A Metric Learning Approach to Graph Edit Costs for Regression

Linlin Jia, Benoit Gaüzère, Florian Yger, Paul Honeine

A Metric Learning Approach to Graph Edit Costs for Regression

Linlin Jia[1,4], Benoit Gaüzère[1,4], Florian Yger[2⋆], and Paul Honeine[3,4]

[1] LITIS Lab, INSA Rouen Normandie, France
[2] LAMSADE, Université Paris Dauphine-PSL, France
[3] LITIS Lab, Université de Rouen Normandie, France
[4] Normandie Université, France

Abstract. Graph edit distance (GED) is a widely used dissimilarity measure between graphs. It is a natural metric for comparing graphs and respects the nature of the underlying space, and provides interpretability for operations on graphs. As a key ingredient of the GED, the choice of edit cost functions has a dramatic effect on the GED and therefore the classification or regression performances. In this paper, in the spirit of metric learning, we propose a strategy to optimize edit costs according to a particular prediction task, which avoids the use of predefined costs. An alternate iterative procedure is proposed to preserve the distances in both the underlying spaces, where the update on edit costs obtained by solving a constrained linear problem and a re-computation of the optimal edit paths according to the newly computed costs are performed alternately. Experiments show that regression using the optimized costs yields better performances compared to random or expert costs.

Keywords: Graph edit distance · Edit costs · Metric Learning.

1 Introduction

Graphs provide a flexible representation framework to encode relationships between elements. In addition, graphs come with an underlying powerful theory. However, the graph space cannot be endowed with the mathematical tools and properties associated with Euclidean spaces. This issue prevents the use of classical machine learning methods mainly designed to operate on vector representations. To learn models on graphs, several approaches have been designed to leverage this flaw and among these, we can cite graph embeddings strategy [17], graph kernels [3, 27] and more recently graph neural networks [8]. Despite their state-of-the-art performances, they seldom operate directly in the graph space, hence reducing the interpretability of the underlying operations.

To overcome these issues, one needs to preserve the property of the graph space. For this purpose, one needs to define a dissimilarity measure in the graph

⋆ with the support of the ANR "Investissements d'avenir" program ANR-19-P3IA-0001 (PRAIRIE 3IA Institute) and grant ESIGMA ANR-17-CE23-0010.

space, in order to constitute the minimal requirement to implement simple machine learning algorithms like the k-nearest neighbors. The most used dissimilarity measure between graphs is the graph edit distance (GED) [10, 26]. The GED of two graphs G_1 and G_2 can be seen as the minimal amount of distortion required to transform G_1 into G_2. This distortion is encoded by a set of edit operations whose sequence constitutes an edit path. These edit operations include nodes and edges substitutions, removals, and insertions. Depending on the context, each edit operation e included in an edit path γ is associated with a non-negative cost $c(e)$. The sum of all edit operation costs included within the edit path defines the cost $A(\gamma)$ associated with this edit path. The minimal cost[5] among all edit paths $\Gamma(G_1, G_2)$ defines the GED between G_1 and G_2, namely

$$\texttt{ged}(G_1, G_2) = \min_{\gamma \in \Gamma(G_1, G_2)} A(\gamma). \tag{1}$$

Evaluating the GED is computationally costly and cannot be done in practice for graphs having more than 20 nodes in general. To avoid this computational burden, strategies to approximate the GED in a limited computational time have been proposed [1] with acceptable classification or regression performances.

An essential ingredient of the GED is the underlying edit cost function $c(e)$, which quantifies the distortion carried by the edit operation e. The values of the edit costs for each edit operation have a major impact on the computation of GED and its performance. Thus, the cost edit function may be different depending on the data encoded by the graph and the one to predict. Generally, they are fixed *a priori* by an expert of the domain, and are provided with the datasets.

However, these predefined costs are not optimal for any prediction tasks, in the same spirit as the no free lunch theorems for machine learning and statistical inference. In addition, these costs may have a great influence on both the prediction performance and the computational time required to compute the graph edit distance. In [9], the authors show that a particular set of edit costs may reduce the problem of computing graph edit distance to well-known problems in graphs like (sub)graph isomorphism or finding the maximum common subgraph of a pair of graphs. This point shows again the importance of the underlying cost function when computing a graph edit distance.

In this paper, we propose a simple strategy to optimize edit costs according to a particular prediction task, and thus avoid the use of predefined costs. The idea is to align the metric in the graph space (namely, the GED) to the prediction space. While this idea has been largely used in machine learning (e.g. with the so-called kernel-target alignment [15]), this is the first time that such a line of attack is investigated to estimate the optimal edit cost. With this distance-preserving principle, we provide a simple linear optimization procedure to optimize a set of constant edit costs. The edit costs resulting from the optimization procedure can then be analyzed to understand how the graph space is structured. The relevance of the proposed method is demonstrated on two regression tasks, showing that the optimized costs lead to a lower prediction error.

[5] Note the GED will be null when comparing two isomorphic graphs.

The remainder of the paper is organized as follows. Section 2 presents related works that aim to compute costs associated with a particular task. Section 3 presents the problem formulation and describes the proposed optimization method. Then, Section 4 presents results from conducted experiments. Finally, we conclude and open perspectives on this work.

2 Related Works

As stated in the introduction, the choice of edit costs has a major impact on the computation of graph edit distance, and thus on the performance associated with the prediction task.

The first approach to design these costs is to set them manually, based on the knowledge on a given dataset/task (when such knowledge is available). This strategy leads, for instance, to the classical edit cost functions associated with the IAM dataset [24]. However, it is interesting to challenge these predefined settings and experiment how they can improve the prediction performance.

In order to fit a particular targeted property to predict, tuning the edit costs and thus the GED can be seen as a subproblem of metric learning. Metric learning consists in learning a dissimilarity (or similarity) measure given a training set composed of data instances and associated targeted properties. For the classical metric learning where each data instance is encoded by a real-valued vector, the problem consists in learning a dissimilarity measure, which decreases (resp. increases) where the vectors have similar (resp. different) targeted properties. Many metric learning works focus on Euclidean data, while only a few addresses this problem on structured data [5]. A complete review for general structured data representation is given in [22]. In the following, we will focus on existing studies to learn edit costs for graph edit distance.

A trivial approach to tune the edit costs is to use a grid search strategy among a predefined range. However, the complexity required to compute graph edit distance and the number of different edit costs forbid such an approach.

String edit distance constitutes a particular case of graph edit distance, associated to a lower complexity, where graphs are restricted to be only linear and sequential. In [25], the authors propose to learn edit costs using a stochastic approach. This method shows a performance improvement, hence demonstrating the interest to tune edit costs; it is however restricted to strings.

Another strategy is based on a probabilistic approach [19–21]. By providing a probabilistic formulation for the common edition of two graphs, an Expectation-Maximization algorithm is used to derive weights applied to each edit operation. The tuning is then evaluated in an unsupervised manner. In [20], the strategy consists in modifying the label space associated with nodes and edges such that edit operations occurring more often will be associated to lower edit costs. Conversely, higher values will be associated with edit operations occurring less often. The learning process was validated on two datasets. However, this approach is computationally too expensive when dealing with general graphs [4].

In [4], the authors propose an interesting way to evaluate whether a distance is a "good" one. This criterion is based on the following concept:

> *a similarity function is* $(\epsilon, \gamma, \tau) - good$ *if a* $1 - \epsilon$ *proportion of examples are on average* 2γ *more similar to reasonable examples of the same class than to reasonable examples of the opposite class, where a* τ *proportion of examples must be reasonable.*

This principle is then derived to define an objective function to optimize. The matrix encoding the edit costs minimizing this objective function is then used to compute edit distances. However, this approach has only been adapted to strings and trees, but not to general graphs.

Another set of methods that address the problem of learning edit costs for GED is proposed in [13, 14]. These methods propose to optimize edit costs to maximize a ground truth mapping between nodes of graphs. This framework requires thus a ground truth mapping, which is not available on many datasets like chemoinformatics.

3 Proposed Method

3.1 Problem formulation

In this section, we propose an optimization procedure to learn edit costs in the context of regression tasks. Consider a dataset $\mathcal{G}$ of N graphs such that each graph $G_k = (V_k, E_k)$, for $k = 1, 2, \dots N$, where V_k represents the set of nodes of G_k labeled by a function $f_v : \mathcal{V} \to \mathcal{L}_v$, and E_k encodes the set of edges of G_k, namely $e_{ij} = (v_i, v_j) \in E_k$ iff an edge connects nodes v_i and v_j in G_k.

The graph edit distance between two graphs is defined as the minimal cost associated to an optimal edit path. Given two graphs G_1 and G_2, an edit path between them is defined as a sequence of edit operations transforming G_1 into G_2. An edit operation e can correspond to a node substitution $e = (v_i \to v_j)$, deletion $e = (v_i \to \varepsilon)$ or insertion $e = (\varepsilon \to v_j)$. Similarly, for edges, we have $(e_{ij} \to e_{ab}), (e_{ij} \to \varepsilon)$, and $(\varepsilon \to e_{ab})$. Each edit operation is associated with a cost characterizing the distortion induced by this edit operation on the graph. These costs can be encoded by a cost function c that associates a positive real value to each edit operation, depending on the elements being transformed.

In this paper, we will restrict ourselves to only constant cost functions. Therefore, we can associate each edit operation to a constant value. Let c_{ns}, c_{ni}, c_{nd}, c_{es}, c_{ei}, $c_{ed} \in \mathbb{R}_+$ be the cost values associated with respectively node substitution, insertion, deletion and edge substitution, insertion, deletion.

As shown in [7], any edit path between two graphs G_1 and G_2 can be encoded as two mapping functions. First, $\varphi : V_1 \to V_2 \cup \varepsilon$ encodes the mapping of G_1's nodes to nodes of G_2. If a node v_i is deleted, we have $\varphi(v_i) = \varepsilon$. Similarly, we denote as φ^{-1} the mapping of V_2 to $V_1 \cup \varepsilon$. For the same edit path, we have thus $\varphi(v_i) = v_j \Rightarrow \varphi^{-1}(v_j) = v_i$. Given a mapping and considering constant cost functions, the cost associated to node operations of an edit path represented by

φ and φ^{-1} is given by:

$$C_v(\varphi, \varphi^{-1}, G_1, G_2) = \sum_{\substack{v_i \in V_1 \\ \varphi(v_i) \neq \varepsilon}} c_{ns} + \sum_{\substack{v_i \in V_1 \\ \varphi(v_i) = \varepsilon}} c_{ni} + \sum_{\substack{v_i \in V_2 \\ \varphi^{-1}(v_i) = \varepsilon}} c_{nd}. \tag{2}$$

The cost associated with edge operations is defined as:

$$C_e(\varphi, \varphi^{-1}, G_1, G_2) = \sum_{\substack{e=(v_i,v_j) \in E_1 \mid \\ \varphi(v_i) \neq \varepsilon \wedge \\ \varphi(v_j) \neq \varepsilon \wedge \\ (\varphi(v_i), \varphi(v_j)) \in E_2}} c_{es} + \sum_{\substack{e=(v_i,v_j) \in E_1 \mid \\ \varphi(v_i) = \varepsilon \vee \\ \varphi(v_j) = \varepsilon \vee \\ (\varphi(v_i), \varphi(v_j)) \notin E_2}} c_{ei} + \sum_{\substack{e=(v_i,v_j) \in E_2 \mid \\ \varphi^{-1}(v_i) = \varepsilon \vee \\ \varphi^{-1}(v_j) = \varepsilon \vee \\ (\varphi^{-1}(v_i), \varphi^{-1}(v_j)) \notin E_1}} c_{ed}. \tag{3}$$

The final cost is given by:

$$C(\varphi, \varphi^{-1}, G_1, G_2) = C_v(\varphi, \varphi^{-1}, G_1, G_2) + C_e(\varphi, \varphi^{-1}, G_1, G_2). \tag{4}$$

Let $\#ns$ be the number of node substitutions, i.e., the cardinality of the subset of V_1 being mapped onto V_2. This number is given by the number of terms of the first sum in Eq. 2, i.e., $\#ns = |\{v_i \in V_1 \mid \varphi(v_i) \neq \varepsilon\}|$. Similarly:

- The number of node deletions is $\#nd = |\{v_i \in V_1 \mid \varphi(v_i) = \varepsilon\}|$;
- The number of node insertions is $\#ni = |\{v_i \in V_2 \mid \varphi^{-1}(v_i) = \varepsilon\}|$;
- The number of edge substitutions is $\#es = |\{e = (v_i, v_j) \in E_1 \mid \varphi(v_i) \neq \varepsilon \wedge \varphi(v_j) \neq \varepsilon \wedge (\varphi(v_i), \varphi(v_j)) \in E_2\}|$;
- The number of node deletions is $\#ei = |\{e = (v_i, v_j) \in E_1 \mid \varphi(v_i) = \varepsilon \vee \varphi(v_j) = \varepsilon \vee (\varphi(v_i), \varphi(v_j)) \notin E_2\}|$;
- The number of node insertions is $\#ed = |\{e = (v_i, v_j) \in E_2 \mid \varphi^{-1}(v_i) = \varepsilon \vee \varphi^{-1}(v_j) = \varepsilon \vee (\varphi^{-1}(v_i), \varphi^{-1}(v_j)) \notin E_1\}|$.

Then, let $\mathbf{x} \in \mathbb{N}^6$ encode the number of each edit operation as $\mathbf{x} = [\#ns, \#nd, \#ni, \#es, \#ed, \#ei]^\top$. Note that these values depend on both graphs being compared and a given mapping between nodes. Similarly, we define a vector representation of the costs associated with each edit operation by $\mathbf{c} = [c_{ns}, c_{nd}, c_{ni}, c_{es}, c_{ed}, c_{ei}]^\top \in \mathbb{R}^6_+$. Given these representations, the cost associated with an edit path, as defined by Eq. 4, can be rewritten as:

$$C(\varphi, \varphi^{-1}, G_1, G_2, \mathbf{c}) = \mathbf{x}^\top \mathbf{c}. \tag{5}$$

Therefore, the graph edit distance between two graphs is defined as:

$$\texttt{ged}(G_1, G_2, \mathbf{c}) = \operatorname*{argmin}_{\varphi, \varphi^{-1}} C(\varphi, \varphi^{-1}, G_1, G_2, \mathbf{c}). \tag{6}$$

3.2 Learning the edit costs

Consider that each graph $G_k \in \mathcal{G}$ is associated with a particular targeted property $y_k \in \mathcal{Y}$, namely the target in regression tasks (e.g. $\mathcal{Y} \subseteq \mathbb{R}$ for real-valued output regression). Furthermore, a distance $\mathrm{d}_\mathcal{Y} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is defined on this targeted property, such as the Euclidean distance when dealing with a vector space $\mathcal{Y}$, namely $\mathrm{d}_\mathcal{Y}(y_i, y_j) = \|y_i - y_j\|_2$.

The main idea behind the proposed method is that the best metric in the graph space is the best aligned one to the target distances (i.e., $\mathrm{d}_{\mathcal{Y}}$). With this distance-preserving principle, we seek to learn the edit cost vector $\mathbf{c}$ by fitting the distances between graphs to the distances between their targeted properties. Ideally, we seek to preserve the GED between any two graphs G_i and G_j and the distance between their targeted properties. Considering the set of N available graphs $G_1, \ldots, G_N$ and their corresponding targets $y_1, \ldots, y_N$, we seek to have

$$\mathtt{ged}(G_i, G_j, \mathbf{c}) \approx \mathrm{d}_{\mathcal{Y}}(y_i, y_j) \qquad \text{for all } i, j = 1, 2, \ldots N. \tag{7}$$

Let $\omega : \mathcal{G} \times \mathcal{G} \times \mathbb{R}_+^6 \to \mathbb{N}^6$ be the function that computes an optimal edit path between G_i and G_j according to the cost vector $\mathbf{c}$ and returns the vector $\mathbf{x}^\star \in \mathbb{R}_+^6$ of numbers of edit operations associated to this optimal edit path, namely $\mathbf{x}^\star = \omega(G_i, G_j, \mathbf{c})$. This function can be any method computing an exact or sub-optimal graph edit distance [1,6].

For any pair of graphs (G_i, G_j), let $\mathbf{x}_{i,j}$ be a vector encoding the number of each edit operation. Let $\mathbf{X} \in \mathbb{N}^{N^2 \times 6}$ be the matrix of the numbers of edit operations for each pair of graphs, namely its $(iN + j)$-th row is $\mathbf{x}_{i,j}^T$. Then, $\mathbf{Xc}$ is the $N^2 \times 1$ vector composed of edit distances computed according to $\mathbf{c}$ and $\mathbf{X}$ between all pairs of graphs. Let $\mathbf{d} \in \mathbb{R}_+^{N^2}$ be a vector of the differences on targeted properties according to $\mathrm{d}_{\mathcal{Y}}$, with $\mathbf{d}(iN + j) = \mathrm{d}_{\mathcal{Y}}(G_i, G_j)$. Therefore, the optimization problem can be rewritten as:

$$\underset{\mathbf{c}}{\operatorname{argmin}}\ \mathcal{L}(\mathbf{Xc}, \mathbf{d}) \qquad \text{subject to } \mathbf{c} > 0, \tag{8}$$

where $\mathcal{L}$ denotes a loss function. Besides the constraint on $\mathbf{c}$ to avoid negative costs, one can also add a constraint to satisfy the triangular inequality, or one to ensure that a deletion cost is equal to an insertion cost [23].

In the case of regression problem, $\mathcal{L}$ can be defined as the sum squares of differences between computed graph edit distances and dissimilarities of the targeted property. Therefore, the final optimization problem is:

$$\underset{\mathbf{c}}{\operatorname{argmin}}\ ||\mathbf{Xc} - \mathbf{d}||_2^2 \qquad \text{subject to } \mathbf{c} > 0. \tag{9}$$

Estimating $\mathbf{c}$ by solving this constrained optimization problem allows to linearly fit graph edit distances to a particular targeted property according to the edit paths initially given by ω. However, changing the edit costs may influence the optimal edit path, and thus its description in terms of the numbers of edit operations. There is thus an interdependence between the function ω computing an optimal edit path according to $\mathbf{c}$, and the objective function optimizing $\mathbf{c}$ according to edit paths encoded within $\mathbf{X}$. To solve this interdependence, we propose an alternated optimization strategy, summarized in Algorithm 1 where $\Omega(G, \mathbf{c})$ denotes the computation of $\omega(G_i, G_j, \mathbf{c}), \forall i, j \in 1 \ldots N$. The two main steps of the algorithm are described next:

– Estimate $\mathbf{c}$ for fixed $\mathbf{X}$ (line 4): This optimization problem is a constrained linear problem that can be resolved using off-the-shelf solvers, such as

Algorithm 1 Main algorithm to optimize costs

1: $\mathbf{c} \leftarrow \text{random}(6)$
2: $\mathbf{X} \leftarrow \Omega(G, \mathbf{c})$
3: **while** not converged **do**
4: $\quad \mathbf{c} \leftarrow \text{argmin}_{\mathbf{c}} ||\mathbf{Xc} - \mathbf{d}||_2^2, \text{subject to } \mathbf{c} > 0$
5: $\quad \mathbf{X} \leftarrow \Omega(G, \mathbf{c})$
6: **end while**

cvxpy [16] and scipy [28]. This optimization problem can also be viewed as a non-negative least squares problem [18]. For a given set of edit operations between each pair of graphs, this step linearly optimizes the constant costs to be applied such that the difference between graph edit distances and distances between targets is minimized.

– Estimate $\mathbf{X}$ for fixed $\mathbf{c}$ (line 5): The modification performed on costs in the previous step may have an influence on the associated edit path. To address this point, the optimization of costs is followed by a re-computation of the optimal edit paths according to the newly computed $\mathbf{c}$ vector encoding the edit costs. This step can be achieved by any method computing graph edit distance. For the sake of computational time, one can choose an approximated version of GED [6, 7].

This alternated optimization is repeated to compute both edit costs and edit operations. Since we do not have theoretical proof of the convergence of this optimization scheme, we limit the number of iterations to 5 in our implementation.

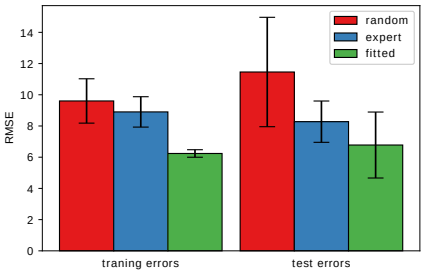
4 Experiments

We conducted experiments[6] on two well-known datasets in chemoinformatics, both composed of molecules and their boiling points. The first dataset is composed of 150 alkanes [11]. An alkane is an acyclic molecule solely composed of carbons and hydrogens. A common representation of such data consists in implicitly encoding hydrogen atoms using the valency of carbon atoms. Such an encoding scheme allows to represent alkanes as acyclic unlabeled graphs. The second dataset is composed of 185 acyclic molecules [12]. In contrast with the previous dataset, these molecules contain several hetero atoms and are thus represented as acyclic labeled graphs.

To evaluate the predictive power of different settings of edit costs, we used a k-nearest-neighbors regression [2] model, where k is the number of the neighbors considered to predict a property. The performances are estimated on ten different random splits. For each split, a test set representing 10% of the graphs in the dataset is randomly selected and used to measure the performance of the prediction. The remaining 90% are used to optimize the edit costs and the value of k, where k is optimized through a 5-fold cross-validation (CV) procedure over the

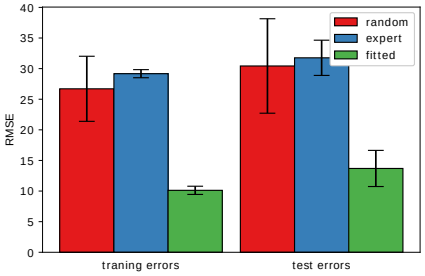
[6] Code available at https://gitlab.insa-rouen.fr/bgauzere/fit-distances.

Method	Train errors	Test errors
random	9.60 ± 1.42	11.46 ± 3.50
expert	8.90 ± 0.97	8.28 ± 1.32
fitted	6.24 ± 0.24	6.78 ± 2.11



(a) Results on Alkane Dataset

Method	Train errors	Test errors
random	26.70 ± 5.32	30.43 ± 7.71
expert	29.17 ± 0.66	31.77 ± 2.88
fitted	10.12 ± 0.67	13.69 ± 2.95



(b) Results on Acyclic Dataset

Fig. 1: Results on each dataset in terms of RMSE for the 10 splits

Table 1: Average and standard deviation of fitted edit costs values

Dataset	c_{ns}	c_{nd}	c_{ni}	c_{es}	c_{ed}	c_{ei}
Acyclic	10.74 ± 0.2	14.26 ± 0.7	14.8 ± 0.6	0.32 ± 0.01	0.23 ± 0.2	0.4 ± 0.2
Alkane	-	26.22 ± 1.0	26.85 ± 0.8	-	0.16 ± 0.1	0.11 ± 0.1

candidate values $\{3, 5, 7, 9, 11\}$. The number of iterations for the optimization of the edit costs is fixed to 5.

The proposed optimization procedure is compared to two other edit costs settings: a random set of edit costs and a predefined cost setting as given in [1]; the latter is the so-called expert costs. Tables in Fig. 1 show the average root mean squared errors (RMSE) obtained for each cost settings over the 10 splits, estimated on the training set and on the test set. The $\pm$ sign gives the 95% confidence interval computed over the 10 repetitions. Figures show a different representation of the same results with error bars modeling the 95% confidence interval. As expected, a clear and significant gain in accuracy is obtained when using fitted costs on the two datasets. These promising results confirm the hypothesis that ad-hoc edit costs may help the graph edit distance catch better targeted properties that are associated to a graph, and thus improve the prediction accuracy while still operating in the graph space.

The fitted values of edit costs are summarized in Table 1. From these results, we can observe that insertion and deletion costs are almost similar, hence showing the symmetry of these operations. Also, one can observe that deletion and insertion costs are more important than substitution costs, which shows that the number of atoms is more important than the atom itself. This is coherent with the chemistry theory [12]. Finally, we can note that costs associated with nodes are higher to the ones associated with edges.

5 Conclusion and future work

In this paper, we introduced a new principle to define optimal graph edit costs of a GED for a given regression task. Based on this principle, we defined the optimization problem of fitting the edit costs to a particular metric, measured for instance on a targeted property to predict. An alternated optimization strategy was proposed to solve this optimization problem. The conducted experiments on two well-known datasets showed that the optimization process leads to a GED with a better predictive power compared to other methods. All these observations confirm that the proposed method helps to fit edit costs and outperforms other methods. There are still several challenges to address in future work. First, a clear and complete comparison to other methods cited in the introduction and related works will be established. Second, we seek to examine other criteria than the distance-preserving criterion, such as the conformal map for instance [**?**]. Third, from a theoretical point of view, we are interested in establishing convergence proof on our alternated optimization strategy, and to extend these proofs to approximate computations of graph edit distances. Fourth, this scheme will be extended to classification problem and non-constant costs to be applicable in most application domains. Considering non-constant costs will need to optimize parametric functions rather than scalar values, hence complexifying the procedure.

References

1. Abu-Aisheh, Z., Gaüzère, B., Bougleux, S., Ramel, J.Y., Brun, L., Raveaux, R., Héroux, P., Adam, S.: Graph edit distance contest: Results and future challenges. Pattern Recognition Letters **100**, 96–103 (2017)
2. Altman, N.S.: An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician **46**(3), 175–185 (1992)
3. Balcilar, M., Renton, G., Héroux, P., Gaüzère, B., Adam, S., Honeine, P.: When spectral domain meets spatial domain in graph neural networks. In: Proceedings of ICML 2020 - Workshop on Graph Representation Learning and Beyond (GRL+ 2020). Vienna, Austria (12 - 18 Jul 2020)
4. Bellet, A., Habrard, A., Sebban, M.: Good edit similarity learning by loss minimization. Machine Learning **89**(1-2), 5–35 (2012)
5. Bellet, A., Habrard, A., Sebban, M.: A survey on metric learning for feature vectors and structured data. arXiv preprint arXiv:1306.6709 (2013)
6. Blumenthal, D.B., Boria, N., Gamper, J., Bougleux, S., Brun, L.: Comparing heuristics for graph edit distance computation. The VLDB Journal **29**(1), 419–458 (2020)
7. Bougleux, S., Brun, L., Carletti, V., Foggia, P., Gaüzère, B., Vento, M.: Graph edit distance as a quadratic assignment problem. Pattern Recognition Letters (2015)
8. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine **34**(4), 18–42 (2017)
9. Bunke, H.: Error correcting graph matching: on the influence of the underlying cost function. IEEE Transactions on Pattern Analysis and Machine Intelligence **21**(9), 917–922 (1999). https://doi.org/10.1109/34.790431

10. Bunke, H., Allermann, G.: Inexact graph matching for structural pattern recognition. Pattern Recognition Letters **1**(4), 245–253 (1983)
11. Cherqaoui, D., Villemin, D.: Use of a neural network to determine the boiling point of alkanes. J. Chem. Soc. Faraday Trans. **90**, 97–102 (1994)
12. Cherqaoui, D., Villemin, D., Mesbah, A., Cense, J.M., Kvasnicka, V.: Use of a Neural Network to Determine the Normal Boiling Points of Acyclic Ethers, Peroxides, Acetals and their Sulfur Analogues. J. Chem. Soc. Faraday Trans. **90**, 2015–2019 (1994)
13. Cortés, X., Conte, D., Cardot, H.: Learning edit cost estimation models for graph edit distance. Pattern Recognition Letters **125**, 256–263 (2019). https://doi.org/10.1016/j.patrec.2019.05.001
14. 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)
15. Cristianini, N., Shawe-Taylor, J., Elisseeff, A., Kandola, J.: On kernel-target alignment. In: Advances in neural information processing systems. pp. 367–373 (2002)
16. Diamond, S., Boyd, S.: Cvxpy: A python-embedded modeling language for convex optimization. The Journal of Machine Learning Research **17**(1), 2909–2913 (2016)
17. Gibert, J., Valveny, E., Bunke, H.: Graph embedding in vector spaces by node attribute statistics. Pattern Recognition **45**(9), 3072–3083 (2012)
18. Lawson, C.L., Hanson, R.J.: Solving least squares problems. SIAM (1995)
19. Neuhaus, M., Bunke, H.: A probabilistic approach to learning costs for graph edit distance. Proceedings ICPR **3**(C), 389–393 (2004)
20. Neuhaus, M., Bunke, H.: Self-organizing maps for learning the edit costs in graph matching. IEEE transactions on systems, man, and cybernetics **35**(3), 503–14 (jun 2005)
21. Neuhaus, M., Bunke, H.: Automatic learning of cost functions for graph edit distance. Information Sciences **177**(1), 239–247 (2007). https://doi.org/10.1016/j.ins.2006.02.013
22. Ontañón, S.: An overview of distance and similarity functions for structured data. Artificial Intelligence Review (2020). https://doi.org/10.1007/s10462-020-09821-w
23. Riesen, K.: Structural pattern recognition with graph edit distance. In: Advances in computer vision and pattern recognition. Springer (2015)
24. Riesen, K., Bunke, H.: Iam graph database repository for graph based pattern recognition and machine learning. In: Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR). pp. 287–297. Springer (2008)
25. Ristad, E.S., N.yianilos, P.: Learning string-edit distance. IEEE Transactions on Pattern Analysis and Machine Intelligence **20**(5), 522–532 (1998). https://doi.org/10.1109/34.682181
26. Sanfeliu, A., Fu, K.S.: A distance measure between attributed relational graphs for pattern recognition. Systems, Man. and Cybernetics **13**(3), 353–362 (1983)
27. Shervashidze, N., Schweitzer, P., Van Leeuwen, E.J., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-lehman graph kernels. Journal of Machine Learning Research **12**(9) (2011)
28. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al.: Scipy 1.0: fundamental algorithms for scientific computing in python. Nature methods **17**(3), 261–272 (2020)