A Recursive Embedding Approach to Median Graph Computation

M. Ferrer[1], D. Karatzas[2], E. Valveny[2], and H. Bunke[3]

[1] Institut de Robòtica i Informàtica Industrial, UPC-CSIC
C. Llorens Artigas 4-6, 08028 Barcelona, Spain
mferrer@iri.upc.edu
[2] Centre de Visió per Computador, Universitat Autònoma de Barcelona
Edifici O Campus UAB, 08193 Bellaterra, Spain
{dimos,ernest}@cvc.uab.cat
[3] Institute of Computer Science and Applied Mathematics, University of Bern
Neubrückstrasse 10, CH-3012 Bern, Switzerland
bunke@iam.unibe.ch

Abstract. The median graph has been shown to be a good choice to infer a representative of a set of graphs. It has been successfully applied to graph-based classification and clustering. Nevertheless, its computation is extremely complex. Several approaches have been presented up to now based on different strategies. In this paper we present a new approximate recursive algorithm for median graph computation based on graph embedding into vector spaces. Preliminary experiments on three databases show that this new approach is able to obtain better medians than the previous existing approaches.

1 Introduction

Graphs are a powerful tool to represent structured objects compared to other alternatives such as feature vectors. For instance, a recent work comparing the representational power of such approaches under the context of web content mining has been presented in [1]. Experimental results show better accuracies of the graph-based approaches over the vector-based methods. Nevertheless, some basic operations such as computing the sum or the mean of a set of graphs, become very difficult or even impossible in the graph domain.

The mean of a set of graphs has been defined using the concept of the median graph. Given a set of graphs, the median graph [2] is defined as the graph that has the minimum sum of distances (SOD) to all graphs in the set. It can be seen as a representative of the set. Thus it has a large number of potential applications primarily enabling many classical algorithms for learning, clustering and classification typically used in the vector domain. However, its computation time increases exponentially both in terms of the number of input graphs and their size [3]. A number of algorithms for the median graph computation have been reported in the past [2,3,4,5], but, in general, they either suffer from a large complexity or they are restricted to specific applications.

A. Torsello, F. Escolano, and L. Brun (Eds.): GbRPR 2009, LNCS 5534, pp. 113–123, 2009.

In this paper we propose a new approximate method based on graph embedding in vector spaces. Graph embedding has been recently used as a way to map graphs into vector spaces [6] using the graph edit distance [7]. In this way we can combine advantages from both domains: we keep the representational power of graphs while being able to operate in a vector space. The median of the set of vectors obtained with this mapping can be easily computed in the vector space. Then, applying recursively the weighted mean of a pair of graphs [8] we go from the vector domain back to the graph domain and we obtain an approximation of the median graph from the obtained median vector. This is the main difference over other embedding-based methods for the median graph computation like [9], where they obtain a graph not corresponding to the median vector of the whole set but the median of just three graphs of the set. We have made experiments on three different graph databases. The underlying graphs have no constraints regarding the number of nodes and edges. The results show that our method obtains better medians, in terms of the SOD, that two other previous methods. With these results at hand, we can think of applying this new approach to the world of real graph-based applications in pattern recognition and machine learning. In addition, our procedure potentially allows us to transfer any machine learning algorithm that uses a median, from the vector to the graph domain.

The rest of this paper is organized as follows. First, the basic concepts are introduced in the next section. Then, we introduce in detail the concept of the median graph and the previous work for its computation in Section 3. In Section 4 the proposed method for the median computation is described. Section 5 reports a number of experiments and present results achieved with our method. Finally, in Section 6 we draw some conclusions.

2 Basic Definitions

2.1 Graph

Given L, a finite alphabet of labels for nodes and edges, a **graph** g is defined by the four-tuple $g = (V, E, \mu, \nu)$ where, V is a finite set of nodes, $E \subseteq V \times V$ is the set of edges, μ is the node labeling function ($\mu : V \longrightarrow L$) and ν is the edge labeling function ($\nu : V \times V \longrightarrow L$). The alphabet of labels is not constrained in any way. For example, L can be defined as a vector space (i.e. $L = \mathbb{R}^n$) or simply as a set of discrete labels (i.e. $L = \{\Delta, \Sigma, \Psi, \cdots\}$). Edges are defined as ordered pairs of nodes, that is, an edge is defined by (u, v) where $u, v \in V$. The edges are directed in the sense that if the edge is defined as (u, v) then $u \in V$ is the source node and $v \in V$ is the target node.

2.2 Graph Edit Distance

The basic idea behind the graph edit distance [7,10] is to define the dissimilarity of two graphs as the minimum amount of distortion required to transform one graph into the other. To this end, a number of distortion or edit operations e,

consisting of the insertion, deletion and substitution of both nodes and edges are defined. Given these edit operations, for every pair of graphs, g_1 and g_2, there exists a sequence of edit operations, or edit path $p(g_1, g_2) = (e_1, \ldots, e_k)$ (where each e_i denotes an edit operation) that transforms g_1 into g_2 (see Figure 1). In general, several edit paths exist between two given graphs. This set of edit paths is denoted by $\wp(g_1, g_2)$. To evaluate which edit path is the best, edit costs are introduced through a cost function. The basic idea is to assign a penalty (or cost) c to each edit operation according to the amount of distortion it introduces in the transformation. The edit distance between two graphs g_1 and g_2, $d(g_1, g_2)$, is the minimum cost edit path that transforms one graph into the other. Since the graph edit distance is a NP-complete problem, in this paper we will use suboptimal methods for its computation [11,12].

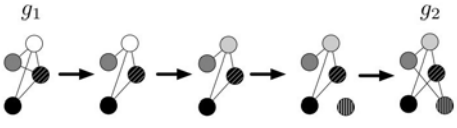

Fig. 1. Example of a possible edit path between two graphs, g_1 and g_2

3 Median Graph

Let U be the set of graphs that can be constructed using labels from L. 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 (SOD) 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 the generalized median graph can only be done in exponential time, both in the number of graphs in S and their size [2]. As a consequence, in real world applications we are forced to use suboptimal methods in order to obtain solutions for the generalized median graph in reasonable time. Such approximate methods [2,4,5,13] apply some heuristics in order to reduce the graph edit distance computation complexity and the size of the search space.

Another alternative is to use the set median graph instead of the generalized median graph. The difference is that, while the search space for the generalized median graph is U, that is, the whole universe of graphs, the search space for the set median graph is simply S, that is, the set of graphs in the given set. It makes the computation of set median graph exponential in the size of the graphs, due to the complexity of graph edit distance, but polynomial with respect to the number

of graphs in S. The set median graph is usually not the best representative of a set of graphs, but it is often a good starting point when searching the generalized median graph.

3.1 Median Graph via Embedding

Graph embedding aims to convert graphs into another structure, such as real vectors, and then operate in the associated space to facilitate certain graph-based tasks, such as matching and clustering.

In this paper we will use a new class of graph embedding procedures based on the selection of some prototypes and graph edit distance computation [6]. For the sake of completeness, we briefly describe this approach in the following.

Assume we have a set of training graphs $T = \{g_1, g_2, \ldots, g_n\}$ and a graph dissimilarity measure $d(g_i, g_j)$ $(i, j = 1 \ldots n;\ g_i, g_j \in T)$. Then, a set $P = \{p_1, \ldots, p_m\} \subseteq T$ of m prototypes is selected from T (with $m \leq n$). After that, the dissimilarity between a given graph of $g \in T$ and every prototype $p \in P$ is computed. This leads to m dissimilarity values, $d_1, \ldots, d_m$ where $d_k = d(g, p_k)$. These dissimilarities can be arranged in a vector $(d_1, \ldots, d_m)$. In this way, we can transform any graph of the training set T into an m-dimensional vector using the prototype set P.

Such kind of embedding has already been used for the approximate median graph computation [9]. The idea behind such an approach is to follow a three step process. Assuming that a set of n graphs $S = \{g_1, g_2, \ldots, g_n\}$ is given, in a first step every graph in S is embedded into a n-dimensional space, i.e. each graph becomes a point in $\mathbb{R}^n$ because in our case the set of prototypes P is the whole set S, and therefore there is no prototype selection. The second step consists of computing the median vector $\boldsymbol{M}$ of all the points obtained in the previous step. Finally, the resulting median vector has to be mapped back to an equivalent graph. This last step of mapping back from the vector space to the graph space presents a difficult problem for a number of reasons. To mention just two, depending on the embedding technique not every point in the (continuous) vector space corresponds to a graph. Secondly it might be that a particular vector presents a one to many relationship to graphs. For instance, to obtain the median graph, in [9] the three closest points to the computed median vector $\boldsymbol{M}$ are used to compute their own median $\boldsymbol{M'}$ (which always falls on the plane defined by them). Using these three points (corresponding to known graphs) and the new median $\boldsymbol{M'}$, the weighted mean approach [8] is used to recover a graph $\bar{g}'$ (corresponding to $\boldsymbol{M'}$), which is taken as an approximation of the median graph $\bar{g}$ of S.

In the next section we present a new recursive approach for computing the median graph for a given set of graphs based on the embedding procedure explained before. The aim of the presented approach is to obtain a graph corresponding to the actual median vector $\boldsymbol{M}$ of the whole set S. We show that, as expected, obtaining a graph corresponding to the real median vector $\boldsymbol{M}$ produces better medians (with a lower SOD to the graphs of the set), than using the graph corresponding to $\boldsymbol{M'}$ as in the approach of [9].

4 A Recursive Embedding Approach

As explained before the difficulty in using graph embedding to calculate the median graph is the mapping from vector space back to the graph space. Here we propose a recursive solution to the problem based on the algorithm of the weighted mean of a pair of graphs [8].

The weighed mean of two graphs g and g' is a graph g'' such that

$$d(g, g'') = a \tag{2}$$

$$d(g, g') = a + d(g'', g') \tag{3}$$

where a, with $0 \leq a \leq d(g, g')$, is a constant. That is, the graph g'' is a graph between the graphs g and g' along the edit path between them. Furthermore, if the distance between g and g'' is a and the distance between g'' and g' is b, then the distance between g and g' is $a + b$.

Assume that we can define a line segment in the vector space that connects two points P_1 and P_2 corresponding to the known graphs g_1 and g_2 such as the calculated median M lies on this line segment. We can then calculate the graph g_M corresponding to the median M as the weighted mean of g_1 and g_2. The problem is thus reduced to creating such a line segment in the vector space. We show here how this can be achieved by recursively applying the weighted mean of a pair of graphs.

Given a set of graphs $S = \{g_1, g_2, \ldots, g_n\}$, we use the graph embedding method described in Section 3.1 to obtain the corresponding n-dimensional points $\{P_1, P_2, \ldots, P_n\}$ in $\mathbb{R}^n$. As long as there are no identical graphs in the set S, the vectors $\boldsymbol{v_i} = \boldsymbol{(P_i - O)}$, where O is the origin of the n-dimensional space defined, will be linearly independent. This arises from the way the coordinates of the points were defined during graph embedding.

Once all the graphs have been embedded in the vector space, the median of the corresponding points is computed. To this end we use the concept of Euclidean Median using the Weiszfeld algorithm [14] as in the case of [9]. The Euclidean median has been chosen as the representative in the vector domain for two reasons. The first reason is that the median of a set of objects is one of the most promising ways to obtain the representative of such a set. The second is that, since the median graph is defined in a very close way to the median vector, we expect the median vector to represent accurately the vectorial representation of the median graph, and then, from the median vector to obtain good median graphs.

Given a set of n linearly independent points in $\mathbb{R}^n$ we can define a hyperplane H_{n-1} of dimensionality *n-1* (e.g. in the case of *n=2*, two points define a unique 1D line, in the case of *n=3*, three points define a unique 2D plane, etc). The normal vector N_{n-1} of the hyperplane H_{n-1} can be calculated from the following set of equations:

$$\begin{aligned}
(\boldsymbol{P_n} - \boldsymbol{P_1}) \cdot \boldsymbol{N_{n-1}} &= 0 \\
(\boldsymbol{P_n} - \boldsymbol{P_2}) \cdot \boldsymbol{N_{n-1}} &= 0 \\
&\vdots \\
(\boldsymbol{P_n} - \boldsymbol{P_{n-1}}) \cdot \boldsymbol{N_{n-1}} &= 0 \\
\|\boldsymbol{N_{n-1}}\| &= 1
\end{aligned} \tag{4}$$

The Euclidean median M_n of these n points will always fall on the hyperplane H_{n-1}. Moreover it will fall within the volume of the *n-1* dimensional simplex with vertices P_i. For *n=4* this is visualised in Figure 2(a). This figure shows the hyperplane H_3 defined by the 4 points $P_i = \{P_1, P_2, P_3, P_4\}$. The Euclidean median M_4 falls in the 3D space defined by the 4 points and specifically within the pyramid (3D simplex) with vertices P_i.

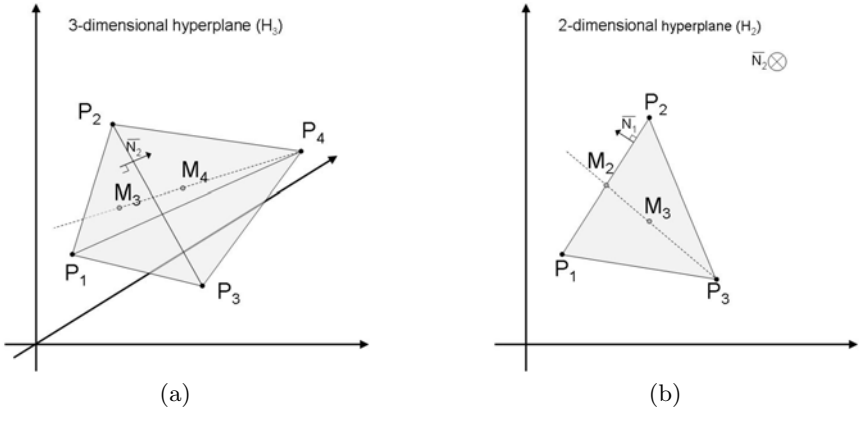

Fig. 2. *a)* The 3D hyperplane defined given four 4D points $\{P_1, P_2, P_3, P_4\}$. *b)* The 2D hyperplane defined by the remaining points $\{P_1, P_2, P_3\}$.

Without loss of generality we can choose any one of the points, say P_n, and create the vector $(\boldsymbol{M_n} - \boldsymbol{P_n})$. This vector will lie fully on the hyperplane H_{n-1}. As mentioned before, in order to use the weighted mean between of a pair of graphs to calculate the graph corresponding to M_n we need to first find a point (whose corresponding graph is known) that lies on the line defined by the vector $(\boldsymbol{M_n} - \boldsymbol{P_n})$, and specifically on the ray extending M_n (so that M_n lies between P_n and the new point).

Let's call H_{n-2} the hyperplane of dimensionality *n-2* defined by the set of points $\{P_1, P_2, \ldots, P_{n-1}\}$, that is all the original points except P_n. Then the intersection of the line defined by the vector $(\boldsymbol{M_n} - \boldsymbol{P_n})$ and the new hyperplane H_{n-2} will be a single point. For the running example of *n=4* this point (M_3) would be the point of intersection of the line $\boldsymbol{P_4} - \boldsymbol{M_4}$ and the 2D plane H_2 defined by the remaining points $\{P_1, P_2, P_3\}$ (see Figure 2(a)).

For the normal vector N_{n-2} of the hyperplane H_{n-2} we can create the following set of *n-1* equations in a similar fashion as before:

$$\begin{aligned} (\boldsymbol{P_{n-1} - P_1}) \cdot \boldsymbol{N_{n-2}} &= 0 \\ (\boldsymbol{P_{n-1} - P_2}) \cdot \boldsymbol{N_{n-2}} &= 0 \\ &\vdots \\ (\boldsymbol{P_{n-1} - P_{n-2}}) \cdot \boldsymbol{N_{n-2}} &= 0 \\ \|\boldsymbol{N_{n-2}}\| &= 1 \end{aligned} \tag{5}$$

Furthermore, we ask that N_{n-2} is perpendicular to N_{n-1} (i.e. it falls within the hyperplane H_{n-1}):

$$\boldsymbol{N_{n-1}} \cdot \boldsymbol{N_{n-2}} = 0 \tag{6}$$

Equations 5 and 6 provide us a set of n equations to calculate N_{n-2}.

Suppose M_{n-1} is the point of intersection of the line defined by $\boldsymbol{(M_n - P_n)}$ and the hyperplane H_{n-2}, then for this point it should be:

$$M_{n-1} = P_n + \alpha\,(M_n - P_n) \tag{7}$$

$$(P_{n-1} - M_{n-1}) \cdot N_{n-2} = 0 \tag{8}$$

Solving the above equations for a, we have:

$$\alpha = \frac{N_{n-2} \cdot (P_{n-1} - P_n)}{N_{n-2} \cdot (M_n - P_n)} \tag{9}$$

Substituting back to 7 we obtain the point M_{n-1}.

We can now follow exactly the same process as before, and assume a new line defined by the vector $\boldsymbol{(M_{n-1} - P_{n-1})}$. Again we can define as M_{n-2} the point of intersection of the above line with the *n-3* dimensional hyperplane H_{n-3} which is defined by the *n-2* points: $\{P_1, P_2, \ldots, P_{n-2}\}$. As an example see Figure 2(b) for *n=4*. In this figure the point M_2 is defined as the intersection of the line defined be $\boldsymbol{(M_3 - P_3)}$ and the 1D hyperplane (line) H_1 defined by the remaining points $\{P_1, P_2\}$.

In the generic case the set of n equations needed to calculate the normal vector N_k of the k dimensional hyperplane H_k are:

$$\begin{aligned} (\boldsymbol{P_{k+1} - P_1}) \cdot \boldsymbol{N_k} &= 0 \\ (\boldsymbol{P_{k+1} - P_2}) \cdot \boldsymbol{N_k} &= 0 \\ &\vdots \\ (\boldsymbol{P_{k+1} - P_k}) \cdot \boldsymbol{N_k} &= 0 \\ \boldsymbol{N_{n-1}} \cdot \boldsymbol{N_k} &= 0 \\ \boldsymbol{N_{n-2}} \cdot \boldsymbol{N_k} &= 0 \\ &\vdots \\ \boldsymbol{N_{k+1}} \cdot \boldsymbol{N_k} &= 0 \\ \|\boldsymbol{N_{k+1}}\| &= 1 \end{aligned} \tag{10}$$

Based on eq. 7, 8 and 9, in the generic case the point M_k can be computed recursively from:

$$M_k = P_{k+1} + \alpha\,(M_{k+1} - P_{k+1}) \tag{11}$$

Where:

$$\alpha = \frac{N_{k-1} \cdot (P_k - P_{k+1})}{N_{k-1} \cdot (M_{k+1} - P_{k+1})} \tag{12}$$

This process is recursively applied until M_2 is sought. The case of M_2 is solvable using the weighted mean of a pair of graphs, as M_2 will lie on the line segment defined by P_1 and P_2 which correspond to known graphs (see Figure 2(b)).

Having calculated M_2 the inverse process can be followed all the way up to M_n. In the next step M_3 can be calculated as the weighted mean of the graphs corresponding to M_2 and P_3. Generally the graph corresponding to the point M_k will be given as the weighted mean of the graphs corresponding to M_{k-1} and P_k. The weighted mean algorithm can be applied continuously until the graph corresponding to M_n is calculated, which is the median graph of the set.

It is important to note that the order of consideration of the points will affect the final solution arrived at. As a result it is possible that one of the intermediate solutions along the recursive path produces a lower SOD to the graphs of the set than the final solution. Thus, the results reported here are based on the best intermediate solutions.

5 Experiments

In this section we provide the results of an experimental evaluation of the proposed algorithm. To this end we have used three graph databases representing Letter shapes, Webpages and Molecules. Table 1 show some characteristics of each dataset. For more information of these databases see [15].

To evaluate the quality of the proposed method, we propose to compare the SOD of the median calculated using the present method (RE) taking the best intermediate solution to the SOD of the median obtained using other existing methods, namely the set median (SM) and the method of [9] (TE). For every database we generated sets of different sizes as shown in Table 1. The graphs in each set were chosen randomly from the whole database. In order to generalize the results, we generated 10 different sets for each size.

Results of the mean value of the SOD over all the classes and repetitions for each dataset are shown in Figure 3. Clearly, the lower the SOD, the better the

Table 1. Summary of dataset characteristics, viz. the size, the number of classes (# classes), the average size of the graphs (∅ nodes) and the sizes of the sets

Database	**Size**	**# classes**	**∅ nodes**	**Number of Graphs in S**
Letter	2,250	15	4.7	15, 25, 35, ..., 75
Webpages	2,340	6	186.1	15, 25, 35, ..., 75
Molecules	2,000	2	15.7	10, 20, 30, ..., 100

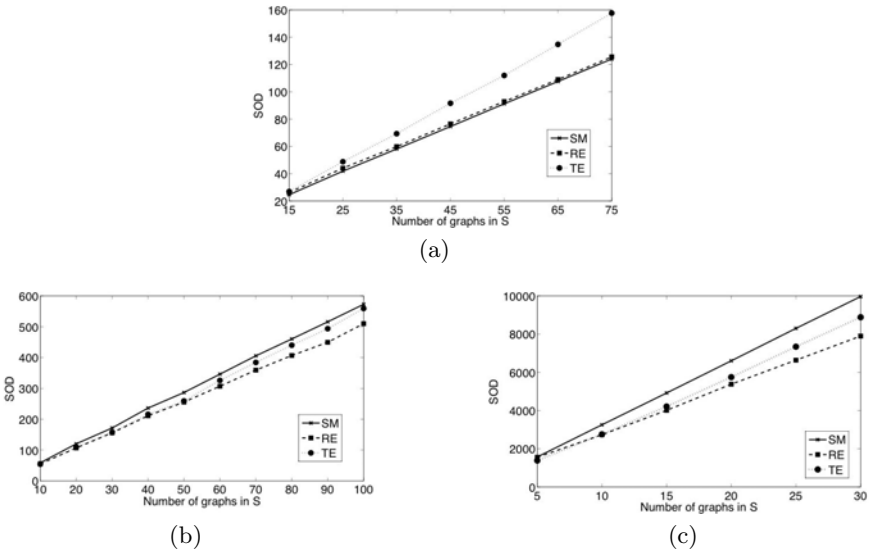

Fig. 3. SOD evolution for the three databases. a) Letter, b) Molecule and c) Webpage.

result. Since the set median graph is the graph belonging to the training set with minimum SOD, it is a good reference to evaluate the median graph quality.

As we can see, the results show that in all cases we obtain medians with lower SOD than those obtained with the TE method. In addition, in two cases (Web and Molecule) we also obtain better results than the SM method. In the case of the Letter database, we obtain slightly worse results than the SM method but quite close to that. Nevertheless, our results do not diverge from the results of the SM method as in the case of the TE method, which means that our proposed method is more robust against the size of the set. With these results we can conclude that our method finds good approximations of the median graph.

6 Conclusions

In the present paper we have proposed a novel technique to obtain approximate solutions for the median graph. This new approach is based on graph embedding into vector spaces. First, the graphs are mapped to points in n-dimensional vector space using the graph edit distance paradigm. Then, the crucial point of obtaining the median of the set is carried out in the vector space, not in the graph domain, which simplifies dramatically this operation. Finally, we proposed a recursive application of the weighted mean of a pair of graphs to obtain the graph corresponding to the median vector. This embedding approach allows us to exploit the main advantages of both the vector and graph representations, computing the more complex parts in real vector spaces but keeping the representational power of graphs. Results on three databases, containing a high

number of graphs with large sizes, show that the medians obtained with our method are, in general, better that those obtained with other methods, in terms of the SOD. For datasets such ones used in this paper, the generalized median could not be computed before, due to the high computational cost of the existing methods. These results show that with this new procedure the median graph can be potentially applied to any application where a representative of a set is needed. Nevertheless, there are still a number of issues to be investigated. For instance, the order in which the points are taken becomes an important topic to be further studied in order to improve the results of the method.

Acknowledgements

This work has been supported by the Spanish research programmes Consolider Ingenio 2010 CSD2007-00018, TIN2006-15694-C02-02 and TIN2008-04998, the fellowship 2006 BP-B1 00046 and the Swiss National Science Foundation Project 200021-113198/1.

References

1. Schenker, A., Bunke, H., Last, M., Kandel, A.: Graph-Theoretic Techniques for Web Content Mining. World Scientific Publishing, USA (2005)
2. Jiang, X., Münger, A., Bunke, H.: On median graphs: Properties, algorithms, and applications. IEEE Trans. Pattern Anal. Mach. Intell. 23(10), 1144–1151 (2001)
3. Münger, A.: Synthesis of prototype graphs from sample graphs. Diploma Thesis, University of Bern (1998) (in German)
4. Hlaoui, A., Wang, S.: Median graph computation for graph clustering. Soft Comput. 10(1), 47–53 (2006)
5. Ferrer, M., Serratosa, F., Sanfeliu, A.: Synthesis of median spectral graph. In: Marques, J.S., Pérez de la Blanca, N., Pina, P. (eds.) IbPRIA 2005. LNCS, vol. 3523, pp. 139–146. Springer, Heidelberg (2005)
6. Riesen, K., Neuhaus, M., Bunke, H.: Graph embedding in vector spaces by means of prototype selection. In: Escolano, F., Vento, M. (eds.) GbRPR 2007. LNCS, vol. 4538, pp. 383–393. Springer, Heidelberg (2007)
7. Bunke, H., Allerman, G.: Inexact graph matching for structural pattern recognition. Pattern Recognition Letters 1(4), 245–253 (1983)
8. Bunke, H., Günter, S.: Weighted mean of a pair of graphs. Computing 67(3), 209–224 (2001)
9. Ferrer, M., Valveny, E., Serratosa, F., Riesen, K., Bunke, H.: An approximate algorithm for median graph computation using graph embedding. In: Proceedings of 19th ICPR, pp. 287–297 (2008)
10. Sanfeliu, A., Fu, K.: A distance measure between attributed relational graphs for pattern recognition. IEEE Transactions on Systems, Man and Cybernetics 13(3), 353–362 (1983)
11. 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)

12. Riesen, K., Neuhaus, M., Bunke, H.: Bipartite graph matching for computing the edit distance of graphs. In: Escolano, F., Vento, M. (eds.) GbRPR 2007. LNCS, vol. 4538, pp. 1–12. Springer, Heidelberg (2007)
13. White, D., Wilson, R.C.: Mixing spectral representations of graphs. In: 18th International Conference on Pattern Recognition (ICPR 2006), Hong Kong, China, August 20-24, pp. 140–144. IEEE Computer Society, Los Alamitos (2006)
14. Weiszfeld, E.: Sur le point pour lequel la somme des distances de n points donnés est minimum. Tohoku Math. Journal (43), 355–386 (1937)
15. Riesen, K., Bunke, H.: IAM graph database repository for graph based pattern recognition and machine learning. In: SSPR/SPR, pp. 287–297 (2008)