A Graph Pre-image Method Based on Graph Edit Distances*

Linlin Jia[1], Benoit Gaüzère[1], and Paul Honeine[2]

[1] LITIS, INSA Rouen Normandie, Rouen, France
[2] LITIS, Université de Rouen Normandie, Rouen, France

Abstract. The pre-image problem for graphs is increasingly attracting attention owing to many promising applications. However, it is a challenging problem due to the complexity of graph structure. In this paper, we propose a novel method to construct graph pre-images as median graphs, by aligning graph edit distances (GEDs) in the graph space with distances in the graph kernel space. The two metrics are aligned by optimizing the edit costs of GEDs according to the distances between the graphs within the space associated with a particular graph kernel. Then, the graph pre-image can be estimated using a median graph method founded on the GED. In particular, a recently introduced method to compute generalized median graphs with iterative alternate minimizations is revisited for this purpose. Conducted experiments show very promising results while opening the computation of graph pre-image to any graph kernel and to graphs with non-symbolic attributes.

Keywords: Pre-image problem · Machine Learning · Graph Kernels · Graph Edit Distance.

1 Introduction

Graph structures have been increasingly attracting attention in pattern recognition and machine learning. While they are able to represent a wide range of data, from molecules to social networks, most machine learning methods operate on Euclidean data. Graph kernels allow bridging the gap between the graph structure and machine learning thanks to the kernel trick. This trick consists in implicitly embedding graphs into a Hilbert space, where kernel methods such as Support Vector Machines can be easily operated. The reverse process of the implicit embedding with kernels, namely the so-called pre-image, continues to intrigue researchers. It corresponds to the mapping of elements from the kernel space back to the input space. Many applications require computing the pre-image, such as denoising or feature extraction with kernel principal component analysis [14]. The challenge of finding the pre-image lies in the fact that the reverse mapping does not exist in general and that most elements in the kernel

* This research was supported by CSC (China Scholarship Council) and the French national research agency (ANR) under the grant APi (ANR-18-CE23-0014).

space do not own valid pre-images in the input space. Consequently, various methods have been developed to approximate the solution, namely, to solve the pre-image problem. We refer interested readers to the tutorial [15].

Solving the pre-image problem for graphs opens the door to many interesting applications, such as molecule synthesis and drug design. However, finding the pre-image as a graph inherits the difficulties of traditional pre-image problems. Additionally, unlike inputs considered by traditional pre-image problems, i.e. vectors which are usually lying in continuous spaces, graphs are discrete structures with a variable and non-ordered number of vertices and edges. Furthermore, multiple labels and attributes can be plugged into each vertex and edge in a graph. Given these structure features, the graph pre-image problem is more challenging to address.

Several pioneering works to construct graphs have been proposed. A method based on the random search is proposed in [3]. It is simple to implement, but has a very high computational complexity and is not applicable to continuous real-valued labels, while the quality of the synthesized graph pre-images is not guaranteed. The methods of [2] and [19] infer a graph from path frequency. However, these methods are either restricted to applying a specific sub-structure of graphs or ignoring vertex and edge labels, which are important information for graphs. All these studies do not fully benefit from discrete optimization that needs to be carried out for graph pre-image. In this paper, we propose a novel pre-image method for graphs. To this end, we bridge the gap between graph edit distances (GEDs) and any given graph kernel, which allows uncovering the relationship between graph space and kernel space. GED is a well-known dissimilarity measure between graphs, based on elementary operations that transform one graph to another. By optimizing the edit costs of these operations according to distances between elements in the kernel space, the metrics of the two aforementioned spaces are aligned, thus allowing constructing the graph pre-image by a median graph method based on GEDs. Specifically, a pre-image problem for the median graph of a graph set is addressed, based on the hypothesis that, benefiting from the alignment of the two metrics, the median of the set of graphs corresponds to the mean of their embeddings in the kernel space. We take advantage of recent advances in GED to solve this problem, where an iterative alternate minimization procedure to generate median graphs is adapted [7].

The remainder of the paper is organized as follows. The next section introduces preliminaries for the paper. Section 3 presents the proposed method in two folds, learning edit costs for GEDs by the distances in kernel space (Section 3.1) and inferring the graph pre-image (Section 3.2). Section 4 gives experiments and analyses. Finally, Section 5 concludes the paper.

2 Preliminaries

2.1 Graphs, Graph Kernels, and Graph Pre-images

A graph $G = (V, E)$ is an ordered pair of disjoint sets, where V is the vertex set and $E \in V \times V$ is the edge set. A graph can have a label set L from a label space

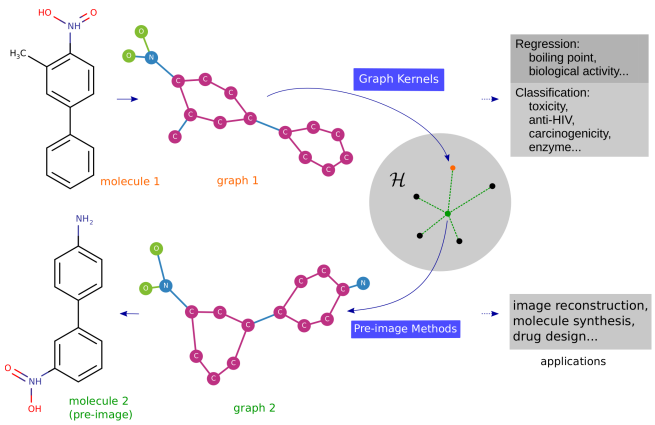

Fig. 1. A graph kernel maps graphs to a kernel space $\mathcal{H}$, while the pre-image provides the reverse procedure, by mapping elements from kernel space back to graphs.

and a labeling function ℓ that assigns a label $l \in L$ to each vertex and/or edge, where l can be symbolic (i.e. discrete values) or non-symbolic (i.e. continuous values). Let φ be the set of vertex labels, Φ the set of edge labels, and n the number of vertices in graph G ($n = |V|$). See [22] for more details.

A positive semi-definite kernel is a symmetric bilinear function that satisfies $\sum_{i=1}^{n} \sum_{j=1}^{n} c_i\, c_j\, k(x_i, x_j) \geq 0$, for all $x_i, \ldots, x_n$ and $c_1, \ldots, c_n \in \mathbb{R}$. These kernels are simply denoted as *kernels* in this paper for conciseness. A kernel corresponds to an inner product between implicit embeddings of input data into an Hilbert space $\mathcal{H}$ (RKHS) thanks to an implicit mapping function $\phi : \mathcal{X} \to \mathcal{H}$.

Graph kernels are kernels defined on graphs. For a given graph kernel, $k(G_i, G_j)$ corresponds to an inner product between the two mapped graphs $\phi(G_i)$ and $\phi(G_j)$ in the kernel space $\mathcal{H}$. More details on graph kernels can be found in [13, 18, 12, 11]. Given a kernel, the mapping $\phi(\cdot)$ remains implicit and is defined by the kernel itself. It does not have to be explicitly known thanks to the kernel trick. However, the reverse map may be interesting and is difficult to compute in general. Indeed, most combinations $\psi = \sum_i \alpha_i \phi(G_i)$ do not have a valid pre-image, namely a graph $G^\star$ such that $\phi(G^\star) = \psi$; The pre-image problem consists in estimating an approximate solution, namely $\widehat{G}$ such that $\phi(\widehat{G}) \approx \psi$ (Fig. 1).

2.2 Graph Edit Distance

The Graph Edit Distance (GED) between two graphs G_i and G_j is defined as the cost of minimal transformation [21]:

$$d_{GED}(G_i, G_j) = \min_{\pi \in \Pi(G_i, G_j)} C(\pi, G_i, G_j), \tag{1}$$

where $\pi(G_i, G_j)$ is a mapping between $V_i \cup \varepsilon$ and $V_j \cup \varepsilon$ encoding the transformation from G_i to G_j [8]. This transformation consists in a series of six elementary

operations: removing or inserting a vertex or an edge, and substituting a label of a vertex or an edge by another label. $C(\pi, G_i, G_j)$ measures the cost associated to π:

$$\begin{aligned} C(\pi, G_i, G_j) = & \sum_{\substack{v \in V(G_j) \\ \pi^{-1}(v) \notin V(G_i)}} c_{vfi}(\varepsilon, v) + \sum_{\substack{u \in V(G_i) \\ \pi(u) \notin V(G_j)}} c_{vfr}(u, \varepsilon) + \sum_{\substack{u \in V(G_i) \\ \pi(u) \in V(G_j)}} c_{vfs}(u, \pi(u)) \\ & + \sum_{\substack{f \in E(G_j) \\ \pi^{-1}(f) \notin E(G_i)}} c_{efi}(\varepsilon, f) + \sum_{\substack{e \in E(G_i) \\ \pi(e) \notin E(G_j)}} c_{efr}(e, \varepsilon) + \sum_{\substack{e \in E(G_i) \\ \pi(e) \in E(G_j)}} c_{efs}(e, \pi(e)), \end{aligned} \tag{2}$$

where $c_{vfr}, c_{vfi}, c_{vfs}, c_{efr}, c_{efi}, c_{efs}$ are the edit cost functions associated to the six edit operations: respectively vertex removal, insertion, substitution and edge removal, insertion and substitution. According to [17], the edit cost functions for graphs with non-symbolic labels can be defined as:

$$\begin{cases} c_{vfi}(\varepsilon, v) = c_{vi}, \quad c_{efi}(\varepsilon, e) = c_{ei}, \quad c_{vfr}(v, \varepsilon) = c_{vr}, \quad c_{efr}(e, \varepsilon) = c_{er}, \\ c_{vfs}(u, v) = c_{vs} \|\ell_v(u) - \ell_v(v)\|, \quad c_{efs}(e, f) = c_{es} \|\ell_e(e) - \ell_e(f)\|, \end{cases} \tag{3}$$

where $c_{vr}, c_{vi}, c_{vs}, c_{er}, c_{ei}, c_{es}$ are the edit costs, namely the coefficients applied to the edit operations. Let $\boldsymbol{c} = [c_{vr}, c_{vi}, c_{vs}, c_{er}, c_{ei}, c_{es}]^\top$ be the edit cost vector.

By definition, the GED can be regarded as a distance measure between graphs. However, the problem of computing the GED is NP-hard [4]. Many methods have been proposed to approximate GED, such as `bipartite` [21] and `IPFP` [8]. For more details on GEDs, we refer interested readers to [21, 4].

3 Proposed Graph Pre-image Method

The main motivation of this work is to address the pre-image problem by building connections between graph and kernel spaces. We propose to align the metrics of the two spaces by optimizing the edit costs such that GEDs approximate the distances in kernel space. Then, once GEDs and kernel distances are similar, we propose to recast the pre-image problem as a graph generation problem, based on the assumption that the median of a set of graphscorresponds to the mean of their embeddings in the kernel space. An iterative alternate minimization method is adapted for this purpose, in which the GEDs with the optimized edit cost distances are used. These two steps are detailed next, and the proposed method is summarized in Algorithm 1.

3.1 Learn Edit Costs by Distances in Kernel Space

When computing GEDs, the choice of edit costs values is essential. In practice, they are determined by domain experts for a given dataset. With our original idea of aligning the GEDs to the kernel metric, we propose to learn the edit costs by the distances of the elements in the kernel space.

On one hand, the distance in $\mathcal{H}$ between two elements $\phi(G_i)$ and $\phi(G_j)$ is:

$$d_{\mathcal{H}}(\phi(G_i), \phi(G_j)) = \sqrt{k(G_i, G_i) + k(G_j, G_j) - 2k(G_i, G_j)}. \tag{4}$$

Algorithm 1 Proposed method

Input: Dataset $\mathbb{G}_N$, graph kernel k, thresholds of stopping criteria (r_{max}, i_{max}).
Output: The approximation of the pre-image.

Initialize randomly $\boldsymbol{c}^{(0)} = [c_{vr}^{(0)}, c_{vi}^{(0)}, c_{vs}^{(0)}, c_{er}^{(0)}, c_{ei}^{(0)}, c_{es}^{(0)}]^\top$.
Compute kernel distances $\boldsymbol{d}_{\mathcal{H}}$ of all pairs of graphs in $\mathbb{G}_N$ with (4).
Let $r = 0$.
while $r < r_{max}$ **do**
  For fixed $\boldsymbol{c}^{(r)}$, estimate $\boldsymbol{W}^{(r)}$ by solving (7) using a GED heuristic (e.g. `bipartite` or `IPFP`).
  For fixed $\boldsymbol{W}^{(r)}$, estimate $\boldsymbol{c}^{(r+1)}$ by solving (7) using constrained linear least square programming (e.g. `CVXPY`).
  $r = r + 1$.
end while
Find set-median $\widehat{G}^{(0)}$ by (9).
Let $i = 0$.
while $i < i_{max}$ **do**
  Compute transformation $\widehat{\pi}_p^{(i+1)}$ by (10) for $\widehat{G}^{(i)}$ with $\boldsymbol{c}^{(r+1)}$.
  Generate $\widehat{G}^{(i+1)}$ by (11) with $\widehat{\pi}_p^{(i+1)}$ and $\boldsymbol{c}^{(r+1)}$.
  $i = i + 1$.
end while
$\widehat{G}^{(i+1)}$ is the graph pre-image.

On the other hand, considering Eq. (1) and the costs defined in Eq. (3), the GED between G_i and G_j is given by:

$$d_{GED}(G_i, G_j) = \boldsymbol{\omega}^\top \boldsymbol{c}, \tag{5}$$

with $\boldsymbol{\omega} = [n_{vr}, n_{vi}, \omega_{vs}, n_{er}, n_{ei}, \omega_{es}]^\top$, where $n_{vr}, n_{vi}, n_{er}, n_{ei}$ are respectively the numbers of vertex removals, insertions, and edge removals, insertions. $\omega_{vs} = \sum_{u \in V(G_i), \pi(u) \in V(G_j)} \|\ell_v(u) - \ell_v(\pi(u))\|$ is the sum of distances of labels between all pairs of vertices; and $\omega_{es} = \sum_{e \in E(G_i), \pi(e) \in E(G_j)} \|\ell_e(e) - \ell_e(\pi(e)\|$ is the sum of distances of labels between all pairs of edges.

A major difficulty, which is not straightforward from (5), is that ω and $\boldsymbol{c}$ are interdependent. For two different edit cost vectors, respective optimal ω may not be equivalent since the costs influence the presence or absence of each edit operation. In addition, ω influences also $\boldsymbol{c}$ since we want to fit GED with kernel distances, i.e., $d_{GED}(G_i, G_j) \approx d_{\mathcal{H}}(\phi(G_i), \phi(G_j))$.

Given a graph space $\mathcal{G}$ of attributed graphs and a kernel space $\mathcal{H}$, we propose to align GEDs in $\mathcal{G}$ with distances in $\mathcal{H}$ between each pair of graphs. In other words, we seek to learn the edit costs of the GED, so that the GED between each pair of graphs in $\mathcal{G}$ is as close as possible to its corresponding distance in $\mathcal{H}$. To achieve this goal, a least squares optimization on graph dataset $\mathbb{G}_N = \{G_1, G_2, \ldots, G_N\} \subset \mathcal{G}$ is considered, namely

$$\arg\min_{\boldsymbol{c}, \boldsymbol{\omega}} \sum_{i,j=1}^{N} \left(d_{GED}(G_i, G_j) - d_{\mathcal{H}}(\phi(G_i), \phi(G_j))\right)^2, \tag{6}$$

with d_{GED} depending on $\boldsymbol{c}$ and $\boldsymbol{\omega}$ as given in (5), where $\boldsymbol{\omega}$ exists for each pair of graphs G_i and G_j in $\mathbb{G}_N$, which will be denoted as $\boldsymbol{\omega}(i, j)$. Moreover, to ensure that the minimum cost edit transformation π in (1) can be found, all edit costs need to be positive, and substituting an element should not be more expensive

than removing and inserting it [21]. Thus, the optimization problem becomes:

$$\arg\min_{\boldsymbol{c},\boldsymbol{W}} \|\boldsymbol{W}^\top \boldsymbol{c} - \boldsymbol{d}_{\mathcal{H}}\|^2 \text{ subject to } \boldsymbol{c} > \boldsymbol{0}, c_{vr} + c_{vi} \geq c_{vs} \text{ and } c_{er} + c_{ei} \geq c_{es}, \quad (7)$$

where $\boldsymbol{W}^\top \in \mathbb{R}^{N^2 \times 6}$ with rows $\boldsymbol{\omega}(i,j)^\top$ and $\boldsymbol{d}_{\mathcal{H}} \in \mathbb{R}^{N^2}$ encoding the GEDs for each pair of graphs of $\mathbb{G}_N$. To solve this constrained optimization problem, we propose an alternating optimization strategy over $\boldsymbol{c}$ and $\boldsymbol{W}$. The optimization problem over $\boldsymbol{c}$, for a fixed $\boldsymbol{W}$, is a constrained linear least square program problem solved using `CVXPY` [10, 1]. Once the edit costs obtained, the weights $\boldsymbol{W}$ are computed by GED heuristics, such as `bipartite` and `IPFP`.

3.2 Generate Graph Pre-image

Given a set of graphs $\mathbb{G}_N \subset \mathcal{G}$, its average point can be easily computed in the kernel space, i.e., $\psi = \sum_{i=1}^{N} \alpha_i \phi(G_i)$ with $\alpha_i = 1/N$. Our objective is to estimate its pre-image, namely the graph $\widehat{G}$ whose image $\phi(\widehat{G})$ is as close as possible to ψ.

With the metric alignment principle (6), $d_{GED}(G_i, G_j) \approx d_{\mathcal{H}}(\phi(G_i), \phi(G_j))$, for all $G_i, G_j \in \mathbb{G}_N$. Therefore, estimating the pre-image is equivalent to estimating the graph median, which can be tackled as the minimization of the sum of distances (SOD) to all the graphs of $\mathbb{G}_N$, namely

$$\widehat{G} = \arg\min_{G \in \mathcal{G}} \sum_{G_{p'} \in \mathbb{G}_N} d_{GED}(G, G_{p'}). \quad (8)$$

A first attempt to solve it is to restrict the solution to the set $\mathbb{G}_N$, namely

$$\widehat{G} = \arg\min_{G_p \in \mathbb{G}_N} \sum_{G_{p'} \in \mathbb{G}_N} d_{GED}(G_p, G_{p'}) = \arg\min_{G_p \in \mathbb{G}_N} \sum_{p'=1}^{N} \min_{\pi_{p'} \in \Pi(G_p, G_{p'})} c(\pi_{p'}, G_p, G_{p'}), \quad (9)$$

where cost $c(\pi_{p'}, G_p, G_{p'})$ consists of two parts, $c_v(\pi_{p'}, \varphi_p, \varphi_{p'})$ and $c_e(\pi_{p'}, A_p, \Phi_p, A_{p'}, \Phi_{p'})$, which are costs of vertex and edge transformation, respectively. This problem can be solved by computing all pairwise GEDs for dataset $\mathbb{G}_N$. The computational complexity is in $\mathcal{O}(aN^2)$, where a is the complexity of computing a GED between two graphs (for instance, by `bipartite` or `IPFP`). The resulting pre-image $\widehat{G}$ is also known as the set-median of $\mathbb{G}_N$.

Despite its simplicity, the set-median can only be chosen from the given dataset $\mathbb{G}_N$, which strongly limits the results. To obtain the pre-image from a bigger space, we take advantage of recent advances in [7] where the proposed iterative alternate minimization procedure (IAM) allows generating new graphs. Next, we revisit this method and adapt it for the pre-image problem. The proposed strategy alternates the optimization over all the $\widehat{\pi}_p$ (i.e., transformations from $\widehat{G}$ to G_p) and over the pre-image estimate $\widehat{G}$, namely

$$\widehat{\pi}_p = \arg\min_{\pi_p \in \Pi(\widehat{G}, G_p)} c(\pi_p, \widehat{G}, G_p) \qquad \forall p \in \{1, \ldots, N\}; \quad (10)$$

$$\widehat{G} = \arg \min_{\substack{\varphi \in \mathcal{H}_v^{\widehat{n}} \\ A \in \{0,1\}^{\widehat{n} \times \widehat{n}} \\ \Phi \in \mathcal{H}_e^{\widehat{n} \times \widehat{n}}}} \sum_{p=1}^{N} c_v(\widehat{\pi}_p, \varphi, \varphi_p) + \tfrac{1}{2} c_e(\widehat{\pi}_p, A, \Phi, A_p, \Phi_p). \tag{11}$$

The resolution of (10) is carried out by solving the GED problem N times with time complexity of $\mathcal{O}(aN)$, and the computation of (11) is detailed in [7], where the vertices and edges are updated separately. The new non-symbolic labels assigned for a vertex v (resp. an edge e) are given by the average values of the corresponding labels of the vertices substituted to v (resp. edges substituted to e). The obtained pre-image $\widehat{G}$ is also known as the generalized median of $\mathbb{G}_N$.

4 Experiments

To perform experiments, we implemented[3] Algorithm 1 in Python. The C++ library `GEDLIB`[4] and its Python interface `gedlibpy` are used as the core implementation to compute graph edit distances and perform IAM algorithm [5]. We implemented a general edit cost function `NonSymbolic`[5] for graphs containing only non-symbolic vertex and/or edge labels and an edit cost function `Letter2` specifically for dataset *Letter-high* based on `NonSymbolic`. In these functions, all edit costs can be freely set, which is more convenient for the optimization proposed in Section 3.1. We have modified the `gedlibpy` accordingly[6]. All experiments were carried out on a computer with 8 CPU cores of Intel Core i7-7920HQ @ 3.10GHz, 32GB memory, and 64-bit operating system Ubuntu 16.04.3 LTS.

Given the *Letter-high* dataset[7], the goal is to compute, for a given kernel, the pre-image of the average of each class of letters, namely $\psi = \sum_{i=1}^{N} \alpha_i \phi(G_i)$ with $\alpha_i = 1/N$. Two graph kernels are considered, the shortest path (SP) kernel [6] and the structural SP kernel [20], both being able to deal with non-symbolic labels; see [16]. In each class (i.e., a set of distortions of a letter), all 150 graphs are chosen to compose the graph set $\mathbb{G}_N$. To estimate the graph edit distances, a multi-start counterpart of `IPFP` (i.e., m`IPFP`) is applied in both procedures of producing set-median and generalized median, where 40 different solutions to the LSAP are chosen [9]. The maximum number of iterations is set to $r_{max} = 6$.

Table 1 exhibits experimental results. Results of two sets of edit costs are presented. The first set of constants is randomly generated for each class of graphs, while the second set is given by domain experts, where $c_{vi} = c_{vr} = 0.675, c_{ei} = c_{er} = 0.425, c_{vs} = 0.75$ and $c_{es} = 0$ [4]. It is worth noting that these expert values take into account prior knowledge of the data, such as setting c_{es} to 0 as graphs in *Letter-high* do not contain edge labels. Moreover, we also give as a baseline a method to generate median graphs (denoted "From median set"), where the median graph is directly chosen from the median set $\mathbb{G}_N$ whose

[3] *https://github.com/jajupmochi/graphkit-learn/tree/master/gklearn/preimage.*
[4] `GEDLIB`: *https://github.com/dbblumenthal/gedlib*
[5] *https://github.com/jajupmochi/gedlib/tree/master/src/edit_ costs*
[6] `gedlibpy` (modified): *https://github.com/jajupmochi/gedlibpy*
[7] *http:// graphkernels.cs.tu-dortmund.de.*

Table 1. Distances in kernel space computed using different methods.

Graph Kernels	Algorithms	$d_{\mathcal{H}}$ GM	Running Times (s)		
			Optimization	Generation	Total
Shortest path (SP)	From median set	0.406	-	-	-
	IAM: random costs	0.467	-	142.59	142.59
	IAM: expert costs	0.451	-	30.31	30.31
	IAM: optimized costs	0.460	5968.92	26.55	5995.47
Structural SP (SSP)	From median set	0.413	-	-	-
	IAM: random costs	0.435	-	30.22	30.22
	IAM: expert costs	0.391	-	29.71	29.71
	IAM: optimized costs	0.394	24.79	25.60	50.39

representation in kernel space is the closest to the true median's (ψ). The average results over all classes are presented for all methods. Column "$d_{\mathcal{H}}$ GM" gives the distances between the embedding of the computed pre-image and the element we want to approximate in the kernel space ($d_{\mathcal{H}}$). The columns "Running Times" give the time to optimize edit costs and generate pre-images.

For the structural SP kernel, when expert and optimized methods are given, applying IAM provides better pre-images than choosing from the median set with respect to $d_{\mathcal{H}}$s of the generalized medians. Compared to $d_{\mathcal{H}}$ of pre-image choosing from median sets, $d_{\mathcal{H}}$ is respectively 5.33% and 4.60% smaller for algorithm with expert and optimized costs. Moreover, $d_{\mathcal{H}}$ of the algorithm with optimized costs is 9.43% smaller than that with random costs and is almost the same as the algorithm with expert costs, which is also the case for the SP kernel. These results show that the algorithm with the optimized costs works better than the one with random costs to generate pre-images as median graphs, and can serve as a method to tune edit costs to help find expert costs, for both median generation problems using IAM and general pre-image problems. Moreover, the running times to optimize edit costs and generate pre-images are acceptable in most cases.

Although these improvements seem trivial, the advantage of our method can be valuated from other aspects. Fig. 2 presents the pre-images generated as the median graphs for each letter of the *Letter-high* dataset using aforementioned methods, which correspond to the eight rows of Table 1, row by row. 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 SP kernel (the first row to the fourth row), it can be seen that the pre-images chosen directly from the median set (the first row) are illegible in almost all cases, while the IAM with random costs provides more legible results, where letters A, K, Y can be easily recognized (the second row). When the expert and optimized costs are used, almost all letters are readable, 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 SP kernel as well (the fifth row to the eighth row).

This analysis indicates that even though the distances $d_{\mathcal{H}}$ are similar, the algorithms applying IAM are able to generate better pre-images, especially when edit costs are optimized. This phenomenon may benefit from the nature of the

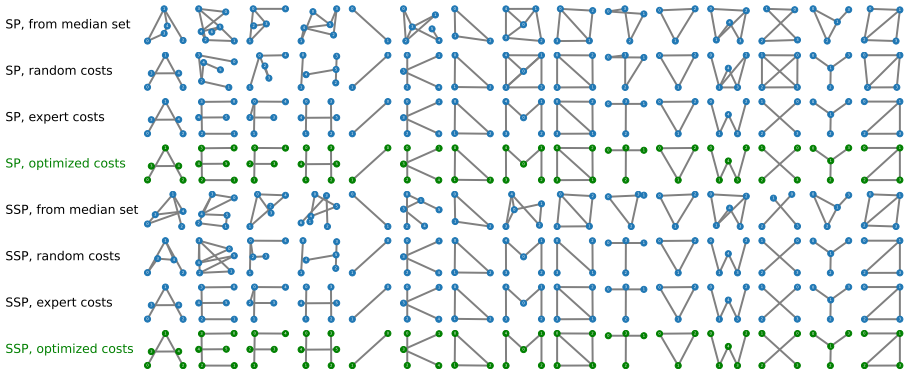

Fig. 2. Pre-images constructed by different algorithms for *Letter-high*, which correspond to the eight rows of Table 1 row by row.

IAM algorithm. In the update procedure (11), the new non-symbolic labels assigned for a vertex v is given by the average values of the corresponding labels of the vertices substituted to v [7]. It provides a "direction" to construct pre-images with respect to features and structures of graphs. For instance, the "x" and "y" attributes on the vertices of the letter graphs presents the coordinates of the vertices. To this end, it makes sense to compute their average values as the new values of a vertex as the vertex will be re-positioned at the middle of all vertices substituted to it.

5 Conclusion and Future Work

In this paper, we proposed a novel method to estimate graph pre-images. This approach is based on the hypothesis that metrics in both kernel space and graph space can be aligned. We first proposed a method to align GEDs to distances in the kernel space. Within the procedure, the edit costs are optimized. Then the graph pre-image was generated by a new method to construct the graph generalized median, where we revisited the IAM algorithm. Our method can generate better pre-images than other methods, as demonstrated on the *Letter-high* dataset. Future work includes generalizing our method to graphs with symbolic labels and constructing pre-images as arbitrary graphs rather than median graphs. The convergence proof of the iterative procedure will be conducted and the non-constant edit costs will be considered. Using state-of-the-art generative graph neural networks to solve the pre-image problem is also interesting.

References

1. Agrawal, A., Verschueren, R., Diamond, S., Boyd, S.: A rewriting system for convex optimization problems. Journal of Control and Decision **5**(1), 42–60 (2018)

2. Akutsu, T., Fukagawa, D.: Inferring a graph from path frequency. In: Annual Symposium on Combinatorial Pattern Matching. pp. 371–382. Springer (2005)
3. Bakır, G.H., Zien, A., Tsuda, K.: Learning to find graph pre-images. In: Joint Pattern Recognition Symposium. pp. 253–261. Springer (2004)
4. Blumenthal, D.B., Boria, N., Gamper, J., Bougleux, S., Brun, L.: Comparing heuristics for graph edit distance computation. The VLDB Journal pp. 1–40 (2019)
5. Blumenthal, D.B., Bougleux, S., Gamper, J., Brun, L.: Gedlib: A c++ library for graph edit distance computation. In: International Workshop on Graph-Based Representations in Pattern Recognition. pp. 14–24. Springer (2019)
6. Borgwardt, K.M., Kriegel, H.P.: Shortest-path kernels on graphs. In: Data Mining, Fifth IEEE International Conference on. pp. 8–pp. IEEE (2005)
7. 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)
8. Bougleux, S., Gaüzère, B., Brun, L.: Graph edit distance as a quadratic program. In: 2016 23rd International Conference on Pattern Recognition (ICPR). pp. 1701–1706 (Dec 2016). https://doi.org/10.1109/ICPR.2016.7899881
9. Daller, É., Bougleux, S., Gaüzère, B., Brun, L.: Approximate graph edit distance by several local searches in parallel. In: 7th International Conference on Pattern Recognition Applications and Methods (2018)
10. Diamond, S., Boyd, S.: CVXPY: A Python-embedded modeling language for convex optimization. Journal of Machine Learning Research **17**(83), 1–5 (2016)
11. Gärtner, T.: A survey of kernels for structured data. ACM SIGKDD Explorations Newsletter **5**(1), 49–58 (2003)
12. Gaüzère, B., Brun, L., Villemin, D.: Graph kernels in chemoinformatics. In: Dehmer, M., Emmert-Streib, F. (eds.) Quantitative Graph Theory Mathematical Foundations and Applications, pp. 425–470. CRC Press (2015), https://hal.archives-ouvertes.fr/hal-01201933
13. Ghosh, S., Das, N., Gonçalves, T., Quaresma, P., Kundu, M.: The journey of graph kernels through two decades. Computer Science Review **27**, 88–111 (2018)
14. Honeine, P.: Online kernel principal component analysis: a reduced-order model. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**(9), 1814 – 1826 (Sep 2012). https://doi.org/10.1109/TPAMI.2011.270
15. Honeine, P., Richard, C.: Preimage problem in kernel-based machine learning. IEEE Signal Processing Magazine **28**(2), 77–88 (2011)
16. Jia, L., Gaüzère, B., Honeine, P.: Graph Kernels Based on Linear Patterns: Theoretical and Experimental Comparisons (Mar 2019), https://hal-normandie-univ.archives-ouvertes.fr/hal-02053946, working paper or preprint
17. Kaspar, R., Horst, B.: Graph classification and clustering based on vector space embedding, vol. 77. World Scientific (2010)
18. Kriege, N.M., Neumann, M., Morris, C., Kersting, K., Mutzel, P.: A unifying view of explicit and implicit feature maps for structured data: systematic studies of graph kernels. arXiv preprint arXiv:1703.00676 (2017)
19. Nagamochi, H.: A detachment algorithm for inferring a graph from path frequency. Algorithmica **53**(2), 207–224 (2009)
20. Ralaivola, L., Swamidass, S.J., Saigo, H., Baldi, P.: Graph kernels for chemical informatics. Neural networks **18**(8), 1093–1110 (2005)
21. Riesen, K.: Structural pattern recognition with graph edit distance. In: Advances in computer vision and pattern recognition. Springer (2015)
22. West, D.B., et al.: Introduction to graph theory, vol. 2. Prentice hall Upper Saddle River (2001)