# The mqr-tree: improving upon a 2-dimensional spatial access method

Marc Moreau, Wendy Osborn[*] and Brad Anderson
Department of Mathematics and Computer Science
University of Lethbridge
4401 University Drive West
Lethbridge, Alberta
T1K 3M4 Canada
{marc.moreau, wendy.osborn}@uleth.ca

## Abstract

*We propose the mqr-tree, a two-dimensional spatial access method that improves upon the 2DR-tree. The 2DR-tree uses two-dimensional nodes so that the relationships between all objects are maintained. The existing structure of the 2DR-tree has many advantages. However, limitations include higher tree height, overcoverage and overlap than is necessary. The mqr-tree improves utilizes a different node organization, set of validity rules and insertion strategy. A comparison versus the R-tree shows significant improvements in overlap and overcoverage, with comparable height and space utilization. In addition, zero overlap is achieved when the mqr-tree is used to index point data.*

## 1. Introduction

Many applications exist today that store and manipulate spatial data. A spatial database [16] contains a large collection of objects that are located in multidimensional space. For example, the Geological Survey of Canada maintains a repository of spatial data for many geoscience applications [5], while the Protein Data Bank [13] contains many three-dimensional protein structures. 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. However, 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. This leads to inefficient searching, both within a node and the structure as a whole, because the only option is a linear search of a node in its entirety.

A recently proposed spatial access method, the 2DR-tree [11, 12], 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. These limitations also lead to high coverage, overcoverage and overlap of MBRs within the 2DR-tree. It is suspected that the primary cause of these limitations lies in the node validity rules and the insertion algorithm.

Therefore, we take another look at the 2DR-tree. We propose the mqr-tree, which improves upon the node organization, validity rules and insertion algorithm of the 2DR-tree. We evaluate the mqr-tree against the R-tree [6]. We show that the mqr-tree achieves significant improvements in overlap and overcoverage, and in many cases comparable tree height over the R-tree. With these improvements accomplished, different searching strategies can be explored that can perform a partial search of nodes.

This paper proceeds as follows. Section 2 presents related work in the area of spatial access methods and their limitations. Section 3 presents the mqr-tree, in particular its node organization and insertion strategy. Section 4 presents special properties of the mqr-tree. Section 5 presents the results of our experimental evaluation versus a benchmark strategy. Finally, the paper concludes and gives research directions in Section 6.

## 2. Related Work

Many approaches for indexing objects based on location are proposed in the literature (see [4, 14, 16] for surveys). These approaches are classified into three categories [4]:

[*] Author to whom all correspondence should be sent. Work was funded by the University of Lethbridge Research Fund.

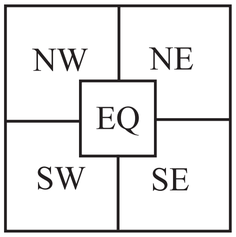

**Figure 1. Node**

| $A_x = B_x$ | $A_x > B_x$ | $A_y = B_y$ | $A_y > B_y$ | Placement |
|---|---|---|---|---|
| 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 |

**Figure 2. Relative orientation of A with respect to B**

main memory methods, point access methods and spatial access methods (spatial access methods). Many important strategies are proposed in all categories. We focus on spatial access methods, since our work is in this category.

Spatial access methods provide uniform access to both point and object data. Also, they remain height-balanced in the presence of a dynamic object set. Many spatial access methods are proposed in the literature [6, 1, 2, 7, 15, 10, 8]. They can be classified [4] into approximation, clipping, and mapping methods.

Approximation methods store a hierarchy of approximations of both objects and the space occupied by subsets of objects. Since the space is not partitioned, approximations can overlap. Many approximation methods are proposed, including the R-tree [6], the $R^*$-tree [1], the X-tree [2] and the 2DR-tree [11, 12]. Clipping methods, such as the $R^+$-tree [15], partition an object into parts so that overlap is avoided. Mapping methods map objects in n-dimensional space into a one-dimensional order. The objects are then stored and retrieved using an access method such as a $B^+$-tree [3]. Approaches that use mapping include Z-ordering [10], the Hilbert R-tree [7], and the Filter tree [8].

No n-dimensional to one-dimensional mapping of spatial data exists that preserves all spatial relationships between objects [4]. A limitation to most hierarchical spatial access methods is their one-dimensional structure. This forces objects in n-dimensional space into a one-dimensional ordering, which results in the loss of spatial relationships. This leads to inefficient searching, both within a node and the structure as a whole, because the only option is a linear search of a node in its entirety. Mapping methods do provide a one-dimensional ordering of objects, but they cannot maintain all spatial relationships.

The 2DR-tree [11, 12], a recently proposed spatial access method, attempts to overcome these limitations by proposing a structure that fits the data as given, instead of forcing n-dimensional data to fit a one-dimensional structure. Two-dimensional nodes are used, which allows spatial relationships between objects to be preserved. Node validity rules are defined and applied during object insertion to ensure that spatial relationships are maintained.

The 2DR-tree has many advantages, including different searching strategies and traversal of space in different dimensions. Limitations of the 2DR-tree include a tree height that is higher than necessary, and a low average space utilization. These limitations also lead to high coverage, overcoverage and overlap of MBRs within the 2DR-tree. We have identified that an improved strategy for the placement of objects within a node, and a corresponding insertion algorithm, will alleviate many of the limitations of the 2DR-tree while preserving the two-dimensional structures and the ability to maintain spatial relationships among objects in each node.

## 3. The mqr-tree

In this section, we present the new approach to organizing objects within a node, updated node validity rules, and the new insertion strategy.

### 3.1. Node Organization and Validity

As with the 2DR-tree, we determine the relative placement of objects in the tree by using the centroids of their MBRs. The origin of a 2DR-tree node is the bottom left-hand node entry. In addition, the 2DR-tree is built from the bottom-up, and therefore all leaf nodes are on the same level. For the mqr-tree, we redefine the origin of a node, and the relative orientation of two centroids. In addition, we relax the requirement that the tree must be height-balanced.

Figure 1 depicts the new node layout. A node contains 5 locations. Each location contains a pointer to either another node or an object. A node must have at least two locations that reference either an object or a subtree. The origin of the node is the now the centre of the node. The centre is defined by the centroid of the **node MBR**, which is an MBR

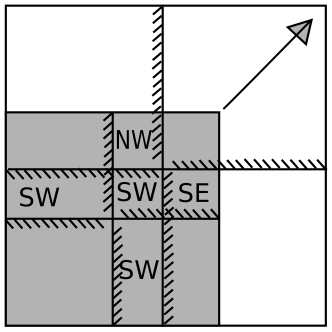

**Figure 3. NorthEast Node MBR expansion**

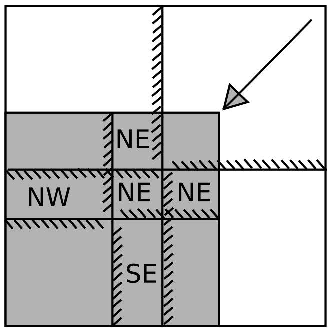

**Figure 4. SouthWest Node MBR contraction**

that contains all objects in the node, and any subtrees of the node. As objects are added to and removed from the node, the centre of the node will change.

Figure 2 depicts the redefined orientations, where $A$ refers to the centroid of a new object, and $B$ refers to the centre of the node. The 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 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 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.

### 3.2. 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 either: 1) an appropriate location is found that is empty and the object reference can be inserted, or 2) a leaf node is reached, and no proper location is available for the new object reference. If the object reference cannot be inserted in the proper location of the leaf node, then a new leaf node is created.

Next, node validity is maintained by removing and 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, some areas within the node MBR may have shifted to another quadrant. In addition, some objects may no longer be in their proper relative node location. Any objects in this situation must be located and moved. Objects are moved by re-inserting them so that they are placed in a proper relative location.

Figure 3 shows a NorthEast expansion of the node MBR from the original area (shaded) to the new area. The MBR is split into four quadrants using hashed lines. The direction of the hash indicates the quadrant in which the object on that line is included. The regions that are labeled represent the destination location for the objects found within that region.The 'EQ' location has been omitted for clarity. Notice that after the MBR is expanded, partial regions that once belonged to the NorthWest, NorthEast and SouthEast quadrants now belong to the region that makes up the SouthWest quadrant. Any objects within these areas are no longer properly located relative to the new node centre, and would have to be relocated. Figure 4 shows the SouthWest contraction in a similar manner. All other expansions and contractions work in a similar manner.

## 4. Properties

In our investigations, we discovered some interesting properties of the mqr-tree index and insertion algorithm:

- Any point, and therefore any MBR centroid, has only one possible location in the tree. This leads to a tree that is independent of the insertion order of all objects.
- The centroid of a node will have the same orientation in its parent as all objects enclosed 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

Now, we present the results of our empirical evaluation of the mqr-tree. Initially, we compared the mqr-tree with the 2DR-tree [9]. We found that the mqr-tree achieved significant improvements in height, space utilization, coverage, overcoverage and overlap. Here, we compare the performance of the mqr-tree insertion algorithm with the R-tree insertion algorithm [6], which is considered one of the benchmark strategies for spatial indexing.

### 5.1. Data Sets

We use both synthetic and real datasets for our comparison. Our synthetic data consist of randomly generated collections of objects that vary in type (i.e. squares, points), size (i.e. 500-10,000 objects) and distribution (i.e. uniform, exponential). Our real data consists of sets of road and railroad data that vary in size from 11,000 to 122,000 line segments. The line data is part of the Digital Chart of the World and obtained from [17].

### 5.2. Tests and Evaluation Criteria

For each dataset, we created 1000 trees using each algorithm. Each tree was built using randomly-ordered data. 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 was calculated for each tree. We discuss 4 of those performance factors below:

- Average space utilization - the average number of MBRs per node. Ideally, the higher the number of MBRs per node, the lower the number of nodes and the lower the tree height. Both the MBRs referencing objects and those encompassing other MBRs are included in this calculation.
- Overcoverage - the amount of whitespace (i.e. area with no objects) that is contained in the MBRs of a spatial index. Ideally, the amount of overcoverage should be zero or very low. A higher overcoverage will results in searches along paths that will lead to no objects.
- Overlap - the amount of space covered by two or more MBRs. Ideally, overlap should If significant overlap exists, then a search may proceed down multiple paths that are covering the same area. This in turn will lead to unnecessary searching, since the extra paths usually do not contain objects required for a query.
- Height - the number of nodes from the root to the leaf node on the longest path of the index. Ideally, the shorter the path, the shorter the search from root to leaf. However, shorter paths may also lead to more overcoverage and overlap, so slightly longer paths may be beneficial overall. For the R-tree, all paths are the same length because the tree is height-balanced. Because the mqr-tree is not height-balanced, in addition to the longest path length, the average path length (i.e. average height) is also recorded.

### 5.3. Results on Synthetic Data

Table 1 displays the results for the data sets consisting of uniformly distributed squares. Note that for the R-tree, the values for all parameters are averaged over all 1000 runs, since these values vary for each tree. For the mqr-tree, 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 the two sets of values for height for the mqr-tree. The first value represents the maximum (i.e. worst-case) height, while the second value in parentheses is the average height (i.e. average path length).

Results show that the mqr-tree achieves a significant improvement over the R-tree in many aspects. In particular, there is a 22-25% decrease in coverage, a 10-36% decrease in overcoverage, and a 17-46% decrease in overlap. In all cases, the improvements increase as the number of objects increase. Although the space utilization of the mqr-tree is 13-8% lower than that of the R-tree, it is still at least 50% overall. In addition, although the maximum tree height of the mqr-tree is higher than that of the R-tree, the difference in tree height decreases to 20% as the number of objects increases. It must also be noted that the average tree height

| #objects | index | #nodes | height | util (%) | coverage | overcoverage | overlap |
|---|---|---|---|---|---|---|---|
| 500 | mqr-tree | 279 | 7(5) | 56 | 290177.89 | 40756.80 | 21028.36 |
| | R-tree | 305 | 4 | 65 | 372130.69 | 45306.94 | 25402.35 |
| 1000 | mqr-tree | 585 | 8(5) | 54 | 631842.02 | 80732.27 | 46784.41 |
| | R-tree | 605 | 5 | 66 | 837206.01 | 91859.61 | 58061.60 |
| 5000 | mqr-tree | 2870 | 9(6) | 55 | 3703689.90 | 388583.38 | 246737.32 |
| | R-tree | 3021 | 6 | 66 | 5161652.44 | 478843.25 | 334332.24 |
| 10000 | mqr-tree | 5786 | 10(7) | 55 | 7861089.36 | 758803.43 | 492034.63 |
| | R-tree | 6017 | 7 | 66 | 10968198.78 | 1005048.34 | 735112.17 |
| 50000 | mqr-tree | 28794 | 11(8) | 55 | 45248114.33 | 3846739.24 | 2531268.33 |
| | R-tree | 30164 | 9 | 66 | 67744586.52 | 5700505.69 | 4394641.83 |
| 100000 | mqr-tree | 57746 | 12(9) | 55 | 95796506.41 | 7715213.25 | 5091928.56 |
| | R-tree | 60368 | 10 | 66 | 148388577.82 | 12014645.01 | 9434519.77 |

**Table 1. Uniform Distribution of Objects**

| #objects | index | #nodes | height | util (%) | coverage | overcoverage | overlap |
|---|---|---|---|---|---|---|---|
| 500 | mqr-tree | 281 | 7(5) | 56 | 183202.42 | 52329.78 | 0 |
| | R-tree | 266 | 4 | 71 | 245727.36 | 57450.22 | 5120.44 |
| 1000 | mqr-tree | 584 | 7(5) | 54 | 412179.74 | 101789.65 | 0 |
| | R-tree | 535 | 4 | 71 | 576901.59 | 115082.14 | 13292.48 |
| 5000 | mqr-tree | 2880 | 9(7) | 55 | 2608566.96 | 503669.43 | 0 |
| | R-tree | 2684 | 6 | 71 | 3818879.85 | 620295.57 | 116626.14 |
| 10000 | mqr-tree | 5758 | 9(7) | 55 | 5683295.67 | 999477.64 | 0 |
| | R-tree | 5375 | 7 | 71 | 8438581.92 | 1284808.36 | 285330.7 |
| 50000 | mqr-tree | 28814 | 11(8) | 55 | 34311970.34 | 5017147.64 | 0 |
| | R-tree | 26889 | 9 | 71 | 54548019.98 | 7079331.94 | 2062184.26 |
| 100000 | mqr-tree | 57737 | 12(9) | 55 | 73778577.04 | 10048704.63 | 0 |
| | R-tree | 53820 | 9 | 71 | 122816655.76 | 15131590.62 | 5082885.99 |

**Table 2. Uniform Distribution of Points**

of the mqr-tree is almost equal to the height of the R-tree. We believe that this is a small price to pay for the significant decrease in coverage, overcoverage and overlap.

Table 2 presents the results for the data sets consisting of uniformly distributed points. The most significant finding in these results is that zero overlap is achieved when an index is constructed for points using the mqr-tree insertion strategy. This is very important because point queries can be executed without having to potentially traverse multiple paths in the tree. In addition, significant reductions in coverage (25-40%) and overcoverage (9-34%) occur. The values for height are similar to those obtained for the object datasets, and therefore we feel these are insignificant compared to the achievement of zero overlap.

## 5.4. Results on Real Data

Table 3 presents the results for the road and railroad data. Here, we also achieve almost no overlap in the mqr-tree. The results show that a reduction of almost 100% is achieved. In addition, we also achieve over 90% in reduction in overlap and overcoverage. The mqr-tree has a higher tree height in the worst case - almost double in some cases. However, the average tree height for the mqr-tree is lower. Given that the overlap and overcoverage of the mqr-tree are much lower, it is expected that more efficient searching will be achieved despite the higher tree height.

## 5.5. Results on Exponential Data Sets

Due to space limitations, we did not include our results on the exponential object and point sets here. However, we found that significant reductions in coverage, overlap and overcoverage are achieved when the mqr-tree is used to index exponential data. In addition, zero overlap for point data was achieved. We did discover that the worst-case height of the mqr-tree is worse than that obtained from indexing the uniformly-distributed data, but that the average

| #objects | index | #nodes | height | util (%) | coverage | overcoverage | overlap |
|---|---|---|---|---|---|---|---|
| 11381 (CArrline) | mqr-tree | 7755 | 13(9) | 49 | 248.10 | 94.13 | 1.01 |
| | R-tree | 5858 | 6 | 73 | 9699.83 | 3777.58 | 3684.48 |
| 21831 (CArdline) | mqr-tree | 14118 | 13(9) | 51 | 352.01 | 97.20 | 4.19 |
| | R-tree | 11192 | 7 | 73 | 21602.31 | 8118.90 | 8025.92 |
| 35074 (CDrrline) | mqr-tree | 23108 | 13(9) | 50 | 4324.47 | 1561.67 | 18.18 |
| | R-tree | 18096 | 8 | 73 | 36064.09 | 13081.80 | 11538.44 |
| 92392 (MXrdline) | mqr-tree | 58849 | 13(10) | 51 | 2038.99 | 553.05 | 16.87 |
| | R-tree | 47116 | 9 | 74 | 96803.79 | 35321.41 | 34785.37 |
| 10060 (MXrrline) | mqr-tree | 6735 | 12(9) | 50 | 1294.95 | 541.89 | 2.20 |
| | R-tree | 5248 | 7 | 72 | 9184.51 | 3474.23 | 2934.55 |
| 121416 (CDrdline) | mqr-tree | 76998 | 15(10) | 52 | 9500.39 | 2925.65 | 59.95 |
| | R-tree | 62399 | 10 | 72 | 134965.07 | 46201.81 | 43336.29 |

**Table 3. Uniform Distribution of Points**

case height is only slightly worse.

## 6. Conclusion and Future Work

We propose the mqr-tree, a two-dimensional index structure that improves upon the structure, validity rules and insertion algorithm of the 2DR-tree. We show through experimental evaluation that the mqr-tree outperforms a benchmark indexing strategy, and achieves no or little overlap. Currently, the mqr-tree is limited to two dimensions.

Future work includes the following. The first is to evaluate the searching ability of the mqr-tree versus benchmark strategies such as the R-tree [6]. 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.

## References

[1] 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.

[2] 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.

[3] D. Comer. The ubiquitous B-tree. *ACM Computing Surveys*, 11:121–37, 1979.

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

[5] Geological Survey of Canada. Geoscience Data Repository, http://gdr.nrcan.gc.ca/index_e.php. (visited March 2006), 2006.

[6] 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.

[7] 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.

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

[9] M. Moreau and W. Osborn. Revisiting 2DR-tree insertion. In *Proceedings of the 2008 Canadian Conference on Computer Science and Software Engineering*, 2008.

[10] J. Orenstein and T. Merrett. A class of data structures for associative searching. In *Proceedings of the Third ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pages 181–90, 1984.

[11] W. Osborn and K. Barker. Searching through spatial relationships using the 2DR-tree. In *Proceedings of the 11th IASTED International Conference on Internet and Multimedia Systems and Applications*, 2006.

[12] 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.

[13] Research Collaboratory For Structural Bioinformatics. Protein data bank, http://www.rcsb.org/pdb. (visited May 2004), 2004.

[14] H. Samet. *The design and analysis of spatial data structures*. Addison-Wesley, 1990.

[15] 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.

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

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