Pattern Recognition Letters
journal homepage: www.elsevier.com

Learning Cost Function for Graph Classification with Open-Set Methods

Rafael de Oliveira Werneck[a], Romain Raveaux[b], Salvatore Tabbone[c], Ricardo da Silva Torres[a,d]

[a]*Institute of Computing, University of Campinas UNICAMP, 13083-852 Campinas, SP, Brazil*
[b]*Université François Rabelais de Tours, 37200 TOURS, France*
[c]*Université de Lorraine-LORIA UMR 7503, Vandoeuvre-lès-Nancy, France*
[d]*Department of ICT and Natural Sciences, NTNU, Norwegian University of Science and Technology, Trondheim, Norway*

ABSTRACT

In several pattern recognition problems, effective graph matching is of paramount importance. In this paper, we introduce a novel framework to learn discriminative cost functions. These cost functions are embedded into a graph matching-based classifier. The learning algorithm is based on an open-set recognition approach. An open-set recognition describes a problem formulation in which the training process does not have access to labeled samples of all classes that may show up during the test phase. We also investigate a set of measures to characterize local graph properties. Performed experiments considering widely used datasets demonstrate that our solution leads to better or comparable results to those observed for several state-of-the-art baselines.

1. Introduction

In several pattern recognition tasks, objects are often represented by means of two main approaches [1]: statistical or structural. In the former, objects are represented as points in n-dimensional space; while in the latter, objects are represented through data structures, which encode their components and relationships. The literature related to classification and retrieval tasks encompasses many more statistical representations. However, structural representations are more powerful, as they provide a single formalism on components and their relations [2]. In this work, we use graphs, one of the most adopted structural representation. In the field of structural pattern recognition, the graph comparison problem is of first importance. Unfortunately, due to the wide variability of patterns, the graph comparison problem is not a trivial task, as it often turns into an error-tolerant graph matching problem. The error-tolerant graph matching problem [3], in turn, is an $\mathcal{NP}$-hard problem [4]. Therefore, there are no exact methods that guarantee to solve the problem in polynomial time.

One successful tool to model the error-tolerant graph matching problem relies on the graph edit distance (GED) [5]. GED is an error-tolerant paradigm to define the similarity between two graphs through the minimum number of edit operations necessary to transform one graph into the other. A sequence of edit operations 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 operation. Consequently, we can define the edit distance between graphs as the edit path with minimum cost. Usually, cost functions are manually designed for each problem, being domain-dependent. Domain-dependent cost functions can be tuned by learning weights associated with them. In this paper, we tackle a more general problem. What can we learn if the cost functions are not given by an expert? Can we extract information from the data to fit a specific goal given by the user?

Different papers address the edit cost learning problem. The contribution presented in [6] is the most related to our proposal. In their work, the authors represent the node assignment as a vector of 24 features. These features are extracted from a node-to-node cost matrix that is used for the original matching process. Then, the assignments derived from the exact graph edit distance computation is used as ground truth. Each node assignment computed is labeled as correct or incorrect where an SVM with a Gaussian kernel classify the assignments computed

DOI: https://doi.org/10.1016/j.patrec.2019.08.010
Corresponding author:
e-mail: rafael.werneck@ic.unicamp.br (Rafael de Oliveira Werneck)

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 focusing on automatically designing generic cost functions in a classification context.

Recent initiatives have been focusing on the proposal of graph representation based on heat-kernel embeddings [7, 8], deep-learning methods [9], quantum walk [10], and generative models [11]. Some of them are detailed below.

Xiao et al. [7] proposed the characterization of the properties of a graph by means of the flow of information across edges. The rate of flow is computed through the Laplacian of the graph. They explored three approaches computed from the heat kernel matrix: zeta function of the heat kernel trace, derivative of the zeta function, and heat-content invariants. Xiao et al. [8] also exploited a heat-kernel formulation based on the Laplacian graph transformation. They presented an embedding scheme to construct a generative model for graph structure. They mapped the nodes of the graphs as points in a vector space, and then computed the correspondence matrix between these points with the Scott and Longuet-Higgins alignment algorithm. Later, they captured any variations in the graph structure through a covariance matrix of the embedding points, so they can construct a point-distribution model using the eigenvalues and eigenvectors of this matrix. This model can be used to measure the distance between a pair of graphs.

Bai et al. [10] developed new graph kernels where the graph structure is examined by means of discrete-time quantum walk. They simulated the evolution of the quantum-walk on each graph, computing their associated density matrix. Later, for a pair of graph, they compute the kernel by the negative exponential of Jensen-Shannon of their density matrix, using a minimum spanning tree of a sparser version of the original graph. Han et al. [11] focused on the problem of representing graphs by edge connectivity. They aimed to learn a generative model to describe the distribution of structural variations present in graphs. Their proposal learns a generative supergraph by the probability distribution over the occurrence of nodes and edges. They encoded the complexity measurement using a Von Neumann entropy, and later they used an EM algorithm to minimize the criterion of correspondence between graphs.

Bai et al. [9] proposed a work to combine graph complexity measures and deep learning networks. Their goal is to compute a representation for each vertex. Later, a single graph feature vector is computed by averaging vertices' representations. For that, they first decompose the graph structure into a family of expansions subgraphs rooted at a vertex, and measured the entropy-based complexities, which is used to build the complexity trace, i.e., the depth-based representation of the root vertex. Next, they perform a clustering using k-means to find prototype representations, which are used to train a deep neural network.

In this paper, we propose to learn a discriminative cost function between the nodes of graphs with no restriction on the graph type, nor on labels for a classification task. On a training set of graphs, a feature vector is extracted from each node of each graph, describing local information on the nodes. Node dissimilarity vectors are obtained by comparing pairs of feature vectors and labeled according to the node pair belonging to graphs of the same class or not. On this basis, a classifier is trained on these node dissimilarity vectors. At the decision stage, when comparing two graphs, a new node pair is given as an input of the classifier, and the class membership probability is output. We use these adapted costs to fill a node-to-node similarity matrix, which encodes our learned matching costs. Based on these costs, we reduce the graph matching problem to 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 [12] in $O(n^3)$ time. Instead of dealing with the graphs as a whole, we exploit their elements (e.g., their node attributes) to guide the weight learning process. Thus, as we increase the number of elements that we use for learning, we can take advantage of only a few graphs in the training process. Our method is, therefore, suitable for problems, which handle small-size training sets, either because they are difficult to obtain, or hard to label.

This paper extends the work presented in [13], by providing a theoretical overview of the introduced graph distance learning framework, as well as by detailing performed experiments related to the parametric evaluation of the proposed approach. We also present an original approach based on an open-set recognition problem formulation, in which the training step does not contain all classes because they are ill-sampled or unknown [14]. The goal is to learn the costs to match nodes from different graphs. The method is based on node-signatures, dissimilarities between node-signatures, a classifier to determine a cost matrix, and a Hungarian algorithm to compute similarities between graphs. Furthermore, this paper presents and discusses for the first time experiments related to the use of open-set classifiers in weight-learning problems associated with graph-classification tasks. To the best of our knowledge, this is the first work to perform such evaluation in the open-set scenario. Finally, another novelty of this work relies on the investigation of complex network measurements in the characterization of local properties of graphs.

Open-set scenario, differently from the closed-set scenario, does not have, *a priori*, training samples from all classes, as these classes might appear in the testing step [15]. Open-set classifiers consider that not all classes are known *a priori* at training time. Therefore, a test sample can belong to a class from the training or it can belong to a class not "seen" during training, i.e., this sample can be considered as *"unknown."* In this paper, we take advantage of this formulation by mapping the distance vector related to nodes belonging to different classes as "unknown." By doing that, learned cost functions are expected to encode more properly existing relations among nodes of vertices of the same class, leading to more discriminant graph matching.

2. Graph Distance Learning Framework

We propose a new framework to learn a discriminative cost function for computing the bipartite graph edit distance between two graphs. In our method, we describe each graph

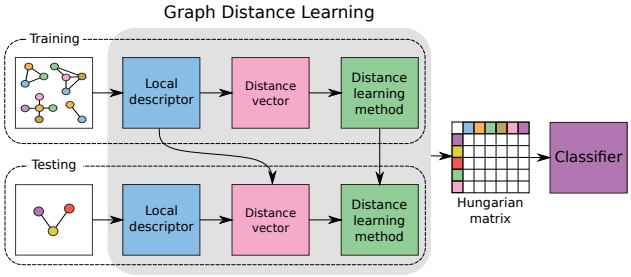

Fig. 1. Schematic overview of the Graph Distance Learning framework.

from a training set using a local descriptor. We extract feature vectors from each node of each graph. Next, we compute node dissimilarity vectors pair-wisely, generating feature vectors. These node dissimilarity vectors are then labeled according to the node pair. If the node pair belongs to the same graph class, the dissimilarity vector received the same label; if not, it is labeled as belonging to an "unknown" class. Later, a distance learning classifier is trained according to the distance vectors. At the decision phase, a graph from the testing set is compared to a graph from the training set. All its nodes are described by a local descriptor and it is computed the dissimilarity vector between test and training samples. These vectors are the input of the distance learning classifier, which returns the class membership probability. These probabilities are the adapted costs used to fill a node-to-node similarity matrix between the two graphs. We use these learned matching costs to approximate the problem of matching graphs as a Linear Sum Assignment Problem (LSAP) between the nodes of two graphs. The LSAP, which aims to find the minimum cost matching between elements of two sets, can be solved by the Hungarian Algorithm [12] in $O(n^3)$ time. Figure 1 shows a schematic view of the proposed Graph Distance Learning framework. In the following, we describe each component of this framework.

2.1. Local descriptor

To describe the graphs of the training and testing sets, we propose the use of local descriptors to characterize local properties of all graph nodes. Then, we can compare them pair by pair, and calculate the matching cost to transform a set of nodes from one graph to the set of nodes of the other graph.

Given a general graph $G = (V, E)$, a local description is defined as:

$$\Gamma(G) = \{\gamma(v) \mid \forall v \in V\}, \tag{1}$$

where $\gamma(v)$ is a local descriptor which encodes local properties of vertex v into a vector.

2.2. Distance vector

Our proposed approach for graph matching consists in finding a minimum distance to transform a local description from one graph into a local description from another graph. To perform that, we use a function to calculate the distance between two local descriptors.

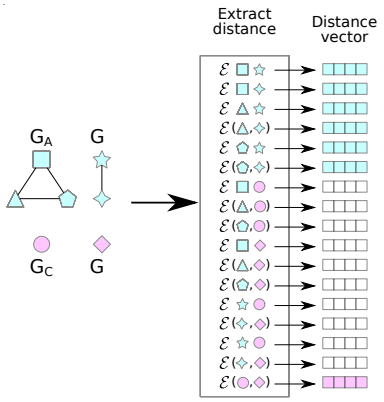

Fig. 2. Illustration of the creation of a distance vector based on node properties of four graphs. When the nodes belong to graphs of the same class (same color – blue and pink distance vectors – in the figure), the distance vector receives the same label. Alternatively, when the nodes belong to graphs of different classes, the distance vector is labeled as "unknown" (white distance vectors).

Let G_I and G_J be two graphs, v_i and v_j two nodes from these graphs, and $\gamma(v_i)$ and $\gamma(v_j)$ be two local descriptions of these nodes. We define a function $\mathcal{E}$ that, using $\gamma(v_i)$ and $\gamma(v_j)$ as inputs, returns a feature vector (d) representing the distance between these two local descriptions.

$$\mathcal{E}(\gamma(v_i), \gamma(v_j)) = d_{ij} \tag{2}$$

To each distance vector d_{ij}, we assign a class label defined in set $\mathcal{L}$. The set containing possible labels (classes) is defined as:

$$\mathcal{L}(d_{ij}) \subset \mathcal{L}(G_I) \cup \mathcal{L}(G_J) \cup \{\text{unknown}\} \tag{3}$$

Figure 2 illustrates the computation of distance vectors based on the properties of vertices belonging to four graphs (graphs G_A, G_B, G_C, and G_D in the figure) and their labeling process.

2.3. Distance learning component

This component of our Graph Distance Learning framework is responsible for learning a cost value related to each distance vector received as input. We propose this component as a function $\mathcal{F}$, in which we obtain the probability of the desired class:

$$\mathcal{F} : \mathcal{D} \rightarrow \mathbb{R}^{|\mathcal{L}|} \tag{4}$$

where $\mathcal{D}$ is the set of all distance vectors computed from vertices of two input graphs.

2.3.1. Hungarian matrix and classification

After we obtain the cost output from the distance learning method, we use these values to populate a cost matrix relative to the combination of each testing graph with each graph from the training set. The cost matrix contains the local description from one testing graph in the rows and the local description from one training graph in the columns. Thus, each entry of the matrix is the cost to transform the description from the row to the description of the column. Thus, the Hungarian algorithm finds the minimum cost assignment between the two sets of signatures.

Finally, the test sample is classified using the k-nearest neighbor (kNN), where the similarity between two graphs is defined by the Hungarian algorithm.

3. Graph Distance Learning Implementation

In this section, we provide an instantiation of the proposed framework, detailing implementation choices.

3.1. Local Descriptor

To describe local information of the graphs in this work, we use information of a graph and their nodes following the node signature:

$$\gamma(v) = \{\alpha_v^G, \theta_v^G, \Delta_v^G, \Omega_v^G\}, \tag{5}$$

where $G = (V, E)$ is a graph defined by vertices in V and edges in E, $v \in V$, and α_v^G, θ_v^G, Δ_v^G, and Ω_v^G are, respectively, the attributes of the node v, the degree of node v, the set of degrees of adjacent nodes to v, and a set of attributes of the incident edges of v [13, 16].

In this paper, we also investigate the use of complex network measurements in the characterization of graph local properties. We use the following complex network measurements:

- Vulnerability (V_n), which presents the difference in performance when the node is removed from the graph [17]: $V_v = \frac{E-E_v}{E}$, where E is the global efficiency of the graph, and E_v is the global efficiency after the removal of node v;
- Clustering coefficient (C_v), which is the fraction of possible triangles that exist including the node [18]: $C_v = \frac{N_\Delta(v)}{N_3(v)}$, where $N_\Delta(v)$ is the number of triangles with node v and $N_3(v)$ is the number of connected triples with v as central node;
- Cyclic coefficient (Θ_v), which measures how cyclic a graph is, defined by the average of the inverse of the sizes of the smallest cycles formed by the node and its neighbors [19]: $\Theta_v = \frac{2}{n_v(n_v-1)} \sum_{w>u} \frac{1}{S_{uvw}} a_{uv} a_{vw}$, where n_v is the number of neighbors of node v, S_{uvw} is the size of smallest circle that passes through nodes u, v, w, and a_{uv} are the elements of adjacency matrix;
- Subgraph centrality (SC_v), which considers the number of subgraphs that constitute a closed walk starting and ending at the given node [20]: $SC_v = \sum_{k=0}^{\infty} \frac{(A^k)_{vv}}{k!}$, where $(A^k)_{vv}$ is the vth diagonal element of the kth power of adjacency matrix A, and $k!$ assures the convergence of the sum and that smaller subgraphs have more weight;
- the average neighbor degree [21]. The degree of a vertex v is the number of edges incident to v.

3.2. Distance vector

Our proposed approach for graph matching consists in finding a minimum distance to transform a node signature from one graph into a node signature from another graph. To perform that, we first need to define a function to calculate the distance between two node signatures, and in our case, a function that is capable of dealing with both numeric and symbolic attributes. We selected the *Heterogeneous Euclidean Overlap Metric* (HEOM) [22] which deals with these attributes, and adapted for our graph local descriptor.

The default HEOM distance function is defined as follow:

$$\text{HEOM}(d_i, d_j) = \sqrt{\sum_a \delta(d_{ia}, d_{ja})^2} \tag{6}$$

for d_i and d_j two heterogeneous feature vectors, where a is each attribute of the vector. $\delta(d_{ia}, d_{ja})$ is also defined as:

$$\delta(d_{ia}, d_{ja}) = \begin{cases} 1 & \text{if } d_{ia} \text{ or } d_{ja} \text{ is missing,} \\ 0 & \text{if } a \text{ is symbolic and } d_{ia} = d_{ja}, \\ 1 & \text{if } a \text{ is symbolic and } d_{ia} \neq d_{ja}, \\ \frac{|d_{ia} - d_{ja}|}{\text{range}_a} & \text{if } a \text{ is numeric} \end{cases} \tag{7}$$

Considering the node signature local descriptor, we define the HEOM distance between two signatures as follow. Considering $A = (V_a, E_a)$ and $B = (V_b, E_b)$ two graphs, $v_a \in V_a$ and $v_b \in V_b$ nodes from these graphs. According to Equation 5, the node signatures of these nodes are: $\gamma(v_a) = \{\alpha_{v_a}^A, \theta_{v_a}^A, \Delta_{v_a}^A, \Omega_{v_a}^A\}$ and $\gamma(v_b) = \{\alpha_{v_b}^B, \theta_{v_b}^B, \Delta_{v_b}^B, \Omega_{v_b}^B\}$. Then, the distance ϵ between two node signatures is:

$$\begin{aligned} \epsilon(\gamma(v_a), \gamma(v_b)) = & HEOM(\alpha_{v_a}^A, \alpha_{v_b}^B) + HEOM(\theta_{v_a}^A, \theta_{v_b}^B) + \\ & HEOM(\Delta_{v_a}^A, \Delta_{v_b}^B) + \\ & \frac{\sum_{i=1}^{|\Omega_{v_a}^A|} HEOM(\Omega_{v_a}^A(i), \Omega_{v_b}^B(i))}{|\Omega_{v_a}^A|} \end{aligned} \tag{8}$$

The goal of the Graph Matching Learning framework is to learn the edit distance between two graphs. For that, we need to define the *distance vector* that will be used in the cost learning process [13]. The function $\mathcal{E}$, which defines the *distance vector*, is based on the ϵ function. Instead of summing the distance of all attributes, $\mathcal{E}$ considers each attribute distance as a bin of the vector. Therefore, we can present the function $\mathcal{E}$ as:

$$\begin{aligned} & \mathcal{E}(\gamma(v_a), \gamma(v_b)) = [HEOM(\gamma(v_a)_i, \gamma(v_b)_i)], \\ & \forall i \in \{0, \cdots, |\gamma(v)|\} \mid \gamma(v)_i \text{ is a attribute of } \gamma(v). \end{aligned} \tag{9}$$

Using complex network measures, the node signature is defined as: $\gamma(v_a) = \{\alpha_{v_a}^A, \theta_{v_a}^A, \Delta_{v_a}^A, \Omega_{v_a}^A, V_{v_a}^A, C_{v_a}^A, \Theta_{v_a}^A, SC_{v_a}^A, AVG_{v_a}^A\}$, and Equation 9 is adapted accordingly.

Later, we label these distance vectors to guide our learning process. We proposed the following formulation [13]. Let $Y = \{y_1, y_2, \cdots, y_l\}$ be a set of l labels associated with the graphs according to the target graph classification problem. In this formulation, a label y_i is assigned to each distance vector built

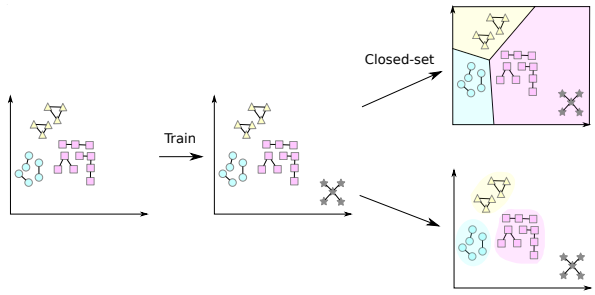

Fig. 3. Differences in the classification of the "X"-shaped test graph from the closed set (upper) and open-set (bottom) approaches. From the closed-set perspective, the test graph is labeled as belonging to the purple class. For the open-set perspective, the same test set is labeled as "unknown".

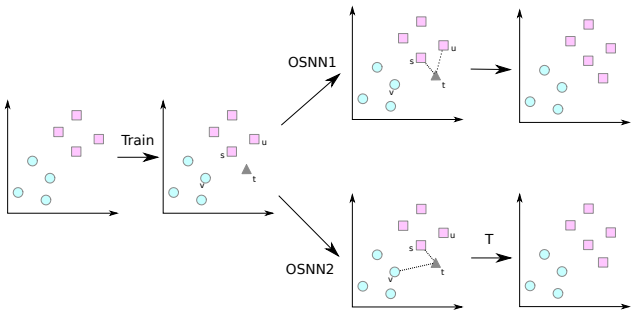

Fig. 4. Differences between OSNN1 and OSNN2 open-set recognition approaches when selecting training neighbors. The OSNN1 approach considers the two closest training neighbors. If they are from the same class, the test sample (in black) is labeled as belonging to this class, otherwise, as "unknown". For the OSNN2 approach, the two nearest neighbors from different classes are selected, and if the ratio of the distances to them is below a threshold defined in the training step, the test sample is labeled with the label of the closest class. Otherwise, it is labeled as "unknown".

based on the node signatures of graphs belonging to the same class y_i. On the other hand, when a distance vector is built from node signatures of graphs belonging to different classes, an "unknown" label (e.g., y_{i+1}) is adopted (see Figure 2).

3.3. Distance Learning Component

In this paper, we present two proposals for learning the graph edit distance between two graphs, using closed-set and open-set formulations.

Figure 3 illustrates graph classification tasks from both the closed-set and open-set perspectives. The test sample is the "X"-shaped graph. From the closed-set perspective, the test graph is labeled as belonging to the purple class, i.e., all test samples will receive one of the labels considering at the training stage. On the other hand, from the open-set perspective, the same test set is labeled as "unknown", i.e., test samples, which are not "close" enough to labeled samples seen at training stage, are considered to belong to an "unknown" category.

3.3.1. Closed-set Formulation

The first approach, the closed-set one, aims to learn how to classify the distance vectors obtained in the previous step. For that, after obtaining the pairwise distance vectors, the vectors from the training set are used to learn a classifier. In this work, we learn the Support Vector Machine (SVM) margin that separates samples of the training set from different classes.

With the margin, we can predict the classes of the graphs in the testing set. First, we extract the local descriptor of each graph of the testing set. Next, we compute the distance vectors considering the node signatures from the test graph with the node signatures from the graphs of the training set. These vectors are projected into the learned feature space and we obtain the probability of a test sample belongs to the training set classes considering the SVM separation hyperplane.

3.3.2. Open-set Formulation

Our second approach is based on an open-set formulation, in which we can classify as "unknown" samples that do not belong to the different class available during the training step.

Scheirer et al. [14] presented a formalization for recognition problems from the open-set perspective. This formalization aims to find a function f, which minimizes the combination of the open space risk R_O and the empirical risk $R_\mathcal{E}$, the later regularized by a constant λ_r:

$$\operatorname{argmin}\{R_O(f) + \lambda_r R_\mathcal{E}(f)\} \tag{10}$$

In this paper, we investigate the use of two recently proposed open-set-based learning methods [15]: Open-Set Nearest Neighbors 1 (OSNN1) and Open-Set Nearest Neighbors 2 (OSNN2).

In the OSNN1 method, during the prediction phase, the two training-set nearest neighbors (s and u) of an input test sample t are selected. If they have the same label, this label is assigned to the test sample, otherwise, the test sample is unknown, i.e., to the test sample the *unknown* label is assigned.

The OSNN2, in turn, labels an input test sample t as follows: it first finds the two training-set nearest neighbors of different labels (s and v, being s the nearest), and then calculates the ratio

$$R = \frac{d(t,s)}{d(t,v)} \tag{11}$$

and assign the label according to the following condition:

$$label(t) = \begin{cases} label(s), & \text{if } R \leq \text{threshold}, \\ unknown, & \text{if } R > \text{threshold}. \end{cases} \tag{12}$$

Figure 4 shows the difference between the two open-set approaches when selecting the closest training neighbors.

4. Experiments

In this section, we present the research questions addressed in our experiments, the datasets used, and the adopted evaluation protocol adopted for each research question.

4.1. Datasets

We select traditional and widely used datasets of the literature to perform our experiments. Each dataset is detailed next.

- **MAO**: Monoamine Oxidase dataset[1] is a dataset that consists of 68 molecules, with 38 molecules that inhibit the monoamine oxidase and 30 that do not. The standard evaluation protocol adopted for this dataset relies on a Leave-one-out cross-validation, where 67 graphs are used for training and the remaining sample is used for testing.
- **PAH**: The Polycyclic Aromatic Hydrocarbons dataset [23] is composed of 94 graphs representing molecules composed only of carbon atoms. All bound in these molecules are aromatics. The typical evaluation protocol used for this dataset relies on a 10-fold cross-validation procedure. In this protocol, we have ≈ 84 graphs per fold.
- **GREC**: The GREC dataset [24] consists of graphs representing architectural and electronic drawings. The nodes are ending points in the drawings, and the graph edges are the lines and arcs. It contains 1100 graphs divided into 22 classes. The default evaluation protocol of this dataset consists of 286 graphs for training, 286 graphs for validation, and 528 graphs for testing.

4.2. Research Questions and Experimental Protocol

In this work, we use different experiment protocols for addressing each research question.

Q1 *What is the impact of the training set size and normalization procedures in the effective performance of the evaluated learning methods?*
In the first question, we want to assess the robustness of the different learning methods with regard to different parameter setting. Recall that our Graph Distance Learning framework relies on the computation of multiple pairwise distance vectors, being therefore computationally costly. We decided, then, to perform experiments using only a subset of the available training sets in our parameter setting investigation. In order to assess the effective performance of the methods for different training set sizes, let s be the number of graphs per class. We vary s in the set $\{2, 5, 10, 20\}$. Also, we use only 10% of the available testing set. The graphs used for training and testing were defined randomly. Our reported results refer to the average effective performance, considering 20 runs using the different randomly selected samples. We also want to assess the impact of different normalization strategies on the effectiveness performance of the evaluated methods. We used the min-max normalization, in which the vectors are normalized between 0 and 1 according to minimum and maximum values observed; and the zscore normalization, in which we use the mean and standard deviation to normalize distance score values. In the Graph Distance Learning method, the selected parameters for the SVM closed-set approach was the default ones (RBF kernel with $C = 0$). Open-set approaches OSNN1 and OSNN2 do not have any parameters to setup. Experiments related to Q1 considered the MAO and PAH datasets, and effectiveness results refer to the average normalized accuracy in the graph classification problems defined for each dataset.

[1] https://brunl01.users.greyc.fr/CHEMISTRY/index.html (As of Jan. 2019).

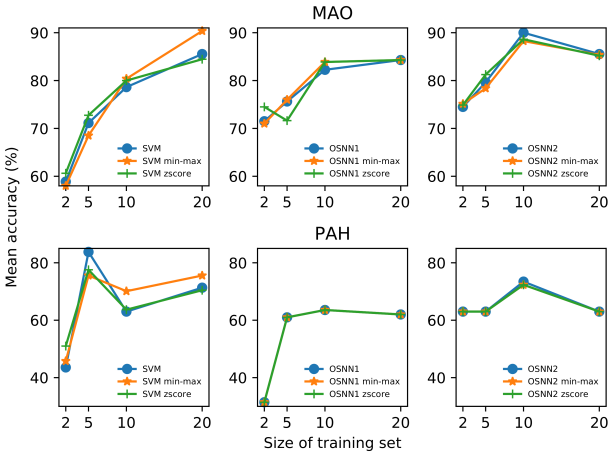

Fig. 5. Evaluation of the different weight learning strategies with regard to the use of normalization procedures and different training set sizes.

Q2 *Which learning method leads to better effectiveness performance?*
Our goal here is to compare the open-set formulations with the SVM-based closed-set solution in the weight cost learning problem. Our evaluation regarding Q2 also considers the use of complex network measurements in the characterization of graph local properties. The experimental protocol is similar to the one described in the previous item. The differences are: we only use the variations of the methods with the best performance observed in the previous experiments, an additional dataset (GREC) is used in our comparisons.

Q3 *How effective are the proposed methods when compared to state-of-the-art solutions?*
Our goal here is to demonstrate that the proposed learning methods yield better or comparable results to those observed for state-of-the-art baselines for different datasets. In order to compare our approach with baselines in the MAO dataset, we consider the evaluation protocol usually employed in the assessment of methods using this dataset (see Section 4.1). We also perform experiments to compare the performance of the incremental increase in the size of the training set.

5. Results and Analysis

5.1. Q1: Impact of normalization and the size of training sets

Figure 5 presents the results observed for the evaluated weight learning methods, considering different normalization strategies (e.g., min-max, and zscore). In this figure, we also assess the robustness of the method with regard to the size of the training set size. The first and the second lines of Figure 5 refer to the MAO and the PAH datasets, respectively. Good results are obtained with just a few graph examples from the training set and as we can observe 10 graphs per class is a good compromise for the open-set methods. Related to the normalization, the min-max normalization obtained the overall best results in our experiments, thus, we will be using this normalization in the next experiments.

Table 1. Best results observed for the different weight learning strategies in terms of normalized accuracy. In all cases, 10 graphs are used for training.

	MAO	PAH	GREC
SVM	80.38	70.11	23.52
OSNN1	83.88	63.56	56.25
OSNN2	88.25	72.33	58.98

Table 2. Best results observed for the different weight learning strategies in terms of normalized accuracy, considering the use of complex network measurements in the characterization of graph local properties. In all cases, 10 graphs are used for training.

	MAO	PAH	GREC
SVM	79.13	55.33	44.20
OSNN1	90.13	93.67	49.66
OSNN2	95.38	84.11	73.52

5.2. *Q2: Identification of the best learning methods*

Table 1 presents the best results observed for the SVM, OSNN1, and OSNN2 learning methods for the MAO, PAH, and GREC datasets, considering only 10 randomly graphs for training. As we can observe, the OSNN2 classifier obtained the best accuracy score considering all datasets. As the OSNN2 classifier considers the distance relation between two classes, it can have a better separation of the classes, leading to a high accuracy score.

We also performed some experiments in which we consider the use of complex network measurements in the local properties of the graph. Table 2 shows that improving the local representation of the nodes, the overall accuracy increases, especially for the OSNN2 classifier.

5.3. *Q3: Comparison with state-of-the-art baselines*

In this comparison with the state of the art, we perform a few experiments considering the same evaluation protocol used by the literature, and a simple modification using fewer graphs per training. Table 3 presents the obtained results of our solution and state-of-the-art approaches in the MAO dataset. We have slightly modified the leave-one-out protocol to assess the impact of different training set sizes. OSNN2(X-Y), in the table, refers to the use of the OSNN2 method, training with X samples of class 0 and Y samples of class 1. As we can see, our results have not yet beaten the state of the art, but it comes as a close third best using only 17 graphs per class in the training set. Our result with all graphs of the training is a little further in the table. This happens mainly because our approach to find the combination of all node signatures results in an overtraining for our classifier, because of the unbalance of the training classes. However, as we can see in Table 1 and 2, we can achieve close or better results using fewer graphs for training.

5.4. *Computational complexity and runtimes*

Let n be the number of training graphs and v_n the total number of vertices in the training graphs. Similarly, let m be the number of testing graphs and v_m, the total number of vertices in the testing graphs.

Table 3. Comparison of our approach with the same evaluation protocol defined in [25] using the MAO dataset.

	MAO
El-Atta et al. [26]	98.5
Mahé et al. [27] [25]	96
Gaüzère et al. Treelet Kernel [25]	94
OSNN2 (17-17)	**92.65**
OSNN2 (15-15)	**91.12**
Riesen et al. [28] [25]	91
Neuhaus and Bunke [29] [25]	90
Gaüzère et al. Normalized Graph Laplacian Kernel [25]	90
Gaüzère et al. Normalized Fast Graph Laplacian Kernel [25]	90
OSNN2 (18-18)	**89.71**
OSNN2 (10-10)	**88.24**
OSNN2 (38-30)	**83.82**
Vishwanathan et al. [30] [25]	82
Suard et al. [31] [25]	80
OSNN2 (5-5)	76.47

At the training phase, the computation complexity of the proposed method depends on the (a) computation of the vertex feature vector representation (local descriptor computation); (b) computation of the distance vectors; and (c) the distance learning. The computational costs of each step of the training phase can be defined as:

(a) Local Descriptor computation: $O(v_n)$;
(b) Distance Vector computation: $O(v_n^2)$;
(c) Distance Learning method: $O(v_n^4)$ as pointed out in [32] for the SVM classifier (closed scenario).

The worst case complexity for training is, therefore, $O(v_n^4)$

The test phase comprises (a) the local descriptor computation for the test set; (b) computation of distance vectors considering test and training graphs; (c) population of the Hungarian matrix using the trained classifier; and (d) computation of the Hungarian Algorithm. The computational costs of each step of the test phase can be defined as:

(a) Local Descriptor computation: $O(v_m)$;
(b) Distance Vector computation: $O(v_n \times v_m)$;
(c) Population of the Hungarian matrix: $O(c \times n \times m)$, where c stands for the probability score computation cost defined by the classifier.
(d) Computation of the Hungarian Algorithm: the Hungarian algorithm computation takes $O(p^3)$ where p is the maximum dimension of the input Hungarian matrix [33]. As this computation is performed $n \times m$, the worst complexity is $O(n \times m \times p^3)$.

Considering the complexity calculated above, we present a few runtimes of our experiments. Table 4 shows the mean runtimes of each iteration in the MAO dataset with Leave-One-Out protocol.

Our proposed approach is somewhat costly because it considers the local descriptions of the graphs to learn the Hungarian matrix cost function. Also, the computation of the Hungarian algorithm itself is quite expensive.

Table 4. Mean runtimes of each iteration in the MAO dataset with the Leave-One-Out protocol.

Method	Runtime (s)
OSNN2 (17-17)	2680 ± 439
OSNN2 (18-18)	3607 ± 671
OSNN2 (20-20)	5384 ± 458
OSNN2 (38-30)	74 391 ± 2029

6. Conclusions

In this work, we introduced new approaches to learn discriminative costs for a bipartite graph edit distance computation between two graphs. We present a generic framework, and then we describe different methods, based on both closed-set and open-set learning paradigms, used to implement the proposed framework. To the best of our knowledge, this is the first work to model the cost function learning process as an open-set recognition problem. Another novelty of this work relies on the investigation of complex network measurements in the characterization of graph local properties, aiming to obtain more effective cost function matrices. Performed experiments considered widely used datasets and evaluation protocols. Achieved results demonstrate that the proposed framework is effective, leading to comparable and better effectiveness results in different graph classification problems when compared with several baselines. One positive property of our solution relies on its capacity of leading to effective results, even when only a few samples ($\approx$ 10 graphs) are used for training.

In our future work, we intend to deepen the investigations of the use of other complex network measurements in the local characterization of graph properties [34]. We also plan to extend our investigation regarding the use and combination of other open-set recognition approaches [35].

Acknowledgments

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. Experiments presented in this paper were carried out using the Grid5000 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` (As of Oct. 2018). This study was financed in part by the Coordenaçãoo de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

[1] H. Bunke, S. Günter, X. Jiang, 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, ICAPR '01, London, UK, 2001, pp. 1–11.

[2] F. B. Silva, R. de O. Werneck, S. Goldenstein, S. Tabbone, R. da S. Torres, Graph-based bag-of-words for classification, Pattern Recognition 74 (Supplement C) (2018) 266 – 285.

[3] H. Bunke, G. Allermann, Inexact graph matching for structural pattern recognition, Pattern Recognition Letters 1 (4) (1983) 245 – 253.

[4] Z. Zeng, A. K. H. Tung, J. Wang, J. Feng, L. Zhou, Comparing stars: On approximating graph edit distance, PVLDB 2 (1) (2009) 25–36.

[5] K. Riesen, Structural Pattern Recognition with Graph Edit Distance - Approximation Algorithms and Applications, Advances in Computer Vision and Pattern Recognition, Springer, 2015.

[6] K. Riesen, M. Ferrer, Predicting the correctness of node assignments in bipartite graph matching, Pattern Recognition Letters 69 (2016) 8–14.

[7] B. Xiao, E. R. Hancock, R. C. Wilson, Graph characteristics from the heat kernel trace, Pattern Recognition 42 (11) (2009) 2589 – 2606.

[8] B. Xiao, E. R. Hancock, R. C. Wilson, A generative model for graph matching and embedding, Computer Vision and Image Understanding 113 (7) (2009) 777 – 789.

[9] L. Bai, L. Cui, X. Bai, E. R. Hancock, Deep depth-based representations of graphs through deep learning networks, Neurocomputing 336 (2019) 3 – 12, advances in Graph Algorithm and Applications.

[10] L. Bai, L. Rossi, L. Cui, Z. Zhang, P. Ren, X. Bai, E. Hancock, Quantum kernels for unattributed graphs using discrete-time quantum walks, Pattern Recognition Letters 87 (2017) 96 – 103, advances in Graph-based Pattern Recognition.

[11] L. Han, R. C. Wilson, E. R. Hancock, Generative graph prototypes from information theory, IEEE Transactions on Pattern Analysis and Machine Intelligence 37 (10) (2015) 2013–2027.

[12] H. W. Kuhn, B. Yaw, The hungarian method for the assignment problem, Naval Res. Logist. Quart (1955) 83–97.

[13] R. de Oliveira Werneck, R. Raveaux, S. Tabbone, R. da Silva Torres, Learning cost functions for graph matching, in: Joint IAPR International Workshop, S+SSPR 2018, Beijing, China, August 17-19, 2018, Proceedings, 2018, pp. 345–354.

[14] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, T. E. Boult, Toward open set recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (7) (2013) 1757–1772.

[15] P. R. Mendes Júnior, R. M. de Souza, R. d. O. Werneck, B. V. Stein, D. V. Pazinato, W. R. de Almeida, O. A. B. Penatti, R. d. S. Torres, A. Rocha, Nearest neighbors distance ratio open-set classifier, Machine Learning 106 (3) (2017) 359–386.

[16] S. Jouili, S. Tabbone, Graph matching based on node signatures, in: A. Torsello, F. Escolano, L. Brun (Eds.), Graph-Based Representations in Pattern Recognition, Springer Berlin Heidelberg, 2009, pp. 154–163.

[17] V. Gol'dshtein, G. Koganov, G. I. Surdutovich, Vulnerability and hierarchy of complex networks, arXiv preprint cond-mat/0409298.

[18] D. J. Watts, S. H. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (1998) 440 EP–.

[19] H.-J. Kim, J. M. Kim, Cyclic topology in complex networks, Phys. Rev. E 72 (2005) 036109.

[20] E. Estrada, J. A. Rodríguez-Velázquez, Subgraph centrality in complex networks, Phys. Rev. E 71 (2005) 056103.

[21] R. Pastor-Satorras, A. Vázquez, A. Vespignani, Dynamical and correlation properties of the internet, Phys. Rev. Lett. 87 (2001) 258701.

[22] D. R. Wilson, T. R. Martinez, Improved heterogeneous distance functions, J. Artif. Int. Res. 6 (1) (1997) 1–34.

[23] B. Gaüzère, L. Brun, D. Villemin, Graph kernel encoding substituents' relative positioning, in: International Conference on Pattern Recognition (ICPR), Stockholm, Sweden, 2014, p. 6 p.

[24] K. Riesen, H. Bunke, Iam graph database repository for graph based pattern recognition and machine learning, in: Structural, Syntactic, and Statistical Pattern Recognition, 2008, pp. 287–297.

[25] B. Gaüzère, L. Brun, D. Villemin, Two new graphs kernels in chemoinformatics, Pattern Recognition Letters 33 (15) (2012) 2038 – 2047, graph-Based Representations in Pattern Recognition.

[26] A. H. A. El-Atta, A. E. Hassanien, Two-class support vector machine with new kernel function based on paths of features for predicting chemical activity, Information Sciences 403-404 (2017) 42 – 54.

[27] P. Mahé, J. Vert, Graph kernels based on tree patterns for molecules, Machine Learning 75 (1) (2009) 3–35.

[28] K. Riesen, M. Neuhaus, H. Bunke, Graph embedding in vector spaces by means of prototype selection, in: F. Escolano, M. Vento (Eds.), Graph-Based Representations in Pattern Recognition, Springer Berlin Heidel-

berg, Berlin, Heidelberg, 2007, pp. 383–393.
[29] M. Neuhaus, H. Bunke, Bridging the Gap Between Graph Edit Distance and Kernel Machines, World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2007.
[30] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, K. M. Borgwardt, Graph kernels, Machine Learning Research 11 (2010) 1201–1242.
[31] F. Suard, A. Rakotomamonjy, A. Bensrhair, Kernel on bag of paths for measuring similarity of shapes, in: ESANN 2007, 15th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 25-27, 2007, Proceedings, 2007, pp. 355–360.
[32] J. Platt, Sequential minimal optimization: A fast algorithm for training support vector machines, Advances in Kernel Methods-Support Vector Learning 208.
[33] E. L. Lawler, Combinatorial Optimiation: Networks and Matroids, 2001.
[34] L. d. F. Costa, F. A. Rodrigues, G. Travieso, P. R. Villas Boas, Characterization of complex networks: A survey of measurements, Advances in physics 56 (1) (2007) 167–242.
[35] M. A. C. Neira, P. R. M. Júnior, A. Rocha, R. da Silva Torres, Data-fusion techniques for open-set recognition problems, IEEE Access 6 (2018) 21242–21265.