Edit Distance Computed by Fast Bipartite Graph Matching[*]

Francesc Serratosa and Xavier Cortés

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

Abstract. We present a new algorithm to compute the Graph Edit Distance in a sub-optimal way. We demonstrate that the distance value is exactly the same than the one obtained by the algorithm called Bipartite but with a reduced run time. The only restriction we impose is that the edit costs have to be defined such that the Graph Edit Distance can be really defined as a distance function, that is, the cost of insertion plus deletion of nodes (or arcs) have to be lower or equal than the cost of substitution of nodes (or arcs). Empirical validation shows that higher is the order of the graphs, higher is the obtained Speed up.

Keywords: Graph Edit Distance, Bipartite Graph Matching, Munkres' algorithm.

1 Introduction

Attributed Graphs have been of crucial importance in pattern recognition throughout more than 3 decades [1], [2], [3], [4], [5], [6], [7], [8], [9] and [10]. 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 run time while querying a graph in a large database [11], [12], [13].

Since its presentation, Bipartite algorithm [14] has been considered one of the best graph-matching algorithms due to it obtains a sub-optimal distance value almost near to the optimal one but with a considerable decrease on the run time. Other algorithms are [15] or [16]. There is an interesting survey in [2].

This paper presents a new algorithm that obtains exactly the same distance value of the Bipartite algorithm but with a reduced run time. The only restriction we impose is that the edit costs have to be defined such that the Graph Edit Distance can be really defined as a distance function, that is, the cost of insertion plus deletion of nodes (or arcs) have to be lower or equal than the cost of substitution of nodes (or arcs). Experimental validation shows a Speed up of 5 on well-known databases. In fact,

[*] This research is supported by the CICYT project DPI2013-42458-P.

higher is the order of graphs, higher is also the Speed up of our algorithm. This property is interesting since in the next years, we will see a need on representing the objects (social nets, scenes, proteins…) on larger structures.

The outline of the paper is as follows, in the next section, we define the attributed graphs and the graph-edit distance. On section 3, we explain how to compute the graph edit distance using the Bipartite algorithm. Finally, on section 4, we present our new method and we schematically show our algorithm. On section 5, we show the experimental validation and we finish the article with some conclusions.

2 Graphs and Graph Edit Distance

In this section, we first define the Attributed Graphs, Cliques and Graph matching and then we explain the Graph Edit Distance.

Attributed Graph and Cliques

Let $\Delta_v$and $\Delta_e$denote the domains of possible values for attributed vertices and arcs, respectively. An attributed graph (over Δ_v and Δ_e) is defined by a tuple $G = (\Sigma_v, \Sigma_e, \gamma_v, \gamma_e)$, where $\Sigma_v = \{v_a \mid a = 1, \dots, n\}$ is the set of vertices (or nodes), $\Sigma_e = \{e_{ab} | a, b \in 1, \dots, n\}$ is the set of arcs (or edges), $\gamma_v: \Sigma_v \to \Delta_v$ assigns attribute values to vertices and $\gamma_e: \Sigma_e \to \Delta_e$ assigns attribute values to arcs. The order of graph G is n.

We define a clique K_a on an attributed graph G as a local structure composed of a node and its outgoing edges $K_a = (v_a, \{e_{ab} | b \in 1, \dots, n\}, \gamma_v, \gamma_e)$.

Error Correcting Graph Isomorphism

Let $G^p = (\Sigma_v^p, \Sigma_e^p, \gamma_v^p, \gamma_e^p)$ and $G^q = (\Sigma_v^q, \Sigma_e^q, \gamma_v^q, \gamma_e^q)$ be two attributed graphs of initial order n and m. To allow maximum flexibility in the matching process, graphs are extended with null nodes [17] to be of order $n + m$. We will refer to null nodes of G^p and 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 by $\hat{\Sigma}_e^p \subseteq \Sigma_e^p$ and $\hat{\Sigma}_e^q \subseteq \Sigma_e^q$.

Bijection $f^{p,q}: \Sigma_v^p \to \Sigma_v^q$, assigns one vertex of G^p to only one vertex of G^q. The bijection between arcs, denoted by $f_e^{p,q}$, is defined accordingly to the bijection of their terminal nodes.

Graph Edit Distance between Two Graphs

One of the most widely used methods to evaluate an error-correcting graph isomorphism is the Graph Edit Distance [18, 19, 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. Then, for every pair of graphs (G^p and G^q), there is a sequence of edit operations, or an edit path $editPath(G^p, G^q) = (\varepsilon_1, \dots, \varepsilon_k)$ (where each ε_i denotes an edit operation) that transform one graph into the other. In general, several edit paths may exist between two given graphs. This set of edit paths is denoted by ϑ. To quantitatively evaluate which edit path is the best, edit cost functions are introduced. 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.

Each $editPath(G^p, G^q) \in \vartheta$ can be related to a univocal graph isomorphism $f^{p,q} \in T$ between the involved graphs. In this way, each edit operation assigns a node of the first graph to a node of the second graph. 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 (Definition 7 of [21]):

$$
\begin{gathered}
EditCost(G^p, G^q, f^{p,q}) = \\
\sum_{\substack{v_a^p \in \Sigma_v^p - \hat{\Sigma}_v^p \\ v_i^q \in \Sigma_v^q - \hat{\Sigma}_v^q}} C_{vs}(v_a^p, v_i^q) + \sum_{\substack{v_a^p \in \Sigma_v^p - \hat{\Sigma}_v^p \\ v_i^q \in \hat{\Sigma}_v^q}} C_{vd}(v_a^p, v_i^q) + \\
\sum_{\substack{v_a^p \in \hat{\Sigma}_v^p \\ v_i^q \in \Sigma_v^q - \hat{\Sigma}_v^q}} C_{vi}(v_a^p, v_i^q) + \sum_{\substack{e_{ab}^p \in \Sigma_e^p - \hat{\Sigma}_e^p \\ e_{ij}^q \in \Sigma_e^q - \hat{\Sigma}_e^q}} C_{es}(e_{ab}^p, e_{ij}^q) + \\
\sum_{\substack{e_{ab}^p \in \Sigma_e^p - \hat{\Sigma}_e^p \\ e_{ij}^q \in \hat{\Sigma}_e^q}} C_{ed}(e_{ab}^p, e_{ij}^q) + \sum_{\substack{e_{ab}^p \in \hat{\Sigma}_e^p \\ e_{ij}^q \in \Sigma_e^q - \hat{\Sigma}_e^q}} C_{ei}(e_{ab}^p, e_{ij}^q)
\end{gathered}
$$

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

where C_{vs} is the cost of substituting node v_a^p of G^p for node $f^{p,q}(v_a^p)$ of G^q, C_{vd} is the cost of deleting node v_a^p of G^p and C_{vi} is the cost of inserting node v_i^q of G^q. Equivalently for edges, C_{es} is the cost of substituting edge e_{ab}^p of graph G^p for edge $f_e^{p,q}(e_{ab}^p)$ of G^q, C_{ed} is the cost of assigning edge e_{ab}^p of G^p to a non-existing edge of G^q and C_{ei} is the cost of 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.

Finally, the Graph Edit Distance is defined as the minimum cost under any bijection in T:

$$EditDist(G^p, G^q) = \min_{f^{p,q} \in T} EditCost(G^p, G^q, f^{p,q})$$

Using this definition, the Graph Edit Distance essentially depends on C_{vs}, C_{vd}, C_{vi}, C_{es}, C_{ed} and C_{ei} functions and several definitions of these functions exist.

We say the optimal bijection, $f^{p,q*}$, is the one that obtains the minimum cost,

$$f^{p,q*} = \operatorname*{argmin}_{f^{p,q} \in T} EditCost(G^p, G^q, f^{p,q})$$

We define the distance and the optimal bijection between two cliques in a similar way as the distance between two graphs since they are local structures of graphs. We name the cost of substituting clique K_a^p by K_i^q as $\boldsymbol{C}_{a,i}$. The cost of deleting clique K_a^p as $\boldsymbol{C}_{a,\varepsilon}$ and the cost of inserting clique K_i^q as $\boldsymbol{C}_{\varepsilon,i}$.

3 Edit Distance Computation by Bipartite Algorithm (BP)

The assignment problem considers the task of finding an optimal assignment of the elements of a set A to the elements of another set B, where both sets have the same cardinality $n = |A| = |B|$. 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 A to the j-th element of B. An optimal 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 $\sum_{i=1}^{n} C_{i,p(i)}$. Munkres' algorithm [22] solves the assignation problem. It is a refinement of an earlier version by Kuhn [23] and is also referred to as Kuhn-Munkres or Hungarian algorithm. The algorithm repeatedly finds the maximum number of independent optimal assignments and in the worst case the maximum number of operations needed by the algorithm is $O(n^3)$. Later, an algorithm to solve this problem applied to non-square matrices where presented [24].

Bipartite, or BP for short [14], is an efficient algorithm for edit distance computation for general graphs that use the Munkres' algorithm. That is, they generalised the original Munkres' algorithm that solve the assignment problem to the computation of the graph edit distance by defining a specific cost matrix. In experiments on artificial and real-world data, authors demonstrate BP obtains an important speed-up of the computation respect other methods while at the same time the accuracy of the approximated distances is not much affected [14]. For this reason, since its publication, it has become one of the most used graph-matching algorithms.

Given attributed graphs G^p and G^q, the $(n+m)x(n+m)$ cost matrix C is defined as follows,

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

Where $\boldsymbol{C}_{a,i}$ denotes the cost of substituting clique K_a^p by K_i^q, $\boldsymbol{C}_{a,\varepsilon}$ denotes the cost of deleting clique K_a^p and $\boldsymbol{C}_{\varepsilon,i}$ denotes the cost of inserting clique K_i^q. On the basis of this cost matrix definition, Munkres' algorithm can be executed to find the minimum cost for all permutations. 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 cliques of graph G^p and columns are related to cliques of G^q. Moreover, it is considered a *correct permutation* the one that $\sum_{a=1}^{n+m} \mathbf{C}_{a,p(a)} < \infty$. That is, all costs are assigned to non-infinitive values.

Note that Munkres' algorithm used in its original form is optimal for solving the assignment problem, but it is suboptimal for solving the graph edit distance. This is due to the fact that cliques are considered individually. The distance values obtained by this method are equal to or smaller than the distance values obtained in an optimal method (with exponential cost). The computational cost of the method is $O((n+m)^3)$.

4 Fast Bipartite Algorithm (FBP)

We define the following edit cost,

$$EditCost'^{(G^p,G^q,f^{p,q})} = \\ EditCost(G^p,G^q,f^{p,q}) - C_{ve\_di}(G^p,G^q)$$

Where,

$$C_{ve\_di}(G^p,G^q) = \sum_{v_a^p \in \Sigma_v^p} C_{vd}(v_a^p, v_\phi^q) + \sum_{v_i^q \in \Sigma_v^q} C_{vi}(v_\phi^p, v_i^q) + \\ + \sum_{e_{ab}^p \in \Sigma_e^p} C_{ed}(e_{ab}^p, e_\phi^q) + \sum_{e_{ij}^q \in \Sigma_e^q} C_{ei}(e_\phi^p, e_{ij}^q)$$

and $v_\phi^q \in \hat{\Sigma}_v^q$, $v_\phi^p \in \hat{\Sigma}_v^p$, $e_\phi^q \in \hat{\Sigma}_e^q$ and $e_\phi^p \in \hat{\Sigma}_e^p$

The new edit cost definition subtracts from the original edit cost the costs of deleting nodes and arcs from G^p and inserting nodes and arcs from G^q. Nodes v_ϕ^q and v_ϕ^p (or arcs e_ϕ^q and e_ϕ^p) represent any extended node (or arc). Note that in cases that nodes or arcs where the extended ones (null nodes o arcs), that is, $v_a^p \in \hat{\Sigma}_v^p$, $v_i^q \in \hat{\Sigma}_v^q$, $e_{ab}^p \in \hat{\Sigma}_e^p$ or $e_{ij}^q \in \hat{\Sigma}_e^q$, then their corresponding cost is zero by definition. Moreover, the subtracted cost C_{ve_di} does not depend on any bijection $f^{p,q}$.

In a similar way than the original edit cost, the optimal bijection, f'^{p,q^*}, is the one that obtains the minimum cost,

$$f'^{p,q^*} = \underset{f^{p,q} \in T}{\operatorname{argmin}} EditCost'(G^p,G^q,f^{p,q})$$

We define D as an $n+m$ vector. The first n positions are filled with the costs of deleting cliques K_a^p that we named $\boldsymbol{C}_{a,\varepsilon}$, and the other m positions are filled with zeros: $D = [\boldsymbol{C}_{1,\varepsilon}, \ldots \boldsymbol{C}_{n,\varepsilon}, 0 \ldots 0]$. Moreover, we define I as an $m+n$ vector. The first m positions are filled with the costs of inserting cliques K_i^q that we named $\boldsymbol{C}_{\varepsilon,i}$, and the other n positions are filled with zeros: $I = [\mathbf{C}_{\varepsilon,1}, \ldots \mathbf{C}_{\varepsilon,m}, 0 \ldots 0]$. Note that zeros in both vectors represent the cost of deleting or inserting null cliques. Besides, it is easy to demonstrate that $C_{ve\_di}(G^p,G^q) = \langle \bar{1}, (I_a + D_a) \rangle$.

With these two vectors, we define two $(\mathrm{n}+\mathrm{m})x(\mathrm{m}+\mathrm{n})$ matrices, $\widehat{D}$ and $\hat{I}$. The first one is obtained by the replication of vector D through columns and the second one is obtained by the replication of vector I through rows.

We are ready to define our cost matrix C' as follows, $\mathrm{C}'=\mathrm{C}-(\widehat{D}+\hat{I})$,

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

Similarly to cost matrix C, this matrix is composed of four quadrants. The dimensions of each quadrant is: $C\mathrm{Q1}' = \mathrm{n\,X\,m}$, $C\mathrm{Q2}' = \mathrm{n\,X\,n}$, $C\mathrm{Q3}' = \mathrm{m\,X\,m}$ and $C\mathrm{Q4}' = \mathrm{m\,X\,n}$. Cells in the first one are filled with the value of substituting the cliques (as in C) but the cost of deleting and inserting the respective cliques is subtracted. The second and third quadrants are composed of infinitive values except at the diagonal that is filled with zeros. All cells on the fourth quadrant have a zero.

On the basis of the new cost matrix C' defined above, Munkres' algorithm [22] can be executed and it finds the optimal permutation p that minimises $\sum_{\mathrm{a}=1}^{\mathrm{n+m}} \mathrm{C}'_{\mathrm{a,p(a)}}$. Note that any correct permutation on p is equivalent to a bijection $\mathrm{f}^{\mathrm{p,q}}$ between nodes of graphs G^{p} and G^{q}.

It was demonstrated in [25] that, on the one hand, the equality $f'^{p,q^*} = f^{p,q^*}$ holds for all pair of graphs G^p and G^q. On the other hand, in the case that $C_{vs} \leq C_{vi} + C_{vd}$ and $C_{es} \leq C_{ei} + C_{ed}$ then the value of $\mathrm{EditCost}'$ is equal to a correct permutation cost of C'. These demonstrations give us a way to compute the $EditCost$ through applying the Munkres' algorithm to matrix C'. Nevertheless, due to the dimensions of C' are the same than the original C, the computational cost would be equivalent. Again, in [25], it was demonstrated that, in the case that $C_{vs} \leq C_{vi} + C_{vd}$ and $C_{es} \leq C_{ei} + C_{ed}$ then minimising a permutation on C' is exactly the same than minimising a permutation on the sub-matrix composed by the first quadrant composed of the first n rows and m columns, that we call it $C\mathrm{Q1}'$.

Algorithm 1 computes the $Fast\ Bipartite$.

Algorithm 1. Fast Bipartite

```
$C\mathrm{Q1}' = \mathrm{Computation\_Cost}(\mathrm{G^p}, \mathrm{G^q})$
    // $C\mathrm{Q1}'$ is the $nXm$ first quadrant of cost matrix $\mathrm{C}'$
$\mathrm{P} = \mathrm{Munkres}(C\mathrm{Q1}')$.
    // P is the $nXm$ permutation matrix
$\mathrm{EditCost}' = \mathrm{Sum}(Sum(P.*\,C\mathrm{Q1}'))$.
    // $.*$ represents the multiplication element by element
$\mathrm{EditCost} = \mathrm{EditCost}' + \mathrm{C_{ve\_di}}$
    // Final distance value
End.
```

As commented, the Munkres algorithm was initially implemented to find the permutation of a quadratic matrix. In case $m \neq n$, matrix $CQ1'$ can be extended with negative values (lower than any original cost). Nevertheless, it is usual the implemented functions of the Munkres' algorithm to automatically enlarge the cost matrix. The worst computational cost of Fast Bipartite is the cost of the Munkres' algorithm, that is: $O(\max(m,n)^3)$. The cost of Bipartite algorithm [14] is $O((m+n)^3)$.

5 Experimental Validation

The goodness of the Bipartite algorithm has been tested in several papers, for this reason, we only want to present the Speed up of our method respect the classical one [14]. We do not present new recognition-ratio tests or correlation tests between the sub-optimal distance and the optimal one since we obtain exactly the same distance value (as described above) when the costs are defined such as $C_{vs} \leq C_{vi} + C_{vd}$ and $C_{es} \leq C_{ei} + C_{ed}$.

We present two different tests. The aim of the first one is to execute again the tests published in paper [14] where the Bipartite algorithm was presented and show the Speed up of our method on these largely used databases known as IAM graph database repository. These databases have been used during some years to test different a types of algorithms related to graphs such as, classification, clustering, prototyping or graph embedding. We do not want to add any comment to these databases since a lot of literature has been written talking about them. The first explanations of these databases can be found in [26] and also they have been commented in [14]. They can be downloaded from the IAPR-TC15 web page [27].

In the experiments, we compared all graphs on the test set respect all graphs on the reference set as authors did in [14]. Then, from this large number of comparisons, we have extracted the mean computational times $\bar{t}_{BP}$ and $\bar{t}_{FBP}$ and the Speed up (table 1). We also show the mean and maximum number of graphs of each database. The source code in Matlab can be downloaded from [28].

Table 1. Mean computational time of Bipartite and Fast Bipartite and Speed Up of Fast Bipartite respect Bipartite

	Mean Order	Max Order	$\bar{t}_{BP}$ (mS)	$\bar{t}_{FBP}$ (mS)	Speed Up
Letter (L)	4.5	8	1.6	1.4	1.08
Letter (M)	4.6	10	2.1	1.8	1.13
Letter (H)	4.7	9	1.8	1.6	1.14
COIL	3.0	11	2.2	1.9	1.18
GREC	12.9	39	29.3	13.1	2.23
Fingerprint	5.4	26	9.8	6.2	1.60
Molecules	9.5	85	391.3	99.7	3.92
Proteins	32.6	126	1460.6	278	5.25

The Speed up is higher than one on the whole experiments therefore it is always worth to use the Fast Bipartite instead of the Bipartite algorithm. Moreover, the higher is the mean and the maximum number of nodes, the higher is the Speed up.

The aim of the second test is to show how the Fast Bipartite algorithm performs when it is not guarantee that $C_{vs} \leq C_{vi} + C_{vd}$ and $C_{es} \leq C_{ei} + C_{ed}$. In some applications, this is a too strong constrain. For instance, palmprint identification is an interesting application since the number of minutiae is around 1000 and it is usual to represent the palmprint image in an attributed graph where nodes represent minutiae and arcs represent the proximity relation (Delaunay triangulation or Nearest-neighbours). Attributes on nodes are the angle of minutiae and edges do not have attributes. The distance between node attributes is the angular distance between the angles of both minutiae. Therefore, if angles are presented by degrees, the maximum distance is 180. For this reason, if we want to fulfil constrain $C_{vs} \leq C_{vi} + C_{vd}$ then we need C_{vi} and C_{vd} to be bigger or equal than 90. Nevertheless, it is usual to consider two minutiae cannot be mapped if the difference between the angles is lower than around 30 degrees. So, from the application point of view we wish $C_{vi} = 30$ and $C_{vd} = 30$. In this second experiment, we show in which extend our algorithm obtains the same distance value obtained by the Bipartite algorithm although the constrain $C_{vs} \leq C_{vi} + C_{vd}$ is not fulfilled.

We used images contained in the Tsinghua 500 PPI Palmprint Database [29]. It is a public high-resolution palmprint database composed of 500 palmprint images of 2040 x 2040 resolution and captured with a commercial palmprint scanner from Hisign. From each person, 16 palmprints are enrolled (8 ones per each hand). We used the algorithm presented in [30] to extract the minutiae from each image. Table 1 shows the mean and max order of the generated graphs as well as the average run time and the Speed up of FBP respect BP. Our database, called Tarragona Palmprint, is public available at [31].

Table 2. Mean computational time of Bipartite and Fast Bipartite and Speed Up of Fast Bipartite respect Bipartite

	Mean Order	**Max Order**	**$\bar{t}_{BP}$ (S)**	**$\bar{t}_{FBP}$ (S)**	**Speed-Up**
Palmprint	987	1505	365,35	40.46	9.03

Table 3 shows the average Distance Error computed as $Distance\ Error = 1 - \frac{Edit\ Distance_{BP}}{Edit\ Distance_{FBP}}$ while computing these two algorithms with several edit costs C_{vi} and C_{vd}. Recall that always holds $Edit\ Distance_{BP} \leq Edit\ Distance_{FBP}$ and $Edit\ Distance_{BP} = Edit\ Distance_{FBP}$ when $C_{vs} \leq C_{vi} + C_{vd}$ and $C_{es} \leq C_{ei} + C_{ed}$. If we wish $C_{vi} = C_{vd} = 30$, as commented above, the average error is lower than 1% and we achieve a Speed Up higher than 9. The run time of these algorithms is independent of the edit costs.

Table 3. Distance between the distance value obtained using BP and FBP when several costs are applied

$C_{vi} = C_{vd}$	10	20	30	40	60	80
Distance Error	0.145	0.019	0.009	0	0	0

6 Conclusions

This paper presents a new algorithm called Fast Bipartite to compute the Graph Edit Distance that obtains exactly the same distance value than the Bipartite algorithm but with a reduced run time. The only restriction we impose is that the edit costs have to be defined such that the insertion plus deletion have to be greater or equal than the substitution. This is a logical restriction since it is needed to be the Edit Distance defined as a distance function. Empirical evaluation shows the Fast Bipartite is always faster than the Bipartite algorithm and higher is the order of both graphs, better is the Speed up we obtain. Moreover, in cases that the application imposes not to hold this restriction, the algorithm also achieves a high Speed up with a reduced error.

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. Edwin, R., Hancock, 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. Serratosa, F., Alquézar, R., Amézquita, N.: A Probabilistic Integrated Object Recognition and Tracking Framework. Expert Systems With Applications 39, 7302–7318 (2012)
7. Solé-Ribalta, A., Serratosa, F.: Graduated Assignment Algorithm for Multiple Graph Matching based on a Common Labelling. International Journal of Pattern Recognition and Artificial Intelligence, IJPRAI 27 (1), 1–27 (2013)
8. Solé, A., Serratosa, F.: Models and Algorithms for computing the Common Labelling of a set of Attributed Graphs. Computer Vision and Image Understanding, CVIU 115(7), 929–945 (2011)
9. Sanromà, G., Alquézar, R., Serratosa, F.: A New Graph Matching Method for Point-Set Correspondence using the EM Algorithm and Softassign. Computer Vision and Image Understanding, CVIU 116(2), 292–304 (2012)
10. Sanromà, G., Alquézar, R., Serratosa, F., Herrera, B.: Smooth Point-set Registration using Neighbouring Constraints. Pattern Recognition Letters, PRL 33, 2029–2037 (2012)

11. 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)
12. Ferrer, M., Valveny, E., Serratosa, F.: Median graphs: A genetic approach based on new theoretical properties. Pattern Recognition 42(9), 2003–2012 (2009)
13. 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)
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. Gold, S., Rangarajan, A.: A Graduated Assignment Algorithm for Graph Matching. IEEE TPAMI 18(4), 377–388 (1996)
16. Rebagliati, N., Solé, A., Pelillo, M., Serratosa, F.: Computing the Graph Edit Distance Using Dominant Sets. In: International Conference on Pattern Recognition, ICPR 2012, Tsukuba, Japan, pp. 1080–1083 (2012)
17. Wong, A., You, M.: Entropy and Distance of Random Graphs with Application to Structural Pattern Recognition. Transaction on Pattern Analysis and Machine Intelligence PAMI-7(5), 599–609 (1985)
18. Sanfeliu, A., Fu, K.-S.: A Distance measure between attributed relational graphs for pattern recognition. IEEE Transactions on Systems, Man, and Cybernetics 13(3), 353–362 (1983)
19. Gao, X., et al.: A survey of graph edit distance. Pattern Analysis and Applications 13(1), 113–129 (2010)
20. 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)
21. Bunke, H.: Error Correcting Graph Matching: On the Influence of the Underlying Cost Function. Trans. on Pattern Analysis and Machine Intelligence 21(9), 917–922 (1999)
22. Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the Society for Industrial & Applied Mathematics 5, 32–38 (1957)
23. Kuhn, H.: The Hungarian method for the assignment problem. Naval Research Logistic Quarterly 2, 83–97 (1955)
24. Bourgeois, F., Lassalle, J.: An extension of the Munkres algorithm for the assignment problem to rectangular matrices. Communications of the ACM 14(12), 802–804 (1971)
25. Serratosa, F.: Fast Computation of Bipartite Graph Matching. Pattern Recognition Letters (2014)
26. 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)
27. `http://iapr-tc15.greyc.fr/links.html`
28. `http://deim.urv.cat/~francesc.serratosa/SW/`
29. Funada, J., et al.: Feature Extraction Method for Palmprint Considering Elimination of Creases. In: Proc. 14th Int. Conf. Pattern Recognition, pp. 1849–1854 (1998)
30. Jain, A.K., Feng, J.: Latent Palmprint Matching. IEEE Trans. on PAMI (2009)
31. `http://deim.urv.cat/~francesc.serratosa/databases/`