On the Relevance of Local Neighbourhoods for Greedy Graph Edit Distance

Xavier Cortés[1(✉)], Francesc Serratosa[1], and Kaspar Riesen[2]

[1] Universitat Rovira i Virgili, Tarragona, Spain
{xavier.cortes,frances.serratosa}@urv.cat
[2] University of Applied Sciences and Arts Northwestern Switzerland, Basel, Switzerland
kaspar.riesen@fhnw.ch

Abstract. Approximation of graph edit distance based on bipartite graph matching emerged to an important model for distance based graph classification. However, one of the drawbacks of this particular approximation is its cubic runtime with respect to the number of nodes of the graphs. In fact, this runtime restricts the applicability of bipartite graph matching to graphs of rather small size. Recently, a new approximation for graph edit distance using greedy algorithms (rather than optimal bipartite algorithms) has been proposed. This novel algorithm reduces the computational complexity to quadratic order. In another line of research it has been shown that the definition of local neighbourhoods plays a crucial role in bipartite graph matching. These neighbourhoods define the local substructures of the graphs which are eventually assigned to each other. In the present paper we demonstrate that the type of local neighbourhood and in particular the distance model defined on them is also highly relevant for graph classification using greedy graph edit distance.

Keywords: Graph edit distance · Graph classification · Greedy assignment · Bipartite graph matching

1 Introduction

Graphs offer a convenient way to formally model objects or patterns, which are composed of complex subparts including the relations that might exist between these subparts. In particular, the nodes of a graph can be employed to represent the individual components of a pattern while the edges might represent the structural connections between these components. Attributed graphs, that is, graphs in which nodes and/or edges are labelled with one or more attributes, have been of crucial importance in pattern recognition throughout more than four decades [1–5]. For instance, graphs are successfully used in the field of recognizing biological patterns where one has to formally describe complex protein structures [6] or molecular compounds [7].

This research is supported by projects DPI2013-42458-P and TIN2013-47245-C2-2-R.

A. Robles-Kelly et al. (Eds.): S+SSPR 2016, LNCS 10029, pp. 121–131, 2016.
DOI: 10.1007/978-3-319-49055-7_11

The definition of an adequate dissimilarity model between two patterns is one of the most basic requirements in pattern recognition. The process of evaluating the dissimilarity of two graphs is commonly referred to as *graph matching*. The overall aim of graph matching is to find a correspondence between the nodes and edges of two graphs. There are several graph matching models available. *Spectral methods*, for instance, constitute an important class of error-tolerant graph matching procedures with a quite long tradition [8–12]. *Graph kernels* constitute a second important family of graph matching procedures and various types of graph kernels emerged during the last decade [13–17]. For an extensive review on these and other graph matching methods developed during the last forty years, the reader is referred to [2–4].

In the present paper we focus on the concept of *graph edit distance* [18–20]. This graph matching model is especially interesting because it is able to cope with directed and undirected, as well as with labelled and unlabeled graphs. However, the major drawback of graph edit distance is its computational complexity which is exponential with respect to the size of the graphs. Graph edit distance belongs to the family of *quadratic assignment problems* (QAPs), and thus, graph edit distance is known to be an NP-complete problem. In order to reduce the computational complexity of this particular distance model, an algorithmic framework for the approximation of graph edit distance was introduced in [21]. The basic idea of this approach is to find an optimal assignment of local neighbourhoods in the graphs, reducing the problem of graph edit distance to a *linear sum assignment problem* (LSAP). Recently, a new approximation framework, which extends [21–25] in terms of finding a suboptimal assignment of local neighbourhoods, has been presented in [27]. This new approach further reduces the computational complexity of graph edit distance from cubic to quadratic order. An algorithm has been presented in which the human can interact or guide the algorithm to find a proper correspondence between nodes or interest points [28, 29]. And other graph matching techniques have been presented such as [30, 31].

Both approaches, i.e. the original approximation [21–25] as well as the faster variant [27], use the same definition of local neighbourhoods, viz. single nodes and their adjacent edges. Yet various other definitions for local structures exist [32]. The present paper investigates the impact of these different local neighbourhoods in the context of the recent quadratic approximation algorithm [27].

2 Definitions and Algorithms

2.1 Graph and Neighbourhood

An attributed graph is defined as a triplet $G = (\Sigma_v, \Sigma_e, \gamma_v)$, where $\Sigma_v = \{v_a | a = 1, \ldots, n\}$ is the set of nodes and $\Sigma_e = \{e_{ab} | a, b \in 1, \ldots, n\}$ is the set of undirected and unattributed edges. In the present paper we investigate graphs with labelled nodes only. Yet, all of our concepts can be extended to graphs with labelled edges. Function $\gamma_v : \Sigma_v \rightarrow \Delta_v$ assigns attribute values from arbitrary domains to nodes. The order of a graph G is equal to the number of nodes n. The number of edges of node v_a is referred to as $E(v_a)$. Finally, the neighbourhood of a node v_a is termed N_{v_a}.

In this paper we focus on three different types of neighbourhoods N_{v_a} depending on the amount of structural information which is taken into account. The first definition of N_{v_a} is the empty set. That is, no structural information of node v_a is taken into account. Formally, $N_{v_a} = \left(\Sigma_v^{N_{v_a}}, \Sigma_e^{N_{v_a}}, \gamma_v\right)$ with $\Sigma_v^{N_{v_a}} = \Sigma_e^{N_{v_a}} = \{\}$. We refer to this type of neighbourhood as *Node*.

The second definition of N_{v_a} is given by $N_{v_a} = \left(\Sigma_v^{N_{v_a}}, \Sigma_e^{N_{v_a}}, \gamma_v\right)$ with $\Sigma_v^{N_{v_a}} = \{\}$ and $\Sigma_e^{N_{v_a}} = \{e_{ab} \in \Sigma_e\}$. In this case we regard the incident edges of v_a as neighbourhood only (without adjacent nodes). This second definition of N_{v_a} is referred to as *Degree* from now on. The third definition of N_{v_a} used in this paper is given by the set of nodes that are directly connected to v_a including all edges that connect these nodes with v_a. Formally, $N_{v_a} = \left(\Sigma_v^{N_{v_a}}, \Sigma_e^{N_{v_a}}, \gamma_v\right)$ with $\Sigma_v^{N_{v_a}} = \{v_b | e_{ab} \in \Sigma_e\}$ and $\Sigma_e^{N_{v_a}} = \{e_{ab} \in \Sigma_e | v_b \in \Sigma_v^{N_{v_a}}\}$. We refer to this definition of a neighbourhood as *Star*. In Fig. 1 an illustration of the three different neighbourhoods (*Node*, *Star* and *Degree*) is shown.

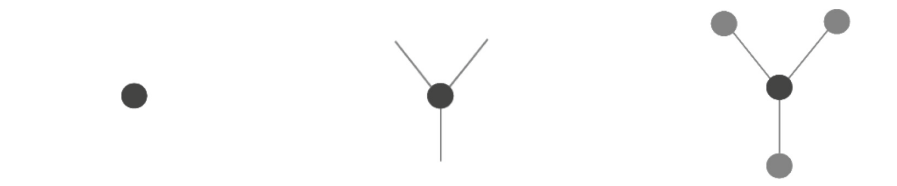
(a) *Node* neighbourhood (b) *Degree* neighbourhood (c) *Star* neighbourhood

Fig. 1. *Node*, *Degree* and *Star* neighbourhoods (shown in light-grey) of a single node (shown in dark-grey).

2.2 Graph Edit Distance

A widely used method to evaluate the dissimilarity between two attributed graphs is graph edit distance [18, 19]. The basic idea of this dissimilarity model is to define a distance between two graphs G^p and G^q by means of the minimum amount of distortion required to transform G^p into G^q. To this end, a number of distortions or edit operations, consisting of insertion, deletion, and substitution of both nodes and edges are employed. Edit cost functions are typically introduced to quantitatively evaluate the level of distortion of each individual edit operation. The basic idea of this is to assign a cost to the edit operations proportional to the amount of distortion they introduce in the underlying graphs.

A sequence $(e_1, \ldots, e_k)$ of k edit operations e_i that transform G^p completely into G^q is called an *edit path* $\lambda(G^p, G^q)$ between G^p and G^q . Note that in an edit path $\lambda(G^p, G^q)$ each node of G^p is either deleted or uniquely substituted with a node in G^q, and likewise, each node in G^q is either inserted or matched with a unique node in G^p. The same applies for the edges.

Let $\Upsilon(G^p, G^q)$ denote the set of all edit paths between two graphs G^p and G^q. The edit distance of two graphs is defined as the sum of cost of the minimal cost edit path among all competing paths in $\Upsilon(G^p, G^q)$.

Optimal algorithms for computing the edit distance are typically based on combinatorial search procedures (such as A* based search techniques). These procedures explore the space of all possible mappings of the nodes and edges of G^p to the nodes and edges of G^q (i.e. the search space corresponds to the set of all edit paths $\Upsilon(G^p, G^q)$). Yet, considering m nodes in G^p and n nodes in G^q, the set of possible edit paths $\Upsilon(G^p, G^q)$ contains $O(m^n)$ edit paths. Therefore, exact graph edit distance computation is exponential in the number of nodes of the involved graphs.

The problem of minimizing the graph edit distance can be reformulated as an instance of a Quadratic Assignment Problem (QAP). QAPs belong to the most difficult combinatorial optimization problems for which only exponential run time algorithms are known to date (QAPs are known to be NP-complete). The *Bipartite Graph Matching* algorithm (BP-GED) [21] is an approximation for the graph edit distance that reduces the QAP of graph edit distance computation to an instance of a Linear Sum Assignment Problem (LSAP). This algorithm first generates a cost matrix C which is based on costs of editing local substructures of both graphs. Formally, the cost matrix is defined by:

$$C = \left[\begin{array}{cccc|cccc} C_{1,1} & C_{1,2} & \cdots & C_{1,m} & C_{1,\varepsilon} & \infty & \cdots & \infty \\ C_{2,1} & C_{2,2} & \cdots & C_{2,m} & \infty & C_{2,\varepsilon} & \cdots & \infty \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{n,1} & C_{n,2} & \cdots & C_{n,m} & \infty & \infty & \cdots & C_{n,\varepsilon} \\ \hline C_{\varepsilon,1} & \infty & \cdots & \infty & 0 & 0 & \cdots & 0 \\ \infty & C_{\varepsilon,2} & \cdots & \infty & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \infty & \infty & \cdots & C_{\varepsilon,m} & 0 & 0 & \cdots & 0 \end{array}\right]$$

Where $C_{i,j}$ denotes the cost of substituting nodes v_i^p and v_j^q as well as their local neighbourhoods $N_{v_i^p}$ and $N_{v_j^q}$. $C_{i,\varepsilon}$ denotes the cost of deleting node v_i^p and its local neighbourhood $N_{v_i^p}$, and $C_{\varepsilon,j}$ denotes the cost of inserting node v_j^q and its local neighbourhood $N_{v_j^q}$.

A linear assignment algorithm can be applied on C in order to find a (optimal) mapping of nodes and their local neighbourhoods. A large number of solvers for linear sum assignment problems exist [26]. The time complexity of the best performing exact algorithms for LSAPs is cubic in the size of the problem.

Any complete assignment of local substructures derived on C can be reformulated as an admissible edit path from $\Upsilon(G^p, G^q)$. That is, the global edge structure from G^p and G^q can be edited with respect to the node operations captured in the mapping of local substructures (this is due to the fact that edit operations on edges always depend on the edit operations actually applied on their adjacent nodes). Eventually, the total cost of all edit operations (applied on both nodes and edges) can be interpreted as a graph edit distance approximation between graphs G^p and G^q (termed BP-GED from now on).

The edit path found with this particular procedure considers the structural information of the graphs in an isolated way only (singular neighbourhoods). Yet, the derived distance considers the edge neighbourhood of G^p and G^q in a global and consistent way and thus the derived distance is in the best case equal to, or in general larger than the exact graph edit distance.

Recently, a new approach which uses an approximation rather than an exact algorithm to solve the assignment of local substructures has been proposed [27]. This new approach employs a greedy assignment algorithm to suboptimally solve the LSAP stated on cost matrix C. The algorithm iterates through C from top to bottom through all rows, and in every row it assigns the current element to the minimum unused element in a greedy manner. More formally, for each row i in the cost matrix $C = (C_{ij})$ the minimum cost entry $\varphi_i = argmin_{\forall \mathrm{j}} c_{ij}$ is determined and the corresponding node edit operation $(v_i^p \rightarrow v_{\varphi i}^q)$ is added to the mapping of local substructures. By removing column φ_i in C one can ensure that every column of the cost matrix is considered exactly once.

The remaining parts of the approximation algorithm are identical with the original framework. That is, based on the found assignment of local substructures an admissible edit path and its corresponding sum of costs is derived. However, as the complexity of this suboptimal assignment algorithm is only quadratic (rather than cubic), the time complexity of the complete graph edit distance approximation, termed *Greedy-GED* from now on, is further reduced.

3 Distance Models for Neighbourhoods

In this section we review and compare various methods proposed in [32] to obtain the individual cost entries $C_{i,j}$, $C_{i,\varepsilon}$, and $C_{\varepsilon,j}$ in the cost matrix. These cost entries depend on two weighted disjoint cost values. The first cost is defined with respect to the nodes, while the second cost takes into account the local neighbourhoods of the nodes. Formally, we define $C_{i,j}$, $C_{i,\varepsilon}$, and $C_{\varepsilon,j}$ according to the following three cases:

(1) If two nodes $v_i^p \in \Sigma_v^p$ and $v_j^q \in \Sigma_v^q$ are mapped to each other, we have

$$C_{i,j} = \beta \cdot C(v_i^p \rightarrow v_j^q) + (1-\beta) \cdot C(N_{v_i^p} \rightarrow N_{v_j^q}) \qquad (1)$$

where $\beta \in]0,1[$ is a weighting parameter that controls what is more important, the cost of the pure node substitution $(v_i^p \rightarrow v_j^q)$ or the cost of substituting the neighbourhoods $N_{v_i^p}$ and $N_{v_j^q}$ of both nodes.

(2) If one node $v_i^p \in \Sigma_v^p$ in G^p is deleted, we have

$$C_{i,\varepsilon} = \beta \cdot k_v + (1-\beta) \cdot C(N_{v_i^p} \rightarrow \varepsilon) \qquad (2)$$

where $\beta \in]0,1[$ (as defined above), k_v refers to a positive constant cost for deleting one node and $C(N_{v_i^p} \rightarrow \varepsilon)$ refers to the cost of deleting the complete neighbourhood of v_i^p

(3) If one node $v_i^q \in \Sigma_v^q$ in G^q is inserted, we finally have (similar to case 2)

$$C_{\varepsilon,j} = \beta \cdot k_v + (1-\beta) \cdot C(\varepsilon \rightarrow N_{v_j^q}) \qquad (3)$$

The cost for node substitutions $C(v_i^p \rightarrow v_j^q)$ is commonly defined with respect to the underlying labelling of the involved nodes. Yet, the definition of an adequate cost model for neighbourhoods is not that straightforward, and this, described in greater detail in the next subsection.

3.1 The Cost of Processing Neighbourhoods

If the structural information of the nodes is not considered, that is we employ the *Node* neighbourhood, we have

$$C(N_{v_i^p} \to N_{v_j^q}) = C(N_{v_i^p} \to \varepsilon) = C(\varepsilon \to N_{v_j^q}) = 0 \tag{4}$$

For *BP-GED* [21] as well as for *Greedy-GED* [27] the same definition of the neighbourhood of a node v has been used, viz. N_v is defined to be the set of incident edges of node v. That is, the *Degree* neighbourhood, as formally described in Sect. 2, is employed. Remember that in our paper unlabeled edges are considered only, and thus, edge substitution is free of cost. Hence, using this definition of a neighbourhood, the cost of substituting two neighbourhoods with each other is given by the difference of the numbers of edges of the involved nodes. Formally,

$$C(N_{v_i^p} \to N_{v_j^q}) = k_e \cdot \left| E(v_i^p) - E\left(v_j^q\right) \right| \tag{5}$$

where k_e refers to a positive constant cost for deleting/inserting edges and *E(.)* refers to the number of edges of a certain node.

Likewise, the deletion and insertion costs of neighbourhoods depend on the number of incident edges $E(v_i^p)/E\left(v_j^q\right)$ of the deleted or inserted node v_i^p and v_j^q, respectively. Formally,

$$C(N_{v_i^p} \to \varepsilon) = k_e \cdot E(v_i^p) \tag{6}$$

$$C(\varepsilon \to N_{v_j^q}) = k_e \cdot E\left(v_j^q\right) \tag{7}$$

We name this particular definition of the cost for processing neighbourhoods as *Degree-cost*. We will use this cost model for neighbourhoods as basic reference system.

Next, four other definitions of the cost for processing neighbourhoods are presented. In contrast with the *Degree*-cost model described above, these definitions are based on the *Star* neighbourhood.

The following definition of the insertion as well as the deletion cost assume that the complete neighbourhood has to be inserted or deleted when the corresponding node is inserted or deleted, respectively. That is, insertion and deletion costs consider the cost of processing all edges that connect the central node and the cost of processing all adjacent nodes. Formally, the complete deletion and insertion costs of neighbourhoods depend on the number of adjacent nodes.

$$C(N_{v_i^p} \rightarrow \varepsilon) = (k_v + k_e) \cdot E(v_i^p) \tag{8}$$

$$C(\varepsilon \rightarrow N_{v_j^q}) = (k_v + k_e) \cdot E\left(v_j^q\right) \tag{9}$$

Where $(k_v + k_e)$ refers to the cost of deleting or inserting one edge and one node.

The substitution cost $C(N_{v_i^p} \rightarrow N_{v_j^q})$ for star neighbourhoods is based on computing a distance between $N_{v_i^p}$ and $N_{v_j^q}$. For this particular computation every adjacent node and its corresponding edge is interpreted as an indivisible entity. That is, a *Star* neighbourhood can be seen as a set of independent entities (nodes with adjacent edge), and thus, the computation of $C(N_{v_i^p} \rightarrow N_{v_j^q})$ refers to an assignment problem between two independent sets. This assignment problem can be solved in the same way as it is done with complete graphs by means of a cost matrix that considers substitutions, insertions and deletions. Formally,

$$C(N_{v_i^p} \rightarrow N_{v_j^q}) = AssignmentCost\left(N_{v_i^p}, N_{v_j^q}\right) \tag{10}$$

In order to compute this assignment cost we use four different optimization algorithms.

The first algorithm is given by an optimal algorithm for general LAPs known as *Hungarian* algorithm [33] that runs in cubic time. The second algorithm solves the assignment of $N_{v_i^p}$ and $N_{v_j^q}$ by means of the *Hausdorff* distance for subsets [34]. This assignment algorithm estimates the distance between two sets of entities by removing the restriction of finding a bijective mapping between the individual elements. In contrast with the *Hungarian* algorithm, *Hausdorff* assignments can be computed in quadratic time.

The third algorithm computes the dissimilarity between $N_{v_i^p}$ and $N_{v_j^q}$ by finding a suboptimal assignment of the individual entities by means of the *Greedy* assignment algorithm [27] described in Sect. 2 (also in quadratic time).

Finally, we propose to use a *Planar* distance model. In this case, the relative position of each neighbourhood node is considered. That is, the only allowed assignments of entities from $N_{v_i^p}$ to entities of $N_{v_j^q}$ are the ones that are generated from cyclic combinations of the neighbours. Formally, the sets $N_{v_i^p}$ and $N_{v_j^q}$ are interpreted as strings and the assignment cost is computed through the *Levenshtein* distance on these strings [34–38].

4 Experimental Evaluation

In the present paper three specific definitions for node neighbourhoods are presented, namely *Node*, *Degree* and *Star*. For the computation of both *Node* and *Degree* no additional algorithm is required. Yet, for *Star* neighbourhood a particular assignment solver is needed. We propose four different algorithms for this task. Hence, in total we have six different cost models to compute the individual entries $c_{i,j} \in C$. Major aim of the present paper is to show the relevance of these costs models in terms of classification accuracy and runtime using *Greedy-GED*. We coded the algorithms using MATLAB 2013 and we conducted the experiments on an i5 processor of 1.60 GHz with 4 GB of RAM and Windows 10.

Table 1 shows the recognition ratio of a 1NN classifier and the mean matching runtime achieved with all models on five graph databases, viz. LETTER LOW, LETTER HIGH, GREC, COIL RAG and AIDS from the IAM graph repository [39]. In our experiments parameter β is fixed to 0.5 and we gauge k_v and k_e through a non-exhaustive trial and error process. The best accuracy and the fastest runtime is highlighted in bold face on each database, while the second best is underlined.

Table 1. Recognition ratio (1NN) and Runtime (Øt) of the three neighbourhoods (and for assignment solvers) on five datasets using *greedy graph edit distance*.

Local neighbourhood	Assignment solver	Database									
		LETTER LOW		LETTER HIGH		GREC		COIL RAG		AIDS	
		1NN (%)	Øt (ms)	1NN (%)	Øt (ms)	1NN (%)	Øt (ms)	1NN (%)	Øt (ms)	1NN (%)	Øt (ms)
Degree	–	95.07	0.6	79.07	0.6	97.54	1.3	96.00	0.4	**95.53**	2.1
Node	–	90.67	**0.4**	60.27	**0.4**	93.37	**0.8**	**96.40**	**0.3**	86.80	**1.2**
Star	Hungarian	98.53	12.5	86.53	13.5	96.40	84.7	95.10	5.8	88.00	162.6
	Hausdorff	**98.67**	1.3	88.40	1.4	96.97	6.9	95.10	0.8	89.00	11.7
	Greedy	**98.67**	1.1	86.67	1.3	96.97	5.6	94.50	0.8	88.60	10.2
	Levenshtein	98.53	1.1	**89.33**	1.7	**97.73**	8.7	94.70	1.0	87.53	17.6

We first focus on the mean run time for one matching in ms (Øt). We observe that on all data sets the Node neighbourhood is the fastest model (as it could be expected) followed by the reference model Degree (which is only slightly slower than the Node neighbourhood). The models that are based on the Star neighbourhood suffer from substantial higher runtimes than Node and Degree (especially on the larger graphs of the GREC and AIDS data sets). Comparing the four assignment solvers with each other, we observe that the optimal (cubic time) Hungarian algorithm provides the highest run times among all competing algorithms, while the differences between the other three algorithms are negligible.

Next, we focus on the recognition rates of the different models. We observe that in three out of five cases the Star neighbourhood achieves the best recognition rates (LETTER LOW and HIGH as well as GREC). The Node neighbourhood and the reference model Degree achieve the best recognition rate on COIL RAG and AIDS, respectively. An interesting observation can be made on the AIDS data set where the reference model Degree significantly outperforms all other methods addressed in the present paper. Overall we conclude that it remains difficult to predict an adequate definition of local neighbourhoods for a given data set and application. Yet, there seems to be a weak tendency that increased neighbourhoods might improve the overall matching accuracy.

5 Conclusions

The fast approximation of graph edit distance is still an open and active area of research in the field of structural pattern recognition. A common model for the approximation of graph distances is based on bipartite graph matching. The basic idea of this approach is to reduce the difficult QAP of graph edit distance computation to an LSAP. This particular algorithmic framework consists of three major steps. In a first step the graphs to be matched are subdivided into individual nodes including local neighbourhoods. Next, in step 2, an algorithm solving the LSAP is employed in order to find an assignment of the nodes (plus local neighbourhoods) of both graphs. Finally, in step 3, an approximate graph edit distance is derived from the assignment of step 2. In the present paper we review six different ways to compute the costs of local neighbourhoods and compare them with each other on five data sets. Although no clear winner can be found in the experimental evaluation, the empirical results suggests to use a larger neighbourhood than it is traditionally employed in the context of bipartite graph matching.

References

1. Sanfeliu, A., Alquézar, R., Andrade, J., Climent, J., Serratosa, F., Vergés, J.: Graph-based representations and techniques for image processing and image analysis. Pattern Recogn. **35** (3), 639–650 (2002)
2. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in pattern recognition. Int. J. Pattern Recogn. Artfi. Intell. **18**(3), 265–298 (2004)
3. Vento, M.: A one hour trip in the world of graphs, looking at the papers of the last ten years. In: Artner, N.M., Haxhimusa, Y., Jiang, X., Kropatsch, W.G. (eds.) GbRPR 2013. LNCS, vol. 7877, pp. 1–10. Springer, Heidelberg (2013)
4. Foggia, P., Percannella, G., Vento, M.: Graph matching and learning in pattern recognition in the last 10 years. Int. J. Pattern Recogn. Artif. Intell. **28**(1), 1450001 (2014)
5. Sanroma, G., Penate-Sanchez, A., Alquezar, R., Serratosa, F., Moreno-Noguer, F., Andrade-Cetto, J., Gonzalez, M.A.: MSClique: multiple structure discovery through the maximum weighted clique problem. PLoS ONE **11**(1), e0145846 (2016)
6. Borgwardt, K., Ong, C., Schönauer, S., Vishwanathan, S., Smola, A., Kriegel, H.-P.: Protein function prediction via graph kernels. Bioinformatics **21**(1), 47–56 (2005)
7. Mahé, P., Ueda, N., Akutsu, T., Perret, J., Vert, J.: Graph kernels for molecular structure-activity relationship analysis with support vector machines. J. Chem. Inf. Model. **45**(4), 939–951 (2005)
8. Luo, B., Wilson, R.C., Hancock, E.R.: Spectral feature vectors for graph clustering. In: Caelli, T.M., Amin, A., Duin, R.P., Kamel, M.S., de Ridder, D. (eds.) SPR 2002 and SSPR 2002. LNCS, vol. 2396, pp. 83–93. Springer, Heidelberg (2002)
9. Robles-Kelly, A., Hancock, E.R.: Graph edit distance from spectral seriation. IEEE Trans. Pattern Anal. Mach. Intell. **27**(3), 365–378 (2005)
10. Kosinov, S., Caelli, T.M.: Inexact multisubgraph matching using graph eigenspace and clustering models. In: Caelli, T.M., Amin, A., Duin, R.P., Kamel, M.S., de Ridder, D. (eds.) SPR 2002 and SSPR 2002. LNCS, vol. 2396, pp. 133–142. Springer, Heidelberg (2002)

11. Wilson, R.C., Hancock, E.R., Luo, B.: Pattern vectors from algebraic graph theory. IEEE Trans. Pattern Anal. Mach. Intell. **27**(7), 1112–1124 (2005)
12. Qiu, H., Hancock, E.R.: Graph matching and clustering using spectral partitions. Pattern Recogn. **39**(1), 22–34 (2006)
13. Ramon, J., Gärtner, T.: Expressivity versus efficiency of graph kernels. In: First International Workshop on Mining Graphs, Trees and Sequences, pp. 65–74 (2003)
14. Borgwardt, K., Petri, T., Kriegel, H.-P., Vishwanathan, S.: An efficient sampling scheme for comparison of large graphs. In: International Workshop on Mining and Learning with Graphs (2007)
15. Vishwanathan, S.V.N., Borgwardt, K., Schraudolph, N.N.: Fast computation of graph kernels. In: Annual Conference on Neural Information Processing Systems, pp. 1449–1456. MIT Press (2006)
16. Gaüzère, B., Brun, L., Villemin, D.: Two new graph kernels and applications to chemoinformatics. In: Jiang, X., Ferrer, M., Torsello, A. (eds.) GbRPR 2011. LNCS, vol. 6658, pp. 112–121. Springer, Heidelberg (2011)
17. Gauzere, B., Grenier, P.A., Brun, L., Villemin, D.: Treelet kernel incorporating cyclic, stereo and inter pattern information in chemoinformatics. Pattern Recogn. **48**(2), 356–367 (2015)
18. Bunke, H., Allermann, G.: Inexact graph matching for structural pattern recognition. Pattern Recogn. Lett. **1**, 245–253 (1983)
19. Sanfeliu, A., Fu, K.: A distance measure between attributed relational graphs for pattern recognition. IEEE Trans. Syst. Man Cybern. (Part B) **13**(3), 353–363 (1983)
20. Solé, A., Serratosa, F., Sanfeliu, A.: On the graph edit distance cost: properties and applications. Int. J. Pattern Recogn. Artif. Intell. **26**(5), 1260004 [21 p.] (2012)
21. Riesen, K., Bunke, H.: Approximate graph edit distance computation by means of bipartite graph matching. Image Vis. Comput. **27**(4), 950–959 (2009)
22. Serratosa, F.: Fast computation of bipartite graph matching. Pattern Recogn. Lett. **45**, 244–250 (2014)
23. Serratosa, F.: Computation of graph edit distance: reasoning about optimality and speed-up. Image Vis. Comput. **40**, 38–48 (2015)
24. Serratosa, F.: Speeding up fast bipartite graph matching trough a new cost matrix. Int. J. Pattern Recogn. Artif. Intell. **29**(2),1550010 [17 p.] (2015)
25. Ferrer, M., Serratosa, F., Riesen, K.: Improving bipartite graph matching by assessing the assignment confidence. Pattern Recogn. Lett. **65**, 29–36 (2015)
26. Burkard, R., Dell'Amico, M., Martello, S.: Assignment Problems. Society for Industrial and Applied Mathematics, Philadelphia (2009)
27. Riesen, K., Ferrer, M., Fischer, A., Bunke, H.: Approximation of graph edit distance in quadratic time. In: Liu, C.-L., Luo, B., Kropatsch, W.G., Cheng, J. (eds.) GbRPR 2015. LNCS, vol. 9069, pp. 3–12. Springer, Heidelberg (2015)
28. Serratosa, F., Cortés, X.: Interactive graph-matching using active query strategies. Pattern Recogn. **48**(4), 1364–1373 (2015)
29. Cortés, X., Serratosa, F.: An interactive method for the image alignment problem based on partially supervised correspondence. Expert Syst. Appl. **42**(1), 179–192 (2015)
30. Sanromà, G., Alquézar, R., Serratosa, F., Herrera, B.: Smooth point-set registration using neighbouring constraints. Pattern Recogn. Lett. **33**, 2029–2037 (2012)
31. Sanromà, G., Alquézar, R., Serratosa, F.: A new graph matching method for point-set correspondence using the EM algorithm and softassign. Comput. Vis. Image Underst. **116** (2), 292–304 (2012)
32. Serratosa, F., Cortés, X.: Graph edit distance: moving from global to local structure to solve the graph-matching problem. Pattern Recogn. Lett. **65**, 204–210 (2015)

33. Kuhn, H.W.: The Hungarian method for the assignment problem. Naval Res. Logistics Q. **2**, 83–97 (1955)
34. Huttenlocher, D.P., Klanderman, G.A., Kl, G.A., Rucklidge, W.J.: Comparing images using the Hausdorff distance. IEEE Trans. Pattern Anal. Mach. Intell. **15**, 850–863 (1993)
35. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions and reversals. Sov. Phys. Dokl. Cybern. Control Theory **10**, 707–710 (1966)
36. Peris, G., Marzal, A.: Fast cyclic edit distance computation with weighted edit costs in classification. Int. Conf. Pattern Recogn. **4**, 184–187 (2002)
37. Serratosa, F., Sanfeliu, A.: Signatures versus histograms: definitions, distances and algorithms. Pattern Recogn. **39**(5), 921–934 (2006)
38. Serratosa, F., Sanromà, G.: A fast approximation of the earth-movers distance between multi-dimensional histograms. Int. J. Pattern Recognit. Artif. Intell. **22**(8), 1539–1558 (2008)
39. Riesen, K., Bunke, H.: IAM graph database repository for graph based pattern recognition and machine learning. In: Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) Structural, Syntactic, and Statistical Pattern Recognition. LNCS, vol. 5342, pp. 287–297. Springer, Heidelberg (2008)