTION

Many applications exist today that store and manipulate spatial data. A spatial database [10] contains a large collection of objects that are located in multidimensional space. An important issue in spatial data management is to efficiently retrieve objects based on their location by using spatial access methods.

An approximation method is a spatial access method that maintains a hierarchy of approximations of both objects and the space occupied by subsets of objects. Approximations are usually represented using a minimum bounding rectangle (MBR). Many approximation strategies have been proposed, including [5, 2, 3, 1, 6, 9, 7, 8]. A comprehensive survey is provided in [4]. Most proposed strategies do not preserve all spatial relationships between objects because the data, which is represented in n-dimensional space, is forced into a 1-dimensional ordering. A recently proposed spatial access method, the 2DR-tree [8], uses two-dimensional nodes, which allows data to be ordered and spatial relationships to be preserved. Limitations of the 2DR-tree include a tree height that is higher than necessary, and a low average space utilization. It is suspected that the primary cause of these limitations lies in the node validity rules and the insertion algorithm.

We take another look at the 2DR-tree. In particular, we revisit the 2DR-tree node structure, validity rules and insertion algorithm. We propose modifications that show promise in reducing tree height and increasing space utilization

C3S2E-08 2008 May 12-13, Montreal, QC, Canada

2. THE INDEX

2.1 Preliminaries

As with the 2DR-tree, we determine the relative placement of objects in the tree by using the centroids of their MBRs. However, we redefine the relative orientation of two centroids, as shown in Table 1. The 4 orientations (NE, SE, SW, NW) are redefined to include centroids that fall on the axes (E, S, W, N, respectively). Also, an equals (EQ) orientation is added, to handle two centroids that overlap.

A node contains 5 locations. Each location can contain a pointer to either another node or an object. A node also stores a **node MBR**, which is an MBR that contains all objects in its subtrees. A node must have at least two locations that reference either an object or a subtree, unless it is the root.

A node is classified as either 'NORMAL' or 'CENTER'. In a NORMAL node, the locations are organized based on the orientations defined above (see figure 1). A NORMAL

$A_x = B_x$	$A_x > B_x$	$A_y = B_y$	$A_y > B_y$	Placement of A
0	0	0	0	SW
0	0	1	0	SW
0	0	0	1	NW
1	0	0	1	NW
0	1	0	0	SE
1	0	0	0	SE
0	1	0	1	NE
0	1	1	0	NE
1	0	1	0	EQ

Table 1: Relative orientation of A with respect to B

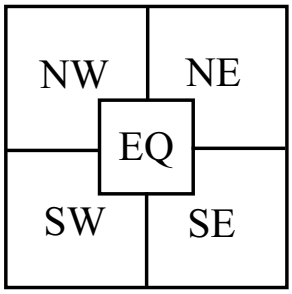

Figure 1: Tree Node

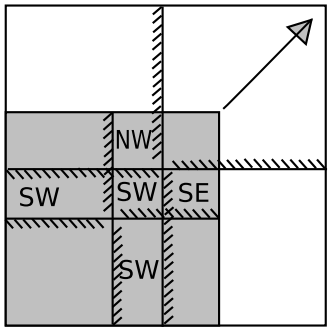

Figure 2: NorthEast Node MBR expansion

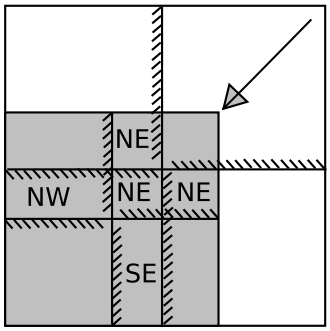

Figure 3: SouthWest Node MBR contraction

node is valid when:

1. The node MBR encloses all the MBRs in the objects or subtrees that the node references, and
2. All objects or subtrees pointed to by a location are in the proper quadrant relative to the node centroid.

In a CENTER node, the locations are organized linearly. A CENTER node only references objects whose centroids are the same as the centroid of the node MBR.

3. THE INSERTION STRATEGY

The new insertion strategy works as follows. Beginning at the root node, the node MBR is adjusted to include the new object. Then, the appropriate location, relative to the centroid of the node MBR, is identified for inserting a reference to the new object. If the location is empty, the reference to the object is inserted. Otherwise, the subtree is traversed in the same manner, until a leaf node is reached. If the object cannot be inserted in the proper location of the leaf node, then a new leaf node is created.

Next, node validity is maintained by reinserting objects that have changed orientation relative to the centroid of the node MBR during the insertion process. When inserting or deleting an object from a node, one of three things will happen to the node MBR:

1. The centroid of the node will not change and therefore all objects remain in their proper orientation,
2. The centroid of the node moves and the area of the node MBR increases,
3. The centroid of the node moves and the area of the node MBR decreases.

   In the latter two cases, objects no longer in their proper relative node location must be located and moved.

Figure 2 shows a NorthEast expansion of the node MBR from the original area (shaded) to the new area. Both MBRs are split into four quadrants using hashed lines. The direction of the hash indicates the quadrant in which object on that line are included. The regions that are labeled represent the destination location for the objects found within that region. The 'EQ' location has been omited for clarity. Figure 3 shows the SouthWest contration in a similar manner. All other expansions and contractions work in a similar manner.

4. PROPERTIES

In our preliminary investigations, we have identified some interesting properties of the modified index and insertion algorithm:

- Any point, and therefore any MBR centroid, has only one possible location in the tree. This leads to a tree which is independent of the insertion order of all objects.
- The centroid of a node will have the same orientation in its parent as does all the objects inclosed by the node MBR.
- The MBR of a location will have less then half of its area outside its quadrant, except for the 'EQ' quadrant. This may lead to a minimizing in overlap.
- With datasets consisting of only points, the overlap of any two MBRs at any level of the tree is zero. There is no area that has the potential to be covered twice.

5. EVALUATION

The new insertion algorithm was compared with the original 2DR-tree insertion algorithm. We used 4 collection of road and rail data from Canada (121416 and 35074 line segments, respectively) and California (21831 and 11381 line segments, respectively), obtained from [11]. For each dataset, we created 1001 trees for each of the old and new strategies. For 1000 trees, each were built using randomly-ordered data. For 1 tree, it was built using a sorted dataset, which was determined to be a worst-case scenario. For each tree, we calculated the number of nodes, height, average space utilization in each node, total coverage of all MBRs, total overcoverage (i.e. whitespace) of all MBRs, and the total overlap between all MBRs.

Table 2 displays the results over the 1000 trees built with randomly-ordered data. Note that for the old strategy, the values for all parameters are averages over all 1000 runs, since all values vary for each tree. For the new strategy, the values are identical for all 1000 trees. As mentioned earlier, the new insertion strategy is independent of the order in which the objects are inserted. The only variation is in how many objects are moved in order to maintain node validity. Also note that tree height for the new insertion strategy is a fractional number. This number represents the average path length from the root to a leaf node.

dataset	algorithm	#nodes	height	sp.util (%)	coverage	overcoverage	overlap
CDrdline	old	224080	29.19	38.5	166097.00	132272.00	129407.00
	new	77000	9.55	51.4	9523.12	2975.79	110.28
CDrrline	old	64651	24.84	38.5	57582.10	14571.20	13027.90
	new	23107	8.68	50.2	4334.23	1574.26	30.90
CArdline	old	40266	23.44	38.5	4163.12	1700.43	1607.44
	new	14118	8.02	51.0	354.68	100.23	7.24
CArrline	old	20905	22.57	38.5	3058.00	423.33	330.22
	new	7760	8.34	49.2	249.00	94.59	1.48

Table 2: Comparison of Insertion Algorithms - Average Cases

dataset	algorithm	#nodes	height	sp.util (%)	coverage	overcoverage	overlap
CDrdline	old	355697	88	33.5	490255.06	79004.80	76139.29
	new	77000	9.55	51.4	9523.12	2975.79	110.28
CDrrline	old	113611	88	32.8	209655.42	10284.43	8741.10
	new	23107	8.68	50.2	4334.23	1574.26	30.90
CArdline	old	64471	38	33.3	5846.69	627.37	534.36
	new	14118	8.02	51.0	354.68	100.23	7.24
CArrline	old	42872	60	31.0	8508.06	610.36	517.26
	new	7760	8.34	49.2	249.00	94.59	1.48

Table 3: Comparison of Insertion Algorithms - Worst Cases

Results show that the new strategy achieves a significant improvement over the old strategy in all aspects. In particular, there is an 11-13% increase in space utilization, with approximately 50% space utilization overall for the new strategy. In addition, there is a 63-66% lower height when the new strategy is used insertion. There are also significant decreases in coverage, overcoverage, overlap and the number of nodes. It must also be noted that the maximum height from the root to a leaf node for the new insertion strategy is 13 (for both California datasets, and the Canada railroad dataset), and 15 (for the Canada road dataset).

Table 3 presents the results for the build using sorted data. The 2DR-tree appears to perform very poorly when indexing ordered data with the original insertion algorithm. However, the new algorithm still performs well, and significantly better than the original insertion algorithm.

6. CONCLUSION AND FUTURE WORK

We propose a new node structure, validity rules and insertion algorithm for the 2DR-tree. Preliminary results show that the new strategy is superior to the old strategy. Currently, the new algorithm is limited to two dimentions.

Future work includes the following. The first is to evaluate the new insertion strategy and node structure versus other existing strategies (such as the R-tree [5]). The second is to extend the node structure to multiple dimensions. The third is to increase the number of locations in a 2-dimensional node. The fourth is to create a bottom-up tree-construction strategy to handle multiple insertions at once. The fifth is to improve the CENTER nodes. The final improvement is a paging strategy that groups nodes based on a high probability that they are retrieved for the same queries.

7. REFERENCES

[1] C. Aggarwal, J. Wolf, P. Yu, and M. Epelman. The S-tree: an efficient index for multidimensional objects. In *Advances in Spatial Databases: 5th International Symposium on Spatial Databases SSD '97*, number 1262 in Lecture Notes in Computer Science, pages 350–73. Springer-Verlag, 1997.

[2] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 322–31, 1990.

[3] S. Berchtold, D. Keim, and H.-P. Kriegel. The X-tree: An index structure for high-dimensional data. In *Proceedings of the 22nd International Conference on Very Large Data Bases*, pages 28–39, 1996.

[4] V. Gaede and O. Günther. Multidimensional access methods. *ACM Computing Surveys*, 30:170–231, 1998.

[5] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 47–57, 1984.

[6] I. Kamel and C. Faloutsos. Hilbert R-tree: an improved r-tree using fractals. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 500–9, 1994.

[7] N. Koudas. Indexing support for spatial joins. *Data and Knowledge Engineering*, 34:99–124, 2000.

[8] W. Osborn and K. Barker. An insertion strategy for a two-dimensional spatial access method. In *Proceedings of the 9th International Conference on Enterprise Information Systems*, 2007.

[9] T. Sellis, N. Roussopoulos, and C. Faloutsos. The R+-tree: a dynamic index for multi-dimensional objects. In *Proceedings of the 13th International Conference on Very Large Data Bases*, 1987.

[10] S. Shekhar and S. Chawla. *Spatial databases: a tour.* Prentice Hall, 2003.

[11] Y. Theodoridis. R-tree Portal, http://www.rtreeportal.org/. (visited March 2008), 2005.