Application of Graph Embedding to solve Graph Matching Problems

E. Valveny, M. Ferrer

Application of Graph Embedding to solve Graph Matching Problems

Ernest Valveny[1] – Miquel Ferrer[1]

Centre de Visió per Computador, Dep. Ciències de la Computació
Universitat Autònoma de Barcelona
Edifici O, Campus UAB, 08193 Bellaterra (Barcelona), Spain

{ernest,mferrer@cvc.uab.cat}

Résumé : *Graphs have very interesting properties for object representation in pattern recognition. However, graph matching algorithms are usually computationally complex. In addition, graphs are harder to manipulate and operate than feature vectors. In the last years, some attempts have been made to combine the best of the graph and the vector domains in order to get the advantages of both worlds. In this paper we review some of these works on graph kernels and graph embedding and we show how graph embedding can be used to obtain accurate and efficient approximations of the median graph. The median graph can be seen as the representative of a set of graphs but its application has been very limited up to now due to computational reasons. With this new approach, we can obtain an approximate median graph using real databases containing large graphs.*

Mots-clés : Graph Matching, Graph Embedding, Graph Kernels, Vector Spaces, Median Graph

1 Introduction

Graphs have been shown as a useful tool for object representation in structural pattern recognition. The power of graphs lies in the fact that they can represent objects in terms of their parts (as nodes) and the relation between these parts (as edges). In addition, we can potentially include any kind of information both in the nodes and in the edges. This is a clear advantage over pattern representations based on feature vectors, which are restricted to the use of unary values. That is, for each object, a set of relevant properties, or features, are computed and arranged in a vector form without any explicit relation between them. Furthermore, the dimensionality of graphs, that is, the number of nodes and edges, can be different for every object even for objects of the same class. Thus, the more complex an object is, the larger the number of nodes and edges can be. This is in contrast to feature vectors where, regardless of the complexity of the object, they have always the same length and structure (a simple list of pre-determined components).

Nevertheless, there are a number of drawbacks with the use of graphs in structural pattern representation. On the one hand, the computational complexity of the algorithms related to graphs is usually high. For instance, the simple task of comparing two graphs, which is commonly referred as graph matching, becomes exponential in the size of graphs, while the computation of the Euclidean distance between two vectors has linear dependence on the size of the involved vectors. Secondly, the repository of algorithmic tools based on graphs is quite limited when compared to the tools available for patterns represented using feature vectors. This is mainly due to the fact that vectors are simple structures with good mathematical properties that can be readily manipulated algebraically.

For this reason, new trends in structural pattern recognition have been proposed merging both worlds in order to extend the available statistical tools to the graph domain [BUN 05]. In this way, graph kernels permit to compute the dot product of the representation of a pair of graphs in a vector space without having to define the explicit transformation between the graphs and the vector space. As a consequence all classification algorithms based on the computation of a dot product, such as Support Vector Machines (SVM) become immediately available for graphs. On the other hand, graph embedding aims to find an explicit transformation between graphs and a vector space. In this way, we can give a semantic interpretation to this transformation. In addition we can also manipulate the vectors resulting from this transformation with all the mathematical machinery that can be applied to vectors. We are not restricted to the dot product.

In this paper, we firstly review the main techniques used to define graph kernels and graph embedding in sections 2 and 3, respectively. Then, in section 4 we show the application of graph embedding to a particular complex graph matching problem : the computation of the median graph. In this section we introduce the concept of median graph as a representative of a set of graphs and then, we describe how it can be efficiently computed using graph embedding. We also show some results of its application to real pattern recognition problems. Finally, in section 5 we state some conclusions and point out some challenges for the future.

2 Graph Kernels

In the last years, there has been an increasing interest in the pattern recognition community for kernel methods[SHA 04]. Kernel methods are based on the formulation of the classification problem in terms of the dot product between two patterns. There exist many pattern recognition algorithms that can be expressed in this way, such as support sector machines, principal component analysis, fisher discriminant analysis or the gaussian mixture modeling. In addition, it has been shown in [COV 65] that a non-linear classification problem is more likely to become linearly separable if we map it to a high dimensional space. A kernel function permits to combine both ideas permitting to compute the dot product in the high dimensional space without need to explicitly define the mapping between the two spaces. They have been shown to outperform other types of classifiers.

A kernel function is basically a similarity function defined in the original pattern space that satisfies the conditions of symmetry and positive definiteness. Under these conditions, it is shown that there exists a pattern space where the kernel function can be interpreted as the dot product of the mapping of the original patterns in the new space. Therefore, we can operate with the dot product in the new space by simply applying the kernel function to the patterns in the original space without need to explicitly find the mapping between the two spaces.

This general procedure can also be applied to graphs. All we need is the definition of a similarity function in the graph domain satisfying the properties of symmetry and positive definiteness. As a result we will be able to compute the dot product in a new pattern space. Therefore, all the machine learning algorithms based on the computation of a dot product (originally developed to be used with feature vectors) become immediately available in the graph domain.

In the literature several alternatives have been proposed to define kernel functions on graphs. A common approach is based on generating random walks on the graphs[KAS 02, KAS 03]. The kernel measures the number of random walks in both graphs that have all or some labels in common. Another family of graph kernels are convolution kernels[HAU 99, BOR 05] where the kernel is obtained as the composition of a set of kernels defined on subparts of the graph, such as shortest paths or edit paths. Diffusion kernels[KON 02] are exponential kernels defined in analogy to the equation of diffusion of the heat of classical physics and based on the Laplacian matrix of the graph. Finally, another family of graph kernels is based on the graph edit distance[NEU 06].

All these graph kernels have been used to extend graph classification to classical classification algorithms, such as SVM, showing a good performance in a number of applications, such as protein classification, shape recognition, fingerprint verification, digit recognition, etc.

3 Graph Embedding

Graph kernels permit to compute the dot product of a pair of graphs in a vector space without having to explicitly define the transformation between the graphs and the vectors. Alternatively, graph embedding aims to explicitly find a mapping between graphs and real vectors in order to be able to operate in the associated space, making easier some typical graph-based tasks, such as matching and clustering. In this way, we can try to define embeddings with a semantic interpretation appropriate to obtain a suitable representation for a particular application. In addition, we can directly manipulate the vector representation of the graphs and we can expand the range of operations that we can compute in the vector domain. We are not restricted to the dot product any more. An example of the usefulness of graph embedding is the computation of the median graph that we will present in the next section. We will show how the embedding permits to compute a median vector in the vector domain and then, using the weighted mean of a pair of graphs we are able to recover the corresponding graph in the graph domain.

Different graph embedding procedures have been proposed in the literature so far. Some of them are based on the spectral graph theory. Others take advantage of typical similarity measures to perform the embedding tasks. In the following, a brief review of some strategies for graph embedding will be outlined.

Several embedding procedures are based on the spectral graph theory. Spectral graph theory is based on the analysis of the spectral decomposition into eigenvalues and eigenvectors of the adjacency matrix or the Laplacian matrix of a graph. The Laplacian matrix is obtained by substracting the adjacency matrix to a diagonal matrix containing the degree of every node of the graph. The spectrum of these matrices conveys interesting properties about the structure and the topology of the graph. This is the reason of using it as the basis to convert graphs into vectors.

A relatively early approach based on the adjacency matrix of a graph is proposed in [LUO 03]. In this work, graphs are converted into a vector representation using some spectral features extracted form the adjacency matrix of a graph. Then, these vectors are embedded into eigenspaces with the use of the eigenvectors of the covariance matrix of the vectors. This approach is then used to perform graph clustering experiments. Another similar approach have been presented in [WIL 05]. This work is similar to the previous one, but in this case they use the coefficients of some symmetric polynomials constructed from the spectral features of the Laplacian matrix, to represent the graphs into a vectorial form. On a recent approach [ROB 07], the idea is to embed the nodes of a graph into a metric space and view the graph edge set as geodesics between pairs of points in a Riemannian manifold. This can be done using the Laplace-Beltrami operator and the Laplacian matrix. Then, the problem of matching the nodes of a pair of graphs is viewed as the alignment of the embedded point sets. In another work[SHO 05] the goal is to obtain a signature to describe shapes using the recursive spectral decomposition of the shock graph representing the skeleton of the shape.

A different approach[BAI 04] is based on applying MDS to a matrix of shortest geodesic distances between nodes of the graph. The embedding is then used for graph matching. For the special case of trees, an embedding has been defined using the super-tree of a set of sample trees[TOR 07]. Then, each tree is embedded in a vector where each component is related to one of the nodes of the super-tree and it only has a value different from zero if the node belongs to the specific tree. The method is used in shape analysis using shock trees extracted from the skeletons of 2D shapes. Random walks, and particularly quantum walks have also been used to embed the nodes of a graph in a vector space[EMM 07]. In this case the embedding is based on the commute time, the expected time for the walk to travel between two nodes.

Another class of graph embedding procedures is based on the selection of some prototypes and the computation of the graph edit distance between the graph and the set of prototypes. This approach was first presented in [RIE 07], and it relies on the work proposed in [PEK 06]. The basic intuition of this work is that the description of the regularities in observations of classes and objects is the basis to perform pattern classification. Thus, from the selection of concrete prototypes, each point is embedded into a vector space by taking its distance to all these prototypes. Assuming these prototypes have been appropriately chosen, each class will form a compact zone in the vector space. An extension to map string representations into vector spaces using a similar approach was later proposed in [SPI 06].

4 Median Graph via Embedding

In this section we will show a particular application of graph embedding using the embedding method introduced in [RIE 07]. We will see how we can obtain good approximations of the median graph manipulating the representation obtained in the vector domain and then, finding the corresponding graph in the graph domain. First, we will briefly introduce the concept of the median graph. Then, we will explain the overall schema of the procedure and we will describe each of the three steps in which it can be decomposed.

4.1 Median Graph

Given a set of graphs, the concept of median graph has been presented as a useful tool to compute a representative of such a set. Let U be the set of graphs that can be constructed using a given set of labels L. Given $S = \{g_1, g_2, ..., g_n\} \subseteq U$, the **generalized median graph** of S is defined as the graph $\bar{g} \in U$ such that its sum of distances (SOD) to all the graphs in S is minimum :

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

The computation of the median graph is not straightforward as all the possible combinations of graphs need to be explored. This makes it exponential in the number and size of graphs. The existing exact and approximate algorithms can only be applied to small sets of graphs with a very few number of nodes.

4.2 General schema of the embedding procedure

In the last section we have shown that the computation of the generalized median graph is a rather complex task. In this section we present the general overview of a novel approach for the approximate computation of the median graph that is faster and more accurate than previous approximate algorithms. It is based on graph embedding in a vector space and it consists of three main steps.

Given a set $S = \{g_1, g_2, ..., g_n\}$ of n graphs, the first step is to embed every graph in S into the n-dimensional space of real numbers, i.e. each graph becomes a point in $\mathbb{R}^n$. The second step consists of computing the median vector using the points obtained in the previous step. Finally, to go from the vector space back to the graph domain, converting the median vector into a graph. The resulting graph is taken as the median graph of S. These three steps are depicted in Figure1.

In the following subsections, these three main steps will be further explained.

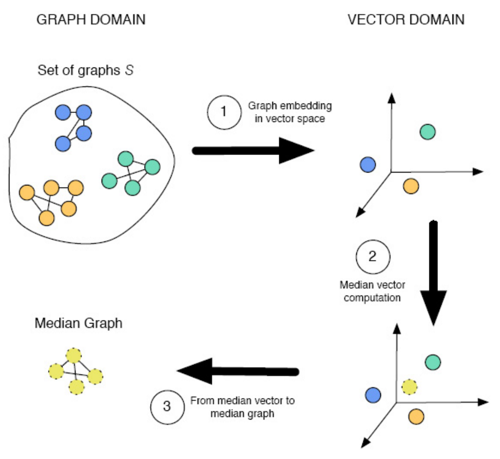
GRAPH DOMAIN
VECTOR DOMAIN
Set of graphs S
1
Graph embedding in vector space
2
Median vector computation
Median Graph
3
From median vector to median graph

FIG. 1 – Overview of the embedding approach.

4.3 Graph Embedding in a Vector Space

As we have shown in Section 3, several techniques for embedding graphs into vector spaces have been proposed. We have used a variation of the novel procedure proposed in [RIE 07] to perform our particular graph embedding.

This procedure is based on computing the distance between every graph and a set of prototypes. In our case, the set of prototypes is exactly the same set of training graphs that are used to compute the median graph. So, we must compute the distance between every pair of graphs in the set S. These distances are arranged in a distance matrix. Each row (column) of the matrix can be seen as an n-dimensional vector. Since each row (column) of the distance matrix is assigned to one graph, such an n-dimensional vector is the vectorial representation of the corresponding graph.

Figure 2 illustrates this procedure. In this example, it is assumed that the first row in the matrix corresponds to the distances of the blue graph in the set to all the graphs in S. This first row is interpreted as an n-dimensional point in a vector space (this n-dimensional space is represented here as a 3D space for obvious reasons).

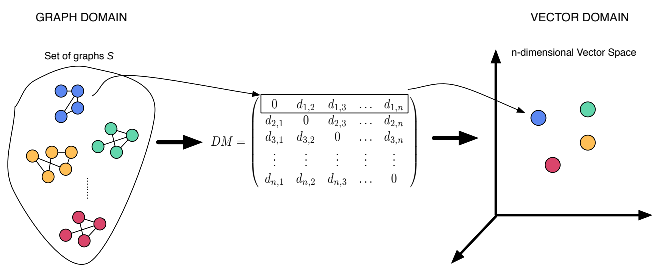

FIG. 2 – Detail of the first step (Graph embedding).

At the end, each graph in S has a corresponding point (n-dimensional vector) in the vector space. What is important to remark here is the meaning of each position in this vector. If a vector $\overrightarrow{v}_i$ corresponds to the graph $g_i \in S$, then the $j-th$ coordinate (with $j = 1 \dots n$) of this vector is the distance from the graph g_i to the graph g_j, that is $d(g_i, g_j)$.

4.4 Median Vector Computation

Once all the graphs have been embedded in the vector space, the median vector is computed. To this end we use the concept of *Euclidean Median*. Given a set X, the *Euclidean Median* is a point $y \in \mathbb{R}^n$ that minimizes the sum of the Euclidean distances to all the points in the set. 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.

The Euclidean median cannot be calculated in a straightforward way. The exact location of the Euclidean median can not be found when the number of elements in X is greater than 5 [BAJ 88]. No algorithm in polynomial time is known, nor has the problem been shown to be NP-hard [HAK 00]. In this work we have used the most common approximate algorithm for the computation of the Euclidean median, that is, the Weiszfeld's algorithm [WEI 37]. It is an iterative procedure that converges to the Euclidean median. To this end, the algorithm first selects an initial estimate solution y (this initial solution is often chosen randomly). Then, the algorithm defines a set of weights that are inversely proportional to the distances from the current estimate to the samples, and creates a new estimate that is the weighted average of the samples according to these weights. The algorithm may finish when a predefined number of iterations is reached, or under some other criteria, such as that the difference between the current estimate and the previous one is less than a established threshold.

4.5 Back to the Graph Domain

In order to obtain the median graph, the last step is to transform the Euclidean median into a graph. Such a graph will be considered as an approximation of the median graph of the set S. To this end we will use two different procedures based on the weighted mean of a pair of graphs [BUN 01] and the edit path between two given graphs. For the sake of completeness the definition of the weighted mean of a pair of graphs is included here.

Definition (Weighted Mean of a Pair of Graphs) Let g and g' be graphs. The weighed mean of 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 in 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 $a+b$. Figure 3 illustrates this idea.

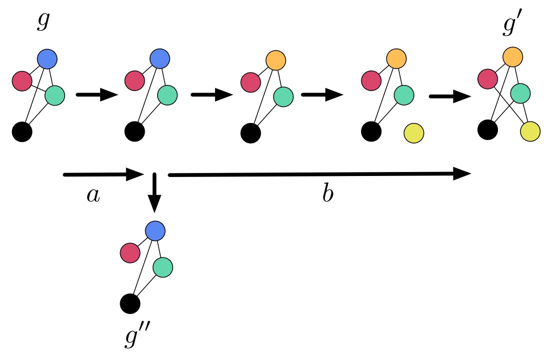

FIG. 3 – Example of the weighted mean of a pair of graphs

We have proposed two different strategies to obtain the median graph from the median vector based on the weighted mean of a pair of graphs. The first approach (called *Linear Interpolation Procedure* or *AELI*) uses two points in the vector space to recover the median graph. It is illustrated in Figure 4. The idea is the following : once the median vector v_m is computed, we choose only its two closest (v_1 and v_2 in Figure 4(a)) points. Then, we compute the median vector of these two points obtaining v'_m (Figure 4(b)). This point v'_m is used to obtain the approximate median graph. To this end, we first compute the distance of each point to v'_m (Figure 4(c)), and then, with these distances we apply the weighted mean of a pair of graphs to obtain g'_m, the approximate median graph (Figure 4(d)).

The second approach (called *Triangulation Procedure* or *AET*) uses the three closest points to the median vector. In this case the procedure to recover the median graph is very similar to that explained before. The only difference is that the median is obtained by triangulation among the three points.

4.6 Experiments

In this section we provide some results of the experimental evaluation of the proposed algorithm. We will show that the median graph obtained with this approach is a good approximation of the real median graph and that the median graph can then be used to extend the classification methods used in the graph domain.

To this end we have a database containing 2340 graph-based representations of web-pages of 6 different classes according to their contents (Business, Health, Politics, Sports, Technology, Entertainment). Nodes of the graph correspond to the most frequently occurring form of the most significant words that appear in the web-page, attributed with their frequency of appearance. Edges represent that two words are adjacent in the document. It has to be noted the large number of graphs in the database and that the size of the graphs is also large, with around 180 nodes in average. The existing methods to compute the median graph could only be applied to much smaller sets of graphs.

In the first experiment (Figure 5) we have computed the

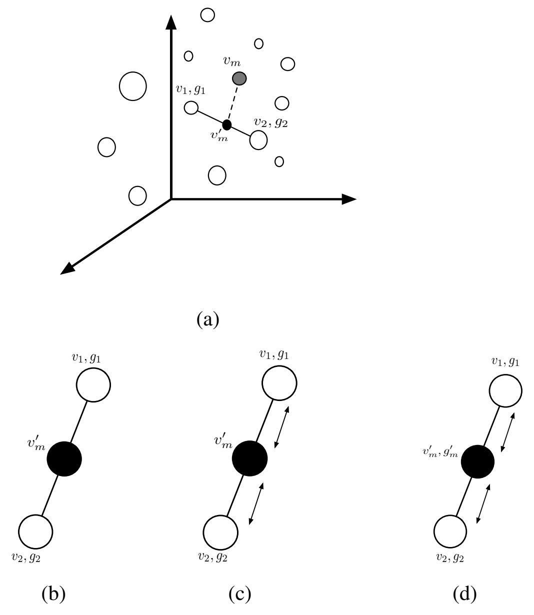

FIG. 4 – Illustration of the linear interpolation procedure.

median graph with an increasing number of graphs in the training set and we have compared the sum of distances (SOD) to all the graphs in the set with the SOD of the set median graph (SM). The set median graph is the graph in the set with minimum SOD and it is usually a good reference to evaluate the accuracy of the median graph. We can see that the SOD of the approximate generalized median is lower than the SOD of the set median for any number of graphs in S. Intuitively, this result could mean that the obtained median is "located" more accurately in the center of the class than the set median. So, we can conclude that our method achieves, in general, good approximations of the generalized median graph, independently of the size of the training set. That means that the generalized median adapts well to the increasing variability and distortion as the number of graphs in the training set increases.

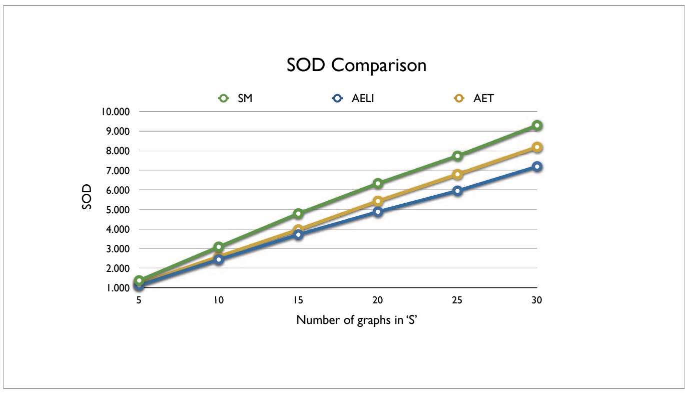

FIG. 5 – SOD evolution on the Webpage dataset

In the second experiment we have applied the median graph to improve the classification of the webpages in the framework of a Nearest-Neighbor classifier. The median graph

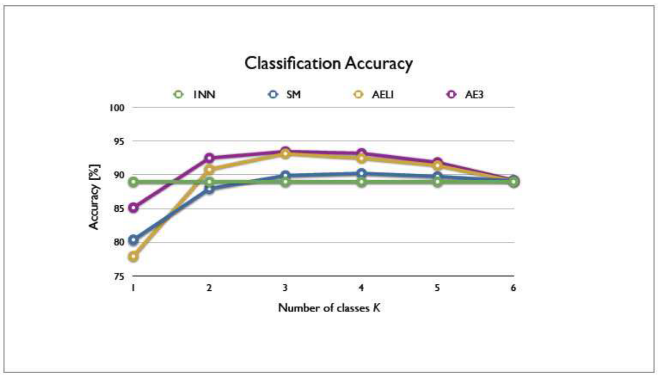

FIG. 6 – SOD evolution on the GREC dataset

is used to obtain a representative of each class. Then, these representatives are used as a filter to reduce the number of classes considered in the Nearest-Neighbor classifier. First, we compare every unknown pattern with the median graph of all the classes. Then, we consider the best K classes according to the distance to the median graph and we perform a Nearest-Neighbor classification using only the elements that belong to the selected K classes. In Figure 6 we show the results with an increasing value of K compared to the reference Nearest-Neighbor classifier. Obviously, for $K = 6$, the results are the same as we are using the elements of all the classes. However, we can see how for smaller values of K we still obtain better results than the Nearest-Neighbor classifier, but requiring a much smaller number of graph comparisons (approximately one third for $K = 2$). This is an important issue when working with graphs as the distance between two graphs is computationally demanding.

5 Conclusions

In this paper we have reviewed several alternatives to combine the graph and the vector domain that have been proposed in the last years in order to keep the advantages of both worlds : the power of representation of graphs and the easiness of manipulation and computation of vector spaces.

In this sense we have briefly analyze the main methods developed in the areas of graph kernels and graph embedding. Both techniques show a promising potential to improve and extend the use of graph-based representations to a large number of classical classification and clustering algorithms in pattern recognition.

As an example we have presented an application of one of the graph embedding approaches to the computation of the median graph. The median graph is useful as a representative of a set of graphs, but its computation is very complex. With the new method we have been able to compute the median graph using a large database of graphs representing webpages with a large number of nodes in each graph. Then, the median has been used to improve the performance of the classical Nearest-Neighbor classifier.

With the use of graph kernels and graph embedding the door to the application of a large number of machine learning algorithms to the graph domain is open. The challenge is how we can take full advantage of it, investigating the relation between these methods and existing classification algorithms in order to find the best kernels and embedding approaches.

Acknowledgement

Work partially supported by the Spanish project TIN2006-15694-C02-02, and by the Spanish research programme Consolider Ingenio 2010 : MIPRCV (CSD2007-00018)

Références

[BAI 04] BAI X., YU H., , HANCOCK E. R., Graph Matching using Spectral Embedding and Alignment, *17th Internarional Conference on Pattern Recognition*, 2004, pp. 398-401.

[BAJ 88] BAJAJ C., The algebraic degree of geometric optimization problems, *Discrete Comput. Geom.*, vol. 3, n^o 2, 1988, pp. 177–191, Springer-Verlag New York, Inc.

[BOR 05] BORGWARDT K., KRIEGEL H.-P., Shortest-path kernels on graphs, *Proc. 5th Int. Conf. on Data Mining*, 2005, pp. 74-81.

[BUN 01] BUNKE H., GÜNTER S., Weighted Mean of a Pair of Graphs, *Computing*, vol. 67, n^o 3, 2001, pp. 209-224.

[BUN 05] BUNKE H., IRNIGER C., , NEUHAUS M., Graph Matching : Challenges and Potential Solutions, ROLI F., VITULANO S., Eds., *ICIAP 2005*, vol. 3617 de *lncs*, pp. 1–10, sv, 2005.

[COV 65] COVER T., Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition, *IEEE Trans. on Electronic Computers*, vol. 14, 1965, pp. 326-334.

[EMM 07] EMMS D., WILSON R., , HANCOCK E., Graph Embedding using Quantum Commute Times, ESCOLANO F., VENTO M., Eds., *Graph-Based Representations in Pattern Recognition, 6th IAPR-TC-15 International Workshop, GbRPR 2007, Alicante, Spain, June 11-13, 2007, Proceedings*, vol. 4538 de *Lecture Notes in Computer Science*, Springer, 2007, pp. 371-382.

[HAK 00] HAKIMI S. L., *Location Theory*, CRC Press., 2000.

[HAU 99] HAUSSLER D., Convolution kernels on discrete structures, Technical Report UCSC-CRL-99-10, 1999.

[KAS 02] KASHIMA H., INOKUCHI A., Kernels for graph classification, *Proc. ICDM Workshop on Active Mining*, 2002, pp. 31-36.

[KAS 03] KASHIMA H., TSUDA K., , INOKUCHI A., Marginalized kernels between labeled graphs, *Proc. 20th Int. Conf. on Machine Learning*, 2003, pp. 321-328.

[KON 02] KONDOR R., LAFFERTY J., Diffusion kernels on graphs and other discrete input spaces, *Proc. 19th Int. Conf. on Machine Learning*, 2002, pp. 315-322.

[LUO 03] LUO B., WILSON R. C., , HANCOCK E. R., Spectral embedding of graphs, *Pattern Recognition*, vol. 36, n^o 10, 2003, pp. 2213-2230.

[NEU 06] NEUHAUS M., BUNKE H., Edit distance-based kernel functions for structural pattern classification, *Pattern Recognition*, , n^o 39, 2006, pp. 1852–1863.

[PEK 06] PEKALSKA E., DUIN R. P. W., , PACLÍK P., Prototype selection for dissimilarity-based classifiers, *Pattern Recognition*, vol. 39, n^o 2, 2006, pp. 189-208.

[RIE 07] RIESEN K., NEUHAUS M., , BUNKE H., Graph Embedding in Vector Spaces by Means of Prototype Selection, ESCOLANO F., VENTO M., Eds., *Graph-Based Representations in Pattern Recognition, 6th IAPR-TC-15 International Workshop, GbRPR 2007, Alicante, Spain, June 11-13, 2007, Proceedings*, vol. 4538 de *Lecture Notes in Computer Science*, Springer, 2007, pp. 383-393.

[ROB 07] ROBLES-KELLY A., HANCOCK E. R., A Riemannian approach to graph embedding, *Pattern Recognition*, vol. 40, n^o 3, 2007, pp. 1042-1056.

[SHA 04] SHAWE-TAYLOR J., CRISTIANINI N., *Kernel Methods for Pattern Analysis*, Cambridge University Pres, Cambridge, 2004.

[SHO 05] SHOKOUFANDEH A., MACRINI D., DICKINSON S. J., SIDDIQI K., , ZUCKER S. W., Indexing Hierarchical Structures Using Graph Spectra, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, n^o 7, 2005, pp. 1125-1140.

[SPI 06] SPILLMANN B., NEUHAUS M., BUNKE H., PEKALSKA E., , DUIN R. P. W., Transforming Strings to Vector Spaces Using Prototype Selection, *Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshops, SSPR 2006 and SPR 2006, Hong Kong, China, August 17-19, 2006, Proceedings*, 2006, pp. 287-296.

[TOR 07] TORSELLO A., HANCOCK E. R., Graph embedding using tree edit-union, *Pattern Recognition*, , n^o 40, 2007, pp. 1393–1405.

[WEI 37] WEISZFELD E., Sur le point pour lequel la somme des distances de n points donnés est minimum, *Tohoku Math. Journal*, , n^o 43, 1937, pp. 355– 386.

[WIL 05] WILSON R. C., HANCOCK E. R., , LUO B., Pattern Vectors from Algebraic Graph Theory, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, n^o 7, 2005, pp. 1112-1124.