Revisiting the DBM-Tree

Marcos R. Vieira[1], Fabio J. T. Chino[2], Agma J. M. Traina[2], Caetano Traina Jr.[2]

[1] University of California at Riverside
mvieira@cs.ucr.edu
[2] University of São Paulo at São Carlos
jun-chino@uol.com.br, {agma,caetano}@icmc.usp.br

Categories and Subject Descriptors: Information Systems [**Miscellaneous**]: Databases

Keywords: indexes, metric access methods, similarity queries

1. INTRODUCTION

In [Vieira et al. 2004] we presented a new dynamic Metric Access Method (MAM) called DBM-tree. This structure, unlike any other MAM, explores the varying density of elements in the dataset that allows creating, in a *controlled* way, *unbalanced* trees. Every dynamic MAM that works with persistent data proposed so far uses the same principle employed in conventional trees, like the B$^+$-tree [Comer 1979]. This principle is to maintain the height of the tree as minimal as possible, and it is widely applied for data domains for which the total ordering property holds. Since the total number of disk operations required to retrieve a register is direct correlated with the height of the tree, one way to perform a search with minimum number of disk accesses is to keep the structure height-balanced.

Unfortunately, the total ordering property does not hold for data in metric domains. This means that *several* subtrees of a MAM must need to be evaluated during a search (breadth-search), making the total number of disk operations required directly correlated also with the amount of node overlap in the tree. This behavior happens for metric and also for domains that holds only the partial ordering (not the total ordering) property, like the spatial domains.

When MAM are built with a very strict constraint on the tree height-balance, there is *little* flexibility for these structures to decrease the overlapping among their nodes. Node overlapping is very common in structures like the M-tree [Ciaccia et al. 1997] and Slim-tree [Traina Jr. et al. 2000], and leads to many *breadth-searches* for each subtree in order to evaluate a query. Therefore, maintaining the height of the tree low is achieved in expense of increasing the overlapping among subtrees. The direct consequence of this is that the performance of MAM deteriorates very fast since several subtrees at each tree level has to be analyzed in order to evaluate a query.

Since the performance of MAM is direct proportional to how many breadth-searches are required to evaluate a search, as well as the cost of the depth-search in each subtree, increasing the node overlapping among subtrees also increases the cost related to evaluate a query. The DBM-tree was the first MAM to have a *tradeoff* between the height-balance of the tree and the number of breadth-searches in order to achieve better query performance. The main idea of the DBM-tree is to keep a "compromise" between the number of disk accesses required to perform a breadth-search in several subtrees and to perform a depth-search in one subtree. In this way, the tree is expected to be deeper in regions of higher density of elements and shallower in lower density regions.

To show some properties of the DBM-tree, we indexed 25,376 geographical coordinates of US cities, villages, boroughs and towns (available at `www.census.gov`) using a DBM-tree. We employed the *MAMViewer* tool [Chino et al. 2005] to visualize a DBM-tree. Figure 1 presents elements (cities) with

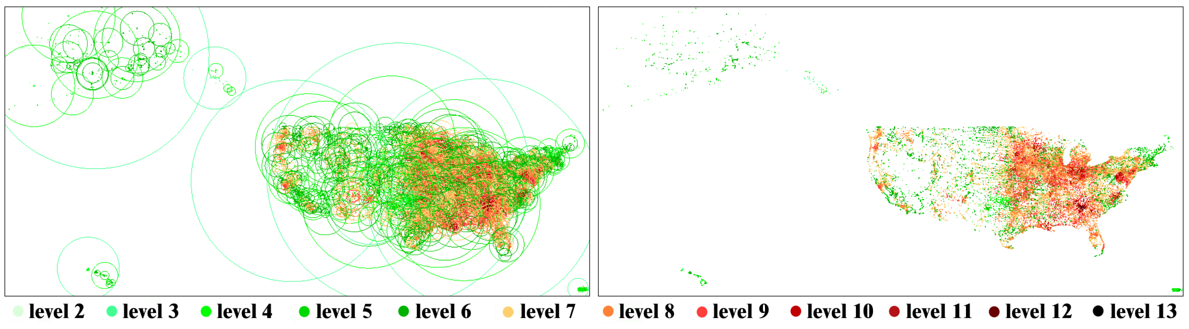

Fig. 1. Visualization of a *DBM-MM* indexing the *USCities* dataset. (a) Elements and the covering node radius; (b) Only the elements. It is possible to verify that the structure is deeper (darker colors) in high-density regions, and shallower (lighter colors) in low-density regions.

different colors representing the different levels in the tree where they are stored. Figure 1(a) shows all elements and the covering radius of each node in the tree. Figure 1(b) shows only the elements in the tree. The elements with darker colors (dark red and black) are in deeper levels than those with lighter colors (light and dark green). As it can be seen, The tree depth is larger in higher density regions (Great Lakes, Atlanta and Washington/Philadelphia/New Jersey/New York areas) and that elements are stored in every level of the structure, as expected. It visually shows that the tree depth is smaller in low density regions of elements (Porto Rico, Hawaii, Alaska and Northwestern states). Moreover, the number of elements at the deepest levels is small, even in the high-density regions.

Extensive experiments performed over synthetic and real datasets showed that the DBM-tree performed better wrt distance calculations and running time to answer similarity queries than both M-tree and Slim-tree. Not surprisingly, the DBM-tree performed less disk accesses than the balanced structures M-tree and Slim-tree. This is due the fact that there is less overlapping in the DBM-tree nodes than both balanced structures.

2. FOLLOW UP RESEARCH

Since the DBM-tree publication in [Vieira et al. 2004], we worked on several research related projects, as well as in MAMs in general. One interesting project related to the DBM-tree is the *MAMView* framework [Chino et al. 2005]. The framework is intended to visualize the data indexed and the operations performed by a MAM (insertion, deletion, query execution, etc.). The *MAMView* has two components: the first one is a set of APIs that are installed in the target MAM under evaluation; the second component is the *MAMViewer* tool which reads the code generated by the first component and displays an iterative visualization of the MAM being evaluated. The *FastMap* algorithm [Faloutsos and Lin 1995] is the core piece in the second component, where elements in metric space are mapped to a 2-dimensional Euclidean space. The *MAMView* framework is available for downloading at `www.gbdi.icmc.usp.br/download`.

The *MAMView* framework, allows MAM developers to better understand the behavior of the DBM-tree, as well as of other structures implemented in the *GBDI Arboretum Library* (`gbdi.icmc.usp.br/arboretum`), allowing developers to interact with the *MAMView* generated visualization of the DBM-tree. Here we show some screenshots of the *MAMViewer* tool. In Figure 2(a), some elements and their representatives are displayed using different colors to represent different levels of the tree. The elements are "connected" to their representatives in Figure 2(b) and their covering radius is presented in Figure 2(c). In Figure 2(d) a rotation was applied in the visualization (c).

A cost model for the DBM-tree that does not rely on the height of tree as well as a new splitting algorithm were proposed in [Vieira et al. 2006]. This cost model is based on some node statistics and

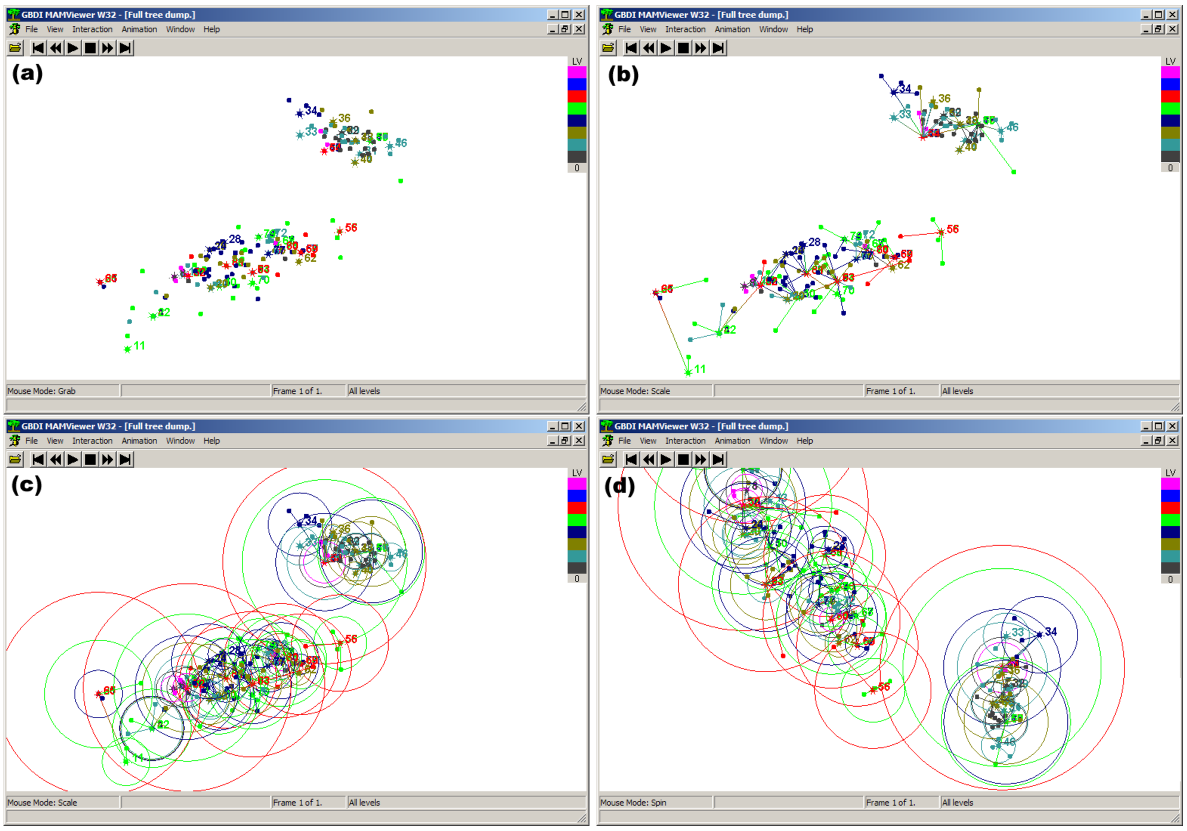

Fig. 2. The *MAMViewer* tool: (a) elements (dots) and representatives (stars with *ids*); (b) elements connected to their representatives represent nodes; (c) covering radius of each node; and (d) a different view of the same visualization as in (c).

the distance distribution of elements indexed by a DBM-tree.

Following the idea that a controlled unbalancing can improve performance, a new structure was investigated, aiming at quickly creating a temporary index in main-memory, which can execute a point query without node overlap. The resulting MAM, called the MM-tree [Pola et al. 2007], showed that, indeed, performing similarity queries in this structure is faster than in the main-memory MAMs that present overlaps. A further MM-tree extension generated the Onion-tree, a structure that allows a point query overlap-free execution over data stored in disk [Carelo et al. 2009] and strengthened that not enforcing, but controlling the height-balancing of a metric tree is in fact the best policy.

3. FUTURE TOPICS FOR RESEARCH

Here we enumerate three directions of future research that can be explored for the DBM-tree. The first one is to explore the idea of extend node, firstly proposed for the X-tree in [Berchtold et al. 1996], for the DBM-tree. The general idea is that, instead of splitting a node covering a high density region into two new ones, a new *extended* node is created and some elements are moved to it. In this way, the extended node can accommodate some elements and the splits can be postponed. Splitting a node can result in node overlapping where the split happens and, possibly increasing the node radii in the upper tree levels. The new tree would thus have smaller overlapping among its nodes, increasing the query performance, at the expense of performing sequential scanning in the extended nodes.

The second possible direction is developing bulk-loading algorithms for the DBM-tree, exploiting the fact that height balancing is not strictly required. As the construction possibilities are larger than for balanced structures, a bulk-loading algorithm can employ novel strategies that can achieve better performance than the bulk-loading algorithms existing for balanced structures (e.g. [Ciaccia and Patella 1998] and [Vespa et al. 2007]). Evaluating re-insertion as an optimizing strategy for the tree is also an interesting investigation to pursue here.

A third research direction is related to the insertion operation in a DBM-tree. In order to find the best node to accommodate a new element, new strategies similar to multi-way insertion [Lokoc and Skopal 2008], [Skopal and Lokoc 2009] could be explored for the DBM-tree. This operation is more expensive to perform when compared with a conventional insertion that takes a single path from the root to a particular node of the tree.

REFERENCES

Berchtold, S., Keim, D. A., and Kriegel, H.-P. The X-tree: An index structure for high-dimensional data. In *Proceedings of the Int'l Conference on Very Large Data Bases*. Morgan Kaufmann, Bombay, India, pp. 28–39, 1996.

Carelo, C. C. M., Pola, I. R. V., de Aguiar Ciferri, C. D., Ciferri, R. R., Traina Jr., C., and Traina, A. J. M. The onion-tree: Quick indexing of complex data in the main memory. In *Proceedings of the East European Conference on Advances in Databases and Information Systems*. Riga, Latvia, pp. 235–252, 2009.

Chino, F. J. T., Vieira, M. R., Traina, A. J. M., and Traina Jr., C. Mamview: A visual tool for exploring and understanding metric access methods. In *Proceedings of the ACM Symposium on Applied Computing*. Santa Fe, New Mexico, USA, pp. 1218–1223, 2005.

Ciaccia, P. and Patella, M. Bulk loading the M-tree. In *Proceedings of the Australasian Database Conference*. Perth, Australia, pp. 15–26, 1998.

Ciaccia, P., Patella, M., and Zezula, P. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the Int'l Conference on Very Large Data Bases*. Athens, Greece, pp. 426–435, 1997.

Comer, D. The ubiquitous B-tree. *ACM Computing Surveys* 11 (2): 121–137, 1979.

Faloutsos, C. and Lin, K.-I. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of the ACM SIGMOD Int'l Conference on Management of Data Conference*. San Jose, USA, pp. 163–174, 1995.

Lokoc, J. and Skopal, T. On reinsertions in M-tree. In *Proceedings of the First Int'l Workshop on Similarity Search and Applications*. IEEE Computer Society, pp. 121–128, 2008.

Pola, I. R. V., Traina Jr., C., and Traina, A. J. M. The mm-tree: A memory-based metric tree without overlap between nodes. In *Proceedings of the East-European Conference on Advances in Databases and Information Systems*. Varna, Bulgaria, pp. 157–171, 2007.

Skopal, T. and Lokoc, J. New dynamic construction techniques for M-tree. *Journal of Discrete Algorithms* 7 (1): 62–77, 2009.

Traina Jr., C., Traina, A. J. M., Seeger, B., and Faloutsos, C. Slim-trees: High performance metric trees minimizing overlap between nodes. In *Proceedings of the Int'l Conference on Extending Database Technology*. Konstanz, Germany, pp. 51–65, 2000.

Vespa, T. G., Traina Jr., C., and Traina, A. J. M. Bulk-loading dynamic metric access methods. In *Proceedings of the Brazilian Symposium on Databases*. João Pessoa, Brazil, pp. 160–174, 2007.

Vieira, M. R., Traina Jr., C., Chino, F. J. T., and Traina, A. J. M. DBM-tree: A dynamic metric access method sensitive to local density data. In *Proceedings of the Brazilian Symposium on Databases*. Brasília, Brazil, pp. 163–177, 2004.

Vieira, M. R., Traina Jr., C., Chino, F. J. T., and Traina, A. J. M. DBM-tree: Trading height-balancing for performance in metric access methods. *Journal of the Brazilian Computer Society* 11 (3): 37–52, 2006.