On the Influence of Node Centralities on Graph Edit Distance for Graph Classification

Xavier Cortés, Francesc Serratosa(✉), and Carlos F. Moreno-García

Universitat Rovira i Virgili, Tarragona, Catalonia, Spain
{francesc.serratosa,xavier.cortes}@urv.cat
carlosfrancisco.moreno@estudiants.urv.cat

Abstract. Classical graph approaches for pattern recognition applications rely on computing distances between graphs in the graph domain. That is, the distance between two graphs is obtained by directly optimizing some objective function which consider node and edge attributes. Bipartite Graph Matching was first published in a journal in 2009 and new versions have appeared to speed up its runtime such as the Fast Bipartite Graph Matching. This algorithm is based on defining a cost matrix between all nodes of both graphs and solving the node correspondence through a linear assignment method. To construct the matrix, several local structures can be defined from the simplest one (only the node) to the most complex (a whole clique or eigenvector structure). In this paper, we propose five different options and we show that the type of local structure and the distance defined between these structures is relevant for graph classification.

Keywords: Graph edit distance · Bipartite graph matching · Fast bipartite graph matching · Levenshtein distance

1 Introduction

Attributed Graphs have been of crucial importance in pattern recognition throughout more than 3 decades [1], [2], [3], [4], [5] and [6]. If elements in pattern recognition are modelled through attributed graphs, error-tolerant graph-matching algorithms are needed that aim to compute a matching between nodes of two attributed graphs that minimizes some kind of objective function. Unfortunately, the time and space complexity to compute the minimum of these objective functions is very high. For this reason, some graph prototyping methods have appeared with the aim of reducing the runtime while querying a graph in a large database [7], [8], [9], [10], [11]. There are two interesting surveys in [2] and [3] about this subject. Recently, Fast Bipartite algorithm (FBP) [12] and Square Fast Bipartite algorithm (SFBP) [13] was presented that solve the graph-matching problem in a similar way. They are variants of Bipartite algorithm (BP) [14] but with a reduced runtime. The three algorithms are based on translating the error-tolerant graph-matching problem into a linear assignation

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

C.-L. Liu et al. (Eds.): GbRPR 2015, LNCS 9069, pp. 231–241, 2015.
DOI: 10.1007/978-3-319-18224-7_23

problem. They are composed of two steps. In the first one, a cost matrix is constructed with the information of both graphs to be compared. Each cell of the matrix represents the distance between a local sub-structure of one of the graphs and the local sub-structure of the other graph. In the second one, a linear assignation algorithm is applied to this matrix to obtain the best (sub-optimal) isomorphism between nodes of both graphs. Several linear assignation algorithms can be used, such as [15] or [16]. Recently, a new research field has appeared where the human or an expert system can interact on the graph matching algorithm to increase the accuracy of the obtained node correspondence [17] and [18].

In this paper, we compare five local sub-structures. The first three are strictly based on considering the local structure as a sub-graph. The other two are based on the spectral information centred at the involved nodes of both graphs.

Results show that the selected local sub-structure has a great impact on the obtained distance value and also on the runtime although in the past little research have been done. There is a related research in [19] that considers the two eigenvector centralities we propose and it studies their impact on the runtime and the accuracy to obtain the exact distance value. In our paper, we test the five centralities in well-known datasets and we obtain the recognition accuracy and also the runtime.

The outline of the paper is as follows. In the next section, we define Attributed graphs, Graph edit distance and we comment how to compute the Graph edit distance using Square Fast Bipartite algorithm (SFBP) [13]. In section 3, we present five local sub-structures and distances between them. In section 4, we show the experimental validation and finally, we conclude the article in section 5.

2 Graphs and Graph Edit Distance

In this section, we first define Attributed graphs with the concept of a neighbour of a node and Error-tolerant graph matching and then we explain the Graph edit distance.

Attributed Graphs

An Attributed graph is defined as a triplet $\mathrm{G} = (\Sigma_\mathrm{v}, \Sigma_\mathrm{e}, \gamma_\mathrm{v})$, where $\Sigma_\mathrm{v} = \{\mathrm{v_a} \mid \mathrm{a} = 1, \ldots, \mathrm{n}\}$ is the set of vertices and $\Sigma_\mathrm{e} = \{\mathrm{e_{ab}} | \mathrm{a}, \mathrm{b} \in 1, \ldots, \mathrm{n}\}$ is the set of undirected and unattributed edges. Function $\gamma_\mathrm{v}: \Sigma_\mathrm{v} \rightarrow \Delta_\mathrm{v}$ assigns attribute values in any domain to vertices. The order of graph G is n. We call $E(v_\mathrm{a})$ to the number of neighbours of node v_a. Finally, we define the neighbours of a node $\mathrm{v_a}$, named $\mathrm{N_a}$, on an attributed graph G, as another graph $\mathrm{N_a} = \left(\Sigma_\mathrm{v}^{\mathrm{N_a}}, \Sigma_\mathrm{e}^{\mathrm{N_a}}, \gamma_\mathrm{v}^{\mathrm{N_a}}\right)$ only composed of nodes connected to them by an edge. Formally, $\Sigma_\mathrm{v}^{\mathrm{N_a}} = \{\mathrm{v_a} | \mathrm{e_{ab}} \in \Sigma_\mathrm{e}\}$, $\Sigma_\mathrm{e}^{\mathrm{N_a}} = \Phi$ and $\gamma_\mathrm{v}^{\mathrm{N_a}}(v_a) = \gamma_\mathrm{v}(v_a)$, $\forall v_a \in \Sigma_\mathrm{v}^{\mathrm{N_a}}$. Finally, we represent as A the adjacency matrix such that $\mathrm{A}[\mathrm{a}, \mathrm{b}] = 1$ if $\mathrm{e_{ab}} \in \Sigma_\mathrm{e}$ and $\mathrm{A}[\mathrm{a}, \mathrm{b}] = 0$ otherwise.

Error Correcting Graph Isomorphism

Let $\mathrm{G^p} = (\Sigma_\mathrm{v}^\mathrm{p}, \Sigma_\mathrm{e}^\mathrm{p}, \gamma_\mathrm{v}^\mathrm{p})$ and $\mathrm{G^q} = (\Sigma_\mathrm{v}^\mathrm{q}, \Sigma_\mathrm{e}^\mathrm{q}, \gamma_\mathrm{v}^\mathrm{q})$ be two Attributed graphs of initial order n and m. To allow maximum flexibility in the matching process, graphs are extended with null nodes to be of order $n + m$. We refer to null nodes of $\mathrm{G^p}$ and $\mathrm{G^q}$ by

$\hat{\Sigma}_v^p \subseteq \Sigma_v^p$ and $\hat{\Sigma}_v^q \subseteq \Sigma_v^q$ respectively. We assume null nodes have indices $a \in [n+1, \dots, n+m\}$ and $i \in [m+1, \dots, n+m\}$ for graphs G^p and G^q, respectively. Let T be a set of all possible bijections between two vertex sets Σ_v^p and Σ_v^q. We define the non-existent or null edges as $\hat{\Sigma}_e^p \subseteq \Sigma_e^p$ and $\hat{\Sigma}_e^q \subseteq \Sigma_e^q$. Isomorphism $f^{p,q}: \Sigma_v^p \to \Sigma_v^q$, assigns one vertex of G^p to only one vertex of G^q. The isomorphism between edges is defined accordingly to the isomorphism of their terminal nodes.

Graph Edit Distance between Graphs

One of the most widely used methods to evaluate an error-correcting graph isomorphism is the Graph edit distance [1], [6], [20]. The dissimilarity is defined as the minimum amount of required distortion to transform one graph into the other. To this end, a number of distortion or edit operations, consisting of insertion, deletion and substitution of both nodes and edges are defined. Edit cost functions are introduced to quantitatively evaluate the edit operations. The basic idea is to assign a penalty cost to each edit operation according to the amount of distortion that it introduces in the transformation. Deletion and insertion operations are transformed to assignations of a non-null node of the first or second graph to a null node of the second or first graph. Substitutions simply indicate node-to-node assignations. Using this transformation, given two graphs G^p and G^q, and a bijection between their nodes, $f^{p,q}$, the graph edit cost is given by:

$$EditCost_{K_v,K_e}(G^p, G^q, f^{p,q}) = \sum_{\substack{v_a^p \in \Sigma_v^p - \hat{\Sigma}_v^p \\ v_i^q \in \Sigma_v^q - \hat{\Sigma}_v^q}} C_{vs}\left(v_a^p, v_i^q\right) + \sum_{\substack{v_a^p \in \Sigma_v^p - \hat{\Sigma}_v^p \\ v_i^q \in \hat{\Sigma}_v^q}} K_v + \sum_{\substack{v_a^p \in \hat{\Sigma}_v^p \\ v_i^q \in \Sigma_v^q - \hat{\Sigma}_v^q}} K_v + \sum_{\substack{e_{ab}^p \in \Sigma_e^p - \hat{\Sigma}_e^p \\ e_{ij}^q \in \hat{\Sigma}_e^q}} K_e + \sum_{\substack{e_{ab}^p \in \hat{\Sigma}_e^p \\ e_{ij}^q \in \Sigma_e^q - \hat{\Sigma}_e^q}} K_e \quad (1)$$

Where $f^{p,q}\left(v_a^p\right) = v_i^q$ and $f_e^{p,q}\left(e_{ai}^p\right) = e_{ij}^q$

where C_{vs} is a function that represents the cost of substituting node v_a^p of G^p by node $f^{p,q}\left(v_a^p\right)$ of G^q. Constant K_v is the cost of deleting node v_a^p of G^p or inserting node v_i^q of G^q. Likewise for the edges, K_e is the cost of assigning edge e_{ab}^p of G^p to a non-existing edge of G^q or assigning edge e_{ab}^q of G^q to a non-existing edge of G^p. Note that we have not considered the cases in which two null nodes or null arcs are mapped, this is because this cost is zero by definition. In the same way, we do not have considered the cost of substituting two edges since they are unattributed and so, its substitution has a null cost. Note the definitions exposed in this paper can be easily generalised by adding attributes on edges.

The Graph edit distance is defined as the minimum cost under any bijection in T:

$$EditDist_{K_v,K_e}(G^p, G^q) = \min_{f^{p,q} \in T}\left\{EditCost_{K_v,K_e}(G^p, G^q, f^{p,q})\right\} \quad (2)$$

The assignment problem considers the task of finding an optimal assignment of the elements of a set P to the elements of another set Q, where both sets have the same cardinality $N = |P| = |Q|$. Let us assume there is a NXN cost matrix C. The matrix elements $C_{i,j}$ correspond to the cost of assigning the i-th element of P to the j-th element of Q. An optimal linear assignment is the one that minimises the sum of the assignment costs and so, the assignment problem can be stated as finding the permutation p that minimises $\sum_{i=1}^{N} C_{i,p(i)}$. There are several algorithms that solve the linear assignation problem [15], [16]. In the worst case, the maximum number of operations needed by these algorithms is $O(N^3)$.

Square Fast Bipartite algorithm (SFBP) [13] is an efficient algorithm to Edit Distance computation for general graphs that in a first step generates a matrix costs and in a second step, applies an optimal linear assignment algorithm on this matrix. The algorithm is similar to Bipartite BP but with a different cost matrix. In fact, SFBP defines two matrices, depending on the order of the graphs (BP only defines one matrix).

If $m \geq n$ then the cost matrix is,

$$C_{m\geq n}^{SFBP} = \boldsymbol{m}\left\{\overbrace{\begin{bmatrix} C_{1,1} & C_{1,2} & \dots & C_{1,m} \\ C_{2,1} & C_{2,2} & & C_{2,m} \\ \vdots & \vdots & & \vdots \\ C_{n,1} & C_{n,2} & \dots & C_{n,m} \\ \hline C_{\varepsilon,1} & C_{\varepsilon,2} & \dots & C_{\varepsilon,m} \\ C_{\varepsilon,1} & C_{\varepsilon,2} & & C_{\varepsilon,m} \\ \vdots & \vdots & & \vdots \\ C_{\varepsilon,1} & C_{\varepsilon,2} & \dots & C_{\varepsilon,m} \end{bmatrix}}^{\boldsymbol{m}}\right.$$

If $m \leq n$ then the cost matrix is,

$$C_{m\leq n}^{SFBP} = \boldsymbol{n}\left\{\overbrace{\left[\begin{array}{ccccc|ccc} C_{1,1} & C_{1,2} & \dots & C_{1,m} & C_{1,\varepsilon} & C_{1,\varepsilon} & \dots & C_{1,\varepsilon} \\ C_{2,1} & C_{2,2} & & C_{2,m} & C_{2,\varepsilon} & C_{2,\varepsilon} & & C_{2,\varepsilon} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ C_{n,1} & C_{n,2} & \dots & C_{n,m} & C_{n,\varepsilon} & C_{n,\varepsilon} & \dots & C_{n,\varepsilon} \end{array}\right]}^{\boldsymbol{n}}\right.$$

Where $C_{a,i}$ denotes the cost of substituting nodes v_a^p and v_i^q and its local sub-structure. $C_{a,\varepsilon}$ denotes the cost of deleting node v_a^p and its local sub-structure, and $C_{\varepsilon,i}$ denotes the cost of inserting node v_i^q and its local sub-structure. Obviously, as described in [14], this minimum cost is a sub-optimal Edit distance value between the involved graphs since cost matrix rows are related to local sub-structures of graph G^p and columns are related to local sub-structures of G^q. The computational cost of the SFBP is $O((\max(n,m))^3)$. Note that the linear assignment algorithms used in its original form are optimal for solving the assignment problem, but they are suboptimal

for finding the Graph edit distance. This is due to the fact that nodes and their local sub-structure are considered individually. If f_m^{p,q^*} is the obtained isomorphism through method or algorithm m, then we have that $EditCost_{K_v,K_e}(G^p, G^q, f_m^{p,q^*}) \geq EditDist_{K_v,K_e}(G^p, G^q)$. In some cases, it is not possible to compute the real distance value $EditDist_{K_v,K_e}(G^p, G^q)$ for runtime reasons due to it has to be computed through an A^* algorithm. In these cases, if we want to evaluate two suboptimal methods to compute the Graph edit distance, the lower the cost, the better the method.

3 Node Centralities and Distances between Them

In this section, different methods are proposed to obtain values $C_{a,i}$, $C_{a,\varepsilon}$ and $C_{\varepsilon,i}$ in the cost matrices used by SFBP algorithm to compute the isomorphism between nodes. The whole values on the cost matrices depend on two weighted disjoint costs. The first one only depends on the nodes and the second one depends on the rest of the local sub-structure.

When the original nodes $v_a^p \in \Sigma_v^p - \hat{\Sigma}_v^p$ and $v_i^q \in \Sigma_v^q - \hat{\Sigma}_v^q$ are mapped:

$$C_{a,i} = \beta \cdot C_{vs}(v_a^p, v_i^q) + (1-\beta) \cdot C_{cs}(v_a^p, v_i^q) \tag{3}$$

When the original node $v_a^p \in \Sigma_v^p - \hat{\Sigma}_v^p$ in G^p is deleted and so mapped to a null node $v_i^q \in \hat{\Sigma}_v^q$ in G^q:

$$C_{a,\varepsilon} = \beta \cdot k_v + (1-\beta) \cdot C_{cd}(v_a^p, v_i^q) \tag{4}$$

When the original node $v_i^q \in \Sigma_v^q - \hat{\Sigma}_v^q$ in G^q is inserted and so mapped from a null node $v_a^p \in \hat{\Sigma}_v^p$ in G^p:

$$C_{\varepsilon,i} = \beta \cdot k_v + (1-\beta) \cdot C_{ci}(v_a^p, v_i^q) \tag{5}$$

As commented in section 2, C_{vs} is a distance function defined through the node attribute values and k_v gauges the importance of deleting or inserting nodes in the matching process. C_{cs} is the cost to substitute the local sub-structure and C_{cd} and C_{ci} are the costs to delete and insert it, respectively. These costs depend on the used local sub-structures. Finally, the weighting parameter β has to be set in a validation or learning process.

We propose the following five node centralities and distances:

-**The degree**: The local sub-structure is composed of a node and its connected arcs. Although it is easily generalizable, in this paper we have considered edges do not have attributes. Therefore, these costs are based on counting the number of edges,

$$\begin{aligned} &C_{cs}(v_a^p, v_i^q) = k_e \cdot \left|E(v_a^p) - E(v_i^q)\right| \text{ where } v_a^p \in \Sigma_v^p - \hat{\Sigma}_v^p \text{ and } v_i^q \in \Sigma_v^q - \hat{\Sigma}_v^q. \\ &C_{cd}(v_a^p, v_i^q) = k_e \cdot E(v_a^p) \text{ where } v_a^p \in \Sigma_v^p - \hat{\Sigma}_v^p \text{ and } v_i^q \in \hat{\Sigma}_v^q. \\ &C_{ci}(v_a^p, v_i^q) = k_e \cdot E(v_i^q) \text{ where } v_a^p \in \hat{\Sigma}_v^p \text{ and } v_i^q \in \Sigma_v^q - \hat{\Sigma}_v^q. \end{aligned} \tag{6}$$

-**The clique**: The local sub-structure is composed of a node, its arcs and the connected nodes. Clearly, we could define more levels of complexity but we decided not to consider them. Some excluded examples are the clique plus the edges of the neighbouring nodes, since the combinations of structures exponentially explode. To increase the complexity of the local sub-structure but not the computational complexity, the spectral centralities are proposed. In the clique structure, the costs on the centralities are defined as follows,

$$C_{cs}\left(v_a^p, v_i^q\right) = EditDistance_{k_v+k_e,0}\left(N_a^p, N_i^q\right) \quad \text{where } v_a^p \in \Sigma_v^p - \hat{\Sigma}_v^p \text{ and } v_i^q \in \Sigma_v^q - \hat{\Sigma}_v^q. \tag{7}$$

Function $EditDistance$ is computed through SFBP in the same way it is done with graphs. A neighbour node and its connecting edge have to be seen as an indivisible structure. Note, the neighbour structures N_a^p and N_i^q are defined as non-connected graphs with only the neighbouring nodes and without edges. Yet, the centrality cost has to consider the cost of the edges that connect the central node with the neighbouring nodes. To do so, the cost of deleting and inserting nodes in function $EditDistance$ is defined as $k_v + k_e$. Due to there are no edges, any cost of deleting and inserting edges could be set at function $EditDistance$ and we imposed 0.

The deletion and insertion centrality costs depend on the number of neighbours,

$$\begin{aligned} C_{cd}\left(v_a^p, v_i^q\right) &= (k_v + k_e) \cdot E\left(v_a^p\right) \text{ where } v_a^p \in \Sigma_v^p - \hat{\Sigma}_v^p \text{ and } v_i^q \in \hat{\Sigma}_v^q. \\ C_{ci}\left(v_a^p, v_i^q\right) &= (k_v + k_e) \cdot E\left(v_i^q\right) \text{ where } v_a^p \in \hat{\Sigma}_v^p \text{ and } v_i^q \in \Sigma_v^q - \hat{\Sigma}_v^q. \end{aligned} \tag{8}$$

-**The planar**: The local sub-structure is exactly the same as the clique but in this case the relative position is considered. Costs C_{cd} and C_{ci} are defined in the same way as above. The difference resides in C_{cs}. The only allowed isomorphisms $f^{N_a^p, N_i^q}$ are the ones that are generated from cyclic combinations of the neighbours. Therefore, sets N_a^p and N_i^q are seen as strings and the cost is computed through the Cyclic Levenshtein distance [21], [22].

$$C_{cs}\left(v_a^p, v_i^q\right) = CyclicLevensh\left(N_a^p, N_i^q, k_v + k_e\right) \quad \text{where } v_a^p \in \Sigma_v^p - \hat{\Sigma}_v^p \;\&\; v_i^q \in \Sigma_v^q - \hat{\Sigma}_v^q. \tag{9}$$

Figure 1 shows an example of the Clique and Planar centralities. Red lines represent the optimal correspondence while considering Clique centrality and green lines while considering Planar centrality. Numbers on the nodes are the attributes $\gamma_v^p\left(v_a^p\right)$ or $\gamma_v^q\left(v_i^q\right)$. Edges do not have attributes. Suppose $C_{cs}\left(v_a^p, v_i^q\right) = \left|\gamma_v^p\left(v_a^p\right) - \gamma_v^q\left(v_a^q\right)\right|$. In the Clique case (red lines), $C_{cs}\left(v_a^p, v_i^q\right) = 1 + 0 + 1 + 0 = 2$. In the Planar case (green lines), $C_{cs}\left(v_a^p, v_i^q\right) = 1 + 17 + 1 + 17 = 36$. Note the restriction to be the labelling cyclic makes the distance value to be larger.

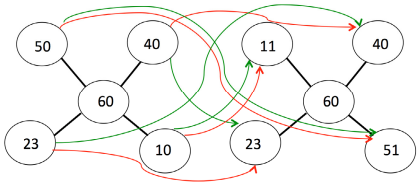

Fig. 1. In red: optimal correspondence in Clique centrality. In green: optimal correspondence in Planar centrality

-**The Eigenvector**: The local sub-structure takes into consideration the degree of the neighbouring nodes considering the eigenvector that has the largest eigenvalue. The eigenvector centralities given a specific node in both graphs are defined by,

$$c_a^p = \frac{1}{\lambda_1^p} \cdot \sum_{v_{a'}^p \in N_a^p} \left| V_{a'}^p \right| \quad \text{and} \quad c_i^q = \frac{1}{\lambda_1^q} \cdot \sum_{v_{i'}^q \in N_i^q} \left| V_{i'}^q \right| \tag{10}$$

where $V_{a'}^p$ and $V_{i'}^q$ are the values of the a'-th and i'-th positions of the eigenvectors with the largest eigenvalues obtained through adjacency matrices A^p and A^q. Besides, λ_1^p and λ_1^q are the largest eigenvalues of these adjacency matrices. Thus, the substitution cost is simply computed as,

$$\begin{aligned} &C_{cs}\left(v_a^p, v_i^q\right) = \left| c_a^p - c_i^q \right| \\ &\text{where } v_a^p \in \Sigma_v^p - \hat{\Sigma}_v^p \;\&\; v_i^q \in \Sigma_v^q - \hat{\Sigma}_v^q. \end{aligned} \tag{11}$$

The deletion and insertion costs are computed assuming that the centrality of a null node is 0.

$$\begin{aligned} C_{cd}\left(v_a^p, v_i^q\right) &= c_a^p \text{ where } v_a^p \in \Sigma_v^p - \hat{\Sigma}_v^p \text{ and } v_i^q \in \hat{\Sigma}_v^q. \\ C_{ci}\left(v_a^p, v_i^q\right) &= c_i^q \text{ where } v_a^p \in \hat{\Sigma}_v^p \text{ and } v_i^q \in \Sigma_v^q - \hat{\Sigma}_v^q. \end{aligned} \tag{12}$$

-**The PageRank**: This centrality is a variation of the Eigenvector centrality. The difference consists in each eigenvector element is normalised by the number of neighbours of the node it represents.

$$c_a^p = \frac{1}{\lambda_1^p} \cdot \sum_{v_{a'}^p \in N_a^p} \frac{\left| V_{a'}^p \right|}{\max\left(1, E\left(V_{a'}^p\right)\right)} \quad \text{and} \quad C_i^q = \frac{1}{\lambda_1^q} \cdot \sum_{v_{i'}^q \in N_i^q} \frac{\left| V_{i'}^q \right|}{\max\left(1, E\left(V_{i'}^q\right)\right)} \tag{13}$$

This centrality is normalised by $\max\left(1, E\left(V_{a'}^p\right)\right)$ instead of $E\left(V_{a'}^p\right)$ to avoid dividing by 0. Then, the substitution, insertion and deletion costs are computed in a similar why than the Eigenvector centrality but using the PageRank centrality.

4 Experimental Validation

Table 1. Recognition ratio of the five proposed centralities given the 5 datasets and different combinations of parameters

	Kv	*Ke*	*β*	*Degree*	*Clique*	*Planar*		*Kv*	*β*	*Eigen Vector*	*Pagerank*
LETTER LOW	**1**	**1**	**0.5**	0.95	0.98	0.98		**1**	**0.9**	0.94	0.94
	1	**0.1**	**0.5**	0.96	**0.99**	0.98		**1**	**0.5**	0.96	0.96
	0.1	**1**	**0.5**	0.96	0.98	0.98		**1**	**0.1**	0.93	0.93
	0.1	**0.1**	**0.5**	0.98	**0.99**	**0.99**		**1**	**0.01**	0.92	0.92
LETTER MED	**1**	**1**	**0.5**	0.87	**0.94**	**0.94**		**1**	**0.9**	0.66	0.65
	1	**0.1**	**0.5**	0.73	**0.94**	**0.94**		**1**	**0.5**	0.85	0.84
	0.1	**1**	**0.5**	0.86	0.89	0.89		**1**	**0.1**	0.84	0.84
	0.1	**0.1**	**0.5**	0.18	0.09	0.09		**1**	**0.01**	0.83	0.83
LETTER HIGH	**1**	**1**	**0.5**	0.79	0.88	**0.90**		**1**	**0.9**	0.65	0.65
	1	**0.1**	**0.5**	0.74	0.88	**0.90**		**1**	**0.5**	0.72	0.71
	0.1	**1**	**0.5**	0.81	0.80	0.80		**1**	**0.1**	0.71	0.74
	0.1	**0.1**	**0.5**	0.38	0.07	0.07		**1**	**0.01**	0.59	0.61
GREC	**10**	**10**	**0.5**	0.96	0.84	0.83		**10**	**0.9**	0.71	0.71
	10	**5**	**0.5**	0.92	0.65	0.63		**10**	**0.5**	0.93	0.93
	5	**10**	**0.5**	0.95	0.39	0.37		**10**	**0.1**	**0.99**	**0.99**
	5	**5**	**0.5**	0.87	0.14	0.14		**10**	**0.01**	0.97	0.96
AIDS	**1**	**1**	**0.5**	0.96	0.84	0.84		**1**	**0.9**	0.83	0.83
	1	**0.1**	**0.5**	0.83	0.82	0.82		**1**	**0.5**	0.91	0.92
	0.1	**1**	**0.5**	0.90	0.82	0.82		**1**	**0.1**	**0.99**	**0.99**
	0.1	**0.1**	**0.5**	0.82	0.81	0.81		**1**	**0.01**	**0.99**	**0.99**

Table 1 and table 2 show the recognition ratio and mean runtime of the five centralities described in section 3 for graph classification using the five graph databases LETTER LOW, LETTER MEDIUM, LETTER HIGH, GREC and AIDS from the IAM repository [23]. Each database consists on a set of different graph instances divided in different classes where each class is composed of a reference set and a test set. We used the K-NN classifier where K = 3 and the graph-matching algorithm SFBP summarised in section 2 with Jonker-Volgenant solver [16]. Note the computation of the edit operations on local structures C_{cs}, C_{cd} and C_{ci} depends on the following parameters: In case of Degree, Clique and Planar, parameters are k_v, k_e and β. In case of Eigenvector and PageRank, parameters are k_v and β. Best results for each database are marked in bold and underlined.

Table 2. Mean runtime spent for a graph classification of the 5 proposed centralities given the 5 datasets and different parameters (I7 3.07 Ghz, Windows, Matlab 2014a)

	Kv	*Ke*	*β*	*Degree*	*Clique*	*Planar*		*Kv*	*β*	*Eigen Vector*	*Pagerank*
LETTER LOW	**1**	**1**	**0.5**	0.61	3.14	0.95		**1**	**0.9**	0.58	0.59
	1	**0.1**	**0.5**	0.56	3.15	0.90		**1**	**0.5**	0.62	0.62
	0.1	**1**	**0.5**	**0.55**	3.09	0.85		**1**	**0.1**	0.76	0.79
	0.1	**0.1**	**0.5**	0.48	3.15	0.80		**1**	**0.01**	2.04	2.30
LETTER MED	**1**	**1**	**0.5**	0.61	3.13	0.94		**1**	**0.9**	0.57	0.58
	1	**0.1**	**0.5**	0.56	3.16	0.91		**1**	**0.5**	0.63	0.64
	0.1	**1**	**0.5**	**0.55**	3.09	0.86		**1**	**0.1**	0.77	0.81
	0.1	**0.1**	**0.5**	0.46	3.16	0.81		**1**	**0.01**	2.04	2.38
LETTER HIGH	**1**	**1**	**0.5**	0.61	3.68	1.24		**1**	**0.9**	0.58	0.59
	1	**0.1**	**0.5**	0.60	3.60	1.26		**1**	**0.5**	0.65	0.69
	0.1	**1**	**0.5**	**0.57**	3.52	1.17		**1**	**0.1**	0.84	0.88
	0.1	**0.1**	**0.5**	0.47	3.51	1.10		**1**	**0.01**	2.40	2.71
GREC	**10**	**10**	**0.5**	0.45	7.34	1.99		**10**	**0.9**	0.49	0.48
	10	**5**	**0.5**	**0.42**	7.48	2.01		**10**	**0.5**	0.53	0.52
	5	**10**	**0.5**	0.42	7.35	1.99		**10**	**0.1**	0.99	0.97
	5	**5**	**0.5**	0.40	7.34	1.97		**10**	**0.01**	4.10	4.35
AIDS	**1**	**1**	**0.5**	0.64	6.29	1.78		**1**	**0.9**	0.41	0.41
	1	**0.1**	**0.5**	0.37	6.27	1.76		**1**	**0.5**	0.55	0.59
	0.1	**1**	**0.5**	0.47	5.94	1.60		**1**	**0.1**	3.04	39.23
	0.1	**0.1**	**0.5**	**0.27**	6.12	1.65		**1**	**0.01**	23.73	21.41

Table 3 summarises the achievements that can be deducted from tables 1 and 2. We realise that the structure that obtains higher recognition ratio clearly depends on the database. We know from [23] that graphs in GREC and AIDS have more variability than graphs in the three LETTER databases. From the runtime point of view, it is clear that Degree is the fastest structure. Considering edit costs, lower are values of k_v and k_e, lower is the runtime. Contrarily, lower is β, higher is the runtime.

Table 3. Summary of the achievements of tables 1 and 2

	Highest recognition ratio	**Lowest runtime**
LETTER LOW	Clique, Planar	Degree
LETTER MEDIUM	Clique, Planar	Degree
LETTER HIGH	Planar	Degree
GREC	Eigenvector/PageRank	Degree
AIDS	Eigenvector/PageRank	Degree

5 Conclusions

In this paper, we have shown the relevance of the node centrality on graph edit distance for graph classification. We have presented five different centralities, viz. Clique, Degree, Planar, Eigenvector and PageRank. We have defined them and we have shown their recognition ratio and runtime in 5 different databases and different configurations of Edit costs. With respect to the runtime, it is clear that the Degree is the best. But it is not so clear which one is the best with respect to the recognition ratio. It seems that the method that presents a best balance between recognition ratio and runtime is the Degree centrality but clearly, it depends on the application or database. For this reason, the decision to use a particular centrality can require an accurate application-dependent analysis to maximize the specific goals.

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 Recognition 35(3), 639–650 (2002)
2. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty Years Of Graph Matching In Pattern Recognition. IJPRAI 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: Kropatsch, W.G., Artner, N.M., Haxhimusa, Y., Jiang, X. (eds.) GbRPR 2013. LNCS, vol. 7877, pp. 1–10. Springer, Heidelberg (2013)
4. Hancock, E.R., Wilson, R.C.: Pattern analysis with graphs: Parallel work at Bern and York. Pattern Recognition Letters 33(7), 833–841 (2012)
5. Serratosa, F., Cortés, X., Solé-Ribalta, A.: Component Retrieval based on a Database of Graphs for Hand-Written Electronic-Scheme Digitalisation. Expert Systems With Applications, ESWA 40, 2493–2502 (2013)
6. Solé, A., Serratosa, F., Sanfeliu, A.: On the Graph Edit Distance cost: Properties and Applications. International Journal of Pattern Recognition and Artificial Intelligence 26(5) (2012)
7. Serratosa, F., Alquézar, R., Sanfeliu, A.: Estimating the Joint Probability Distribution of Random Vertices and Arcs by Means of Second-Order Random Graphs. In: Caelli, T.M., Amin, A., Duin, R.P.W., Kamel, M.S., de Ridder, D. (eds.) SPR 2002 and SSPR 2002. LNCS, vol. 2396, pp. 252–262. Springer, Heidelberg (2002)
8. Ferrer, M., Valveny, E., Serratosa, F.: Median graphs: A genetic approach based on new theoretical properties. Pattern Recognition 42(9), 2003–2012 (2009)
9. Ferrer, M., Valveny, E., Serratosa, F.: Median graph: A new exact algorithm using a distance based on the maximum common subgraph. Pattern Recognition Letters 30(5), 579–588 (2009)
10. Serratosa, F., Alquézar, R., Sanfeliu, A.: Estimating the Joint Probability Distribution of Random Vertices and Arcs by Means of Second-Order Random Graphs. In: Caelli, T.M., Amin, A., Duin, R.P.W., Kamel, M.S., de Ridder, D. (eds.) SPR 2002 and SSPR 2002. LNCS, vol. 2396, pp. 252–262. Springer, Heidelberg (2002)
11. Serratosa, F., Sanfeliu, A.: Function-Described Graphs applied to 3D object recognition. In: Del Bimbo, A. (ed.) ICIAP 1997. LNCS, vol. 1310, pp. 701–708. Springer, Heidelberg (1997)

12. Serratosa, F.: Fast Computation of Bipartite Graph Matching. Pattern Recognition Letters 45, 244–250 (2014)
13. Serratosa, F.: Speeding up Fast Bipartite Graph Matching trough a new cost matrix, International Journal of Pattern Recognition and Artificial Intelligence 29(2) (2015)
14. Riesen, K., Bunke, H.: Approximate graph edit distance computation by means of bipartite graph matching. Image Vision Comput. 27(7), 950–959 (2009)
15. Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the Society for Industrial & Applied Mathematics 5, 32–38 (1957)
16. Jonker, R., Volgenant, T.: A shortest augmenting path algorithm for dense and sparse linear assignment problems. Computing 38, 325–340 (1987)
17. Cortés, X., Serratosa, F.: An Interactive Method for the Image Alignment problem based on Partially Supervised Correspondence. Expert Systems With Applications 42(1), 179–192 (2015)
18. Serratosa, F., Cortés, X.: Interactive Graph-Matching using Active Query Strategies. Pattern Recognition 48, 1360–1369 (2015)
19. Riesen, K., Bunke, H., Fischer, A.: Improving Graph Edit Distance Approximation by Centrality Measures. In: International Congress on Pattern Recognition (2014)
20. Cortés, X., Serratosa, F.: Learning Graph-Matching Edit-Costs based on the Optimality of the Oracle's Node Correspondences. Pattern Recognition Letters (2015)
21. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady, Cybernetics and Control Theory 10, 707–710 (1966)
22. Peris, G., Marzal, A.: Fast Cyclic Edit Distance Computation with Weighted Edit Costs in Classification. In: ICPR 2002, vol. 4, pp. 184–187 (2002)
23. 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)