A Comparison between Structural and Embedding Methods for Graph Classification*

Albert Solé-Ribalta, Xavier Cortés, and Francesc Serratosa

Universitat Rovira i Virgili - Spain
{albert.sole,xavier.cortes,francesc.serratosa}@urv.cat

Abstract. Structural pattern recognition is a well-know research field that has its birth in the early 80s. Throughout 30 years, structures such as graphs have been compared through optimization of functions that directly use attribute values on nodes and arcs. Nevertheless, in the last decade, kernel and embedding methods appeared. These new methods deduct a similarity value and a final labelling between nodes through representing graphs into a multi-dimensional space. It seems that lately kernel and embedding methods are preferred with respect to classical structural methods. However, both approaches have advantages and drawbacks. In this work, we compare structural methods to embedding and kernel methods. Results show that, with the evaluated datasets, some structural methods give slightly better performance and therefore, it is still early to discard classical structural methods for graph pattern recognition.

1 Introduction and Literature Review

Classical graph approaches for pattern recognition applications rely on computing distances between graphs on the graph domain. That is, the distance between two graphs is obtained by directly optimizing some objective function which consider node and edge attributes. However, in the graph domain these distances cannot be computed in polynomial time. To overcome this problem, a large set of approximation algorithms have been developed since the 80 [1]. Undoubtedly, pattern recognition applications which work on the graph domain need to rely on graph class prototypes to represent sets of data. The synthesis of these representatives is usually done using either a sequential or hierarchical synthesis. In this type of synthesis, the graph prototype is constructed in an iterative fashion. These iterative approaches start from a model computed using two graphs and they iteratively refine the current model by sequentially considering all the graphs in the training set. This process of synthesis, similar to generative models, relies on the fact that the model is able to capture the global information also in the initial steps. If this is not the case, usually the process derives in a bad model and so performance of the application at hand decreases. To overcome this drawback of classical models one can rely on

* We acknowledge Consolider Ingenio (CSD2007-00018) and CICYT (DPI2010-17112).

G.L. Gimel' farb et al. (Eds.): SSPR & SPR 2012, LNCS 7626, pp. 234–242, 2012.

new procedures to synthesize the prototype such as a Common Labelling solutions [2–4]. Traditional graph prototypes seems that lately have been move apart in favour of graph embeddings and kernels. These methods allow exploiting existing classical pattern recognition algorithms such as Bayes Classifiers, SVM or K-Means. Both methods, graph embeddings and kernels, rely on the same intuition, in both cases the graph is somehow encoded in a vector which correspond to a point in a multi-dimensional vector space. The main difference between kernels and embeddings rely on how the embeddings are performed. In graph embeddings, one knows the destination space and the transformation function. In this way, one explicitly does the embedding by transforming the graph into a vector. Since in graph embeddings the destination space is known, the similarity/distance/dissimilarity function is directly applied to vectors in the destination vector space. On the contrary, graph kernels go one step forward by embedding this initial vectorial representation in a larger vectorial space. This destination space is usually unknown. However, by using the so-called kernel trick, one is able to compute distances on the destination space by using the dot product between the initial vectorial representations of the graphs. Graph embeddings and graph kernels share the basic intuition. That is, they rely in encoding the graph in a some-dimensional vector space and performing operations on that space. Consequently, they share some drawbacks. Two of the main drawbacks are the following. The first one is related to the fact that usually, both approaches rely on a non-complete representation of the graphs such as [5–7], the Laplacian matrix [8] or the walk kernel [9], sacrificing, in some cases, performance for speed. The second one is related to the unfolding of the embedding in case graph information needs to be used or represented. This process might be of crucial importance if learned or classified data must be shown to the user, to either perform supervised learning or just validation. This unfolding process might be of great difficulty [10] if not impossible. The objective of this paper is manifold. On the one hand, we introduce the graph prototype synthesis from the Common Labelling point of view. This new formulation defines in a clear and uniform way the synthesis of graph under a global framework. On the other hand, we present a comparative study of state-of-the-art methods for graph classification based on graph embeddings and common labelling synthesis for graph prototyping.

2 Graph Prototypes

In this section, the graph prototypes that are evaluated are introduced, and a synthesis based on the common labelling is defined. To this aim, we start by giving some notation and basic definitions.

Attributed Graphs. Given vertex attribute domain Δ_v and edge attribute domain Δ_e, we define an attributed graph with a four-tuple $G^p = (\Sigma_v^p, \Sigma_e^p, \gamma_v^p, \gamma_e^p) \subseteq \mathcal{H}$, where $\Sigma_v^p = \{v_k^p | k = 1, , N\}$ is the set of vertices, $\Sigma_e^p = \{e_{ij}^p | i, j \in 1, , N, i \neq j\}$ is the set of arcs and $\gamma_v^p : \Sigma_v^p \to \Delta_v$ and $\gamma_e^p : \Sigma_e^p \to \Delta_e$ assign attribute values to vertices and arcs respectively. Since the article is focussed on graph prototyping

for pattern recognition, usually, a training set of elements is provided for learning this prototype. In this case, we denote the training set of prototypes by symbol $S = \{G^1, ..., G^P\}$, we consider that all graphs in S have the same order. If this is not the case, graphs can be extended with null nodes and arcs. These null nodes and arcs are labelled with a special attribute $\emptyset \in \Delta_v$ for nodes and $\emptyset \in \Delta_e$ for edges. This is the usual mechanism to deal with graphs of different cardinality, the reader is referred to [11].

2.1 Generalized Median Graph

Given a set of graph $S = \{G^1, ..., G^P\}$ and a dissimilarity/distance function $d(\cdot, \cdot)$, the Median Graph $\overline{M}$ [12] is the attributed graph that minimizes the sum of distances between it and all the graphs in the set of Attributed Graphs. That is,

$$\overline{M} = arg \min_{M \in \mathcal{H}} \sum_{G^p \in S} d(G^p, M) \tag{1}$$

Notice that the Generalized Median Graph is usually not a member of the set, and in general, more than one Generalized Median Graph may exist for a given set of graphs. The computation of a Median Graph is at least NP-complete since the general graph matching problem it is. Nevertheless, several suboptimal methods to obtain approximate solutions for the Median Graph, in reasonable time, have been presented [2, 3, 5, 10, 12–14]. These methods apply some heuristic functions in order to reduce the complexity of the graph distance computation and the size of the search space. Most of the existing methods to compute the Generalized Median Graph use a classical synthesis, that is, either sequential or hierarchical. Here, we present a synthesis based on the Common Labelling framework [4]. In this way, given a Common Labelling $H = \{h^1, ..., h^P\}$ where $h^p : \Sigma_v - > L$, a Median Graph $\overline{M} = (\overline{\Sigma}_v, \overline{\Sigma}_e, \overline{\gamma}_v, \overline{\gamma}_e)$ is defined as another Attributed Graph where attributes on nodes and arcs are computed as:

$$\overline{\gamma}_v(\overline{v}_i) = \frac{1}{\zeta_v} \sum_{G^p \in S} (1 - \delta(h^{p^{-1}}(\overline{v}_i), \emptyset)) \gamma_v(h^{p^{-1}}(\overline{v}_i)) \tag{2}$$

$$\overline{\gamma}_e(\overline{e}_{i,j}) = \frac{1}{\zeta_e} \sum_{G^p \in S} (1 - \delta(\epsilon_{i,j}^{G^p}, \emptyset)) \epsilon_{i,j}^{G^p} \tag{3}$$

where,

$$\epsilon_{i,j}^{G^p} = \gamma(e_{h^{p^{-1}}(\overline{v}_i), h^{p^{-1}}(\overline{v}_j)}) \tag{4}$$

$$\zeta_v = \sum_{G^p \in S} (1 - \delta(h^{p^{-1}}(\overline{v}_i), \emptyset)), \zeta_e = \sum_{G^p \in S} (1 - \delta(\epsilon_{i,j}^{G^p}, \emptyset) \tag{5}$$

and δ represents the Kronecker Delta function and $h^{p^{-1}}$ the inverse function of h^p. The main idea of (2) and (3) is that the attribute values of the Median Graph is the mean of the values of all the nodes or arcs of the Attributed Graphs that their nodes or arcs have been matched to a concrete node of the virtual structure L. Moreover, in the case that there does not exist a node or arc in the Attributed Graph, its value is not considered to compute the mean.

2.2 Set Median Graph

The Set Median Graph [12] is an alternative to Generalized Median Graphs. The difference between the two models consists in the search space where the Median Graph is searched. The search space for the Median Graph is the domain of the Attributed Graphs $\mathcal{H}$. In contrast, the search space for the Set Median Graph is restricted the set of graphs that represents S. The computation of Set Median Graph is, in general, exponential with respect to the order of the graphs, due to the complexity of graph isomorphism problem, but quadratic with respect to the number of graphs in S. In some applications, Set Median Graphs are preferred to Median Graphs due to two main reasons. Firstly, practical evaluations show that the capacity of Set Median Graphs to represent a set is almost similar to the capacity of Median Graphs [10]. Secondly, the synthesis (using the whole set of graphs or incrementally) is less computationally demanding. The Set Median Graph is the Attributed Graph in the set such that it has the minimum distance between it and the rest of the Attributed Graphs. Note the computation of the set median does not need any labelling only just graph-to-graph distances.

2.3 Closure Graphs

The closure graph [15] is a graph prototype originally applied to graph databases. In the closure graph, attributes of nodes or edges in the training set are represented with a set of attributes. Closure graphs are restricted to discrete attributes, if this is not the case, an extra discretization step must be performed in order to discretize them. Closure graphs need more physical space than median graphs. Formally, a closure graph $\overline{M} = (\overline{\Sigma}_v, \overline{\Sigma}_e, \overline{\gamma}_v, \overline{\gamma}_e)$ is a graph where node and arc attributes are a set of domains of the nodes and arcs in the training set, $\overline{\Delta}_v = \{\Delta_v, \Delta_v, ..., \Delta_v\}$ and $\overline{\Delta}_e = \{\Delta_e, \Delta_e, ..., \Delta_e\}$. We synthetize a Closure Graph from a set of Attributed Graphs S and a Common Labelling φ as follows:

$$\gamma_v(\overline{v}_i) = \{a | a = \gamma(h^{p^{-1}}(\overline{v}_i)), 1 \leq p \leq P, \gamma(h^{p^{-1}}(\overline{v}_i)) \neq \emptyset\} \tag{6}$$

$$\gamma_e(\overline{e}_{i,j}) = \{b_k | b_k = \epsilon_{i,j}^{G_p}, \epsilon_{i,j}^{G_p} \neq \emptyset\} \tag{7}$$

The basic intuition of (6) and (7) is that nodes and edges of the Closure graph can take all values that nodes and edges of the training set have taken. In the case that there does not exist a node or arc in the Attributed Graph, its value is not considered. Practical evaluations show that Median Graphs and Closure Graphs tend to generalize too much the set that they represent; allowing graphs that are distant from the ones that have not been used to synthesize them. To alleviate this weakness, the following probabilistic models have been defined.

2.4 First Order Random Graphs

A First-Order Random Graph (FORG) [11] is a model graph that contains first-order probabilities on nodes and arcs to describe a set of Attributed Graphs. To

deal with the first-order probabilities, there is a random variable associated with each vertex or arc, which represents the attribute information of the corresponding graph nodes and arcs in the set of Attributed Graphs. This random variable has a one-dimensional probability density function defined over the same attribute domain of the Attributed Graphs, including a null value, $\emptyset$, that denotes the non-instantiation of a FORG graph node or arc in an Attributed Graphs. This was the first probabilistic model that appeared in the literature to represent a set of Attributed Graphs. It assumes that the Attributed Graphs in a set or cluster have similar local parts. Nevertheless, in practical applications, some graphs can be quite different despite of belonging to the same class. For this reason, in several applications representing a set of attributed graphs with only first order probabilities seems to be too restrictive. A First Order Random Graph $\overline{M} = (\overline{\Sigma}_v, \overline{\Sigma}_e, P_v, P_e)$ is a graph where the node and arc attribute domains are random variables with values in Δ_v and Δ_e. Probabilities at the nodes and arcs of the FORG are related to the number of times the values have appeared at the nodes or arcs of the Attributed Graphs related to this node or arc. In the case of the nodes,

$$p_i(a) = \frac{1}{P} \sum_{\forall p \in 1..P} \delta(h^{p^{-1}}(\overline{v}_i), a) \tag{8}$$

where P corresponds to the number of graphs given to construct the prototype. Arc probabilities are computed in an equivalent form.

2.5 Function Described Graphs

A Function Described Graph (FDG) [16, 17] is a model graph that appeared with the aim of overcoming the representational power of FORGs. It contains first-order probabilities of attributes and second-order structural information to describe a set of Attributed Graphs. The first order information was represented in the same way than FORGs trough probability density functions. The second-order structural information represents qualitative information of the second-order joint probability of each pair of vertices or arcs. This information is represented by binary relations called Antagonisms, Occurrences and Existences between nodes and arcs. FDGs increased the representational power at the cost of increasing also the required physical space. Two nodes or arcs are antagonistic if they have never taken place together in any graph used to synthesise the FDG although these two nodes or arcs are included in the FDG as different elementary parts. There is an occurrence relation between two nodes or arcs of the FDG if always that one of related nodes or arcs in the graph has appeared; also the other node or arc of the same graph has appeared. Finally, there is an existence relation between two nodes or arcs if all the graphs in the class described by the FDG have at least one of the two nodes or arcs. A Function-Described Graph $\overline{M} = (\overline{\Sigma}_v, \overline{\Sigma}_e, P_v, P_e, R_v)$ is a graph where $\overline{\Sigma}_v$, $\overline{\Sigma}_e$, P_v and P_e are defined exactly in the same way than FORGs; including, in addition, a set of binary relations R. See [16] for the construction of these binary relations.

2.6 Second Order Random Graphs

A Second-Order Random Graph (SORG) [18] is a probabilistic model closely related to FDGs. The main difference lies in the fact that the second-order structural information is not defined as binary relations but with the specific information of the second-order joint probability. Thus, the physical space needed to represent SORGs is much higher than FDGs but also its ability to represent the set of Attributed Graphs increases. In [17], it is shown how to convert a SORG to an FDGs simply by analysing the second-order probabilities and deciding if the binary relations hold. A Second-Order Random Graph $\overline{M} = (\overline{\Sigma}_v, \overline{\Sigma}_e, P_v, P_e)$ is a graph where $\overline{\Sigma}_v$ and $\overline{\Sigma}_e$ are defined similarly to FORGs but in P there are first order and also second order probability densities. See [18] for the construction of these second-order probability densities given a Common Labelling.

3 Performance Evaluation

3.1 Datasets

To evaluate the structural graph prototypes for graph classification and compare its results with embedding methods, we have selected three datasets form the repository presented in [19]. The datasets are Letter HIGH and LOW and GREC. Table 1 summarizes the characteristics of the datasets. The **Letter HIGH / LOW** database contains examples of the 15 capital letters (classes) of the Roman alphabet which are composed of straight lines. Each class contains 150 graph examples. The straight lines are represented by edges and the terminal points of the lines are represented by the nodes. The **GREC** database is composed of a set of 150 symbols from architecture, electronics and other technical fields. We have used a subset of 22 different symbols (classes) which are composed only of straight lines. These images are converted into graphs by assigning a node to each junction or terminal point and an edge to each line.

Table 1. Summary of graph data set characteristics, viz. the size of the training (tr), the validation (va) and the test set (te), the number of classes (Ω), the label alphabet of both nodes and edges, the average number of nodes and edges (mean nodes/edges)

Dataset	Size (tr, va, te)	$\lvert\Omega\rvert$	V labels	E labels	mean nodes	mean edges
Let. L-H	750,750,750	15	(x,y) coord.	None	5	9
GREC	286,286,528	22	(x,y) coord.	None	11	23

3.2 Results

Table 2 presents classification accuracies achieved with different structural [11, 12, 15, 16, 18] and embedding [7, 10, 20] methods. The embedding methods in [7, 20] are based on representing each graph with a vector of distances which related each graph to a set of prototypes extracted from the training set. Different prototype selection methods are presented: sps-c, bps-c and k-cps-c.

Once selected the prototypes and the embedding is performed, training is done using a SVM. In the original article, these methods are compared with the K-NN classifier under the Graph Edit Distance. The method in [10] is addressed to compute the Median Graph to later apply a K-NN classifier. The Median Graph is computed in an iterative form using an embedding space, the selected embedding is inspired in [7, 20]. For the structural methods on the Letter dataset, in addition to the standard procedure of constructing a single prototype to represent the class, we also analyzed the effect of representing the class using several prototypes. Specifically, we also test the system using 5 prototypes to represent each class of the Letters datasets (elements have been chosen randomly). Intuitively, representing the class using several prototypes may increase performance since the representer should reduce overgeneralization of the elements it represents. It is important to highlight that, results extracted from [7, 20] may not be evaluated with the same version of the dataset presented here, since, even the dataset we used is downloaded from the same website cited in [7, 20], the number of training, validation and test elements do not seem to correspond. Even though, we assume that results can be compared since the results we obtained with the reference K-NN (Table 2 (T4)) method seem to correlate with

Table 2. Comparative study between embedding and structural methods for graph classification. Numbers indicate: (T1) results extracted from [7], (T2) results extracted from [20], (T3) distance computed with the Graduated Assigment [21], (T4) results extracted from [10], (T5) with 5 prototypes to model the class and (T6) with 1 prototype to model the class.

Alg. type	Method	Let. LOW	Let. HIGH	GREC
Ref. System	**K-NN (T1)**	91.1	61.6	86.1
	K-NN (T2)	89.1	-	-
	K-NN (T3)	94.3	82.2	95.0
	K-NN (T4)	-	-	97.9
Embed. Mthds	**Embed. Kernel(T1)**	91.8	74.3	89.2
	sps-c (T2)	92.3	-	-
	bps-c (T2)	92.9	-	-
	k-cps-c (T2)	92.0	-	-
	Set Median (T4)	-	-	76.7
	Generalized Median (T4)	-	-	78.5
Struct. Mthds	**Generalized Median**	90.3 (T5)	70.0 (T5)	90.9 (T5)
		89.2 (T6)	70.9 (T6)	-
	Set Median	96.4 (T5)	74.5 (T5)	80.5 (T5)
		96.3 (T6)	68.9 (T6)	-
	Closure Graph	93.6 (T5)	49.1 (T5)	57.0 (T5)
		69.2 (T6)	22.1 (T6)	-
	FORG	93.9 (T5)	80.1 (T5)	85.8 (T5)
		92.3 (T6)	79.2 (T6)	-
	FDG	93.9 (T5)	81.6 (T5)	85.8 (T5)
		92.3 (T6)	81.7 (T6)	-
	SORG	94.0 (T5)	80.9 (T5)	91.2 (T5)
		92.8 (T6)	79.9 (T6)	-

the results obtained in [7, 20]. Under these considerations, obtained results show that structural/classical methods achieve recognition ration on the same range as new methodologies based on graph embeddings. Some structural methods, such as the Generalized Median Graph, obtain less classification ration than the embedding methodologies. However, more advanced structural methods, such as First Order Random Graphs, Function Described Graphs, Second Order Graphs and Closure Graphs, obtain greater recognition ration, even if the classification algorithm, i.e. the K-NN, is much simpler than the SVM used in [7, 20]. With respect to the embedding method in [10], which also uses a K-NN classifier, we see that the improvement of structural methods is notable. More advanced classification algorithms may further increase the recognition ratio. With respect to the approach with different representers per class on structural algorithms, it is possible to note a general tendency on the decrease of the recognition ration. This tendency increases in Closure Graphs. Thus, we could conclude that, in these datasets, it is better to use all the information available to generate a single prototype. One could think that some clustering schema to group elements, instead of random selection, should improve results. However, since, in this particular dataset, noise on elements is uniformly distributed, a random selection algorithm is as valid as any other grouping scheme.

4 Conclusions and Discussion

In this article, we presented a comparison between the two main approaches to perform graph classifications. The comparison is focussed in structural and embedding methods for graph data. The main objective of embedding methods is to encode/embed each graph as a point in some-dimensional vector space. In this way, they take the advantage of classical classification mechanisms for vector spaces to work with graph data in the embedding space. Once vectors are embedded in the vectorial space, operations between them are usually done in polynomial time. However, in most of the cases the process embedding requires non-polynomic computations. The main drawback of embedding methods is that usually vectorial representations of graphs are not complete, so they do not contain all the information available. On the other hand, most of the structural methods required either non-polynomic or approximated algorithms to compute the distance between two graphs. However, good approximation methods exist. Results presented in this article, show that classical graph prototype with combination with advanced synthesis mechanisms, such as the common labelling, improve or at least give the same classification accuracy than embedding methods. In addition, if embedding is done in non-polynomic time the classification phase is faster with structural methods since the queried graph do not need to be embedded and fewer graph-to-graph comparisons are required.

References

1. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in pattern recognition. IJPRAI 18(3), 265–298 (2004)

2. Mukherjee, L., Singh, V., Peng, J., Xu, J., Zeitz, M., Berezney, R.: Generalized median graphs and applications. Journal of Combinatorial Optimization 17, 21–44 (2009)
3. Hlaoui, A., Wang, S.: Median graph computation for graph clustering. Soft Computing - A Fusion of Foundations, Methodologies and Applications 10, 47–53 (2006)
4. 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)
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. Gibert, J., Valveny, E., Bunke, H.: Graph embedding in vector spaces by node attribute statistics. Pattern Recognition 45(9), 3072–3083 (2012)
7. Riesen, K., Bunke, H.: Graph classification based on vector space embedding. IJPRAI 23(6), 1053–1081 (2009)
8. Kondor, R.I., Lafferty, J.D.: Diffusion kernels on graphs and other discrete input spaces. In: ICML, pp. 315–322 (2002)
9. Gärtner, T., Flach, P.A., Wrobel, S.: On graph kernels: Hardness results and efficient alternatives. In: COLT, pp. 129–143 (2003)
10. 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)
11. 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 PAMI-7(5), 599–609 (1985)
12. Jiang, X., Müunger, A., Bunke, H.: On median graphs: Properties, algorithms, and applications. IEEE Trans. Pattern Anal. Mach. Intell. 23(10), 1144–1151 (2001)
13. Ferrer, M., Valveny, E., Serratosa, F.: Median graphs: A genetic approach based on new theoretical properties. Pattern Recognition 42(9), 2003–2012 (2009)
14. Ferrer, M., Serratosa, F., Valveny, E.: Evaluation of Spectral-Based Methods for Median Graph Computation. In: Martí, J., Benedí, J.M., Mendonça, A.M., Serrat, J. (eds.) IbPRIA 2007, Part II, LNCS, vol. 4478, pp. 580–587. Springer, Heidelberg (2007)
15. He, H., Singh, A.K.: Closure-tree: An index structure for graph queries. In: ICDE, p. 38 (2006)
16. Serratosa, F., Alquézar, R., Sanfeliu, A.: Synthesis of function-described graphs and clustering of attributed graphs. IJPRAI 16(6), 621–656 (2002)
17. Serratosa, F., Alquézar, R., Sanfeliu, A.: Function-described graphs for modelling objects represented by sets of attributed graphs. Pattern Recognition 36(3), 781–798 (2003)
18. Sanfeliu, A., Serratosa, F., Alquézar, R.: Second-order random graphs for modeling sets of attributed graphs and their application to object learning and recognition. IJPRAI 18(3), 375–396 (2004)
19. 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)
20. Bunke, H., Riesen, K.: Towards the unification of structural and statistical pattern recognition. Pattern Recognition Letters 33(7), 811–825 (2012)
21. Gold, S., Rangarajan, A.: A graduated assignment algorithm for graph matching. IEEE Trans. Pattern Anal. Mach. Intell. 18(4), 377–388 (1996)