Learning Cost Functions for Graph Matching

Rafael de O Werneck, Romain Raveaux, Salvatore Tabbone, Ricardo da S Torres

▶ **To cite this version:**

Rafael de O Werneck, Romain Raveaux, Salvatore Tabbone, Ricardo da S Torres. Learning Cost Functions for Graph Matching. Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), Aug 2018, Beijing, China. hal-01889964

HAL Id: hal-01889964
https://espci.hal.science/hal-01889964

Submitted on 8 Oct 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Learning Cost Functions for Graph Matching

Rafael de O. Werneck*[1], Romain Raveaux[2],
Salvatore Tabbone[3], and Ricardo da S. Torres[1]

[1] Institute of Computing, University of Campinas, Campinas, SP, Brazil
[2] Université Franois Rabelais de Tours, 37200 TOURS, France
[3] Université de Lorraine-LORIA UMR 7503, Vandoeuvre-lès-Nancy, France
{rafael.werneck,rtorres}@ic.unicamp.br
romain.raveaux@univ-tours.fr
tabbone@loria.fr

Abstract. During the last decade, several approaches have been proposed to address detection and recognition problems, by using graphs to represent the content of images. Graph comparison is a key task in those approaches and usually is performed by means of graph matching techniques, which aim to find correspondences between elements of graphs. Graph matching algorithms are highly influenced by cost functions between nodes or edges. In this perspective, we propose an original approach to learn the matching cost functions between graphs' nodes. Our method is based on the combination of distance vectors associated with node signatures and an SVM classifier, which is used to learn discriminative node dissimilarities. Experimental results on different datasets compared to a learning-free method are promising.

Keywords: Graph matching, Cost learning, SVM

1 Introduction

In the pattern recognition domain, we can represent objects using two methods: statistical or structural [4]. On the later, objects are represented by a data structure (*e.g.*, graphs, trees), which encodes their components and relationships; and on the former, objects are represented by means of feature vectors. Most methods for classification and retrieval in the literature are limited to statistical representations [17]. However, structural representation are more powerful, as the object components and their relations are described in a single formalism [18]. Graphs are one of the most used structural representations. Unfortunately, graph comparison suffers from high complexity, often an NP-hard problem requiring exponential time and space to find the optimal solution [5].

One of the widely used method for graph matching is the graph edit distance (GED). GED is an error-tolerant graph matching paradigm that defines

* Thanks to CNPq (grant #307560/2016-3), CAPES (grant #88881.145912/2017-01), FAPESP (grants #2016/18429-1, #2017/16453-5, #2014/12236-1, #2015/24494-8, #2016/50250-1, and #2017/20945-0), and the FAPESP-Microsoft Virtual Institute (#2013/50155-0, #2013/50169-1, and #2014/50715-9) agencies for funding.

the similarity of two graphs by the minimum number of edit operations necessary to transform one graph into another [3]. A sequence of edit operations that transforms one graph into another is called edit path between two graphs. To quantify the modifications implied by an edit path, a cost function is defined to measure the changes proposed by each edit operation. Consequently, we can define the edit distance between graphs as the edit path with minimum cost.

The possible edit operations are: node substitution, edge substitution, node deletion, edge deletion, node insertion, and edge insertion. The cost function is of first interest and can change the problem being solved. In [1,2], a particular cost function for the GED is introduced, and it was shown that under this cost function, the GED computation is equivalent to the maximum common subgraph problem. Neuhaus and Bunke [14], in turn, showed that if each elementary operation satisfies the criteria of a metric distance (separability, symmetry, and triangular inequality) then the GED is also a metric.

Usually, cost functions are manually designed and are domain-dependent. Domain-dependent cost functions can be tuned by learning weights associated with them. In Table 1, published papers dealing with edit cost learning are tabulated. Two criteria are optimized in the literature, the matching accuracy between graph pairs or an error rate on a classification task (classification level). In [13], learning schemes are applied on the GED problem while in [11,6], other matching problems are addressed. In [11], the learning strategy is unsupervised as the ground truth is not available. In another research venue, different optimization algorithms are used. In [12], Self-Organizing Maps (SOMs) are used to cluster substitution costs in such a way that the node similarity of graphs from the same class is increased, whereas the node similarity of graphs from different classes is decreased. In [13], Expectation Maximization algorithm (EM) is used for the same purpose. An assumption is made on attribute types. In [7], the learning problem is mapped to a regression problem and a structured support vector machine (SSVM) is used to minimize it. In [8], a method to learn scalar values for the insertion and deletion costs on nodes and edges is proposed. An extension to substitution costs is presented in [9]. The contribution presented in [16] is the nearest work to our proposal. In that work, the node assignment is represented as a vector of 24 features. These numerical features are extracted from a node-to-node cost matrix that is used for the original matching process. Then, the assignments derived from exact graph edit distance computation is used as ground truth. On this basis, each node assignment computed is labeled as correct or incorrect. This set of labeled assignments is used to train an SVM endowed with a Gaussian kernel in order to classify the assignments computed by the approximation as correct or incorrect. This work operates at the matching level. All prior works rely on predefined cost functions adapted to fit an objective of matching accuracy. Little research has been carried out to automatically design generic cost functions in a classification context.

In this paper, we propose to learn a discriminative cost function between nodes with no restriction on graph types nor on labels for a classification task. On a training set of graphs, a feature vector is extracted from each node of each graph

Table 1. Graph matching learning approaches.

Ref.	Graph matching problem	Supervised	Criterion	Optimization method
[12]	GED	Yes	Recognition rate	SOM
[13]	GED	Yes	Recognition rate	EM
[8,9]	GED	Yes	Matching accuracy	Quadratic programming
[6]	Other	Yes	Matching accuracy	Bundle
[7]	Other	Yes	Matching accuracy	SSVM
[11]	Other	No	Matching accuracy	Bundle

thanks to a node signature that describes local information in graphs. Node dissimilarity vectors are obtained by pairwise comparison of the feature vectors. Node dissimilarity vectors are labeled according to the node pair belonging to graphs of the same class or not. On this basis, an SVM classifier is trained. At the decision stage, two graphs are compared, a new node pair is given as an input of the classifier, and the class membership probability is outputted. These adapted costs are used to fill a node-to-node similarity matrix. Based on these learned matching costs, we approximate the matching graph problem as a Linear Sum Assignment Problem (LSAP) between the nodes of two graphs. The LSAP aims at finding the maximum weight matching between the elements of two sets and this problem can be solved by the Hungarian algorithm [10] in $O(\mathrm{n}^3)$ time.

The paper is organized as follow: Section 2 presents our approach for local description of graphs, and the proposed approaches to populate the cost matrix for the Hungarian algorithm. Section 3 details the datasets and the adopted experimental protocol, as well as presents the results and discussions about them. Finally, Section 4 is devoted to our conclusions and perspectives for future work.

2 Proposed Approach

In this section, we present our proposal to resolve the graph matching problem as a bipartite graph matching using local information.

2.1 Local Description

In this work, we use node signatures to obtain local descriptions of graphs. In order to define the signature, we use all information of the graph and the node. Our node signature is represented by the node attributes, node degree, attributes of incident edges, and degrees of the nodes connected to the edges.

Given a general graph $G = (V, E)$, we can define the node signature extraction process and representation, respectively, as:

$$\Gamma(G) = \{\gamma(n) | \forall n \in V\}$$

$$\gamma(n) = \{\alpha_n^G, \theta_n^G, \Delta_n^G, \Omega_n^G\}$$

where α_n^G is the attributes of the node n, θ_n^G is the degree of node n, Δ_n^G is the set of degrees of adjacent nodes to n, and Ω_n^G is a set of attributes of the incident edges of n.

2.2 HEOM Distance

One of our approaches to perform graph matching consists on finding the minimum distance to transform the node signatures from one graph into the node signatures from another graph. To calculate the distance between two node signatures, we need a distance metric capable of dealing with numeric and symbolic attributes. We selected the *Heterogeneous Euclidean Overlap Metric* [19] (HEOM) and we provided an adaptation for our graph local description.

The HEOM distance is defined as:

$$HEOM(i, j) = \sqrt{\sum_{a=0}^{n} \delta(i_a, j_a)^2}, \tag{1}$$

where a is each attribute of the vector, and $\delta(i_a, j_a)$ is defined as:

$$\delta(i_a, j_a) = \begin{cases} 1 & \text{if } i_a \text{ or } j_a \text{ is missing,} \\ 0 & \text{if } a \text{ is symbolic and } i_a = j_a, \\ 1 & \text{if } a \text{ is symbolic and } i_a \neq j_a, \\ \frac{|i_a - j_a|}{\text{range}_a} & \text{if } a \text{ is numeric.} \end{cases} \tag{2}$$

In our approach, we define the distance between two node signatures as follow. Let $A = (V_a, E_a)$ and $B = (V_b, E_b)$ be two graphs and $n_a \in V_a$ and $n_b \in V_b$ be two nodes from these graphs. Let $\gamma(n_a)$ and $\gamma(n_b)$ be the signature of these nodes, that is:

$$\gamma(n_a) = \{\alpha_{n_a}^A, \theta_{n_a}^A, \Delta_{n_a}^A, \Omega_{n_a}^A\}$$

and

$$\gamma(n_b) = \{\alpha_{n_b}^B, \theta_{n_b}^B, \Delta_{n_b}^B, \Omega_{n_b}^B\}.$$

The distance ϵ between two node signatures is:

$$\begin{aligned} \epsilon(\gamma(n_a), \gamma(n_b)) = HEOM(\alpha_{n_a}^A, \alpha_{n_b}^B) + HEOM(\theta_{n_a}^A, \theta_{n_b}^B) + \\ HEOM(\Delta_{n_a}^A, \Delta_{n_b}^B) + \frac{\sum_{i=1}^{|\Omega_{n_a}^A|} HEOM(\Omega_{n_a}^A(i), \Omega_{n_b}^B(i))}{|\Omega_{n_a}^A|} \end{aligned} \tag{3}$$

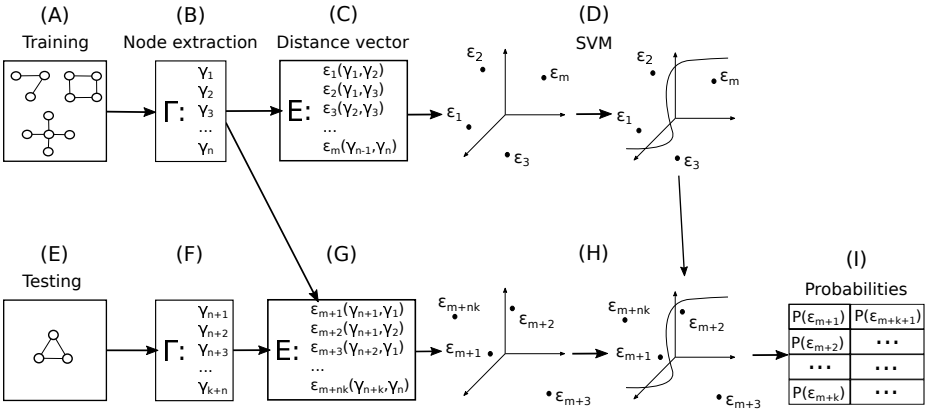

Fig. 1. Proposed SVM approach to compute the edit cost matrix.

2.3 SVM-based Node Dissimilarity Learning

We propose an SVM approach to learn the graph edit distance between two graphs. In this approach, we first define a *distance vector* ϵ' between two node signatures. Function ϵ' is derivated from ϵ, but instead of summing up the distance related to all structures, the function considers each structure distance score as a value of a bin of the vector. This distance vector is composed of the HEOM distance between each structure of the node signature, i.e., the distance between the node attribute, node degree, degrees of the nodes connected to the edges, and attributes of incident edges are components of the vector, i.e.,

$$\epsilon'(\gamma(n_a), \gamma(n_b)) = [HEOM(\gamma(n_a)_i, \gamma(n_b)_i)]\,,$$
$$\forall i \in \{0, \cdots, |\gamma(n)|\} \mid \gamma(n)_i \text{ is a component of } \gamma(n).$$

To each distance vector ϵ', a label is assigned. These labels guide the SVM learning process. We propose the following formulation to assign labels to distance vectors. Let $Y = \{y_1, y_2, \ldots, y_l\}$ be the set of l labels associated with graphs. In our formulation, denominated multi-class, distance vectors, which are associated with node signatures extracted from graphs of the same class (say y_i), are labeled as y_i. Otherwise, a novel label y_{l+1} is used, representing that the distance vectors were computed from node signatures belonging to graphs belonging to different classes.

Figure 1 illustrates the main steps of our approach. Given a set of training graphs (step A in the figure), we first extract the node signatures from all graphs (B), and compute the pairwise distance vectors (C). We then use the labeling procedure described above to assign labels to distance vectors defined by node signatures extracted from graphs of the training set and use these labeled vectors to train an SVM classifier (D).

2.4 Graph Classification

At testing stage, each one of the graphs from the test set (E) has its node signatures extracted (F). Again, distance vectors are computed, now considering node signatures from the test and from the training set (G). With the distance vectors, we can project them into the learned feature space and obtain the probability of a test sample that belongs to the training set classes considering the SVM hyperplane of separation (H). These probabilities are used to populate a cost matrix for each graph in the training set (I), in such a way that, for each node signature from the test graph (row) and each node signature from the training graph (column), we create a matrix of probabilities for each combination of test and training graphs. This matrix is later used in the Hungarian algorithm. As the resulting cost matrices encodes probabilities, we compute the maximum cost path using the Hungarian algorithm instead of the minimum. The test sample classification is based on the k-nearest neighbor (kNN) graphs found in the training set, where graph similarity is defined by the Hungarian algorithm.

3 Experimental Results

In this section, we describe the datasets used in the experiments, we present our experimental protocol, and how our method was evaluated. At the end, we present our results and discuss them.

3.1 Datasets

In our paper, we perform experiments in three labeled datasets from the IAM graph database [15]: Letter, Mutagenicity, and GREC.

The **Letter** database compromises 15 classes of distorted letter drawings. Each letter is represented by a graph, in which the nodes are ending points of lines, and edges are the lines connecting ending points. The attributes of the node are its position. This dataset has three sub-datasets, considering different distortions (low distortion, medium distortion, and a high distortion).

Mutagenicity is a database of 2 classes representing molecular compounds. In this database, the nodes are the atoms and the edges the valence of the linkage.

GREC database consists of symbols from architectural and electronic drawings represented as graphs. Ending points are represented as nodes and lines and arcs are the edges connecting these ending points. It is composed of 22 classes.

3.2 Experimental Protocol

Considering that the complexity and computational time to calculate the distance vectors for the SVM method is soaring, we decide to perform preliminary experiments where we randomly selected two graphs of each class from the training set to be our training, and for our test, we selected 10% of the testing graphs from each class. As we are selecting randomly the training and testing sets, we

need to perform more experiments to obtain an average result, to avoid any bias a unique experiment selecting training and testing sets can have. Thus, we performed each experiments 5 times to obtain our results. To evaluate our approach, we present the mean accuracy score and the standard deviation of a k-NN classifier ($k = 3$). Table 2 presents detailed information about the datasets.

Table 2. Informations about the datasets.

	Datasets				
	Letter-LOW	Letter-MED	Letter-HIGH	Mutagenicity	GREC
# graphs	750	750	750	1500	286
# classes	15	15	15	2	22
# graphs per class	50	50	50	830/670	13
# graphs in learning	30	30	30	4	44
# distance vectors	$\approx 10,000$	$\approx 10,000$	$\approx 10,000$	$\approx 14,000$	$\approx 130,000$
# graphs in testing	75	75	75	129/104	44

3.3 Results

In our first experiments, to provide a baseline, we performed the graph matching using the HEOM distance function between the node signatures to populate the cost matrix. We also populated the cost matrix with random values between 0 and 1 for comparison. Table 3 shows these results for the chosen datasets. The HEOM distance approach shows improvement over a simple random selection of values.

Table 3. Accuracy results for HEOM distance and random population of the cost matrix in the graph matching problem (in %).

Approach	Datasets				
	Letter-LOW	Letter-MED	Letter-HIGH	Mutagenicity	GREC
Random	0.53 ± 0.73	1.60 ± 2.19	1.60 ± 1.12	54.85 ± 4.22	1.36 ± 2.03
HEOM distance	40.53 ± 11.72	15.73 ± 3.70	10.93 ± 3.70	49.44 ± 10.69	52.27 ± 7.19

As we can see in Table 3, the HEOM distance presents a better result than the random assignment of weights, except for the Mutagenicity dataset, which is the only dataset with two classes. In this case, the obtained results are similar, considering the standard deviation of the executions (±4.22 for Random approach, and ±10.69 for the HEOM approach).

Next, we run experiments using the proposed multi-class SVM approach to compare with the results obtained using the HEOM distance in the cost matrix. We used default parameters for the SVM for the training step (RBF kernel, $C = 0$). We also present results of experiments in which we normalize

the distance vector, using min-max (normalizing between 0 and 1) and zscore (normalization using the mean and standard deviation) normalizations. Table 4 shows the mean accuracy of the experiments made.

Table 4. Mean accuracy (in %) for the HEOM distance and SVM multi-class approach in the graph matching problem. The best results for each dataset are show in bold.

		Datasets				
		Letter-LOW	Letter-MED	Letter-HIGH	Mutagenicity	GREC
HEOM distance		**40.53 ± 11.72**	15.73 ± 3.70	10.93 ± 3.70	49.44 ± 10.69	**52.27 ± 7.19**
SVM Multi-class		30.67 ± 5.50	**28.00 ± 9.80**	18.93 ± 5.77	**71.24 ± 29.50**	18.64 ± 6.89
	min-max	33.33 ± 7.12	20.27 ± 6.69	14.40 ± 5.02	63.26 ± 15.61	20.00 ± 7.43
	zscore	37.87 ± 9.83	21.87 ± 1.52	**20.27 ± 8.56**	64.12 ± 7.68	30.91 ± 2.59

Table 4 shows us that the SVM approach is promising, obtaining better results for three of the five datasets considered. The improvement in the Mutagenicity dataset was above 20 percentage points from the HEOM distance baseline. As for the other cases, the Letter-LOW dataset had similar results for the HEOM distance and SVM approach (standard deviation of the HEOM is ±11.72 and for the SVM is ±9.83). The GREC dataset was the only dataset with a distant results from the HEOM approach. We discuss that it is because the dataset has more classes than the others, so its "different" class contains more distance vectors combining node signatures of different classes. With this imbalanced distribution, the "different" class shadows the other classes in the SVM classification.

Table 4 also shows that a normalization step can help separate the classes in the SVM, being successful in improving the result of three of five approaches used, specially the zscore normalization, that considers the mean and standard deviation of the vectors.

To better understand our results, we also calculated the accuracy of the SVM classification for the same training used in it. Our experiments shows that the "different" class does not help the learning, especially in the datasets with more classes, as this "different" class overlook the other classes, preventing the classification as the correct class. It also shows the necessity of a bigger training and a validation set to tune the parameters of the SVM. Figure 2 shows a confusion matrix of a classification of the training data in the Letter-LOW dataset.

To improve our results, we propose to ignore the "different" class in the training set. Table 5 shows the accuracy for this new proposal.

As we can see in Table 5, our proposed modifications improved the results obtained in our experimental protocol. The dataset Letter-LOW achieved the best result when we do not consider the "different" class in the training step, avoiding misclassification as "different" class. With this, we show that our proposed approach to learn the cost to match nodes are very promising.

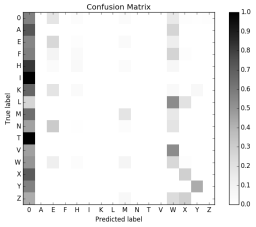

Fig. 2. Classification of the training set for the Letter LOW dataset.

Table 5. Accuracy scores for four datasets (in %).

Modification	Multi-class	Datasets			
		Letter-LOW	Letter-MED	Letter-HIGH	GREC
Without "different" class		37.87 ± 5.88	**34.13 ± 9.78**	**29.07 ± 4.36**	38.18 ± 8.86
	min-max	30.13 ± 6.34	30.13 ± 9.31	27.47 ± 7.92	35.45 ± 2.03
	zscore	**44.80 ± 5.94**	25.87 ± 0.73	29.07 ± 5.99	41.82 ± 7.11

4 Conclusions

In this paper, we presented an original approach to learn the costs to match nodes belonging to different graphs. These costs are later used to compute a dissimilarity measurement between graphs. The proposed learning scheme combines a node-signature-based distance vector and an SVM classifier to produce a cost matrix, based on which the Hungarian algorithm computes graph similarities. Performed experiments considered the graph classification problem, using k-NN classifiers built based on graph similarities. Promising results were observed for widely used graph datasets. These results suggest that our approach can also be extended to use similar methods based on local vectorial embeddings and can be exploited to compute probabilities as estimators of matching costs.

For future work, we want to perform experiments considering all training and testing sets to compare with our results presented in this paper, and also make a complete study on the minimum training set necessary to achieve a good performance not only in classification, but also in retrieval tasks.

Acknowledgments. Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER, and several Universities, as well as other organizations (see `https://www.grid5000.fr`).

References

1. Brun, L., Gaüzère, B., Fourey, S.: Relationships between Graph Edit Distance and Maximal Common Unlabeled Subgraph. Tech. rep. (Jul 2012)

2. Bunke, H.: On a relation between graph edit distance and maximum common subgraph. Pattern Recognition Letters **18**(8), 689 – 694 (1997)
3. Bunke, H., Allermann, G.: Inexact graph matching for structural pattern recognition. Pattern Recognition Letters **1**(4), 245 – 253 (1983)
4. Bunke, H., Günter, S., Jiang, X.: Towards bridging the gap between statistical and structural pattern recognition: Two new concepts in graph matching. In: Proceedings of the Second International Conference on Advances in Pattern Recognition. pp. 1–11. ICAPR '01, Springer-Verlag, London, UK (2001)
5. Bunke, H., Riesen, K.: Recent advances in graph-based pattern recognition with applications in document analysis. Pattern Recognition **44**(5), 1057 – 1067 (2011)
6. Caetano, T.S., McAuley, J.J., Cheng, L., Le, Q.V., Smola, A.J.: Learning graph matching. IEEE Trans. Pattern Anal. Mach. Intell. **31**(6), 1048–1058 (2009)
7. Cho, M., Alahari, K., Ponce, J.: Learning graphs to match. In: IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013. pp. 25–32 (2013)
8. Cortés, X., Serratosa, F.: Learning graph-matching edit-costs based on the optimality of the oracle's node correspondences. Pattern Recognition Letters **56**, 22–29 (2015)
9. Cortés, X., Serratosa, F.: Learning graph matching substitution weights based on the ground truth node correspondence. IJPRAI **30**(2) (2016)
10. Kuhn, H.W., Yaw, B.: The hungarian method for the assignment problem. Naval Res. Logist. Quart pp. 83–97 (1955)
11. Leordeanu, M., Sukthankar, R., Hebert, M.: Unsupervised learning for graph matching. International Journal of Computer Vision **96**(1), 28–45 (2012)
12. Neuhaus, M., Bunke, H.: Self-organizing maps for learning the edit costs in graph matching. IEEE Trans. Systems, Man, and Cybernetics, Part B **35**(3), 503–514 (2005)
13. Neuhaus, M., Bunke, H.: Automatic learning of cost functions for graph edit distance. Information Sciences **177**(1), 239 – 247 (2007)
14. Neuhaus, M., Bunke, H.: Bridging the Gap Between Graph Edit Distance and Kernel Machines. World Scientific Publishing Co., Inc., River Edge, NJ, USA (2007)
15. 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.) Structural, Syntactic, and Statistical Pattern Recognition. pp. 287–297. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
16. Riesen, K., Ferrer, M.: Predicting the correctness of node assignments in bipartite graph matching. Pattern Recognition Letters **69**, 8–14 (2016)
17. de Sa, J.M.: Pattern Recognition: Concepts, Methods, and Applications. Springer Science & Business Media (2001)
18. Silva, F.B., de O. Werneck, R., Goldenstein, S., Tabbone, S., da S. Torres, R.: Graph-based bag-of-words for classification. Pattern Recognition **74**(Supplement C), 266 – 285 (Feb 2018)
19. Wilson, D.R., Martinez, T.R.: Improved heterogeneous distance functions. J. Artif. Int. Res. **6**(1), 1–34 (Jan 1997)