Graph Indexing and Retrieval based on Median Graphs

Francesc Serratosa, Albert Solé-Ribalta & Enric Vidiella

Universitat Rovira i Virgili, Computer Science Department, Spain
{francesc.serratosa,albert.sole}@urv.cat,
{enric.vidiella}@estudiants.urv.cat

Abstract. Metric indexing is used to organize large databases of Attributed Graphs and define fast queries on these sets. We present a proposal where indexing is based on an m-tree in which routing nodes of the m-tree act as prototypes of the sub-cluster of Attributed Graphs that routing nodes represent. In the classical schemes, the information kept in the tree nodes of the sub-cluster is a selected Attributed Graph from the sub-set. Depending on the sub-cluster and the application, it is difficult to select a good representative of the sub-clusters. To that aim, we propose to use Median Graphs as the main information kept in the routing nodes of the m-tree. Experimental validation shows that in the database queries, the decrease of the nodes explored in the m-tree while using a Generalized Median Graph is about 20% respect using a selected Attributed Graph.

Keywords: Graph database, m-tree, graph organization, graph prototype, graph indexing.

1 Introduction

Index structures are fundamental tools of database technology, they are used to obtain efficient access to large collections of images. Traditional database systems manage global properties of images, such as histograms, and many techniques for indexing one-dimensional data sets have been defined. Since a total order function over a particular attribute domain always exists, this ordering can be used to partition the data and moreover, exploited to efficiently support queries. Several multi-dimensional indexes have appeared, such as, color, texture, shape, with the aim of increasing the efficiency in executing queries on sets of objects characterized by multi-dimensional features. Once again, ordering systems of individual orthogonal dimensions are used for partitioning the search space, so these methods can, in fact, be considered as direct extensions of the one-dimensional case.

Effective access to image databases requires queries addressing the expected appearance of searched images [1]. To this end, it is needed to represent the image as a set of entities and relations between them. The effectiveness of retrieval may be improved by registering images as structural elements rather than global features [2]. In the most practiced approach to content-based image retrieval, the visual appearance of each spatial entity is represented independently by a vector of features. Mutual relationships between entities can be taken into account in this retrieval process through a cascade filter, which evaluates the similarity in the arrangement of entities after these have been retrieved on the basis of their individual features [3]. To

overcome these systems, local entities and mutual relationships have to be considered to have the same relevance and to be defined as parts of a global structure that captures mutual dependencies. In this case, the model of content takes the shape of an Attributed Graph (AG). The attributes of the vertices of the AGs represent the features of the local entities and the attributes of the arcs of the AGs represent the features of the relationships.
While the distance between two sets of independent features can be computed in polynomial time, the exact distance between two AGs is computed in exponential time, respect the number of nodes of the AGs. For this reason, few contributions, of practical interest, have been proposed supporting the application of AGs to content-based retrieval from image databases [4] and [5].

Out of the specific context of content-based image retrieval, the problem of comparing an input graph against a large number of model graphs has been addressed in several approaches. In some applications, the classes of objects are represented explicitly by a set of prototypes, which means that a huge amount of model AGs must be matched with the input AG and so the conventional error-tolerant graph matching algorithms must be applied to each model-input pair sequentially. As a consequence, the total computational cost is linearly dependent on the size of the database of model graphs and exponential (or polynomial in subgraph methods) with the size of the AGs. For applications dealing with large databases, this may be prohibitive. To alleviate these problems, some attempts have been designed with the aim of reducing the computational time of matching the unknown input patterns to the whole set of models from the database. Those approaches assume that the AGs that represent a cluster or class are not completely dissimilar in the database and in this way only one structural model is defined from the AGs that represent the cluster; as a consequence only one comparison is needed for each cluster [6], [7] and [8].
In this paper, we show an indexing scheme implemented by an m-tree in which the cluster knowledge embedded in each node of the m-tree is represented by a Median Graph. In the experimental section, we have compared our scheme with a similar one in which the cluster information was represented by one of the AGs of the cluster [4]. We show that the use of Median Graphs instead of AGs in the m-tree scheme makes the queries more efficient. In the next section, we comment the related work and introduce our method. In section 3, we give some definitions related to AGs and Median Graphs. In sections 4 and 5, we first present the metric trees and then this technique is applied to AGs. In section 6, we experimentally evaluate our model. We finish the paper drawing some conclusions and presenting the future work.

2 Related Work and our Proposal

Some indexing techniques have been developed for graph queries. We discern these techniques in two categories. In the first ones, the index is based on several tables and filters [9], [10]. In the second ones, the index structure is based on metric trees [4], [11], [12].
In the first group of techniques, the ones that are not based on trees, we emphasize the method developed by Shasha *et. al.* [9] called GraphGrep. GraphGrep is based on a table in which each row stands for a path inside the graph (up to a threshold length) and each column stands for a graph. Each entry in the table is the number of occurrences of the path in the graph. Queries are processed in two phases. The filtering phase generates a set of candidate graphs for which the count of each path is

at least that of the query. The verification phase verifies each candidate graph by subgraph isomorphism and returns the answer set. More recently, Yan *et. al.* [10] proposed GIndex that uses frequent patterns as index features. These frequent patterns reduce the index space as well as improve the filtering rate. The main drawback of these models is that the construction of the indices requires an exhaustive enumeration of the paths or fragments with high space and time overhead. Moreover, since paths or fragments carry little information about a graph, the lost of information at the filtering step seems to be unavoidable.
Considering the second group, the first time that metric trees were applied to graph databases was done by Berreti *et. al.* [4]. Attributed Graphs were clustered hierarchically according to their mutual distances and indexed by m-trees [13]. Queries are processed in a top-down manner by routing the query along the index tree. Each node of the index tree represents a cluster and it has one of the graphs of the cluster as a representative. The graph matching problem, in the tree construction and at query time, was solved by an extension of the A* algorithm that uses a look-ahead strategy plus a stopping threshold. Latter, Lee *et. al.* [11] used this technique to model graphical representations of foreground and background scenes in videos. The resulting graphs were clustered using the edit-distance metric, and similarity queries were answered using a multi-level index structure.
More recently, He and Singh [12] proposed what they called a Closure-tree. It uses a similar structure than the one presented by Berreti [4] but, the representative of the cluster was not one of the graphs but a graph prototype (called closure graph) that could be seen as the union of the AG that compose the cluster. Figure 1 shows the closure of 3 graphs. The structurally similar nodes that have different attributes in the graphs are represented in the closure graph with only one node but with more than one attribute. Closure trees have two main drawbacks. First, they can only represent discrete attributes at nodes of the AGs. Second, they tend to generalize to much the set that represent, allowing AGs that have not been used to synthesize the closure graph.

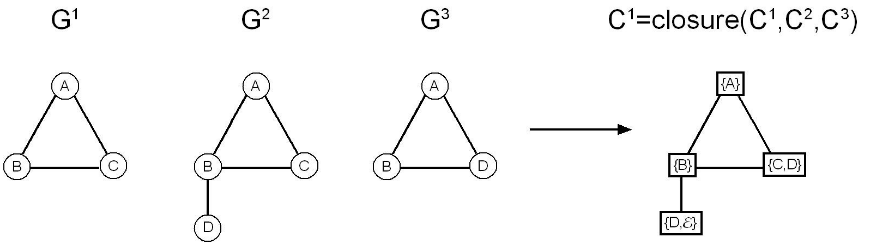

Fig. 1: Example of a Closure obtained by 3 AGs.

Our proposal is to use Median Graphs as a representative of the sub-clusters in the routing nodes of the metric trees instead of an AG representative [4] or a closure graph [12]. On one hand, we wish to find a better representative of the sub-set and on the other hand, we wish to use continuous attribute values.

3 Graph Preliminaries

Given an alphabet of labels for the nodes and arcs of the AGs, *L*, we define *U* as the set of all AGs that can be constructed using labels from *L*. Moreover, we assume there is a distance function between AGs *d*.

Given $S = \{g_1, g_2, ..., g_n\} \subseteq U$, the *Generalized Median Graph* $\bar{g}$ of *S* is defined as,

$$\bar{g} = \arg\min_{g \in U} \sum_{g_i \in S} d(g, g_i) \tag{1}$$

That is, the generalized median graph $\bar{g}$ of *S* is a graph $g \in U$ that minimizes the sum of distances to all the graphs in S. Notice that $\bar{g}$ is usually not a member of *S*, and in general, more than one generalized median graph may exist for a given set *S*. The computation of a generalized median graph is a NP-complete problem. Nevertheless, several suboptimal methods to obtain approximate solutions for the generalized median graph in reasonable time have been presented [14], [15] and [16]. These methods apply some heuristic functions in order to reduce the complexity of the graph distance computation and the size of the search space.

An alternative to the generalized median graph but less computationally demanding is the *Set Median Graph*.

$$\bar{g} = \arg\min_{g \in S} \sum_{g_i \in S} d(g, g_i) \tag{2}$$

The difference between the two models consists in the search space where the median is looked for. As it is shown in (1), the search space for the generalized median graph is *U*, that is, the whole universe of graphs. In contrast, the search space for the set median graph is simply *S*, that is, the set of given graphs. It makes the computation of set median graph exponential in the size of the graphs, due to the complexity of graph edit distance, but quadratic with respect to the number of graphs in *S*.

4 Database Indexing based on m-trees

A metric tree [13], m-tree, is a tree of nodes, each containing a fixed maximum number of *m* entries, < *node* > *:= {< entry >}*^{m}. In turn, each entry is constituted by a routing element *H*; a reference to the root r^H of a sub-index containing the element in the so-called covering region of *H*; and a radius d^H providing an upper bound for the distance between *H* and any element in its covering region, < *entry* > *:= {H,* r^H, d^H*}*. During retrieval, triangular inequality is used to support efficient processing of range queries. That is, queries seeking for all the elements in the database which are within a given range of distance from a query element *G*. To this end, the distance between *G* and any element in the covering region of a routing element *H* can be lower-bounded using the radius r^H and the distance between *G* and *H*.

To perform range queries in Metric Trees, the tree is analyzed in a top down fashion. Specifically, if d_{max} is the range of the query and G is the query graph, the following conditions are employed, at each node of the tree, to check whether all the elements in the covering region of *H*, sub^H, can be discarded or accepted. The conditions are based on the evaluation of the distance between the routing element and the graph query *d(G,H)*.

If condition (3) holds, we will reject all elements deeper from the routing element.

$$d(G,H) \geq d_{\max} + r^H \quad \Rightarrow \textit{No element in } sub^H \textit{ is acceptable} \tag{3}$$

In a similar manner, the following condition checks whether all the elements in the covering region of *H*, sub^H, fall within the range of the query. In this case, all the elements in the region can be accepted:

$$d(G,H) \leq d_{\max} - r^H \quad \Rightarrow \textit{Every element in } sub^H \textit{ is acceptable} \tag{4}$$

In the critical case that neither of the two inequalities holds, the covering region of H, sub^H, may contain both acceptable and no acceptable elements, and the search must be repeated on the sub index sub^H.

5 Graph Indexing based on Median Graphs

In this section, we first present the qualities of the Median Graphs as routing elements and second, the method used to obtain a metric tree based on Median Graphs.

Accordingly to the definition of the Median Graphs, they are supposed to be the best representatives of a set of graphs, due to they represent a graph which minimizes the sum of distances to all other graphs of the set. The advantages of using Median Graphs as routing elements in an m-tree are manifold. The main effect of using them is the reduction of the overlap between sub-clusters, due to the radius of the covering region can be more tightly adjusted. In fact, if we use the Generalized Median Graphs as a routing element, the radius of the covering region has to be equal or lower than the radius of the covering region represented by a Set Median Graph.

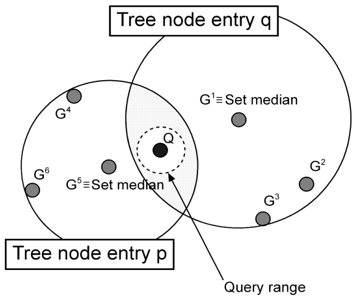

Fig. 2.1: Clusters represented by a Set Median.

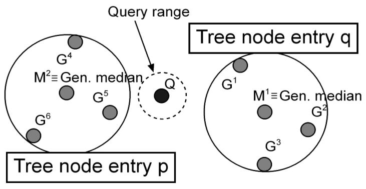

Fig. 2.2: Clusters represented by a Generalized Median.

Figures 2.1 and 2.2 show the same 6 elements in two sub-clusters and the radius of their covering regions. The representative of sub-clusters in figure 2.1 is the Generalized Median Graph and in figure 2.2 is the Set Median Graph. Suppose a hypothetical query graph Q with a query range represented by the outer doted circle. The execution of the search will behave very different on both representations. In the Set Median approach, neither entry p nor q holds for equations (3) and (4), so the sub^q and sub^p must be explored. However, due to the better representation that the Generalized Median provides, (3) holds for both tree node entries p and q. Consequently, it can be assumed that none of the entries contain any desired graph. Thus, they can be discarded and not explored.

We provide a general construction methodology from which we are able to construct a metric tree independently of the type of the routing element; a Generalized Median Graph or a Set Median Graph. Given an AG set, it is crucial to obtain the same structure of the m-tree for both types of routing elements, since we want to compare its representational power in similar conditions. We use a non-balanced tree constructed through a hierarchical clustering algorithm and complete linkage clustering. In this way, given a set of graphs, we first compute the distance matrix over the whole set and then we construct a dendogram. We obtain a set of partitions that clusters the AGs with the dendogram using some horizontal cuts. With these partitions we generate the m-tree and we synthesize a Generalized Median or a Set

Median. Figure 3.1 shows an example of a dendogram. The AGs G^i are placed on the leaves of the dendogram and the Generalized Medians or Set Medians M^i are placed on the junctions between the cuts and the horizontal lines of the dendograms. Figure 3.2 shows the obtained m-tree.

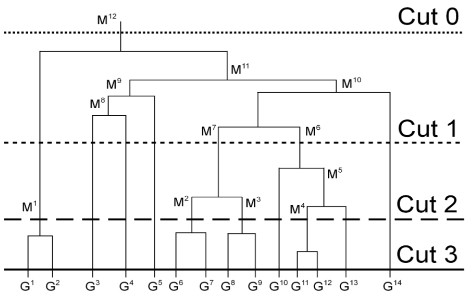

Fig. 3.1: Example of a dendogram.

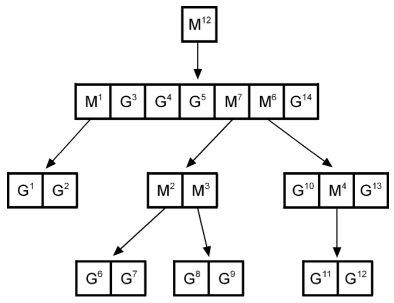

Fig. 3.2: The obtained m-tree.

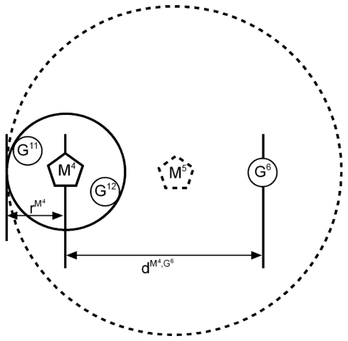

Fig. 4.1: Second radius computation rule.

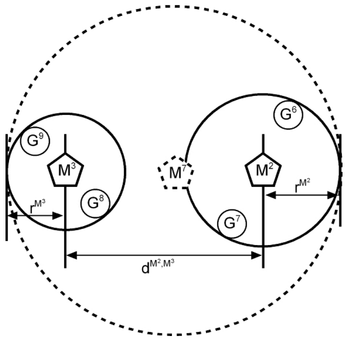

Fig. 4.2: Third radius computation rule.

Computing the m-tree based on the Generalized Median Graph

At each node of the m-tree, we have to compute a Generalized Median, we use the method presented in [17]. With the aim of reducing the computational cost of computing these Medians, we compute them as pairwise consecutive computations of the Medians obtained in lower levels of the tree. For instance, to compute M^7, which appears at Figure 3.1, we only use M^2 and M^3 Medians. That is, we assume that:

$$M^7 \cong \overline{(M^2,M^3)} \cong \overline{(\overline{(G^6,G^7)},\overline{(G^8,G^9)})} \tag{5}$$

The covering region radius r^p of the Generalized Median M^p is computed applying three rules, depending whether the type of the descendant of M^p in the dendogram is another Median (that is, a routing node of the m-tree) or an AG (that is, a leaf of the m-tree):

- When both descendants are AGs (G^a and G^b):

$$r^p = Max(Dist(M^p,G^a),Dist(M^p,G^b)) \tag{6}$$

- When a descendant is a Median (M^a) and the other is an AG(G^b):

$$r^p = Max(Dist(M^p,M^a)+r^a,Dist(M^p,G^b)) \tag{7}$$

- When both descendants are Medians (M^a and M^b):

$$r^p = Max(Dist(M^p, M^a) + r^a, Dist(M^p, M^b) + r^b) \tag{8}$$

Fig. 4.1 and 4.2 illustrate the second and third rule, respectively. In the first case, $Dist(M^5,M^4) + r^{M4}$ is greater than $Dist(M^5,G^6)$, and in the second case $Dist(M^7,M^3) + r^{M3}$ is greater than $Dist(M^7,M^2)+r^{M2}$.

Computing the m-tree based on the Set Median Graph

At each node of the m-tree, it is desired to compute the Set Median. Given the distance matrix of the whole set of AGs, the computation of the Set Median given a sub-set is simply performed by adding the pre-computed distances between the involved AGs. For instance, to compute M^7 that appears at Figure 3.1, we use the distances between the AGs G^6, G^7, G^8 and G^9.

The covering region radius r^p of the Set Median M^p is computed as the maximum distance between M^p and any of the AGs in the sub-set.

6 Evaluation

To evaluate the performance of both model, we used two indices. The first index is addressed to evaluate the quality of the tree. The lower is the overlap between the covering regions of sibling nodes, the higher is the quality of the m-tree since they are more discriminative and therefore the time to compute the query reduces.

Given two sibling nodes, we define the overlap of their covering regions as follows,

$$S(i,j) = \begin{cases} \dfrac{(R_i + R_j)}{d(i,j)} & if\ \dfrac{(R_i + R_j)}{d(i,j)} > 1 \\ 0 & Otherwise \end{cases} \tag{9}$$

Given a node of the m-tree, their own overlap is computed as the normalized overlap between their children. The radius of the sub-clusters that the children represent is obtained from the parameter d^H in their m-tree nodes.

$$S_g = \sum_{i=1}^{E} \sum_{j=i+1}^{E} S(N_i, N_j) \Big/ \binom{E}{2} \tag{10}$$

where E is the number of entries of the m-tree node. Finally the general overlap of an m-tree is computed as,

$$S = \sum S_g \Big/ numberOfNodes \tag{11}$$

The second index, called *access ratio,* is addressed to evaluate the capacity of the m-tree to properly route the queries. Given a query element, this index is the number of accessed nodes and leaves of the m-tree. That is, the number of comparisons required between the queried AG and the median graphs (in the case of nodes of the m-tree) plus the number of comparisons between the queried AG and the AGs (in the case of leaves of the m-tree). This value is normalized by the number of AGs used to generate the m-tree.

$$access\ ratio = number\ of\ comparisons \Big/ number\ of\ elements \tag{12}$$

In the evaluation phase, we used the Letter database created at the University of Bern [18]. It is composed by 15 classes and 150 AGs per class representing the Roman alphabet. Nodes are defined over a two-dimensional domain that represents its plane position (x, y). Edges have a binary attribute that represents the existence of a line between two terminal points.

We constructed 12 different m-trees per each letter (or class) varying the number of dendogram partitions *{4, 7, 10 , 12}* and the number of AGs that represent each class, that is, the AGs that are used to generate the m-tree *{30, 50, 100}*. Therefore, we analyzed 15x12=180 m-trees with the Generalized Median Graph as routing elements and other 180 m-trees with the Set Median Graph as routing elements. Figures 5.1 and 5.2 show the general overlap (11) of the m-trees depending on the number of partitions and the number of AGs per class. Figure 5.3 shows the difference between the Set Median and the Generalize Median.

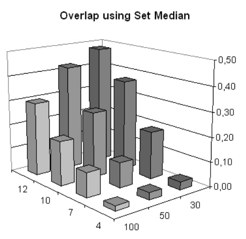

Fig. 5.1: overlap using Set Median.

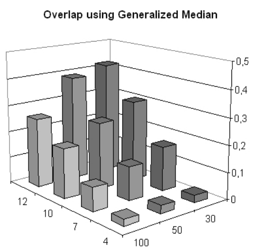

Fig. 5.2: overlap using Generalized Median.

		Database size		
	Difference	30	50	100
Num. Cuts	4	0,0039	-0,0019	-0,0046
	7	0,027	-0,0247	0,0128
	10	0,0657	0,0154	-0,0055
	12	0,0616	0,0335	0,0398

Fig. 5.3: overlap difference of both methods.

The overlap index is slightly lower when the Generalize Median is used than when the Set Median is used. The difference increases when the number of AGs per partition decreases since it is statistically more difficult to find a good representative using the Set Median.

To analyze our model through the access ratio (12) we generated several queries on the above m-trees. Each test was carried out by 9 queries in which we used 9 different AGs. 3 of these AGs were used to create the m-tree, 3 AGs where not used to create the m-tree but belong to the same letter and 3 AGs belong to other letters. Figures 6 to 8 show the access ratio of these queries on m-trees with Generalized Median, Set Median and the difference between the Generalized and the Set. In these figures, we applied the following query ranges (section 4) of $d_{max} = \{D_{max}/8, D_{max} / 4, D_{max}/2\}$, respectively, where D_{max} is the maximum distance of any two AGs of the m-tree.

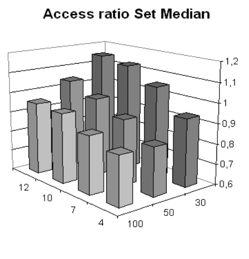

Fig. 6.1: access ratio using $d_{max} = D_{max}/2$

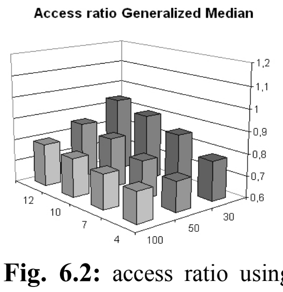

Fig. 6.2: access ratio using $d_{max} = D_{max}/2$.

		Database size		
	Difference	30	50	100
Num. Cuts	4	0,16	0,10	0,10
	7	0,20	0,15	0,13
	10	0,23	0,16	0,17
	12	0,21	0,19	0,16

Fig. 6.3: difference of access ratio.

Analyzing the experimental results, we conclude that the Generalized Median decreases the number of accesses in about 20%. As a consequence, we conclude that the Generalized Median has better representational power than the Set Median as routing objects in the m-trees. Note that the access ratio of some experiments on the Set Median is higher than one. That means that without any indexing structure, the run time would be lower.

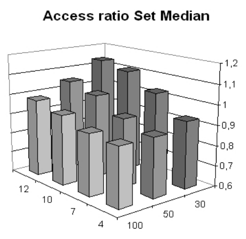

Fig. 7.1: Access ratio using $d_{max} = D_{max}/4$

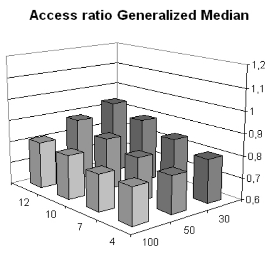

Fig. 7.2: Access ratio using $d_{max} = D_{max}/4$.

		Database size		
	Difference	30	50	100
Num. Cuts	4	0,13	0,12	0,13
	7	0,19	0,14	0,14
	10	0,22	0,18	0,15
	12	0,20	0,17	0,17

Fig. 7.3: difference of access ratio.

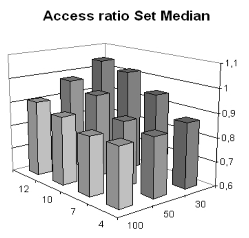

Fig. 8.1: Access ratio using $d_{max} = D_{max}/8$

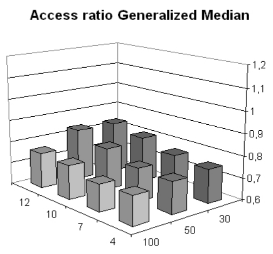

Fig. 8.2: Access ratio using $d_{max} = D_{max}/8$.

		Database size		
	Difference	30	50	100
Num. Cuts	4	0,12	0,10	0,11
	7	0,18	0,12	0,12
	10	0,21	0,15	0,13
	12	0,21	0,14	0,15

Fig. 8.3 difference of access ratio.

7 Conclusions

We have presented a graph indexing technique based on metric trees and Median Graphs. Furthermore, we have compared the use of the Generalized Median Graph and the Set Median Graph as routing elements in the m-trees. We arrive at the conclusion that the construction of the m-tree is computationally harder using the Generalized Median Graph but better performance can be obtained while using them as routing elements. Experimental validation on a real database shows that the general overlap of the m-trees is lower when using the Generalized Medians instead of Set Median. Moreover, we have verified that the number of comparisons done while performing the queries is lower in the Generalized Medians than the Set Medians and so, the run time is also lower. With these results, we conclude that it is preferably to use Generalized Medians as routing elements in m-trees instead of Set Medians.

References

1. Gudivada, V.N., Raghavan, V.V.: Special issue on Content Based Image Retrieval Systems. In: Computer, vol. 28, no. 9, (1995).
2. Tao, Y., Grosky, W.I.: Spatial Colour Indexing: A Novel approach for Content-Based Image Retrieval. In: Proc. IEEE international conference Multimedia computing and Systems, (1999).

3. Smith, J.R., Samet, H.: VisualSEEk: A Fully Automated Content-Based Image Query System. In: Proc. ACM Multimedia, pp. 87-98, (1996),
4. Berretti, S., Del Bimbo, A., Vicario, E.: Efficient Matching and Indexing of Graph Models in Content-Based Retrieval. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 10, pp. 1089-1105, (2001).
5. Zhao, J.L., Cheng, H.K.: Graph Indexing for Spatial Data Traversal in Road Map Databases. In: Computers & Operations Research, vol. 28, pp: 223-241, (2001).
6. Serratosa, F., Alquézar, R., Sanfeliu, A.: Function-described graphs for modeling objects represented by attributed graphs. In: Pattern Recognition, vol. 36 no. 3, pp. 781-798, (2003).
7. Serratosa, F., Alquézar, R., Sanfeliu, A.: Synthesis of Function-Described Graphs and clustering of Attributed Graphs. In: International Journal of Pattern Recognition and Artificial Intelligence, vol. 16, no. 6, pp. 621-655, (2002).
8. Sanfeliu, A., Serratosa, F., Alquézar, R.: Second-Order Random Graphs for modeling sets of Attributed Graphs and their application to object learning and recognition. In: International Journal of Pattern Recognition and Artificial Intelligence, vol. 18, no. 3, pp. 375--396, (2004).
9. Shasha ,D., Wang, J.T.L., Giugno, R.: Algorithmics and applications of tree and graph searching. In: ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp. 39-52, (2002).
10. Yan, X., Yu, P.S., Han, J.: Graph indexing: a frequent structure-based approach. In: ACM SIGMOD international conference on Management of data, pp. 335-346, (2004).
11. Lee, S.Y., Hsu, F.: Spatial Reasoning and Similarity Retrieval of Images using 2D C-Strings Knowledge Representation. In: Pattern Recognition, vol. 25 no. 3, pp. 305-318, (1992).
12. He, H., Singh, A.K.: Closure-Tree: An Index Structure for Graph Queries. In: Proc. International Conference on Data Engineering, pp. 38, (2006).
13. Ciaccia, P., Patella, M., Zezula, P.: M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In: Proc. 23rd VLDB Conference, pp. 426-435, (1997).
14. Jiang, X., Münger, A., Bunke, H.: On median graphs: Properties, algorithms and applications. In: IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 23, no. 10, pp: 1144-1151, (2001).
15. Ferrer, M., Valveny, E., Serratosa, F., Riesen, K., Bunke, H.: Generalized Median Graph Computation by Means of Graph Embedding in Vector Spaces. In: Pattern Recognition, vol. 43, no. 4, pp. 1642-1655, (2010).
16. Ferrer, M., Valveny, E., Serratosa, F.: Median graphs: A genetic approach based on new theoretical properties. In: Pattern Recognition, vol. 42, no. 9, pp. 2003-2012, (2009).
17. Neuhaus, M., Riesen, K., Bunke, H.: Fast Suboptimal Algorithms for the Computation of Graph Edit Distance. In: In: Proc. IAPR International Workshops on Structural and Syntactic Pattern Recognition and Statistical Techniques in Pattern Recognition, LNCS 4109, pp. 163-172, (2006)
18. Riesen, K., Bunke, H.: IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning. In: Proc. IAPR International Workshops on Structural and Syntactic Pattern Recognition and Statistical Techniques in Pattern Recognition, LNCS 5342, pp. 287-297, (2009).