Bridging Distinct Spaces in Graph-based Machine Learning *

Linlin Jia[1][0000−0002−3834−1498], Xiao Ning[2][0000−0002−5785−2226], Benoit Gaüzère[3][0000−0001−9980−2641], Paul Honeine[4][0000−0002−3042−183X], and Kaspar Riesen[1][0000−0002−9145−3157]

[1] Institute of Computer Science, University of Bern, Bern 3012, Switzerland
linlin.jia@unibe.ch
[2] State Key Laboratory of Bioelectronics, School of Biological Science and Medical Engineering, Southeast University, 2 Sipailou, Nanjing, 210096, PR China
[3] The LITIS Lab, INSA Rouen Normandie, Rouen, France
[4] The LITIS Lab, Université de Rouen Normandie, Rouen, France

Abstract. Graph-based machine learning, encompassing Graph Edit Distances (GEDs), Graph Kernels, and Graph Neural Networks (GNNs), offers extensive capabilities and exciting potential. While each model possesses unique strengths for graph challenges, interrelations between their underlying spaces remain under-explored. In this paper, we introduce a novel framework for bridging these distinct spaces via GED cost learning. A supervised metric learning approach serves as an instance of this framework, enabling space alignment through pairwise distances and the optimization of edit costs. Experiments reveal the framework's potential for enhancing varied tasks, including regression, classification, and graph generation, heralding new possibilities in these fields.

Keywords: Graph-based Machine Learning · Graph Spaces Alignment · Graph Edit Distances (GEDs) · Metric Learning

1 Introduction

Graph structures have emerged as critical tools for tackling complex challenges across a range of disciplines, including chemoinformatics [1], bioinformatics [2], social network analysis [3], computer vision [4], and others. In these fields, intricate relationships can be elegantly represented through graphs, allowing a better understanding and analysis of complex systems. The benefits of graph structures are amplified when combined with machine learning techniques, paving the way for a new era of data exploration and knowledge discovery. In particular, graph-based machine learning techniques such as graph embedding strategies [5], graph kernels [6, 7], and Graph Neural Networks (GNNs) [8] have shown promise in

* Supported by Swiss National Science Foundation (SNSF) Project No. 200021_188496.

their ability to harness the complexity and interconnectedness of graph structures. Benefiting from distinct designs and the ability to capture specific information, each of these techniques offer their unique strengths and perspectives when addressing graph-related challenges. However, despite the impressive performance of these models, there remains a critical gap in their application. Specifically, these models typically project the graph structure into different spaces, leading to the potential loss of valuable graph properties and operations, which, in turn, results in a disconnect between the graph space and the learning technique employed. On the other hand, as stated in [9], the use of metric distances, such as graph edit distances (GEDs), directly in a graph space is often practically insufficient, as they are seldom computationally tractable. This limit drives researchers to embed graphs into other spaces, which reduces the interpretability of the underlying operations. Moreover, the connections among these spaces themselves remain unrevealed, keeping the underlying theories and potential applications behind the mist.

To bridge this gap and to ensure the preservation of the essential properties and operations of the graph space, the concept of GED can be leveraged [10]. The key to GED's utility lies in its ability to operate directly within the graph space, maintaining the graph's structural integrity while enhancing the interpretability of operations. Additionally, GED allows for the direct construction of new graphs according to specific edit paths [11], further underscoring its applicability and usefulness. The successful application of GED necessitates the appropriate optimization of edit costs, which have a major impact on the computation of GED and its performance. As the values of edit costs may differ depending on the data encoded by the graph and tasks, methods for performing this crucial step are recently trending to a prosperity. Recognizing the need for an improved approach to bridge graph spaces, we take advantage of these methods and turn to a more comprehensive exploration of edit costs.

In this paper, we propose a novel framework designed to overcome the existing challenges of bridging graph spaces. This framework leverages cutting-edge optimization methods to effectively manage edit costs, thereby enhancing the applicability and efficiency of GED. The innovative nature of this framework goes beyond the integration of improved optimization methods. Its true value lies in its ability to serve as a unifying platform that bridges disparate graph spaces, bringing together the strengths of various graph learning models while addressing their shortcomings. This positions our framework as a promising solution that could advance graph-based machine learning, opening up new avenues for exploration and application in numerous fields.

The remainder of this paper is organized as follows: Section 2 discusses related work, including current methods for bridging graph spaces and existing approaches to edit cost learning. Section 3 presents our proposed framework, along with a detailed discussion of GEDs and an application of Supervised Metric Learning. Section 4 explores potential applications of the framework, followed by an examination of our experimental results in Section 5. Finally, we conclude with a summary of our findings and future work in Section 6.

2 Related Work

The study between distinct graph-based spaces keeps attracting the attention of researchers. In this paper we emphasize on spaces induced by GEDs, graph kernels, and GNNs. In [12], the random walk graph kernel is extended by the GEDs, where the latter is used to evaluate global matching information to guide the local similarity evaluation of the kernel. In [13], an approach is proposed to turn an existing dissimilarity measure (e.g. GED) into a similarity measure, thus construct graph kernels directly from GED. In [14], GED is applied to encode the cross information between treelets. In [15], the graph and graph kernel spaces are aligned through GED to conduct a pre-image problem. Meanwhile, GEDs and GNNs are often bounded by the optimization of edit costs or learning and redesign of edit distances [16].

Connections between graph kernels and GNNs have been established in recent work as well. Graph kernels have be applied in GNN settings as convolutional filters, first-layer representation selection, and pre-train strategies [17]. In converse, graph kernels designs inspired by GNNs are proposed as well [18]. Moreover, The equivalent expressiveness of 1-dimensional Weisfeiler-Leman graph isomorphism heuristic (1-WL) and GNNs are theoretically exhibited in [19].

Despite the thriving of the field, each proposed model operates only in specific spaces or with specific machine learning methods. Through the GED cost learning, however, a universal approach can be systematically established, operating on multiple graph-based spaces. Various edit cost choosing approaches have been proposed in the literature. Manual settings are the most straightforward approach, based on the knowledge on a given dataset/task [20]. To challenge these settings and adapt the method to situations without such prior knowledge, one can tune the edit costs by grid search. However, the time complexity of the GED computation and the number of edit costs restrict its usage. Another commonly-used strategy is to fit edit costs with a particular targeted property, generally a prediction target. This problem can be seen as a sub-problem of generalized metric learning, which consists in learning a dissimilarity (or similarity) measure given a training set composed of data instances and associated targeted properties. In contrast to a valid metric, a generalized metric or a pseudometric (e.g. GED), does not strictly adhere to at least one of the following conditions: non-negativity, identity of indiscernibles, symmetry, positive definiteness, and triangle inequality [21]. For the sake of brevity, we use the term "metric" to denote "generalized metric" in this paper.

One set of strategies is based on a probabilistic approach [22]. 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. However, this approach is computationally too expensive when dealing with general graphs [23]. Another class of strategies optimizes edit costs by maximizing the similarity between the computed mapping and a ground-truth mapping between the vertices of the graphs [24]. This framework thus requires a ground truth mapping, which is not available on most datasets such as the ones in the chemoinformatics domain. To

address this shortcoming, multiple supervised strategies are proposed. In [25], the edit costs are optimized by minimizing the difference between the GED and the distances between the prediction targets. While in [26], genetic algorithms are applied to optimize the costs for classification tasks.

3 Proposed Framework

In this section, we propose first a general framework bridging graph-based spaces through Graph Edit Cost Learning (GECL), and then an instance of this framework taking advantage of supervised metric learning. With these details, we validate GECL's innovations in cost learning and graph embedding spaces bridging, highlighting its versatile applications.

3.1 Graph Edit Distances and Edit Costs

To introduce the GECL framework, we first introduce the necessary preliminaries of the graph edit distances (GEDs). The GED between two graphs is defined as the minimal cost associated with an optimal edit path. Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, 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 an insertion, removal, or substitution of vertices or edges. 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 $c(e)$ that associates a non-negative real value to each edit operation, depending on the elements being transformed. The total cost of an edit path π is the sum of the costs of all edit operations in the path:

$$c(\pi) = \sum_{e \in \pi} c(e). \tag{1}$$

Then, the GED between G_1 and G_2 is defined as the minimum cost associated with any possible edit path: the cost of the cheapest edit path:

$$\texttt{ged}(G_1, G_2) = \min_{\pi \in \mathit{\Pi}(G_1, G_2)} c(\pi), \tag{2}$$

where $\mathit{\Pi}(G_1, G_2)$ denotes the set of all possible edit paths from G_1 to G_2. The computation of the exact GED is a NP-hard problem [21]. In practice, a sub-optimal approximation algorithm is often used and a cost matrix C is often defined for the edit operations. Then the task of GED computation can be formulated as an optimization problem that minimizes the total edit cost:

$$\min_{\pi \in \mathit{\Pi}(G_1, G_2)} \sum_{e \in \pi} C_e. \tag{3}$$

In the case of G_1 and G_2, the cost matrix C is expressed as:

$$C = \begin{bmatrix} C_s & C_{r_{G_1}} \\ C_{i_{G_2}} & 0 \end{bmatrix} \tag{4}$$

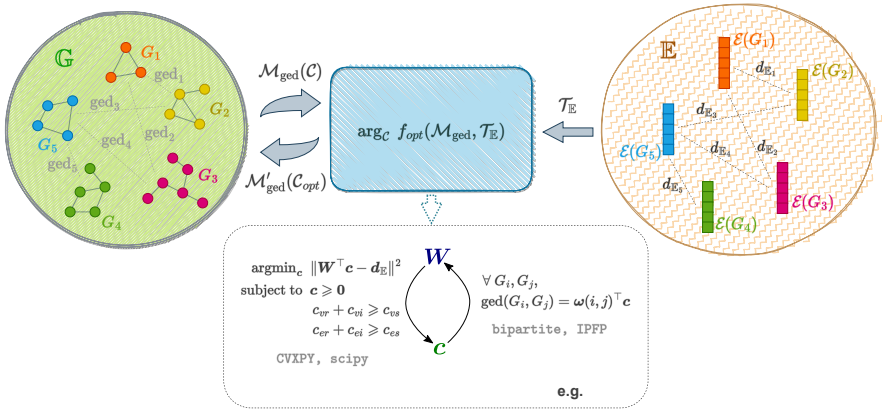

Fig. 1. The GECL Framework.

where:

- C_s is the $n_1 \times n_2$ substitution cost matrix with elements $C_{s_{ij}} = c_s(v_{i_{G_1}}, v_{j_{G_2}})$, i.e., the cost of substituting node $v_{i_{G_1}}$ in G_1 with node $v_{j_{G_2}}$ in G_2.
- $C_{r_{G_1}}$ is the $n_1 \times n_1$ matrix with diagonal elements $C_{r_{G_{1_{ii}}}} = c_r(v_{i_{G_1}})$, i.e., the cost of removing node $v_{i_{G_1}}$ in G_1; the off-diagonal elements are set to ∞.
- $C_{i_{G_2}}$ is the $n_2 \times n_2$ matrix with diagonal elements $C_{i_{G_{2_{jj}}}} = c_i(v_{j_{G_2}})$, i.e., the cost of inserting node $v_{j_{G_2}}$ in G_2, and off-diagonal elements are set to ∞.

Thus, the matrix C represents all possible edit operations and their associated costs, which are utilized in calculating the GED.

3.2 The GECL Framework

The cost matrix C plays an important role in GEDs, actively affecting its computation and performance, thus reshaping the structure of the underlying graph space. Taking advantage of this merit, we propose the GECL framework, and formalize it as follows (See Fig. 1):

We define a space of graphs $\mathbb{G}$ as all possible graphs whose vertex and edge labels are defined by a label alphabet of a domain. Given a dataset $\mathcal{G} \subset \mathbb{G}$ of N graphs such that each graph $G_k = (V_k, E_k)$, for $k = 1, 2, \ldots, N$, we define the set of pairwise GED $\mathcal{M}_{\texttt{ged}} = \{\texttt{ged}(G_i, G_j, C_{ij}) \mid i, j \in 1, 2, \ldots, N\}$ as a metric associated with space $\mathbb{G}$. Meanwhile, we define a target measurement $\mathcal{T}_{\mathbb{E}}$ associated with an embedding space $\mathbb{E}$. The GECL framework aims at aligning the two spaces by optimizing a target function f_{opt}. The problem can then be formalized as

$$\arg_{\mathcal{C}} \ f_{opt}(\mathcal{M}_{\texttt{ged}}, \mathcal{T}_{\mathbb{E}}), \tag{5}$$

where $\mathcal{C} = \{C_{ij} \mid i, j \in 1, 2, \ldots, N\}$ is the set of cost matrices between all pairs of graphs. After optimizing $\mathcal{C}$, the GEDs are recomputed accordingly.

The measurement $\mathcal{T}_{\mathbb{E}}$ and the target function f_{opt} endow the possibility of varied forms. Given $\mathcal{T}_{\mathbb{E}}$ the distances between prediction targets or ground truths,

many edit costs optimization strategies fall into this framework, including the ones introduced in Section 2 (e.g., [26]). In the next section, we take advantage of a supervised metric learning strategy in [25] and define an instance of the framework.

3.3 GECL Based on Supervised Metric Learning

In many applications, the cost functions are restricted to constants, namely, each type of edit operation is associated to a constant value. Let c_{vs}, c_{vi}, c_{vr}, c_{es}, c_{ei}, $c_{er} \in \mathbb{R}$ be the cost values associated with vertex and edge substitutions, insertions, removals, respectively. The cost associated with edit operations of an edit path represented by π is given by:

$$C(\pi, G_1, G_2) = C_v(\pi, G_1, G_2) + C_e(\pi, G_1, G_2), \tag{6}$$

where $C_v(\pi, G_1, G_2)$ is the cost associated with vertex operations, namely

$$C_v(\pi, G_1, G_2) = \sum_{\substack{v \in V_2 \\ \pi^{-1}(v)=\varepsilon}} c_{vi} + \sum_{\substack{v \in V_1 \\ \pi(v)=\varepsilon}} c_{vr} + \sum_{\substack{v \in V_1 \\ \pi(v)\neq\varepsilon}} c_{vs}, \tag{7}$$

and $C_e(\pi, G_1, G_2)$ is the one associated with edge operations, namely

$$C_e(\pi, G_1, G_2) = \sum_{\substack{e=(v_i,v_j)\in E_2 \mid \\ \pi^{-1}(v_i)=\varepsilon \vee \\ \pi^{-1}(v_j)=\varepsilon \vee \\ (\pi^{-1}(v_i),\pi^{-1}(v_j))\notin E_1}} c_{ei} + \sum_{\substack{e=(v_i,v_j)\in E_1 \mid \\ \pi(v_i)=\varepsilon \vee \\ \pi(v_j)=\varepsilon \vee \\ (\pi(v_i),\pi(v_j))\notin E_2}} c_{er} + \sum_{\substack{e=(v_i,v_j)\in E_1 \mid \\ \pi(v_i)\neq\varepsilon \wedge \\ \pi(v_j)\neq\varepsilon \wedge \\ (\pi(v_i),\pi(v_j))\in E_2}} c_{es}. \tag{8}$$

Let $n_{vs} = |\{v_i \in V_1 \mid \pi(v_i) \neq \varepsilon\}|$ be the number of vertex substitutions, that is, the cardinality of the subset of V_1 being mapped onto V_2. Similarly:

- The number of vertex removals is $n_{vr} = |\{v_i \in V_1 \mid \pi(v_i) = \varepsilon\}|$;
- The number of vertex insertions is $n_{vi} = |\{v_i \in V_2 \mid \pi^{-1}(v_i) = \varepsilon\}|$;
- The number of edge substitutions is $n_{es} = |\{e = (v_i, v_j) \in E_1 \mid \pi(v_i) \neq \varepsilon \wedge \pi(v_j) \neq \varepsilon \wedge (\pi(v_i), \pi(v_j)) \in E_2\}|$;
- The number of vertex removals is $n_{ei} = |\{e = (v_i, v_j) \in E_1 \mid \pi(v_i) = \varepsilon \vee \pi(v_j) = \varepsilon \vee (\pi(v_i), \pi(v_j)) \notin E_2\}|$;
- The number of vertex insertions is $n_{er} = |\{e = (v_i, v_j) \in E_2 \mid \pi^{-1}(v_i) = \varepsilon \vee \pi^{-1}(v_j) = \varepsilon \vee (\pi^{-1}(v_i), \pi^{-1}(v_j)) \notin E_1\}|$.

Then, define $\mathbf{x} = [n_{vi}, n_{vr}, n_{vs}, n_{ei}, n_{er}, n_{es}]^\top \in \mathbb{N}^6$ to represent the count of each edit operation. It is important to note that these values are dependent on both graphs under comparison as well as a specified vertex mapping. In a similar fashion, we create a vector representation $\mathbf{c} = [c_{vi}, c_{vr}, c_{vs}, c_{ei}, c_{er}, c_{es}]^\top \in \mathbb{R}^6$ to represent the costs associated with each edit operation. With these vector representations, we can express the cost associated with an edit path, as defined by (6), in a compact form:

$$C(\pi, G_1, G_2, \mathbf{c}) = \mathbf{x}^\top \mathbf{c}. \tag{9}$$

The GED between two graphs is consequently defined as:

$$\texttt{ged}(G_1, G_2, \mathbf{c}) = \min_{\pi} C(\pi, G_1, G_2, \mathbf{c}). \tag{10}$$

In this framework, we assume that each graph $G_k \in \mathcal{G}$ maps to a specific element, or "embedding", $f_{\mathbb{E}}(G_k)$ in an embedding space $\mathbb{E}$. Furthermore, a distance $\mathrm{d}_{\mathbb{E}} : \mathbb{E} \times \mathbb{E} \to \mathbb{R}$ is defined over these embeddings. The core principle of this framework is that the most effective metric in the GED space aligns the most accurately with the distances within the embedded space (i.e., $\mathrm{d}_{\mathbb{E}}$). Guided by this principle of distance preservation, we aim to determine the edit cost vector $\mathbf{c}$ by aligning the GEDs between graphs with the distances between their respective embeddings. It is then ideal to preserve the GED between any two graphs G_i and G_j and the distance between their embeddings (see Fig. 1). That is to say, given a set of N available graphs $G_1, \ldots, G_N$ and their corresponding embeddings $f_{\mathbb{E}_i}, \ldots, f_{\mathbb{E}_j}$, we seek to have

$$\texttt{ged}(G_i, G_j, \mathbf{c}) \approx \mathrm{d}_{\mathbb{E}}(f_{\mathbb{E}_1}, f_{\mathbb{E}_N}) \qquad \forall\ i, j = 1, 2, \ldots N. \tag{11}$$

Given any pairs of graphs $(G_i, G_j) \in \mathcal{G}$ and a cost vector $\mathbf{c}$, we can define $\mathbf{x}_{i,j} = \omega(G_i, G_j, \mathbf{c})$. Here, $\omega : \mathcal{G} \times \mathcal{G} \times \mathbb{R}^6_+ \to \mathbb{N}^6$ is the function that computes an optimal edit path between G_i and G_j according to $\mathbf{c}$, and the vector $\mathbf{x}_{i,j} \in \mathbb{R}^6_+$ denotes the numbers of edit operations associated with this optimal edit path. Function ω and vector $\mathbf{x}_{i,j}$ can be obtained by any method computing an exact or sub-optimal GED [27]. We further define a matrix $\mathbf{X} \in \mathbb{N}^{N^2 \times 6}$ to gather the numbers of edit operations for each pair of graphs, where $\mathbf{X}_{iN+j,:} = \mathbf{x}^T_{i,j}$. Namely, the $(iN + j)$-th row of $\mathbf{X}$ is $\mathbf{x}^T_{i,j}$. Consequentially, the product $\mathbf{Xc}$ is a $N^2 \times 1$ vector comprising of edit distances between all pairs of graphs computed according to $\mathbf{c}$ and $\mathbf{X}$. Let the vector $\mathbf{d} \in \mathbb{R}^{N^2}$ the differences on embeddings according to $\mathrm{d}_{\mathbb{E}}$, where $\mathbf{d}(iN + j) = \mathrm{d}_{\mathbb{E}}(f_{\mathbb{E}_1}, f_{\mathbb{E}_N})$. With these definitions, the optimization problem can be expressed as

$$\underset{\mathbf{c}}{\operatorname{argmin}}\ \mathcal{L}(\mathbf{Xc}, \mathbf{d}) \qquad \text{subject to } \mathbf{c} > 0, \tag{12}$$

where $\mathcal{L}$ denotes a loss function and the constraint on $\mathbf{c}$ ensures non-negative costs. Besides this constraint, one can also integrate a constraint to satisfy the triangular inequality, or one to ensure that a removal cost is equal to an insertion cost [28].

By defining the loss function $\mathcal{L}$ as the mean square error between computed GEDs and dissimilarities between the embeddings, the optimization problem can be rewitten as:

$$\underset{\mathbf{c}}{\operatorname{argmin}}\ ||\mathbf{Xc} - \mathbf{d}||_2^2 \qquad \text{subject to } \mathbf{c} > 0. \tag{13}$$

Solving this constrained optimization problem estimates $\mathbf{c}$ which allows to linearly fit GEDs to a specific target embedding space according to the edit paths initially given by ω. However, changes to the edit costs may affect the optimal

Algorithm 1.1 Optimization of constant edit costs according to given embeddings

1: $\mathbf{c} \leftarrow$ random(6)
2: $\mathbf{X} \leftarrow [\omega(G_1, G_1, \mathbf{c}) \;\; \omega(G_1, G_2, \mathbf{c}) \;\; \cdots \;\; \omega(G_N, G_N, \mathbf{c})]^\top$
3: **while** not converged **do**
4: $\quad \mathbf{c} \leftarrow \text{argmin}_{\mathbf{c}} \, ||\mathbf{X}\mathbf{c} - \mathbf{d}||_2^2$, subject to $\mathbf{c} > 0$
5: $\quad \mathbf{X} \leftarrow [\omega(G_1, G_1, \mathbf{c}) \;\; \omega(G_1, G_2, \mathbf{c}) \;\; \cdots \;\; \omega(G_N, G_N, \mathbf{c})]^\top$
6: **end while**

edit path, and consequently its description in terms of the number of edit operations. This leads to 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 address this interdependence, we propose an alternated optimization strategy, summarized in Algorithm 1.1 (See the example shown in Fig. 1). The two main steps of the algorithm are:

- **Estimate c for fixed X** (refer to line 4 in Algorithm 1.1): The given optimization issue is a constrained linear problem, which can also be viewed as a non-negative least squares problem [29]. This step linearly optimizes the constant costs for a specified set of edit operations between each graph pair, which is done by minimizing the difference between GEDs and the distances between their corresponding embeddings. Several readily available solvers can be used to tackle this problem, such as `CVXPY` [30] and `scipy` [31].

- **Estimate X for fixed c** (refer to line 5 in Algorithm 1.1): As discussed before, the changes made to costs in the preceding step may affect the associated edit path. To account for this, we follow up the cost optimization with a re-calculation of the optimal edit paths in line with the newly computed $\mathbf{c}$ vector which encodes the edit costs. Any method capable of computing GED can accomplish this step. For efficiency, one might opt for an approximated version of GED [27].

These optimization steps are alternately repeated to compute both edit costs and edit operations until the loss value no longer decreases or a iteration number limit is reached. Since the theoretical convergence proof of this optimization scheme has yet not been proposed, we limit the number of iterations to 5 in our implementation, which turns out to be sufficient in the conducted experiments.

The embeddings of the graphs and the distances between them may vary according to the embedded spaces. We consider the following embeddings in the rest of the paper:

– For a target space, the corresponding embeddings are the targets themselves. Since a target space $\mathbb{Y}$ is often a vector space, some off-the-shelf distances on $\mathbb{Y}$ are:

  - The *Euclidean* distance: $\mathrm{d}_{\mathbb{Y}}(y_i, y_j) = \|y_i - y_j\|_2$.
  - The *Manhattan* distance: $\mathrm{d}_{\mathbb{Y}}(y_i, y_j) = \|y_i - y_j\|_1$.

  we use the *Euclidean* distance in our experiments.

– For a graph kernel space $\mathcal{H}$, the distance between two elements $\phi(G_i)$ and $\phi(G_j)$ in $\mathcal{H}$ is

$$d_{\mathcal{H}}(\phi(G_i), \phi(G_j)) = \sqrt{k(G_i, G_i) + k(G_j, G_j) - 2k(G_i, G_j)}, \tag{14}$$

where $k(G_i, G_j)$ is a graph kernel between graphs G_i and G_j.
– For GNNs, we construct the embedding space by extracting the activations before the last layer. As these embeddings are often vectors or tensors, we apply Euclidean and Manhattan distances to them as well.

4 Applications of The Framework

The proposed Graph Edit Cost Learning (GECL) framework promises applications on varied tasks:

Predictions: Leveraging the interrelated spaces in graph structures, GECL can potentially introduce knowledge learned by other machine learning models (e.g., graph kernels, GNNs), thus improving the predictability and accuracy of GED-based prediction models on both regression and classification tasks.

Graph generation: With the learnt knowledge encoded in the edit costs, GECL can advance the generation of graphs. Specific areas of focus include median graph generation, which benefits from enhanced graph space understandability, and the pre-image problem, which benefits from the connection with the graph kernel space. Furthermore, the generation of Matching-Graphs [11] can be refined through the GECL framework, broadening its application scope.

Graph Matching Problems: GECL opens up new possibilities in addressing graph matching problems by providing a comprehensive view of the graph spaces. It may facilitate the detection and matching of similar patterns across disparate graphs, thereby enhancing the resolution of such problems.

Incorporation of Multi-level Information: the integration of the information at the node and graph level becomes more seamless with the GECL framework. A straightforward strategy would be to set the embedding space as the one of graph-level representations (e.g., vectors).

These potential applications foresee an exciting potential of powering the field of graph-based machine learning by the GECL framework.

5 Experiments

In this section, we exhibit the usage of our framework on two distinct applications: prediction tasks including regression and classification, and a graph pre-image generation task. First, we introduce the datasets used in the experiments.

Table 1. Results on each dataset in terms of RMSE for regression and accuracy (in %) for classification, measured on the test sets. The "-" notation indicate that the method is not suitable for the dataset.

Datasets	Random	Expert	Target	Path	Treelet	WLSubtree	GCN	GAT
Alkane	13.4±3.5	10.6±1.6	**5.9±0.7**	6.4±0.8	**5.9±0.7**	8.2±1.2	7.4±0.8	8.2±1.0
Acyclic	29.2±4.4	30.4±3.8	15.0±3.6	**13.0±3.6**	16.8±3.1	14.3±3.6	14.0±3.3	14.6±3.5
Redox ΔG_{red}^{PBE0}	25.3±9.5	36.2±13.3	24.8±7.0	20.1±5.8	**19.4±5.5**	22.1±6.2	26.1±6.4	21.0±6.0
Redox ΔG_{ox}^{PBE0}	24.4±7.1	40.0±11.5	26.8±10.1	26.4±6.1	**25.8±7.8**	26.7±6.3	28.5±8.4	26.9±7.8
MAO	80.0±9.9	74.3±10.6	80.0±13.8	81.4±9.7	81.4±10.8	84.3±7.5	84.3±7.5	**87.1±10.2**
PAH	69.0±11.4	71.0±9.2	68.0±8.1	68.0±7.4	**74.0±6.0**	71.0±10.9	67.0±11.2	68.0±6.6
MUTAG	80.0±7.1	81.6±6.2	78.9±6.6	82.6±6.2	**84.7±5.5**	81.1±6.7	80.0±5.8	80.5±5.9
Monoterpens	71.4±3.9	71.7±5.2	70.7±6.3	71.0±5.5	70.0±5.2	**72.4±5.3**	70.3±6.3	69.7±5.6
PTC_MR	56.3±7.1	56.0±4.4	59.4±4.7	55.7±6.0	55.1±5.8	**60.0±3.2**	57.4±8.3	54.0±5.4
Letter-high	84.3±8.7	84.3±2.4	**91.8±0.9**	90.1±2.1	-	-	82.6±1.2	82.6±1.2

5.1 Datasets

We conducted experiments[5] on benchmark datasets through multiple fields. Four datasets are evaluated for the regression problem: *Alkane* and *Acyclic* are molecule dataset associated with the boiling point prediction. *Redox* is the newly generated small molecule dataset aiming at predicting the Redox potential, where *Redox* ΔG_{red}^{PBE0} and *Redox* ΔG_{ox}^{PBE0} represent respectively the reduction and oxidation potential targets[6]. Six datasets associated with classification problems are considered. *MAO*, *PAH*, *MUTAG*, *Monoterpens*, and *PTC_MR* are chemical molecules associated with different classification problems. *Letter-high* involves graphs of highly distorted letter drawings where the task is to classify each graph to the proper letter. These datasets cover unlabeled graphs, graphs with discrete and continuous vertex attributes and edge attributes.

5.2 On Prediction Tasks

To evaluate the predictive power of GED empowered with knowledge of different target spaces, we used a k-nearest-neighbors regression model [32], 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 prediction performance. A 5-fold cross-validation (CV) procedure is performed on the remaining 90%. Pairwise distances in the embedded space are computed on the training fold, and then the edit costs are optimized accordingly, and the value of k is optimized through the CV over the candidate values $\{3, 5, 7, 9, 11\}$. The number of iterations to optimize the edit costs is fixed to 5. The GEDs are estimated by the `bipartite` heuristics [27]. Three graph kernels (i.e., path kernel, treelet kernel, and WL-subtree kernel) and two GNNs (i.e., GCN and

[5] `https://github.com/jajupmochi/ged-cost-learn-framework/tree/master/`.

[6] We thank the COBRA lab (Chimie Organique Bioorganique : Réactivité et Analyse) and the ITODYS lab (Le laboratoire Interfaces Traitements Organisation et DYnamique des Systèmes) for providing this dataset.

GAT) are applied to derive embeddings. The proposed optimization procedure is compared to two other edit costs settings: the first is a random set of edit costs; the second is a predefined cost setting given in [27], namely the so-called expert costs with $c_{vi} = c_{vr} = c_{ei} = c_{er} = 3$, $c_{vs} = c_{es} = 1$.

Table 1 shows the average root mean squared errors (RMSE) obtained for each cost settings over the 10 splits, estimated on the test set. The $\pm$ sign gives the 95% confidence interval computed over the 10 repetitions. The first four datasets are associated with regression problems, where lower values indicate better results; while the remaining datasets are for classification tasks, with higher values indicating better performance. The best prediction for each dataset over all methods is marked green and bold. As expected, a clear and significant gain in accuracy is obtained by using the proposed framework to introduce knowledge from embedded spaces. Compared with using random and expert costs, the improvements can be up to 57% for regression problems and 17% for classification problems. Five best performances are achieved by treelet kernels, two by WL-subtree kernel, one by path kernel, one by GAT, and two directly on prediction targets. For regression problems, optimizing edit costs through prediction targets can achieve better results than random or expert costs in most cases. When embedded spaces are considered, further precision can be achieved. For classification problems, using prediction targets cannot optimize the edit costs in general. This is due to the fact that the targets (i.e., classes) are symbolic values and even boolean for binary classification. As a result, the corresponding distances have limited values (e.g., only one and zero in the case of binary classification). The only exception is on *Letter-high*, which may due the fact that there are 15 classes on this dataset, thus providing more information for the metric learning. Meanwhile, methods using random and expert costs are beaten by applying embedded spaces on all these data sets in accuracy. This promising result confirms the hypothesis that the edit cost optimized by embedded spaces can capture their underlying information, and thus improve the prediction accuracy while still operating in the graph space.

5.3 Graph Generations: A Pre-image Example

By fixing the embedding space as a kernel space, we align the distances in it and graph space as proposed in Section 3.3. The pre-image problem can then be recast as a graph generation problem [21]. We first optimize edit cost constants according to the embedded kernel space, and then we use these optimized edit costs, and take advantage of recent advances where the proposed iterative alternate minimization procedure (IAM) allows generating new graphs [33]. We choose the *Letter-high* dataset to show the potential of our method. Spaces derived from two graph kernels are chosen as the embedded spaces, namely the shortest path kernel and the structure shortest path kernel. The choice is made on the fact that these two graph kernels can tackle continuous vertex attributes. Notice that this specific application is already exhibited in our previous work [15], which serves here as an illustration of the capabilities of our framework.

Table 2. Distances $d_{\mathbb{E}}$ in embedding spaces computed using different methods.

Embedding Spaces	From median set	Random costs	Expert costs	Optimized costs (ours)
Shortest Path (SP)	0.406	0.467	0.451	**0.460**
Structural SP (SSP)	0.413	0.435	0.391	**0.394**

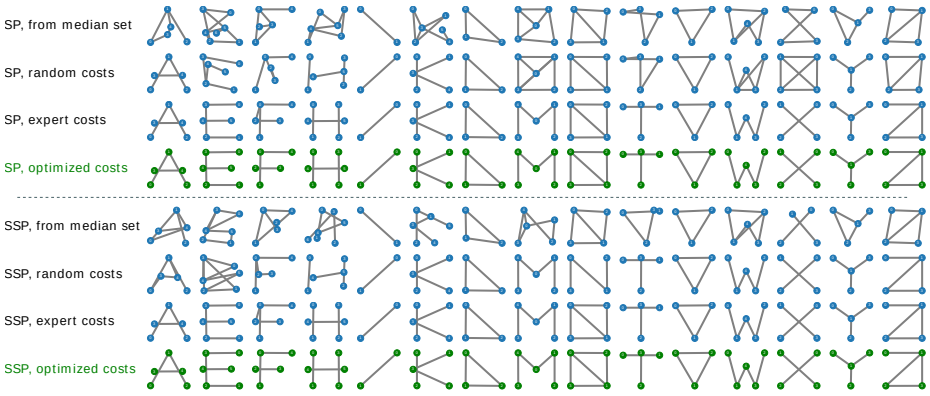

Fig. 2. Pre-images constructed by different algorithms for *Letter-high* with shortest path (SP) and structural shortest path (SSP) kernels.

Table 2 shows the distance $d_{\mathbb{E}}$ between generated graphs and the "true" median graphs in embedding spaces. "from median set" is a reference method which takes the set median graph from a graph set as the pre-image of its median. Our framework produces smaller or competitive $d_{\mathbb{E}}$ compared to other methods. Moreover, the advantage of our framework can be evaluated from a more intuitive aspect. Fig. 2 presents the pre-images generated as the median graphs for each letter in the *Letter-high* dataset using the aforementioned methods, Vertices are drawn according to coordinates determined by their attributes "x" and "y". In this way, plots of graphs are able to display the letters that they represent, which are possible to be recognized by human eyes. When using the shortest path (SP) kernel (first to fourth rows), it can be seen that when the expert and optimized costs are used, almost all letters are readable, compared to the first two methods, despite that the pre-images of letter F are slightly different (the third and fourth rows). The same conclusion can be derived for the structure shortest path kernel as well (fifth to eighth rows). This analysis indicates that the proposed algorithms are able to generate better pre-images, especially when edit costs are optimized. It provides a "direction" to construct pre-images with respect to the features and structures of graphs. The experiment effectively showcases the capabilities of the GECL framework in handling graph generation tasks.

6 Conclusion and Future Work

In this paper, we proposed the GECL framework which combines the varied graph-based spaces. We showed that this framework has the potential to learn

knowledge from embedded spaces by optimizing the edit costs of GEDs. A specific case based on supervised metric learning was proposed as an example to show the power of the framework. Experiments on two different tasks were performed, namely regression and classification predictions and the pre-image problem. Both results showed that the GECL framework outperforms the methods using predefined experts edit costs, bringing promising possibilities to these fields.

In this paper, we only exhibit a special example of the framework. While multiple state-of-the-art metric learning strategies and edit cost learning algorithms can be included in the framework, it would be interesting to tackle their abilities. Furthermore, as described in Section 4, exploring the application of the framework in more tasks would be a substantial work to do. Finally, we plan to seek out deeper underlying relationships between different graph-based spaces through a deep study of this framework.

References

1. Trinajstic, N.: Chemical graph theory. Routledge (2018)
2. Yi, H.C., You, Z.H., Huang, D.S., Kwoh, C.K.: Graph representation learning in bioinformatics: trends, methods and applications. Briefings in Bioinformatics **23**(1), bbab340 (2022)
3. Tabassum, S., Pereira, F.S., Fernandes, S., Gama, J.: Social network analysis: An overview. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **8**(5), e1256 (2018)
4. Jiao, L., Chen, J., Liu, F., Yang, S., You, C., Liu, X., Li, L., Hou, B.: Graph representation learning meets computer vision: A survey. IEEE Transactions on Artificial Intelligence **4**(1), 2–22 (2022)
5. Goyal, P., Ferrara, E.: Graph embedding techniques, applications, and performance: A survey. Knowledge-Based Systems **151**, 78–94 (2018)
6. Kriege, N.M., Johansson, F.D., Morris, C.: A survey on graph kernels. Applied Network Science **5**(1), 1–42 (2020)
7. Jia, L., Gaüzère, B., Honeine, P.: Graph kernels based on linear patterns: theoretical and experimental comparisons. Expert Systems with Applications **189**, 116095 (2022)
8. Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., Sun, M.: Graph neural networks: A review of methods and applications. AI open **1**, 57–81 (2020)
9. Grattarola, D., Zambon, D., Livi, L., Alippi, C.: Change detection in graph streams by learning graph embeddings on constant-curvature manifolds. IEEE Transactions on neural networks and learning systems **31**(6), 1856–1869 (2019)
10. Bunke, H., Allermann, G.: Inexact graph matching for structural pattern recognition. Pattern Recognition Letters **1**(4), 245–253 (1983)
11. Fuchs, M., Riesen, K.: A novel way to formalize stable graph cores by using matching-graphs. Pattern Recognition **131**, 108846 (2022)
12. Neuhaus, M., Bunke, H.: A random walk kernel derived from graph edit distance. In: Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops, SSPR 2006 and SPR 2006, Hong Kong, China, August 17-19, 2006. Proceedings. pp. 191–199. Springer (2006)

13. Neuhaus, M., Bunke, H.: Bridging the gap between graph edit distance and kernel machines, vol. 68. World Scientific (2007)
14. Gaüzère, B., Brun, L., Villemin, D.: Graph kernels: Crossing information from different patterns using graph edit distance. In: Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshop, SSPR&SPR 2012, Hiroshima, Japan, November 7-9, 2012. Proceedings. pp. 42–50. Springer (2012)
15. Jia, L., Gaüzère, B., Honeine, P.: A graph pre-image method based on graph edit distances. In: Proceedings of the IAPR Joint International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (S+SSPR). Venice, Italy (21 - 22 Jan 2021)
16. Riba, P., Fischer, A., Lladós, J., Fornés, A.: Learning graph edit distance by graph neural networks. Pattern Recognition **120**, 108132 (2021)
17. Feng, A., You, C., Wang, S., Tassiulas, L.: Kergnns: Interpretable graph neural networks with graph kernels. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 6614–6622 (2022)
18. Du, S.S., Hou, K., Salakhutdinov, R.R., Poczos, B., Wang, R., Xu, K.: Graph neural tangent kernel: Fusing graph neural networks with graph kernels. Advances in neural information processing systems **32** (2019)
19. Morris, C., Ritzert, M., Fey, M., Hamilton, W.L., Lenssen, J.E., Rattan, G., Grohe, M.: Weisfeiler and leman go neural: Higher-order graph neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 4602–4609 (2019)
20. 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 and Structural and Syntactic Pattern Recognition. pp. 287–297. Springer (2008)
21. Jia, L.: Bridging graph and kernel spaces: a pre-image perspective. Ph.D. thesis, Normandie (2021)
22. Neuhaus, M., Bunke, H.: A probabilistic approach to learning costs for graph edit distance. Proceedings ICPR **3**(C), 389–393 (2004)
23. Bellet, A., Habrard, A., Sebban, M.: Good edit similarity learning by loss minimization. Machine Learning **89**(1-2), 5–35 (2012)
24. 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
25. Jia, L., Gaüzère, B., Yger, F., Honeine, P.: A metric learning approach to graph edit costs for regression. In: Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops, S+ SSPR 2020, Padua, Italy, January 21–22, 2021, Proceedings. pp. 238–247. Springer (2021)
26. Garcia-Hernandez, C., Fernández, A., Serratosa, F.: Learning the edit costs of graph edit distance applied to ligand-based virtual screening. Current topics in medicinal chemistry **20**(18), 1582–1592 (2020)
27. 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)
28. Riesen, K.: Structural pattern recognition with graph edit distance. In: Advances in computer vision and pattern recognition. Springer (2015)
29. Lawson, C.L., Hanson, R.J.: Solving least squares problems. SIAM (1995)
30. Diamond, S., Boyd, S.: Cvxpy: A python-embedded modeling language for convex optimization. The Journal of Machine Learning Research **17**(1), 2909–2913 (2016)

31. 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)
32. Altman, N.S.: An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician **46**(3), 175–185 (1992)
33. Boria, N., Bougleux, S., Gaüzère, B., Brun, L.: Generalized median graph via iterative alternate minimizations. In: International Workshop on Graph-Based Representations in Pattern Recognition. pp. 99–109. Springer (2019)