Graph Database Retrieval Based on Metric-Trees*

Francesc Serratosa, Xavier Cortés, and Albert Solé-Ribalta

Universitat Rovira i Virgili, Departament d'Enginyeria Informàtica i Matemàtiques, Spain
{francesc.serratosa,xavier.cortes,albert.sole}@urv.cat

Abstract. M-trees are well-know structures used to speed-up queries in databases. In this paper, we evaluate the applicability of m-trees to graph databases. In classical schemes based on metric-trees, the routing information kept in a metric-tree node is a selected element from the sub-cluster that represents. Nevertheless, defining a graph that represents a set of graphs is not a trivial task. We evaluate different graphs-class prototype as routing nodes in the metric tree. The considered prototypes are: Median Graphs, Closure Graphs, First-Order Random Graphs, Function-Described Graphs and Second-Order Random Graphs.

Keywords: Metric-tree, Graph Indexing, Median Graph, First-Order Random Graph, Function-Described Graph, Second-Order Random Graph.

1 Introduction

Indexing structures are fundamental tools in database technology; they are used to obtain efficient access to large collections of elements. Traditional image database systems manage global properties of images, such as histograms [1]. 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 it can be exploited to efficiently support queries. Several multi-dimensional indexes have appeared, such as, colour, texture, shape, with the aim of increasing the efficiency in executing queries on sets of objects characterized by multi-dimensional features.

Effective access to image databases requires queries addressing the expected appearance of searched images [2]. 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 [3, 4]. 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. Thus, local entities and mutual relationships may be considered to have the same relevance and to be defined as parts of a global structure that captures mutual

* This research is supported by Consolider Ingenio 2010: project CSD2007-00018 & by the CICYT project DPI 2010-17112.

dependencies [5]. In this case, the model of content takes the structure of an Attributed Graph.

While the distance between two sets of independent features can be computed in polynomial time [6,7], the exact distance between two graphs is computed in exponential time with respect to the number of nodes of the graphs. Although some sub-optimal solutions have been presented to compare a pair of graphs, in which, the computational complexity is reduced to polynomial cost, few contributions of practical interest have been proposed supporting the application of graphs to content-based retrieval from image databases [8, 9].

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 graphs, which means that a huge amount of model graphs must be matched with the input graph 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 number of model graphs and exponential (or polynomial if suboptimal methods are used) with the size of the graphs. 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 graphs 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 graphs that represent the cluster. These structures are called *Graph-Class Prototypes*. In the classification process, only one comparison is needed for each cluster.

In this paper, we evaluate an indexing scheme, modelled by an m-tree, in which the cluster knowledge embedded in each node of the m-tree is represented by one of the six Graph-Class Prototypes presented in the literature. The different representations of Graph-Class Prototypes are: 1) Set Median Graph [8]; 2) Generalise Median Graph [10, 11, 12, 13] synthesised through a hierarchical method [14], synthesised through a genetic algorithm [15] or synthesised through an extension of the Graduated Assignment algorithm [16]; 3) First-Order Random Graphs [17]; 4) Function-Described Graphs [18, 19]; 5) Second-Order Random Graphs [20]; 6) Closure Graphs [21]. Moreover, we evaluate two types of graph queries; the ones that the user imposes the number of graphs to be queried and the ones that the user imposes the maximum distance between the query graph and the returned graphs. It is not the aim of this paper to explain the structural representation of each graph prototype but to evaluate its representational power in metric trees. Some of the methods presented in this paper have been presented in [14, 23] but only applied to Median Graphs. The aim of this paper is to evaluate the representational power of the Graph-Class Prototypes presented in the literature.

In this paper, we have performed more experiments with more databases and we have put together both types of queries and we have used more Graph-Class Prototypes with the aim of obtaining a more general results and conclusions.

The rest of the paper is organised as follows. In section 2, we comment the few approaches that have been presented for indexing Attributed Graphs. In chapter 3, we introduce metric-trees. In section 4, we explain the methods used to synthesise the graph prototypes. In section 5, we experimentally evaluate the graph prototypes as routing elements of m-trees. We finish the paper drawing some conclusions.

2 Indexing Databases of Graphs

Some indexing techniques have been developed for graph queries. We divide these techniques into two categories. In the first ones, the index is based on several tables and filters [25, 26]. In the second ones, the index structure is based on m-trees [8,21,27].

In the first group of techniques, the ones that are not based on trees, we emphasize the method developed by Shasha *et. al.* [26] 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 compounds to the number of occurrences of a particuar 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. Since indexing schemes based on paths do not ensure graph isomorphism, in a verification phase, each candidate is strictly compared to the query graph and only isomorphic graphs are returned. More recently, Yan *et. al.* [25] proposed G_{Index} that uses frequent patterns as indexing 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 that increases the memory and time requirements of the model. 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 Berretti *et. al.* [8]. Attributed graphs were clustered hierarchically according to their mutual distances and indexed by m-trees [22]. 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. A drawback of this method is that the computational cost is exponential respect the number of nodes in the graphs. Lee *et. al.* [27] 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 [21] proposed what they called a Closure-tree. It uses a similar structure than the one presented by Berretti [8] 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 Attributed Graphs that compose the cluster. 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 attributed graphs. Second, they tend to generalize too much the set of graphs they represent, allowing graphs that have not been used to synthesize the closure graph.

Finally, Median Graphs have been used as a new prototype to represent Attributed Graphs in [14, 15]. More specifically, in [14], they defined queries in which a maximum distance between the query and the graphs was considered. And in [23], they performed k-nearest neighbour queries.

3 Database Indexing Based on Metric-Trees

A metric-tree (m-tree) [22] is a scheme to partition a database in a hierarchical set of clusters, collecting similar objects. Each cluster has a routing object and a radius providing an upper bound for the maximum distance between the reference object and any other object in the cluster. Triangle inequality can be used during the access to the database to prune clusters that are bound out of an assigned range from the query.

Formally, a metric-tree is a tree of nodes. Each node contains a fixed maximum number of m entries, $< node > := \{< entry >\}^m$. In turn, each entry is constituted by a routing element M; a reference to the father r^H of a sub-index containing the element in the so-called covering region of M; and a radius d^M providing an upper bound for the distance between M and any element in its covering region, $< entry > := \{M, r^M, d^M\}$. During retrieval of an element Q, triangular inequality is used to support efficient processing of queries. To this end, the distance between Q and any element in the covering region of a routing element M can be max-bounded using the radius d^M plus the distance between Q and M.

Two different types of queries can be performed to databases organised by m-trees: *k-Nearest-Neighbour* queries [23] and *Similarity* queries [14]. The aim of the k-Nearest Neighbour Queries is to retrieve the k elements in a database that have minimum distance between them and the query element. On the contrary, the aim of the Similarity queries is to retrieve all the elements in the database which its distance to the queried element is lower than a threshold $d_{\max}$.

The m-tree can be constructed using different schemes for the insertion of a new element and the selection of the routing element [22]. In this paper, we use a general construction methodology from which we are able to construct an m-tree independently of the type of the routing element. We use a non-balanced tree constructed through a hierarchical clustering algorithm and complete linkage clustering [24]. In this way, given a set of graphs, the distance matrix over the whole set is computed and then a dendogram is constructed. Using the dendogram and some horizontal cuts, a set of partitions that clusters the graphs in the database is obtained. With these partitions the m-tree is generated. Finally, the information on the routing elements in the m-tree is inserted, M and d^M.

In our case, M is a Graph-Class Prototype and d^M is the maximum distance between the Graph Prototype and any of the graphs in the covering region. Figure 1 shows an example of a dendogram. Elements G^i are placed on the leaves of the

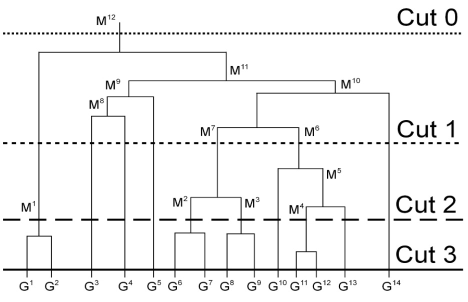
Cut 0
Cut 1
Cut 2
Cut 3
M1 M2 M3 M4 M5 M6 M7 M8 M9 M10 M11 M12
G1 G2 G3 G4 G5 G6 G7 G8 G9 G10 G11 G12 G13 G14

Fig. 1. Example of a dendogram

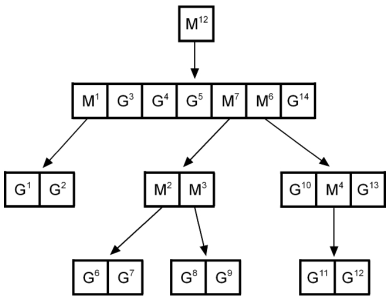

Fig. 2. The obtained m-tree

dendogram and the routing elements M^j are placed on the junctions between the cuts and the horizontal lines of the dendograms. Dendogram of figure 1 defines 4 different partitions. Figure 2 shows the obtained m-tree. Note that in some tree nodes, there are Class-Graph Prototypes (M^j) together with original graphs (G^i).

4 Synthesis of Graph Prototypes Related to Metric-Trees

Two types of methods exist to generate Graph Prototypes from a given set of graphs [18, 20]. We assume the structure of the metric-tree has been computed (section 3) and we have to compute the Graph Prototype and the radius of the cluster d^M. The first method is based on a hierarchical synthesis. The second one is based on a Global Synthesis based on a Common Labelling [28, 29, 30].

In the Hierarchical method, each Graph Prototype is computed only using two Graph Prototypes or Attributed Graphs at a time. Therefore, a Common Labelling is not needed and Graph Prototypes are computed as pairwise consecutive computations of other Graph Prototypes obtained in lower levels of the tree.

In the Global Synthesis, each Graph Prototype is computed using the whole set of Attributed Graphs in the cluster that the m-tree node represents, independently of whether the m-tree node has other nodes as descendants in the tree. The first step of this method computes a Common Labelling from the Attributed Graphs of the sub-cluster and the second step obtains the Graph Prototype. In this paper, we have used two different Common Labelling algorithms, which have the main feature that are independent of the prototype graph to be synthesised. The first one is based on the Graduated Assignment [16] and the second one is based on a genetic algorithm [15].

Note that the Set Median is a special prototype since it does not need to be synthesised. The Set Median is the graphs of the cluster that has the minimum distance between it and the other graphs.

5 Practical Evaluation

Test Parameters: In each test, only one m-tree is constructed with 50 graphs of the reference set. The parameters used to construct each m-tree are:

- Evaluated Datasets: COIL, Letter (low), Letter (high) and GREC.
- Type of routing element: Set Median [8], Hierarchical Median [14], Genetic Median [15], Graduated Assignment Median [16], Closure Graph [21], Function Described Graph [18], First Order Random Graph [17] and Second Order Random Graphs [20].
- Number of dendogram partitions: 7. The partitions are the number of cuts used to generate the m-tree (section 3). This number also corresponds to the levels of the m-tree. Distances to set the cuts are (see figure 1); distance of cut 0 = D_{max}, distance of cut 1 = $D_{max}\cdot 6/7$, distance cut 2 = $D_{max}\cdot 5/7$, ... distance of cut 6 = $D_{max}/7$. Where D_{max} is the maximum distance of any two graphs of the m-tree.

Parameters for each query are:

- Graph query. The graph has been extracted from the test set.
- Number of queries: 50. Results values are the mean of these 50 queries.
- Metric-tree.
    - If k-NN query: Number of elements to be retrieved: **k** = 3.
    - If similarity query: Range of the query: $\mathbf{d_{max}} = 0.6\cdot D_{max}$.

Evaluation Indices: Three indices have been used: Access ratio, Precision and Recall. Access ratio evaluates the capacity of the m-tree to properly route the queries [14, 23]. It is obtained as the normalised number of accessed nodes and leaves of the m-tree given a query. Precision is the fraction of retrieved documents that are relevant to the search and Recall is the fraction of the documents that are relevant to the query that are successfully retrieved. Ground truth of precision and recall are computed by exhaustive search of the elements in the dataset. Note that Precision and Recall values depend on the construction of the m-tree due to we use sub-optimal algorithms to synthesise the prototypes and to compute d^M. In addition, at query time, since distances are also sub-optimally computed, the algorithm may violate triangle inequality restrictions and so return not accurate results.

Datasets: COIL, Letter (low), Letter (high) and GREC (presented in [31]). The 72 images of each element of COIL dataset have been clustered in 4 classes instead of one class. Each class is composed by 18 consecutive images.

Results: Tables 1 and 2 show the access ratio in nearest neighbour and similarity queries. Lower is the access ratio faster is the query. Besides, if the access ratio is greater than 1, the number of comparisons done using the m-tree is higher than if there was no m-tree and the graphs of the whole database where all compared. This situation does not appear in the K-nn queries, which means that it is worth to structure the database in an m-tree. On the contrary, some values of table 2 (similarity query) are greater than 1. To reduce this problem, d_{max} whole have to be reduced but we preferred to use the same value for all the experiments for the compactness of the result values. Besides, some cells of table 2 have value 0.02. This is because, given a query, only the root node of the m-tree is explored. Considering that the m-tree has been built using 50 graphs, the value comes from 0.02 = 1/50. Again, this problem could be solved by adapting d_{max} value to each Graph-Class Prototype.

In general, Median Graphs are the prototypes with better access ratio. So, they obtain faster queries (except for the commented extreme values, 0.02).

Table 1. Access ratio on nearest neighbour queries

Access Ratio (k=3)	**Synthesis**	**COIL**	**Letter L**	**Letter H**	**GREC**
Set Median	----	0.54	0.34	0.35	0.45
Generalise Median	Hierarchical	0.38	0.31	0.33	0.37
Generalise Median	Genetic	0.40	0.38	0.36	0.57
Generalise Median	Graduated Assignment	0.44	0.34	0.37	0.40
Closure Graph		0.65	0.33	0.57	0.80
FORG		0.42	0.35	0.42	0.47
SORG		0.35	0.33	0.36	0.38
FDG		0.45	0.36	0.45	0.56

Table 2. Access Ratio on similarity queries

Access Ratio (similarity)	**Synthesis**	**COIL**	**Letter L**	**Letter H**	**GREC**
Set Median	----	0.02	1.03	1.46	0.06
Generalise Median	Hierarchical	0.47	0.99	1.34	1.07
Generalise Median	Genetic	0.92	1.65	1.66	0.78
Generalise Median	Graduated Assignment	0.89	0.98	1.30	0.94
Closure Graph		0.02	0.02	0.02	0.02
FORG		0.02	0.02	0.02	0.02
SORG		0.02	2.27	2.78	0.02
FDG		0.02	0.47	0.04	0.02

Tables 3 and 4 show the mean precision. SORGs and Closures are the prototypes that obtain the best results although there are other prototypes with similar values. In general, prototypes computed using the Graduated Assignment obtains better results than the Hierarchical and Genetic synthesis. Considering values on tables 1 and 3 (k-NN), we can conclude that the probabilistic prototypes are slower but obtain greater precision. And considering values on tables 2 and 4 (similarity), we realise that the fact that some queries only explore the root node penalises the obtained precision.

Table 3. Precision on nearest neighbour queries

Precision (k=3)	**Synthesis**	**COIL**	**Letter L**	**Letter H**	**GREC**
Set Median	----	0.46	0.76	0.42	0.48
Generalise Median	Hierarchical	0.37	0.62	0.34	0.22
Generalise Median	Genetic	0.11	0.16	0.22	0.34
Generalise Median	Graduated Assignment	0.32	0.94	0.46	0.32
Closure Graph		0.62	0.98	0.38	0.59
FORG		0.58	0.86	0.42	0.57
SORG		0.65	0.81	0.50	0.37
FDG		0.48	0.84	0.44	0.53

Table 4. Precision on similarity queries

Precision (similarity)	**Synthesis**	**COIL**	**Letter L**	**Letter H**	**GREC**
Set Median	----	0.75	0.99	0.99	0.78
Generalise Median	Hierarchical	0.81	0.99	0.99	0.98
Generalise Median	Genetic	0.89	1	0.99	0.92
Generalise Median	Graduated Assignment	0.99	0.99	0.99	0.97
Closure Graph		0.75	0.62	0.74	0.77
FORG		0.75	0.62	0.74	0.77
SORG		0.75	1	1	0.77
FDG		0.75	0.82	0.78	0.77

Finally, tables 5 and 6 show the recall results. In general, probabilistic prototypes obtain greater recall than non-probabilistic ones, except in some cases. Note that in cases that the access ratio is 0.02, the recall is always 1. This is because, if all graphs of the database are accepted, then the recall has to be 1 by definition.

Table 5. Recall on nearest neighbour queries

Recall (k=3)	**Synthesis**	**COIL**	**Letter L**	**Letter H**	**GREC**
Set Median	----	0.26	0.58	0.32	0.50
Generalise Median	Hierarchical	0.24	0.48	0.26	0.22
Generalise Median	Genetic	0.06	0.12	0.18	0.34
Generalise Median	Graduated Assignment	0.18	0.72	0.36	0.34
Closure Graph		0.38	0.76	0.30	0.60
FORG		0.38	0.66	0.32	0.58
SORG		0.40	0.62	0.38	0.38
FDG		0.30	0.64	0.32	0.54

Table 6. Recall on similarity queries

Recall (similarity)	**Synthesis**	**COIL**	**Letter L**	**Letter H**	**GREC**
Set Median	----	1	0.99	1	0.99
Generalise Median	Hierarchical	1	1	0.98	0.90
Generalise Median	Genetic	0.92	1	1	0.96
Generalise Median	Graduated Assignment	1	0.98	0.98	0.90
Closure Graph		1	1	1	1
FORG		1	1	1	1
SORG		1	1	1	1
FDG		1	0.60	0.90	1

Table 7 summarises the results presented in the last 6 tables. Each value is the average of the eight corresponding values. Statistically best values are bolded. FORGs obtain the fastest queries (lower access ratio). Generalise Median (with Graduated Assignment) and SORGs obtain the greatest Precision. Closure Graphs, FORGs and SORGs obtain the greatest Recall. Finally, SORGs obtain the best F-measure.

Table 7. Average results of Access Ratio, Precision, Recall and F-measure

	Synthesis	Access	Precision	Recall	F-measure
Set Median	----	0.53	0.70	0.70	0.70
Generalise Median	Hierarchical	0.65	0.66	0.63	0.64
Generalise Median	Genetic	0.84	0.57	0.57	0.57
Generalise Median	Graduated Assignment	0.70	**0.74**	0.68	0.71
Closure Graph		0.30	0.68	**0.75**	0.71
FORG		**0.21**	0.66	**0.74**	0.70
SORG		0.81	**0.73**	**0.72**	**0.72**
FDG		0.29	0.64	0.32	0.67

6 Conclusions

We have evaluated a graph indexing technique based on metric-trees and several Graph-Class Prototypes. Specifically, we have studied the behaviour of Graph-Class Prototypes as routing elements of m-trees. Several papers have been published that compare the accuracy of the evaluated prototypes. In this paper, we evaluated the goodness of those prototypes on speeding-up queries on graph databases. The evaluation has been performed using four different datasets with different characteristics. We see from the practical validation that probabilistic prototypes seem to achieve better results on k-nn queries. On the contrary, the Generalise Median together with the Set Median seem to give better results on similarity queries at the cost of giving a larger access ratio. Up to now, Set Median Graphs and Closure Graphs where the only prototypes used as routing elements of metric trees. The general conclusion of this work is that other existing Graph-Class Prototypes can also be successfully used as routing elements of metric trees in graph databases.

References

1. Konstantinidis, K., Gasteratos, A., Andreadis, I.: Image retrieval based on fuzzy colour histogram processing. In: Optics Communications, vol. 248, pp. 375–386 (2005)
2. Datta, R., Joshi, D., Li, J., Wang, J.Z.: Image Retrieval: Ideas, Influences, and Trends of the New Age. ACM Computing Surveys 40(2), Article 5 (2008)
3. Jouili, S., Tabone, S.: Hypergraph-based image retrieval for graph-based representation. Pattern Recognition 45(11), 4054–4068 (2012)
4. Lebrun, J., Gosselin, P., Philipp, S.: Inexact graph matching based on kernels for object retrieval in image databases. Image and Vision Computing 29(11), 716–729 (2011)

5. Le Saux, B., Bunke, H.: Feature Selection for Graph-Based Image Classifiers. In: Marques, J.S., Pérez de la Blanca, N., Pina, P. (eds.) IbPRIA 2005. LNCS, vol. 3523, pp. 147–154. Springer, Heidelberg (2005)
6. Gold, S., Rangarajan, A.: A Graduated Assignment Algorithm for Graph Matching. Transactions on Pattern Analysis and Machine Intelligence 18(4), 377–388 (1996)
7. Neuhaus, M., Riesen, K., Bunke, H.: Fast Suboptimal Algorithms for the Computation of Graph Edit Distance. In: Yeung, D.-Y., Kwok, J.T., Fred, A., Roli, F., de Ridder, D. (eds.) SSPR 2006 and SPR 2006. LNCS, vol. 4109, pp. 163–172. Springer, Heidelberg (2006)
8. Berretti, S., Del Bimbo, A., Vicario, E.: Efficient Matching and Indexing of Graph Models in Content-Based Retrieval. IEEE Transactions on Pattern Analysis and Machine Intelligence 23(10), 1089–1105 (2001)
9. Zhao, J.L., Cheng, H.K.: Graph Indexing for Spatial Data Traversal in Road Map Databases. Computers & Operations Research 28, 223–241 (2001)
10. Jiang, X., Münger, A., Bunke, H.: On median graphs: Properties, algorithms and applications. IEEE Trans. on PAMI 23(10), 1144–1151 (2001)
11. Ferrer, M., Valveny, E., Serratosa, F., Riesen, K., Bunke, H.: Generalized Median Graph Computation by Means of Graph Embedding in Vector Spaces. Pattern Recognition 43(4), 1642–1655 (2010)
12. Ferrer, M., Valveny, E., Serratosa, F.: Median graphs: A genetic approach based on new theoretical properties. Pattern Recognition 42(9), 2003–2012 (2009)
13. Ferrer, M., Valveny, E., Serratosa, F.: Median graph: A new exact algorithm using a distance based on the maximum common subgraph. Pattern Recognition Letters 30(5), 579–588 (2009)
14. Serratosa, F., Solé-Ribalta, A., Vidiella, E.: Graph Indexing and Retrieval Based on Median Graphs. In: Martínez-Trinidad, J.F., Carrasco-Ochoa, J.A., Kittler, J. (eds.) MCPR 2010. LNCS, vol. 6256, pp. 311–321. Springer, Heidelberg (2010)
15. Bunke, H., Munger, A., Jiang, X.: Combinatorial search versus genetic algorithms: A case study based on the generalized median graph problem. Pattern Recognition Letter 20, 1271–1277 (1999)
16. Solé-Ribalta, A., Serratosa, F.: Graduated Assignment Algorithm for Finding the Common Labelling of a Set of Graphs. In: Hancock, E.R., Wilson, R.C., Windeatt, T., Ulusoy, I., Escolano, F. (eds.) SSPR&SPR 2010. LNCS, vol. 6218, pp. 180–190. Springer, Heidelberg (2010)
17. Wong, A.K.C., You, M.: Entropy and distance of random graphs with application to structural pattern recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 7, 599–609 (1985)
18. Serratosa, F., Alquézar, R., Sanfeliu, A.: Function-described graphs for modeling objects represented by attributed graphs. Pattern Recognition 36(3), 781–798 (2003)
19. Serratosa, F., Alquézar, R., Sanfeliu, A.: Synthesis of Function-Described Graphs and clustering of Attributed Graphs. International Journal of Pattern Recognition and Artificial Intelligence 16(6), 621–655 (2002)
20. Sanfeliu, A., Serratosa, F., Alquézar, R.: Second-Order Random Graphs for modeling sets of Attributed Graphs and their application to object learning and recognition. Intern. Journal of Pattern Recognition and Artificial Intelligence 18(3), 375–396 (2004)
21. He, H., Singh, A.K.: Closure-Tree: An Index Structure for Graph Queries. In: Proc. International Conference on Data Engineering, p. 38 (2006)
22. 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)

23. Serratosa, F., Solé-Ribalta, A., Cortés, X.: K-nn Queries in Graph Databases Using M-Trees. In: Real, P., Diaz-Pernil, D., Molina-Abril, H., Berciano, A., Kropatsch, W. (eds.) CAIP 2011, Part I. LNCS, vol. 6854, pp. 202–210. Springer, Heidelberg (2011)
24. Fernández, A., Gómez, S.: Solving Non-uniqueness in Agglomerative Hierarchical Clustering Using Multidendrograms. Journal of Classification (25), 43–65 (2008)
25. 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)
26. Shasha, D., Wang, J.T.L., Giugno, R.: Algorithmics and applications of tree and graph searching. In: ACM SIGMOD-SIGACT-SIGART, pp. 39–52 (2002)
27. Lee, S.Y., Hsu, F.: Spatial Reasoning and Similarity Retrieval of Images using 2D C-Strings Knowledge Representation. Pattern Recognition 25(3), 305–318 (1992)
28. Lozano, M.A., Escolano, F., Bonev, B., Suau, P., Aguilar, W., Saez, J.A., Cazorla, M.A.: Region and constellations based categorization of images with unsupervised graph learning. Image and Vision Computing 27, 960–978 (2009)
29. Solé-Ribalta, A., Serratosa, F.: Graduated Models and Algorithms for computing the Common Labelling of a set of Attributed Graphs. In: CVIU, vol. 115 (7), pp. 929–945 (2011)
30. Solé-Ribalta, A., Serratosa, F.: A Probabilistic Framework to Obtain a Common Labelling between Attributed Graphs. In: Vitrià, J., Sanches, J.M., Hernández, M. (eds.) IbPRIA 2011. LNCS, vol. 6669, pp. 516–523. Springer, Heidelberg (2011)
31. Riesen, K., Bunke, H.: IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) S+SSPR 2008. LNCS, vol. 5342, pp. 287–297. Springer, Heidelberg (2008)