A Deep Neural Network Architecture to Estimate Node Assignment Costs for the Graph Edit Distance

Xavier Cortés, Donatello Conte, Hubert Cardot, Francesc Serratosa

A Deep Neural Network Architecture to Estimate Node Assignment Costs for the Graph Edit Distance

Xavier Cortés[1], Donatello Conte[1], Hubert Cardot[1] & Francesc Serratosa[2]

[1] LiFAT, Université de Tours, Tours, France
[2] Universitat Rovira i Virgili, Tarragona, Catalonia, Spain
{xavier.cortes, donatello.conte, hubert.cardot}@univ-tours.fr, francesc.serratosa@urv.cat

Abstract. The problem of finding a distance and a correspondence between a pair of graphs is commonly referred to as the Error-tolerant Graph matching problem. The Graph Edit Distance is one of the most popular approaches to solve this problem. This method needs to define a set of parameters and the cost functions aprioristically. On the other hand, in recent years, Deep Neural Networks have shown very good performance in a wide variety of domains due to their robustness and ability to solve non-linear problems. The aim of this paper is to present a model to compute the assignments costs for the Graph Edit Distance by means of a Deep Neural Network previously trained with a set of pairs of graphs properly matched. We empirically show a major improvement using our method with respect to the state-of-the-art results.

1 Introduction

Graphs are defined by a set of nodes (local components) and edges (the structural relations between them), allowing to represent the connections that exist between the component parts of an object. Due to this, graphs have become very important to model objects that require this kind of representation. In fields like cheminformatics, bioinformatics, computer vision and many others, graphs are commonly used to represent objects [1].

One of the key points in pattern recognition is to define an adequate metric to estimate distances between two patterns. The Error-tolerant Graph Matching tries to address this problem. In particular, the Graph Edit Distance (GED) [2] is an approach to solve the Error-tolerant Graph Matching problem by means of a set of edit operations including insertions, deletions and node assignments, also referred to as node substitutions. On the other hand, Deep Neural Networks (DNNs) have become a very powerful tool applied in several domains due to their ability to find models.

The aim of this paper is to propose a new way to estimate node assignment costs for GED, using a DNN trained with a set of graphs correspondences properly labelled. The document is organized as follows: in Section 2 are presented the definitions to understand the paper, in Section 3 is presented the state-of-the-art, in Section 4 we describe the architecture and de details of our model while Section 5 shows the experimental results. Finally, the conclusions are presented in Section 6.

2 Definitions and Methods

2.1 Attributed Graph

Formally, we define an attributed graph as a quadruplet $G = (\Sigma_v, \Sigma_e, \gamma_v, \gamma_e)$, where $\Sigma_v = \{v_i \mid i = 1, \dots, n\}$ is the set of nodes, $\Sigma_e = \{e_{ij} | i, j \in 1, \dots, n\}$ is the set of edges connecting pairs of nodes, γ_v is a function to map nodes to their attributed values and γ_e maps the structure of the nodes.

2.2 Graphs Correspondence

We define a correspondence between two graphs G^p and G^q as a set of assignments f: $\Sigma_v^p \to \Sigma_v^q$ that univocally relate the nodes of G^p to the nodes of G^q. Where $f(v_i^p) = v_j^q$ if exist the assignment $v_i^p \to v_j^q$.

2.3 Node Assignment Costs for the Graphs Edit Distance

The basic idea of the GED [2] between two graphs G^p and G^q, is to find the minimum cost to transform completely G^p into G^q by means of a set of edit operations, including insertions, deletions and node assignments, commonly referred to as *editpath*. Cost functions are introduced to quantitatively evaluate the level of distortion that each edit operation introduces.

$$c\left(v_i^p \to v_j^q\right) = c_v\left(v_i^p \to v_j^q\right) + c_e\left(v_i^p \to v_j^q\right) \quad (1)$$

The cost of an assignment edit operation (1) is typically given by the distance measure between the nodes attributes $c_v\left(v_i^p \to v_j^q\right) = \text{local\_distance}\left(\gamma_v^p(v_i^p), \gamma_v^q\left(v_j^q\right)\right)$ and by the cost of substituting the local structures $c_e\left(v_i^p \to v_j^q\right) = \text{structural\_distance}\left(\gamma_e^p(v_i^p), \gamma_e^q\left(v_j^q\right)\right)$. These cost functions estimate the degree of separation between a pair of nodes v_i^p and v_j^q belonging to graphs G^p and G^q. The Euclidean distance is a common way to estimate the local_distance between the nodes attributes, while in [3] are presented different metrics to estimate the structural_distance. Our model, as we will see, automatically learns the costs of these assignations from a set of training correspondences previously labeled without having to define the cost functions.

In order to allow the maximum flexibility in the matching process and taking into account that graphs can have different cardinality and that a node that appears in G^p could not be in G^q, graphs can be extended with null nodes adding penalty costs when an existing node of one graph is assigned to a null one of the other graph. In this

paper we do not consider this option since we focus on the problem of node assignments comparing our results with other works that face the same problem, as in [4, 5]. However, our model can be easily combined with other models that consider null nodes by adding penalty costs for insertions and deletions.

2.4 Hamming Distance

The hamming distance is a metric to compare graph correspondences used typically to assess the correctness of a correspondence comparing the correspondence that we are evaluating with respect to the ground-truth one. This metric evaluates the ratio between the number of correct assignments and the total number of assignments in the evaluated correspondence. Formally:

Let $f: \Sigma_v^p \rightarrow \Sigma_v^q$ the automatic correspondence and $f': \Sigma_v^{p'} \rightarrow \Sigma_v^{q'}$ the ground-truth correspondence between two graphs G^p and G^q with cardinality n (graphs can be extended with null nodes to manage insertions or deletions of nodes), the hamming distance is formally defined as:

$$\Delta^h(f, f') = \frac{\sum_{i=1}^{n}(1 - \delta\left(f(v_i^p), f'(v_i^p)\right))}{n} \tag{2}$$

Where, δ is the Kronecker Delta function:

$$\delta(a, b) = \begin{cases} 0, \text{if } a \neq b \\ 1, \text{if } a = b \end{cases} \tag{3}$$

2.5 Deep Neural Networks

DNNs are a computational model inspired by the neural networks existing in many biological organisms [6]. They have become very popular in many fields due to its adaptability and learning capacity.

The classical architecture of a DNN consists of an input layer, an output layer and a cascade of multiple hidden layers in the middle. Each layer contains several neurons connected with the neurons of the previous layer. The connections between neurons have different weights fixing the strength of the signal at the connection. Each neuron executes an activation function having as inputs the values of the connections with the previous layer and sending the output to the neurons of the next layer. The signal path goes from the input layer to the output layer. Depending on the connections weights and the bias values, the output can be different given the same input.

During the training process the learning algorithm adjust the weights and bias according to the values of a training set trying to minimize the error between the given inputs and the expected outputs.

3 State of the Art

The distance value of the GED depends on the edit costs, in particular c_v (distance between the nodes attributes), c_e (distance between the local structures) and the penalties costs for insertions and deletions. Typically, these costs must be defined and parameterized aprioristically. Depending on how these parameters and costs functions are defined the performance in terms of hamming distance between the automatically deduced correspondence and a ground truth correspondence or graphs classification accuracy, can be different.

Recently, in order to maximize the performance of different Error-Tolerant Graph Matching approaches, some researchers have focused their work on automatically learn the parameters and the cost functions instead of using the traditional trial-error method.

We can divide the learning methods in three main groups depending on the objective function. The first group [7-10] addresses the recognition ratio for graph classification, while the second group [4, 5, 11, 12] targets the hamming distance. Finally, there is a special case in [13] that does not learn the parameters to estimate the costs but tries to predict if an assignment between nodes is correct or not depending on the values of the costs matrix (the matrix with the costs of each edit operation). Moreover, another subdivision can be considered depending if the methods try to learn the assignments costs or the insertions and deletions. The aim of our paper is to propose a model to estimate only the assignments costs minimizing the hamming distance, as in [4, 5]. As we have commented before, our model can be combined with other models that consider nodes insertions and deletions but we do not address this particularity in this paper.

4 Proposed Architecture

In this section we describe a new architecture based on DNNs to estimate assignments costs (section 2.3) between a pair of nodes by means of a DNN (section 2.5) in order to minimize the hamming distance (section 2.4).

$$c\left(v_i^p \rightarrow v_j^q\right) = DNN\left(v_i^p \rightarrow v_j^q\right) \tag{4}$$

4.1 Node Assignment Embedding

The first step of our model consists of transforming the local and structural information of both nodes into a set of inputs for the network. In this section we show how to embed this information into an input vector.

Let G^p and G^q two attributed graphs, $\gamma_v^p = \{v_i^p \rightarrow \Psi_i^p \mid i = 1 \dots n\}$ a function that assigns t attribute values from an arbitrary domain to each node of G^p, where $\Psi_i^p \in \mathbb{R}^t$ is defined in a metric space of $t \in \mathbb{R}$ dimensions and $\gamma_e^p = \{v_i^p \rightarrow$

$E(v_i^p) \mid i = 1 \ldots n\}$ where $E(.)$ refers to the number of edges of a certain node (the Degree centrality [3]). And similar for γ_v^q and γ_e^p in G^q.

Vector $x^{i \to j} = [\gamma_v^p(v_i^p), \gamma_e^p(v_i^p), \gamma_v^q(v_j^q), \gamma_e^q(v_j^q)] \in \mathbb{R}^{(t+1)\cdot 2}$ is the embedded representation of the assignment $v_i^p \to v_j^q$ where each position of the vector $x^{i \to j}$ corresponds to one of the values of the input layer of the DNN that estimates the assignment cost between the node v_i^p of G^p and the node v_j^q of G^q (Fig. 1).

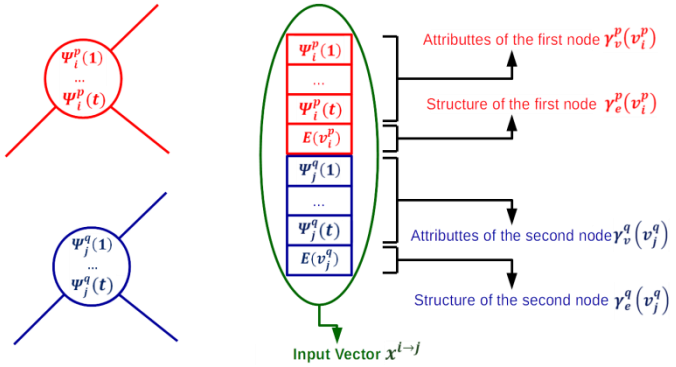

Fig. 1. An illustration showing the embedding process of two nodes (red and blue) into an input vector.

4.2 Network Architecture

The topology we propose is a classical topology for parameters fitting consisting of a multi-layer network using the sigmoid activation function for the hidden layers and a linear function for the output layer (Fig. 2). In the experimental section we shown the results achieved with different configurations changing the number of neurons and the number hidden layers.

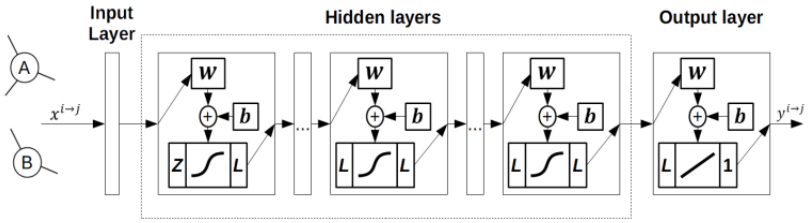

Fig. 2. DNN architecture for node assignments costs. Z is the number of inputs (size of the vector $x^{i \to j}$). L the number of neurons of each hidden layer, w the weights and b the bias.

The input of the network representing the nodes to be assigned is the vector $x^{i \to j} \in \mathbb{R}^{(t+1)\cdot 2}$ (defined in section 4.1) and the output is a real value theoretically defined within a cost range from zero to one viz. $y^{i \to j} = \{c \in \mathbb{R} : 0 \leq c \leq 1\}$. Zero is the expected value when there is no penalty for the assignment and one is the maximum expected value penalizing a node assignment.

4.3 Training the Model

We manage the problem of training the DNN as a supervised learning problem. The training set has K observations. Each observation is composed of a triplet consisting of pair of graphs and the correspondence that relates its nodes $\{G^{p^k}, G^{q^k}, f^k\}$. The ground-truth correspondences f^k must be provided by an oracle according to the problem (images, fingerprints, letters…).

a) b)

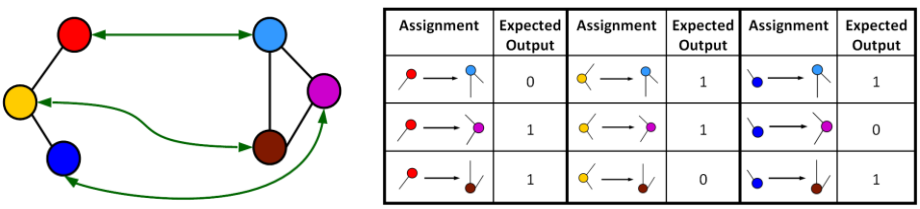

Fig. 3. (a) Correspondence between a pair of graphs. Colored circles: Nodes. Black lines: Edges. Green arrows: Graphs correspondence. (b) Set of all possible node assignments and expected DNN outputs given the correspondence in (a).

Then, assuming that the assignment cost must be low if two nodes are matched and high in the opposite case and taking into account that the outputs range goes from zero to one (section 4.2), we propose to feed the learning algorithm with a set of R inputs-outputs pairs $\left\{x^{v_i^{p^r} \to v_j^{q^r}}, o^r\right\}$ that we deduce from the training set $\{G^{p^k}, G^{q^k}, f^k\}$. Where $v_i^{p^r}$ and $v_j^{q^r}$ are two nodes belonging to graphs G^{p^k} and G^{q^k} respectively. $x^{v_i^{p^r} \to v_j^{q^r}}$ are the inputs of the DNN representing the assignment between $v_i^{p^r}$ and $v_j^{q^r}$ (section 4.1). And o^r is the expected output, zero if $f^k\left(v_i^{p^r}\right) = v_j^{q^r}$ and one otherwise.

In Fig. 3.b, we show the expected outputs between nodes when the ideal correspondence is the correspondence shown in Fig. 3.a. Zero when there is an assignment in the gorund-truth correspondence and one when not. Note that there are more cases in which the expected output must be one because the correspondences between graphs are bijective by definition in our framework. That means, each node of G^{p^k} is assigned to a single node of G^{q^k} while it is unassigned to all the other nodes. For this reason and in order to prevent unbalancing problems we propose to oversample the positive assignments between nodes (when the expected output is zero) repeating them in the set of inputs-outputs that feeds the learning algorithm $n - 1$ times, where n is the graphs cardinality.

The training algorithm used to learn the bias and weights of the network is the Leveberg-Marquardt [14].

4.4 Graph Matching Algorithm

The graph matching method we propose is inspired by the Bipartite-GED [15] which is one of the most popular methods used to reduce the computational complexity of the GED problem to a Linear Sum Assignment Problem (LSAP). First, we build a cost matrix in which each cell corresponds to the cost of an assignment. The algorithm fills the values of this matrix with the DNN outputs. Our algorithm does not extend the matrix for insertions and deletions since we only consider the assignments between nodes. The process of assigning nodes can be solved as a LSAP on C matrix. In our experiments we used the Hungarian [16] solver. The final step is to sum the costs of the solution provided by the solver.

Algorithm: Neural Graph Matching
Input: Graph G1, G2; DNN network;
Output: Correspondences Co; Cost Ct;

```
1:  Initialisation:
2:  foreach Node NodeI of G1
3:      foreach Node NodeJ of G2
4:          x:=inputVector(NodeI,NodeJ);
5:          y:=computeCosts(network,x);
6:          C(I,J) = y;
7:      end
8:  end
9:  [Co, Ct] = solveLSAP(C);
```

Algorithm 1. Learning Graph Matching methods.

5 Experiments

We divided the experimental section in three parts. First, we describe the database used in the experiments. Second, we show the resultant costs matrix using different network configurations. Finally, we present the hamming distance results using our model compared with the state-of-the-art algorithms that face the same kind of problem.

5.1 Databases

The HOUSE-HOTEL database described in detail in [17] consists of two sequences of frames showing two computer modeled objects, 111 frames of a HOUSE and 101 frames of a HOTEL, rotating on its own axis. Each frame of these sequences has the same 30 salient points identified and labelled. Each salient point represents a node of the graph and it is attributed by 60 Context Shape features. They triangulated the set of salient points using the Delaunay triangulation to generate the structure of the graphs. They made three sets of frames pairs taking into account different baselines (number of frames of separation in the video sequence). One set was used to learn, another to validate and the third one to test the model. Since the salient points are labelled we know the ground-truth correspondence between the nodes of the graphs.

5.2 Costs Matrix

This section shows the heatmaps of the resultant costs matrix (C matrix in 4.4) using our model. The aim of this experiment is to find a cost matrix minimizing the costs when the nodes must be assigned and maximizing the costs when not. Since we know the ground-truth correspondence we can deduce the ground-truth cost matrix. Fig. 4.a shows the results using a single hidden layer while Fig. 4.b shows the same results using 5 hidden layers and Fig. 4.c shows the results using 10 hidden layers with different configurations of numbers of neurons per layer. Blue color represents low costs values while yellow color represents high costs values. The experiment was performed using the first pair of graphs of the test set in the HOUSE sequence separated by 90 frames and the model has been trained with all the graphs separated by 90 frames in the training set.

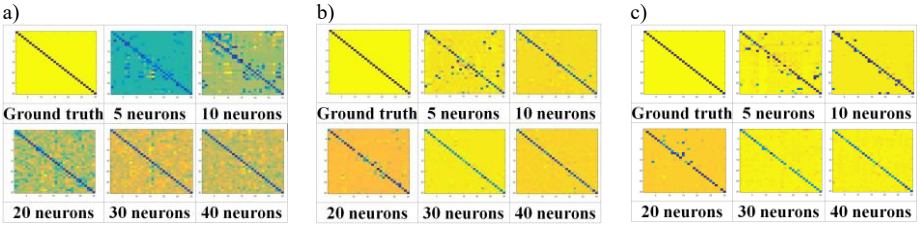

Fig. 4. Costs matrix heatmaps between two graphs corresponding to the HOUSE dataset (90 frames of separation) using (a) 1 hidden layer, (b) 5 hidden layers and (c) 10 hidden layers.

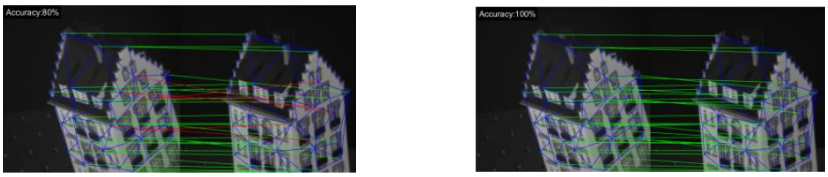

Fig. 5. Correspondences found between two graphs of the HOTEL sequence using our model. Left: single-layer and 10 neurons per layer, *Right*: five-layers and 10 neurons per layer. Blue lines are the edges between these nodes. Green lines: correct assignments. Red lines: incorrect assignments.

We observe how the model tends to separate better the correct assignments from the incorrect ones when we increase the number of neurons and layers until reaching a point where the improvement is no longer increasing and even it could decrease. This can be explained because when we increase the network complexity, the model is able to find deeper non-linear correlations between the attributes that feature the nodes, but reached a critical point, could present overfitting problems due to there are more neurons than the ones that can be justified by the data.

Fig. 5 shows the obtained correspondences computing a cost matrix with a single-layer (*left*) and with five-layers (*right*) of 10 neurons each layer in order to illustrate the performance of the model with different network configurations in terms of matching accuracy.

5.3 Hamming Distance Results

The main goal of our model is to reduce the hamming distance performing the GED. In the following experiment we show the hamming distance results between the correspondence found by our model and the ground-truth correspondence. In Table 1, we compare our results with respect to the state-of-the-art, note that smaller values mean better performance. We train, validate and test the model using different pairs of graphs as we described in section 5.1. The baseline of our experiments is the number of frames of separation in the video sequence. Since the objects are in motion, consecutive frames are more similar than the distant ones. Therefore, the problem tends to be more complex when we increase the number of frames of separation. A single-layer network with 30 neurons per layer has been enough to reduce the hamming distance to zero for all the experiments, however, in Fig. 4, we show how deeper networks tend to increase the gap between the costs, generally separating better the correct assignments from the incorrect ones. The achieved results using our model represent a major improvement with respect to the previously presented results. We discuss the results in the next section.

Table 1. Hamming distance results on House and Hotel datasets.

HOUSE			
#Frames	**[4]**	**[5]**	***Our model***
90	0.14	0.24	**0**
80	0.14	0.18	**0**
70	0.13	0.10	**0**
60	0.09	0.06	**0**
50	0.19	0.04	**0**
40	0.02	0.02	**0**
30	0.02	0.01	**0**
20	0.01	0	**0**
10	0	0	**0**

HOTEL			
#Frames	**[4]**	**[5]**	***Our model***
90	0.09	0.21	**0**
80	0.17	0.18	**0**
70	0.14	0.15	**0**
60	0.13	0.16	**0**
50	0.09	0.07	**0**
40	0.07	0.04	**0**
30	0.04	0.02	**0**
20	0.02	0	**0**
10	0	0	**0**

** Results obtained with 1 layer of 30 neurons*

6 Conclusions

We have presented a new model to estimate assignment costs for the Graphs Edit Distance using a Deep Neural Network. We experimentally show that our model is able to find the ideal solution independently of the number of frames of separation. These results represent a major improvement with respect to the previous state-of-the-art results, in particular, when the number of frames of separation is large. This means that the model can manage important distortions in the representations when it tries to find the best correspondence. We conclude that the improvement is because using neural networks allows to find multiple correlations between nodes attributes when performing the matching and our model is not limited by having to define a particular distance metric aprioristically since it learns the costs functions.

We consider that this work represents an important step to define the costs functions for node assignments in the problem of the Graph Edit Distance. However it is necessary to train the network with a set of examples properly labeled. The next step is to expand the model including insertions and deletions costs.

Acknowledgments. This work is part of the LUMINEUX project supported by the Region Centre-Val de Loire (France) and by the Spanish projects TIN2016-77836-C2-1-R and ColRobTransp MINECO DPI2016-78957-R AEI/FEDER EU; and also, the European project AEROARMS, H2020-ICT-2014-1-644271.

References

1. D. Conte, P. Foggia, C. Sansone, M. Vento. “Thirty Years Of Graph Matching In Pattern Recognition”. International Journal of Pattern Recognition and Artificial Intelligence, Vol. 18, No. 3 pp: 265-298, 2004.
2. H. Bunke, G. Allermann. “Inexact graph matching for structural pattern recognition”. Pattern Recognition Letters, 1(4): p. 245-253, 1983.
3. Francesc Serratosa, Xavier Cortés. “Graph Edit Distance: Moving from global to local structure to solve the graph-matching problem”. Pattern Recognition Letters 65: 204-210, 2015.
4. T.S. Caetano, J.J. McAuley, L. Cheng, Q.V. Le, A.J. Smola. “Learning Graph Matching”. IEEE Trans. Pattern Anal. Mach. Intell. 31(6): 1048-1058, 2009.
5. X. Cortés, F. Serratosa. “Learning Graph Matching Substitution Weights Based on the Ground Truth Node Correspondence”. IJPRAI 30(2), 2016.
6. J. Schmidhuber, J. "Deep Learning in Neural Networks: An Overview". Neural Networks. 61: 85–117. 2015.
7. R. Raveaux, M. Martineau, D. Conte, G. Venturini. “Learning Graph Matching with a Graph-Based Perceptron in a Classification Context”. GbRPR 2017: 49-58. 2017.
8. M. Neuhaus, H. Bunke. “Self-organizing maps for learning the edit costs in graph matching”. IEEE Trans. on Sys., Man, and Cybernetics, Part B 35(3), pp. 503-514, 2005.
9. M. Neuhaus, H. Bunke. “Automatic learning of cost functions for graph edit distance”. Inf. Sci. 177(1), pp. 239-247, 2007.
10. M. Leordeanu, R. Sukthankar, M. Hebert. “Unsupervised Learning for Graph Matching”. International Journal of Computer Vision 96(1): 28-45, 2012.
11. F. Serratosa, A. Solé-Ribalta, X. Cortés. “Automatic Learning of Edit Costs Based on Interactive and Adaptive Graph Recognition”. GbRPR 2011: 152-163, 2011.
12. X. Cortés, F. Serratosa. “Learning graph-matching edit-costs based on the optimality of the oracle's node correspondences”. Pattern Recognition Letters 56: 22-29, 2015.
13. K. Riesen, M. Ferrer. “Predicting the correctness of node assignments in bipartite graph matching”. Pattern Recognition Letters 69: 8-14. 2016.
14. C. Kanzow, N. Yamashita, M. Fukushima "Levenberg-Marquardt methods with strong local convergence properties for solving nonlinear equations with convex constraints," JCAM, 172(2):375-97, 2004.
15. K. Riesen, H. Bunke. “Approximate graph edit distance computation by means of bipartite graph matching”. Image and Vision Computing, vol. 27, no. 4, pp. 950–959, 2009.
16. H. W. Kuhn. “The Hungarian Method for the assignment problem”. Naval Research Logistics Quarterly, 2: 83–97, 1955.
17. C. Francisco Moreno-García, X. Cortés, F. Serratosa. “A Graph Repository for Learning Error-Tolerant Graph Matching”. S+SSPR 2016: 519-529. 2016.