Evaluating the Spatial Indexing of Dense Point Sets

Wendy Osborn
Department of Mathematics and Computer Science
University of Lethbridge
Lethbridge, Alberta, T1K 3M4
Canada
Email: wendy.osborn@uleth.ca

Marc Moreau
Department of Mathematics and Computer Science
University of Lethbridge
Lethbridge, Alberta, T1K 3M4
Canada
Email: marc.moreau@uleth.ca

***Abstract*—In this paper, we present an evaluation of the mqr-tree as a spatial access method for handling high-density point regions, such as world co-ordinates. Although previous work in spatial access methods focused on indexing objects of arbitrary size and performing region searches on them, recent applications that require the management of co-ordinate data also require that high-density point data be managed effectively by spatial access methods. The mqr-tree has shown promise in effectively managing point data. A comparison of the mqr-tree versus the benchmark R-tree shows that the mqr-tree can index high-density point regions effectively. In addition, searching dense point regions using the mqr-tree requires far fewer disk accesses than the R-tree when point density is very high.**

I. Introduction

A spatial data repository such as a spatial database system [1], [2] manages objects, such as points, lines, and regions, which are located in multidimensional space. An important issue for any spatial data repository is the ability to efficiently retrieve one or more objects based on their locations. To facilitate efficient access to spatial data, one can use a spatial access method [3], [4], [5] to index the objects based on their location. Historically, research on spatial access methods has focused on region-query retrieval of arbitrary objects that are represented with minimum bounding rectangles (MBRs). However, more recent applications of spatial data management require the management and retrieval of point data, such as longitude and latitude co-ordinates. For example, an application of exact match location searching comes from a mobile information system such as a tourist information system [6], [7]. In a tourist information system, the location of items of interest, such as historic buildings and other tourist facilities, are tracked using co-ordinates (i.e. points). If a system is tracking items of interest in a high-interest area, this can lead to a very dense set of points that must be managed for efficient search and retrieval. In addition, as a user (i.e. point query) moves around, the items of interest that are near them must be obtained continually so they can be presented. This requires very efficient processing of point queries, so the user is not waiting for the information they seek.

Most indexes that have been proposed specifically for handling point data use a recursive space partitioning technique [8], [9], [10]. Although simple to implement, these approaches manage space that does not contain any points. Spatial access methods, on the other hand, only manage subregions of space that contain objects, and not the entire space [4], [11], [12], [3], [13], [5]. One drawback of most spatial access methods is the overlap of these subregions, which can lead to multiple-path (and therefore, inefficient) searching. This is especially noticeable when performing point searches (both exact-match and overlap-point) [13]. However, a recently proposed spatial access method, the mqr-tree [5], has been shown to efficiently index point data with zero overlap of the subregions that contain then. The application of the mqr-tree in a tourist information system [7] revealed the potential of the mqr-tree to effectively manage very dense sets of points.

Therefore, this work presents an in-depth evaluation of the mqr-tree for indexing very dense sets of points so that efficient exact-match point queries are possible. This work not only evaluates the number of disk accesses (i.e. page hits) required by the mqr-tree for exact-match searches in a dense point set, but also presents some general statistics of the mqr-tree - specifically, the coverage and overlap of the subregions managed by the mqr-tree - and relates them to the resulting page hit values that are obtained from using the mqr-tree.

The remainder of this paper proceeds with some background on both the mqr-tree and the R-tree, which is used for comparison purposes in the evaluation. Then, the mqr-tree is evaluated for effective exact-match point searches. A discussion of the results of the evaluation is presented before the paper concludes with future research directions.

II. Background

In this work, the mqr-tree is compared against the R-tree. The R-tree is chosen for comparison because it is considered a benchmark structure in the area of spatial access methods. We briefly summarize both data structures here.

A. R-tree

The R-tree [4] was the first approximation-based spatial access method to be proposed. It is based on the B+-tree [14] and can index a set of spatial objects that consists of both points and objects of non-zero size. All nodes of the R-tree contain entries of the form:

$$(MBR, ptr)$$

In a leaf node, MBR is the approximation that represents a spatial object, and ptr is a pointer to the actual spatial

978-1-4799-0033-6/13/$31.00 ©2013 IEEE

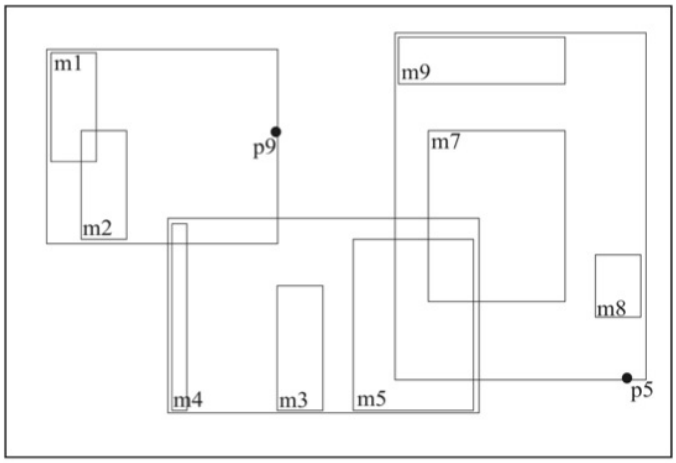

Fig. 1. Example Dataset (from [3])

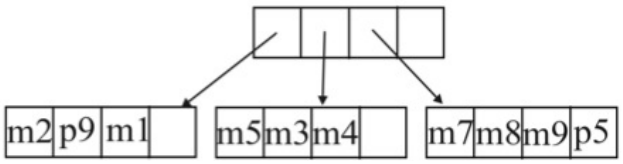

Fig. 2. R-tree

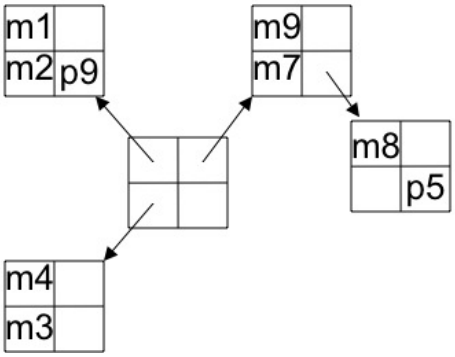

Fig. 3. mqr-tree

object on secondary storage. In a non-leaf node, ptr points to a subtree and MBR is the approximation that encloses all MBRs in the subtree referenced by ptr.

Figure 2 depicts an R-tree that is indexing the set of objects in Figure 1. As you can see, the left-most leaf node contains entries that would point to objects m1, m2 and p9, while an entry in the root node contains the MBR that encloses those objects. The same situation takes place for the other leaf nodes.

A point search - both exact-match and overlap - takes place in the following way. First, the query point is tested for overlap with all MBRs in the root node. For any MBRs that overlap the query point, the search continues in the corresponding subtrees in the same manner until zero or more leaf nodes are reached. For those leaf nodes, the objects are tested for overlap or exact match with the point, and any that pass the test are retrieved for the result.

B. mqr-tree

The mqr-tree [5] is a recent addition to research in spatial access methods. The main difference between the mqr-tree and the R-tree is that nodes are organized in quadrants (NW,NE,SE, SW and CENTRE) in the mqr-tree instead of the linear organization used by the R-tree. This allows for spatial relationships between objects to be maintained. Groups of objects are placed in a node based on their relationship to the centre of the MBR that encompass them. MBRs in non-leaf nodes are organized in the same manner. Figure 3 depicts an mqr-tree that is indexing the set of objects in Figure 1. Note that CENTRE in each node is left out for clarity. Notice that the objects m1,m2 and p9 are placed in the northwest, southwest and southeast locations respectively in the node because their centres are located northwest, southwest and southeast of the centre of the MBR that contains them. Also notice for m9,m7,m8 and p5, that m8 and p5 are both southeast of the centre of the MBR that contains them, therefore they are placed in a separate node and organized separately.

This organization also provides support for non-linear point search strategies to be implemented. Because a point query can only fall into one quadrant (NW,NE,SE,SW and CENTRE), the MBR that encompasses the objects (or MBRs) can be partitioned recursively, with the point tested for which part of the MBR it falls into. This eliminates having to test every entry for overlap with the point query. Note that this strategy only works when the mqr-tree is indexing points - it does not work when indexing objects or querying with regions.

III. Evaluation

In this section, we present the results of our performance evaluation of the mqr-tree versus the R-tree. The primary objective of the evaluation is to compare the two data structures on the average number of disk accesses (i.e. page hits) it takes to perform an exact-match point query. It should be noted that most of the query points used in the evaluation do not exist in the data sets. Therefore, the queries are being performed to determine how many disk accesses are required to determine that the points do not exist in the data sets.

A secondary objective is to determine if any relationships exist between coverage, overcoverage and overlap of the data structures, and the number of disk accesses required for locating a point.

A. Data Sets and Tests

The data used for the evaluations consisted of randomly generated points that reside in specified square units of space, between 1x1 units and 100x100 units. Point sets were created for both constructing the trees and performing the searches. Overall, 30 sets of 10,000 points were created for index construction, with 30 more corresponding sets of 20 points created for performing the searches. Randomly-generated data sets were chosen for the evaluation, instead of real world point sets, so that the density of the point sets could be better controlled.

index	1x1	2x2	3x3	4x4	5x5	6x6	7x7	8x8	9x9	10x10
mqr-tree	6	6	6	6	6	6	6	6	6	6
r-tree	1705	1242	658	532	363	269	175	156	115	102

Fig. 4. Results for 1x1 to 10x10 Area Size

For each pair of 10,000 and 30 point sets, 100 R-trees and mqr-trees are constructed. To be able to fairly compare both data structures, a node size of 5 entries is used in both cases.

For each tree, the coverage, overcoverage, overlap, space utilization and tree height were recorded. Also for each tree, 20 searches are performed, and the number of disk accesses that is required for locating a point is recorded. For disk accesses, the worst case is assumed here - that is, each node is fetched from secondary storage every time it is required.

The average number of disk accesses per search is calculated over all 100 trees. Then, the average coverage, overcoverage, overlap, tree height, space utilization, and number of disk accesses for each data set is calculated.

B. Results

The results for the average number of required disk accesses will be presented here. Any relationships between disk accesses and the other statistics will be presented in Discussion (Section IV). The results for the area sizes of 1x1 to 10x10 are presented first, followed by 11x11 to 20x20, then 10x10 to 100x100.

Figure 4 depicts the search results for the regions of sizes 1x1 to 10x10. Here, we see that the mqr-tree significantly outperforms the R-tree when searching in very dense point regions. In particular, when searching in a region of 1 square unit (1x1) that is indexed with an mqr-tree, the mqr-tree only requires 5 page hits on average to determine whether or not the point being searched on is in the index, where the R-tree requires approximately 1700 page hits to determine whether a point exists in the index. In addition, where the mqr-tree maintains a consistent average number of page hits to locate a point, regardless of the density of the points in the space they occupy, in the case of the R-tree the average number of page hits required to locate a point decreases as the density of the point set decreases.

Figure 5 shows the search results for the regions of sizes 11x11 to 20x20. Here, we observe the same trends as in Figure 4. Specifically, the mqr-tree once again maintains an almost constant number of page hits to obtain a point (or determine that it is not in the index), while the R-tree continues its decline on the number of page hits required to locate a point.

To see if a crossover point exists where the R-tree begins to outperform the mqr-tree with respect to page accesses, a few more tests were run. Figure 6 shows the results of the last set of searches on regions of sizes 10x10 to 100x100. The results show that after the 30x30 region size, the results for the r-tree begin to plateau. The average number of disk accesses required for point queries in the last few regions in the diagram (i.e. 70x70 to 100x100) is 10.

Tests were also carried out with point sets in regions from 100x100 to 1000x1000. Due to space limitations, the results

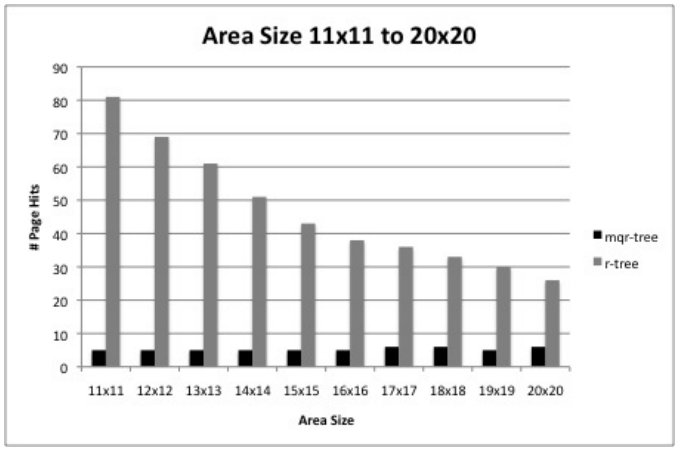

Fig. 5. Results for 11x11 to 20x20 Area Size

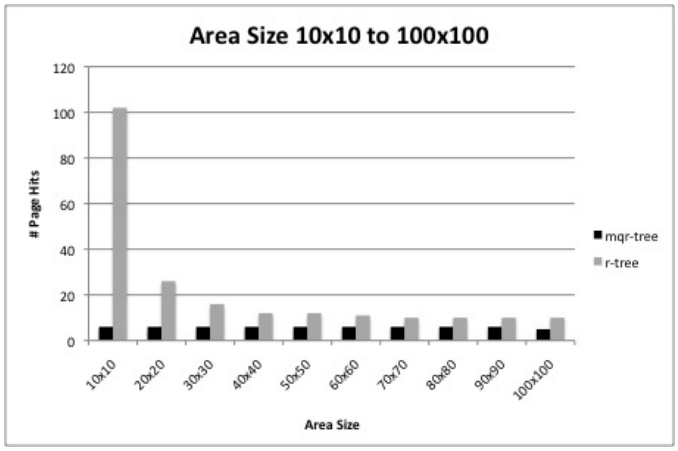

Fig. 6. Results for 10x10 to 100x100 Area Size

are not included here. However, their results showed the a similar pattern of 6 required disk accesses for the mqr-tree, and 9-10 disk accesses required for the R-tree, with no improvement of the R-tree over the mqr-tree occurring.

IV. Discussion

Through the evaluation above, it is discovered that the mqr-tree requires a similar number of disk accesses per point search, regardless of the density of the set of points. Also, that the number of disk accesses required by the R-tree decreases until a constant number is reached. In this section, an attempt is made to identify any relationships between the other statistics the define the spatial access methods - namely, tree height, overlap, coverage and overcoverage - and the number of disk accesses required for performing exact-match point queries. Figure 7 presents a table of the remaining statistics for the mqr-tree and R-tree that correspond to the 1x1 to 10x10 area sizes

index	height	coverage	overcov	overlap	sp.util
mqr-tree	10(7.45)	5.68	1.00	0.00	55
r-tree	6	1683.63	1032.13	1031.13	80
mqr-tree	10(7.45)	22.73	4.00	0.00	55
r-tree	6	5354.06	3063.07	3059.07	80
mqr-tree	10(7.45)	51.02	9.00	0.00	55
r-tree	6	7222.63	3694.12	3685.12	79
mqr-tree	10(7.45)	90.89	16.00	0.00	55
r-tree	6	8260.45	3952.21	3936.21	77
mqr-tree	10(7.44)	141.67	24.99	0.00	55
r-tree	6	8929.46	4097.91	4072.91	77
mqr-tree	10(7.45)	204.09	35.99	0.00	55
r-tree	6	9297.34	4167.27	4131.28	77
mqr-tree	10(7.45)	278.02	48.99	0.00	55
r-tree	6	9567.21	4227.89	4178.90	77
mqr-tree	10(7.45)	364.12	63.97	0.00	55
r-tree	6	9708.18	4252.75	4188.78	77
mqr-tree	10(7.45)	461.26	80.97	0.00	55
r-tree	6	9751.47	4263.62	4182.64	77
mqr-tree	10(7.45)	566.87	99.97	0.00	55
r-tree	6	9771.36	4259.49	4159.52	77

Fig. 7. Statistics for 1x1 to 10x10 Area Size

in Figure 4. This table contains the coverage, overcoverage, overlap, node space utilization, and height of all trees. For the tree height of the mqr-tree, the first number is the maximum height of the tree, while the number in brackets is the average height (or, the longest path and the average path length, respectively).

The first observation that can be made is that the number of disk accesses required by the mqr-tree is less than both the average and maximum height of the tree. Because no overlap of subregions exists for any mqr-tree, the conclusion can be drawn that a one-path point search is achieved. Although the height of the R-tree is smaller than that of the mqr-tree in all cases, the ratio of the number of disk accesses to tree height for the R-tree is significantly higher, which is a result of the search having to traverse multiple paths to locate a point (or not find it!). Therefore, although the mqr-tree has a higher maximum height, it is less than 2x the height of the R-tree but results in a significantly improved number of pages accesses.

With respect to the other statistics, it appears that the decrease in the number of disk accesses required by the R-tree is directly related to the increase in coverage, overcoverage and overlap of its subregions. At first glance, this may not make any sense. However, a closer look at the coverage and overlap shows that for the smaller area sizes, the difference between and coverage and overlap is smaller than the difference between these values in the larger area sizes. This means that more covered area in the smaller area sizes is actually covered multiple times, which leads to more multiple-path searching than in the R-trees that have more covered area that is not subjected to overlap.

V. Conclusion

In this paper, we present an evaluation of the mqr-tree as a spatial access method for handling high-density point regions. Although previous work in spatial access methods focused on indexing objects of arbitrary size and performing region searches on them, recent applications that require the management of co-ordinate data require that high-density point data be managed effectively by spatial access methods as well. The mqr-tree has shown promise in this regard. A comparison of the mqr-tree versus the benchmark R-tree shows that the mqr-tree can index dense point regions effectively. In addition, searching dense point regions using the mqr-tree requires far fewer disk accesses than the R-tree when point density is very high.

A future direction of research based on this work is the following. It was noted that most exact-match point searches do not actually find a matching point. This is due to the fact that in a mobile application such as a tourist information provider, the user (i.e. point query) is not standing on top of the item of interest. Therefore, an extension of the mqr-tree to support nearest-neighbour queries is desirable, so that the mqr-tree can be more useful for such mobile applications in the future.

References

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

[2] P. Rigaux, M. Scholl, and A. Voisard, *Spatial databases: with application to GIS*. Morgan-Kauffman, 2001.

[3] V. Gaede and O. Günther, "Multidimensional access methods," *ACM Computing Surveys*, vol. 30, pp. 170–231, 1998.

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

[5] M. Moreau and W. Osborn, "mqr-tree: a 2-dimensional spatial access method," *Journal of Computer Science and Engineering*, vol. 15, pp. 1–12, 2012.

[6] A. Hinze, A. Voisard, and G. Buchanan, "Tip: Personalizing information delivery in a tourist information system," *J. of IT & Tourism*, vol. 11, no. 3, pp. 247–264, 2009.

[7] W. Osborn and A. Hinze, "TIP spatial index: efficient access to digital libraries in a context-aware mobile system," in *Proceedings of the 23rd Australasian Database Conference*, 2012.

[8] J. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, pp. 509–517, 1975.

[9] J. Bentley and J. Friedman, "Data structures for range searching," *ACM Computing Surveys*, vol. 11, pp. 397–409, 1979.

[10] J. Nievergelt, H. Hintenberger, and K. Sevcik, "The grid file: an adaptable, symmetric multikey file structure," *ACM Transactions on Database Systems*, vol. 9, pp. 38–71, 1984.

[11] 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*, 1990, pp. 322–31.

[12] 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.

[13] W. Osborn, "The 2DR-tree: a 2-dimensional spatial access method," Ph.D. dissertation, University of Calgary, University of Calgary, 2005.

[14] D. Comer, "The ubiquitous B-tree," *ACM Computing Surveys*, vol. 11, pp. 121–37, 1979.