Tasting the Time: How M-Tree Broke the "27 Club" Curse

Paolo Ciaccia[1,*], Marco Patella[1], Fausto Rabitti[2] and Pavel Zezula[3]

[1]Alma Mater Studiorum Universitá di Bologna, viale del Risorgimento, 2, 40134 - Bologna, Italy
[2]ISTI - CNR, Via G. Moruzzi, 1, 56124 - Pisa, Italy
[3]Masaryk University, Botanickà 554/68a, 602 00 - Brno, Czech Republic

Abstract

The M-tree turned 27 this year, since it was first published in 1997, at SEBD [5] and VLDB [6]. Differently from the likes of Jim Morrison, Jimi Hendrix, and Amy Winehouse, it is still alive and kicking, receiving dozens of downloads and citations every year.[1] In this paper, we offer a quick overview of the context in which the M-tree was created and show how it helped starting a whole family of metric trees.

In the mid of 90's we were involved in the Esprit LTR HERMES project on high-performance multimedia storage management, funded by the European Community. At that time we faced a couple of basic issues dealing with the problem of indexing multimedia data: (i) the high dimensionality of feature vectors characterizing the content of such data, and (ii) the huge variety of distance-based criteria used for assessing the similarity of two multimedia objects. Both issues ruled out the possibility of using at-the-time established solutions available for spatial data, such as the R-tree [1]. And even novel proposals, such as the X-tree [2], able to alleviate the first issue, were not able to deal at all with more sophisticated distance functions that were not based on a Cartesian coordinate space.

The abstraction of *metric space* was the 1st ingredient underlying the design of the M-tree. A metric space is a pair (U, d), where U is the objects' domain, and d is a distance function that obeys the metric postulates: (i) *symmetry*: $\forall x, y \in U \; : \; d(x, y) = d(y, x)$; (ii) *non-negativity*: $\forall x, y \in U$: if $x \neq y$ then $d(x, y) > 0$, $d(x, x) = 0$; (iii) *triangle inequality*: $\forall x, y, z \in U \; : \; d(x, y) \leq d(x, z) + d(z, y)$. Even without any notion of objects' coordinates (as used by spatial indices), a metric index can effectively prune part of the search space by exploiting above properties, in particular the triangle inequality.

Although at that time some proposals of metric indices were already available, such as the VP-tree [3] and the GNATree [4], they were based on a main-memory implementation, thus unsuitable for large, disk-resident datasets, the common case for multimedia data. The 2nd ingredient/requirement for the design of the M-tree was therefore to devise a paged and balanced organization, along the successful lines adopted by B^+-trees and R-trees.

Since its first introduction, in 1997 at SEBD [5] and VLDB [6], M-tree has been extended with:

[1]"M-tree gets a lot of citations because it is easily beaten." Benjamin Bustos, personal communication.

SEBD 2024: 32nd Symposium on Advanced Database Systems, June 23-26, 2024, Villasimius, Sardinia, Italy

*Corresponding author.
paolo.ciaccia@unibo.it (P. Ciaccia); marco.patella@unibo.it (M. Patella); fausto.rabitti@isti.cnr.it (F. Rabitti); zezula@fi.muni.cz (P. Zezula)
0000-0002-1794-6244 (P. Ciaccia); 0000-0003-2655-0759 (M. Patella); 0000-0001-5438-6760 (P. Zezula)

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

- an efficient bulk loading technique [7];
- cost models for estimating search costs [8] and [10];
- techniques for evaluating complex queries, where multiple similarity predicates are defined on a single feature [9] or on multiple features (M^2-tree) [13];
- efficient algorithms for solving probably approximately correct (PAC) queries [11] and [12];
- a technique to correctly solve queries using a user-defined distance, different from the one used to build the actual index (QIC-M-tree) [14].

The two original papers were seminal in generating a whole family of metric access methods, sharing the general M-tree structure. Besides the already cited M^2-tree and QIC-M-tree, several techniques have been proposed to improve the search performance of M-tree, among which the slim-tree [15], the PM-tree [16], the M^+-tree [17], the BM^+-tree [18], and the M^*-tree [19]. Finally, the NM-tree [20] is able to deal with distance functions that do not satisfy the metric postulates, in particular the triangle inequality.

In our view, one of the reasons for M-tree success is the fact that, since the very beginning, its source code has been freely available for research purposes at http://www-db.disi.unibo.it/Mtree/. The code of the original M-tree implementation was written in C++ and it is based on the GiST library [21]. A parallel version was presented in [22]. M-tree is now also available, among others, in PostgreSQL, again exploiting GiST [23], as a plugin for the Secondo DBMS (https://secondo-database.github.io/content_plugins.html) and the ELKI data mining environment (https://elki-project.github.io/), and in the SurrealDB multi-model database (https://surrealdb.com/). It is finally interesting to note that, in a conference held exactly on the same days as VLDB 1997, a homonymous abstract data type, generalizing a quadtree for parallel adaptive computations, was proposed [24]: such data structure however did not stand the test of time, rapidly fading into oblivion.

References

[1] A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. SIGMOD 1984: 47–57.

[2] S. Berchtold, D.A. Keim, and H.-P. Kriegel. The X-tree : An Index Structure for High-Dimensional Data. VLDB 1996: 28–39.

[3] J.K. Uhlmann. Satisfying General Proximity/Similarity Queries with Metric Trees. Information Processing Letters 40(4): 175–179 (1991).

[4] S. Brin. Near Neighbor Search in Large Metric Spaces. VLDB 1995: 574–584.

[5] P. Ciaccia, M. Patella, F. Rabitti, and P. Zezula. Indexing Metric Spaces with M-Tree. SEBD 1997: 67–86.

[6] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. VLDB 1997: 426–435.

[7] P. Ciaccia and M. Patella. Bulk Loading the M-Tree. ADC'98: 15–26.

[8] P. Ciaccia, M. Patella, and P. Zezula. A Cost Model for Similarity Queries in Metric Spaces. PODS 1998: 59–68.

[9] P. Ciaccia, M. Patella, and P. Zezula. Processing Complex Similarity Queries with Distance-Based Access Methods. EDBT 1998: 9–23.
[10] P. Ciaccia, A. Nanni, and M. Patella. A Query-sensitive Cost Model for Similarity Queries with M-Tree. ADC'99: 65–76.
[11] P. Ciaccia and M. Patella. PAC Nearest Neighbor Queries: Using the Distance Distribution for Searching in High-Dimensional Metric Spaces. SEBD 1999: 259–273.
[12] P. Ciaccia and M. Patella. PAC Nearest Neighbor Queries: Approximate and Controlled Search in High-Dimensional and Metric Spaces. ICDE 2000: 244–255.
[13] P. Ciaccia and M. Patella. The M^2-tree: Processing Complex Multi-Feature Queries with Just One Index. DELOS 2000.
[14] P. Ciaccia and M. Patella. Searching in Metric Spaces with User-Defined and Approximate Distances. ACM TODS 27(4): 398–437 (2002).
[15] C. Traina, A.J.M. Traina, B. Seeger, and C. Faloutsos. Slim-Trees: High Performance Metric Trees Minimizing Overlap Between Nodes. EDBT '00: 51–-65.
[16] T. Skopal, J. Pokorný and V. Snasel. PM-Tree: Pivoting Metric Tree for Similarity Search in Multimedia Databases. ADBIS 2004: 803–815.
[17] X. Zhou, G. Wang, J.X. Yu, and G. Yu. M^+-Tree: A New Dynamical Multidimensional Index For Metric Spaces. ADC'03: 161–-168.
[18] X. Zhou, G. Wang, X. Zhou, and G. Yu. BM^+-Tree: A Hyperplane-Based Index Method for High-Dimensional Metric Spaces. DASFAA'05: 398.–409.
[19] T. Skopal and D. Hoksza. Improving the Performance of M-Tree Family by Nearest-Neighbor Graphs. ADBIS 2007: 172–188.
[20] T. Skopal and J. Lokoč. NM-Tree: Flexible Approximate Similarity Search in Metric and Non-Metric Spaces. DEXA 2008: 312–325.
[21] J.M. Hellerstein, J.F. Naughton and A. Pfeffer. Generalized Search Trees for Database Systems. VLDB 1995: 562–573.
[22] P. Zezula, P. Savino, F. Rabitti, G. Amato, and P. Ciaccia, Processing M-trees with Parallel Resources, RIDE 1998: 147–154.
[23] I. Donkó, J. Szalai-Gindl, G. Gombos, and A. Kiss. An Implementation of the M-Tree Index Structure for PostgreSQL Using GiST. Informatics'2019: 189–194.
[24] Q. Wu, A.J. Field, and P.H.J. Kelly. M-Tree: A Parallel Abstract Data Type for Block-Irregular Adaptive Applications. Euro-Par'97: 638–649.