Quasimetric Graph Edit Distance as a Compact Quadratic Assignment Problem

David B. Blumenthal*, Évariste Daller†, Sébastien Bougleux†, Luc Brun†, and Johann Gamper*
*Free University of Bozen-Bolzano, Faculty of Computer Science, 39100 Bozen-Bolzano, Italy
†Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France

Presented at the 24th IAPR International Conference on Pattern Recognition (ICPR), Beijing, China
p. 934–939, IEEE, 2018 (https://doi.org/10.1109/ICPR.2018.8546055)

***Abstract*—The graph edit distance (GED) is a widely used distance measure for attributed graphs. It has recently been shown that the problem of computing GED, which is a NP-hard optimization problem, can be formulated as a quadratic assignment problem (QAP). This formulation is useful, since it allows to derive well performing approximative heuristics for GED from existing techniques for QAP. In this paper, we focus on the case where the edit costs that underlie GED are quasimetric. This is the case in many applications of GED. We show that, for quasimetric edit costs, it is possible to reduce the size of the corresponding QAP formulation. An empirical evaluation shows that this reduction significantly speeds up the QAP-based approximative heuristics for GED.**

I. Introduction

The graph edit distance is a very flexible distance measure for attributed graphs, which is widely used in the pattern recognition community. It can be viewed as the minimum cost that has to be paid for transforming one graph into another. In the entire paper, we consider (directed or undirected) attributed, simple, loopless graphs $G = (V^G, E^G)$ and $H = (V^H, E^H)$ and assume w. l. o. g. that $V^G = \{1, \dots, n\}$ and $V^H = \{1, \dots, m\}$. An *edit path* $P = (\mathsf{o}_i)_{i=1}^r$ between G and H is a sequence of edit operations that transforms G into H, i. e., satisfies $(\mathsf{o}_r \circ \dots \circ \mathsf{o}_1)(G) = H$. There are six edit operations, which are summarized in Table I. Each edit operation o comes with a positive edit cost $c(\mathsf{o})$.

TABLE I
Edit Operations and Edit Costs

edit operation o	edit cost $c(\mathsf{o})$
substitute node $i \in V^G$ with node $k \in V^H$	$c_V(i,k)$
remove isolated node i from V^G	$c_V(i,\epsilon)$
insert isolated node k into V^H	$c_V(\epsilon,k)$
substitute edge $(i,j) \in E^G$ with edge $(k,l) \in E^H$	$c_E((i,j),(k,l))$
remove edge (i,j) from E^G	$c_E((i,j),\epsilon)$
insert edge (k,l) between $i,k \in V^H$ into E^H	$c_E(\epsilon,(k,l))$

Note that the edit costs c_V and c_E exclusively depend on the node and edge attributes of G and H: For instance, if two nodes $i, j \in V^G$ have the same attribute, then $c_V(i,k) = c_V(j,k)$ holds for all $k \in V^H$. Therefore, c_V and c_E can be viewed as implicit encodings of the attributes of G and H.

Definition 1 (Graph Edit Distance): The *graph edit distance* (GED) between two graphs G and H is defined as

$$\mathrm{GED}(G,H) = \min_{P \in \Psi(G,H)} \sum_{\mathsf{o} \in P} c(\mathsf{o}),$$

where $\Psi(G,H)$ is the set of all edit paths between G and H.

Since the exact computation of GED is NP-hard [1], exact algorithms can only cope with very small graphs [2]–[6]. For this reason, research has mainly focused on the task of devising approximative heuristics that compute lower or upper bounds for GED. Established approximative approaches compute bounds for GED by using beam search [7], genetic algorithms [8], Hausdorff matching [9], local search [1], [10], [11], linear programming [6], [12], or transformations to variants of the linear sum assignment problem [1], [12]–[21].

Recently, a new paradigm for computing upper bounds for GED has been proposed [22]–[24]. Its backbone is a reduction of the problem of computing GED to the following version of the quadratic assignment problem:

Definition 2 (Quadratic Assignment Problem): Given a matrix $\mathbf{C} \in \mathbb{R}^{(N \cdot M) \times (N \cdot M)}$ with $N \leq M$, the *quadratic assignment problem* (QAP) asks to solve the problem

$$\operatorname*{arg\,min}_{\mathbf{X} \in \Pi_{N,M}} Q(\mathbf{X}, \mathbf{C}) \stackrel{\text{def.}}{=} \operatorname*{arg\,min}_{\mathbf{X} \in \Pi_{N,M}} \mathrm{vec}(\mathbf{X})^T \mathbf{C}\, \mathrm{vec}(\mathbf{X}),$$

where $\Pi_{N,M}$ is the set of maximal partial permutation matrices between $\{1, \dots, N\}$ and $\{1, \dots, M\}$, and vec is a matrix-vectorization operator. The cost $Q(\mathbf{X}^\star, \mathbf{C})$ of an optimal solution $\mathbf{X}^\star$ for $\mathbf{C}$ is denoted by $\mathrm{QAP}(\mathbf{C})$.

The new paradigm starts out by constructing a matrix $\mathbf{C} \in \mathbb{R}^{(n+m)^2 \times (n+m)^2}$ such that $\mathrm{QAP}(\mathbf{C}) = \mathrm{GED}(G,H)$. In a second step, state-of-the-art QAP-methods are run on $\mathbf{C}$. These methods then yield upper bounds for GED that are tighter than all other available upper bounds. However, they are quite slow, especially in comparison to the approaches based on the linear sum assignment problem. In order to tackle this problem, it has been shown that, alternatively, GED can be formulated as an instance $\mathbf{C}' \in \mathbb{R}^{(n+1)(m+1) \times (n+1)(m+1)}$ of the quadratic assignment problem with edition (QAPE), a variant of QAP where the assignment constraints for some nodes are relaxed [25]. Yet, the drawback of this approach is that some QAP-methods are not applicable to QAPE and that even those that

are have to be customized manually, which requires a thorough theoretical understanding on the side of the implementer.

In this paper, we alleviate these shortcomings for settings where the triangle inequalities

$$c_V(i,k) \leq c_V(i,\epsilon) + c_V(\epsilon,k) \tag{1}$$
$$c_E((i,j),(k,l)) \leq c_E((i,j),\epsilon) + c_E(\epsilon,(k,l)) \tag{2}$$

are satisfied for all $(i,k) \in V^G \times V^H$ and all $((i,j),(k,l)) \in E^G \times E^H$, i. e., for settings where the edge and node edit costs c_V and c_E are quasimetric. Note that (2) can be assumed to hold w. l. o. g.: Since edge substitutions can always be replaced by a removal and an insertion, we can enforce (2) by setting $c_E((i,j),(k,l)) = \min\{c_E((i,j),(k,l)), c_E((i,j),\epsilon) + c_E(\epsilon,(k,l))\}$. This is not true of (1), which effectively constrains the node edit costs c_V. However, (1) is met in many application scenarios [26], [27].

We show that, if (1) and (2) hold, GED can be reduced to an instance $\widehat{\mathbf{C}} \in \mathbb{R}^{(n\cdot m)\times(n\cdot m)}$ of QAP. Note that $\widehat{\mathbf{C}}$ is significantly smaller than the matrix $\mathbf{C} \in \mathbb{R}^{(n+m)^2\times(n+m)^2}$ employed by the baseline approach [22], [23] and slightly smaller than the instance $\mathbf{C}' \in \mathbb{R}^{(n+1)(m+1)\times(n+1)(m+1)}$ used by the QAPE-based reduction [25]. Furthermore, unlike the QAPE-based reduction, the reduction proposed in this paper allows the use of off-the-shelf QAP-methods for upper-bounding GED. An empirical evaluation shows that using $\widehat{\mathbf{C}}$ instead of $\mathbf{C}$ or $\mathbf{C}'$ significantly reduces the runtime of QAP-based methods for upper-bounding GED, while the tightness of the obtained upper bound deteriorates only slightly.

The remainder of this paper is organized as follows: In Section II, the QAP-formulation $\mathbf{C}$ of GED proposed in [22], [23] is summarized briefly and notations that are used throughout the paper are introduced. In Section III, it is shown how to construct the smaller formulation $\widehat{\mathbf{C}}$ for quasimetric edit costs. In Section IV, the results of an empirical evaluation of the effect of this size-reduction are reported. Section V concludes the paper.

II. A QAP-Formulation of GED

In this section, we summarize the original QAP-formulation of the problem of computing GED proposed in [22], [23]. Note that, for the sake of presentation, we slightly alter the notations at some points.

The original QAP-formulation works on enlarged graphs $(V^G \cup \mathcal{E}^G, E^G)$ and $(V^H \cup \mathcal{E}^H, E^H)$. $\mathcal{E}^G = \{\epsilon_1 = n+1, \ldots, \epsilon_m = n+m\}$ and $\mathcal{E}^H = \{\epsilon_1 = m+1, \ldots, \epsilon_n = m+n\}$ contain m and n isolated dummy nodes, respectively. Let $\mathbf{X} = (x_{i,k}) \in \Pi_{(n+m),(m+n)}$ be a permutation matrix. $\mathbf{X}$ can be written in the following form:

$$\mathbf{X} = \begin{array}{c} \\ V^G \\ \mathcal{E}^G \end{array} \begin{array}{c} \begin{array}{cc} V^H & \mathcal{E}^H \end{array} \\ \begin{bmatrix} \mathbf{X}^{\mathrm{s}} & \mathbf{X}^{\mathrm{r}} \\ \mathbf{X}^{\mathrm{i}} & \mathbf{X}^{0} \end{bmatrix} \end{array}$$

The north-west quadrant $\mathbf{X}^{\mathrm{s}}$ encodes node substitutions, the south-west quadrant $\mathbf{X}^{\mathrm{i}}$ encodes node insertion, the north-east quadrant $\mathbf{X}^{\mathrm{r}}$ encodes node removals, and the south-east quadrant $\mathbf{X}^0$ encodes assignments of dummy nodes to dummy nodes. Since nodes can be inserted and removed only once, assignments $x_{i,\epsilon_j} = 1$ and $x_{\epsilon_l,k} = 1$ with $i \neq j$ and $k \neq l$ are forbidden. The notation $i \nrightarrow k$ is used to denote that assigning $i \in V^{G+\epsilon}$ to $k \in V^{H+\epsilon}$ is forbidden, where $V^{G+\epsilon} = V^G \cup \mathcal{E}^G$ and $V^{H+\epsilon} = V^H \cup \mathcal{E}^H$.

Next, the matrices $\mathbf{C}_V, \mathbf{C}_E, \mathbf{C} \in \mathbb{R}^{(n+m)^2\times(n+m)^2}$ are constructed. For the ease of notation, we use two-dimensional indices $(i,k) \in V^{G+\epsilon} \times V^{H+\epsilon}$ for referring to their elements, and a special vectorization operator vec that first concatenates the columns of $V^G \times V^H$, then the columns of $\mathcal{E}^G \times V^H$, followed by the columns of $V^G \times \mathcal{E}^H$, and the columns of $\mathcal{E}^G \times \mathcal{E}^H$.

$\mathbf{C}_V = (c^V_{(i,k),(j,l)})$ contains the costs of the node edit operations. It is defined as

$$c^V_{(i,k),(j,l)} = \begin{cases} 0 & \text{if } i \neq j \vee k \neq l \vee i \nrightarrow k \\ c'_V(i,k) & \text{otherwise,} \end{cases} \tag{3}$$

where c'_V is defined as

$$\begin{aligned} c'_V(i,k) = {} & \delta_{i\in V^G}\delta_{k\in V^H} c_V(i,k) \\ & + (1-\delta_{i\in V^G})\delta_{k\in V^H} c_V(\epsilon,k) \\ & + \delta_{i\in V^G}(1-\delta_{k\in V^H}) c_V(i,\epsilon), \end{aligned} \tag{4}$$

and $\delta_{\texttt{true|false}}$ maps `true` to 1 and `false` to 0.

$\mathbf{C}_E = (c^E_{(i,k),(j,l)})$ contains the costs of the edge edit operations. It is defined as

$$c^E_{(i,k),(j,l)} = \begin{cases} \omega & \text{if } i \nrightarrow k \vee j \nrightarrow l \\ c'_E(i,k,j,l) & \text{otherwise,} \end{cases} \tag{5}$$

where $\omega = 1 + \sum_{i\in V^G} c_V(i,\epsilon) + \sum_{k\in V^H} c_V(\epsilon,k) + \sum_{(i,j)\in E^G} c_E((i,j),\epsilon) + \sum_{(k,l)\in E^H} c_E(\epsilon,(k,l))$ is a very large number and c'_E is defined as follows:

$$\begin{aligned} c'_E(i,k,j,l) = {} & \delta_{(i,j)\in E^G}\delta_{(k,l)\in E^H} c_E((i,j),(k,l)) \\ & + (1-\delta_{(i,j)\in E^G})\delta_{(k,l)\in E^H} c_E(\epsilon,(k,l)) \\ & + \delta_{(i,j)\in E^G}(1-\delta_{(k,l)\in E^H}) c_E((i,j),\epsilon) \end{aligned} \tag{6}$$

Finally, the QAP-formulation $\mathbf{C} = (c_{(i,k),(j,l)})$ of GED is defined as $\mathbf{C} = \frac{1}{2}\mathbf{C}_E + \mathbf{C}_V$, if G and H are undirected, and as $\mathbf{C} = \frac{1}{2}(\mathbf{C}_E + \mathbf{C}_E^T) + \mathbf{C}_V$, otherwise. It is shown that the following holds:

Theorem 1 (Cf. equation (16) in [23]): If, given two graphs G and H and positive edit costs c_V and c_E, the matrix $\mathbf{C} \in \mathbb{R}^{(n+m)^2\times(n+m)^2}$ is constructed as specified in the equations (3), (4), (5), and (6), then it holds that $\mathrm{GED}(G,H) = \mathrm{QAP}(\mathbf{C})$.

III. Reducing the Size of the QAP-Formulation

In this section, we show how to reduce the size of the QAP-formulation from $(n+m)^2 \times (m+n)^2$ to $(n\cdot m)\times(n\cdot m)$. We first state the main theorem which presents the reduction principle. This reduction principle assumes that the graphs are undirected. In order to avoid redundancies, we do not present the directed case, which is very similar and can easily be derived from the undirected one.

Theorem 2 (Reduction Principle): Given two graphs G and H and quasimetric edit costs c_V and c_E, let $\mathbf{C} \in \mathbb{R}^{(n+m)^2 \times (n+m)^2}$ be the matrix defined in the equations (3), (4), (5), and (6). Let $\widehat{\mathbf{C}} = (\widehat{c}_{(i,k),(j,l)}) \in \mathbb{R}^{(n\cdot m)\times(n\cdot m)}$ be defined as follows:

$$\begin{aligned}\widehat{c}_{(i,k),(j,l)} = c_{(i,k),(j,l)} - \frac{3}{2}\Big[&\delta_{n<m}\delta_{(k,l)\in E^H} c_E(\epsilon,(k,l)) \\ &+ \delta_{n>m}\delta_{(i,j)\in E^G} c_E((i,j),\epsilon)\Big] \\ &- \delta_{i=j}\delta_{k=l}\Big[\delta_{n<m} c_V(\epsilon,k) + \delta_{n>m} c_V(i,\epsilon)\Big]\end{aligned} \tag{7}$$

Then it holds that

$$\begin{aligned}\mathrm{QAP}(\mathbf{C}) = \mathrm{QAP}(\widehat{\mathbf{C}}) &\\ + \delta_{n<m}\Big[\sum_{(k,l)\in E^H} c_E(\epsilon,(k,l)) + \sum_{k\in V^H} c_V(\epsilon,k)\Big] &\\ + \delta_{n>m}\Big[\sum_{(i,j)\in E^G} c_E((i,j),\epsilon) + \sum_{i\in V^G} c_V(i,\epsilon)\Big],&\end{aligned} \tag{8}$$

which, by Theorem 1, implies that $\widehat{\mathbf{C}}$ is a QAP-formulation of GED.

Note that, by showing the correctness of the reduction principle, we show that, if the edit costs c_V and c_E are quasimetric, the following propositions hold:

- If $n \leq m$, then there is an optimal edit path between G and H that contains no node removals and hence no edge removals induced by node removals.
- If $n \geq m$, then there is an optimal edit path between G and H that contains no node insertions and hence no edge insertions induced by node insertions.

In Section III-A, we prove that the reduction principle stated in Theorem 2 is correct. In Section III-B, we explain how to turn it into a paradigm for (suboptimally) computing GED.

A. Proving the Correctness of the Reduction Principle

Throughout this section, we assume w. l. o. g. that $n \leq m$. The case $n \geq m$ is analogous. The first observation, of which we will make continuous use, is that, by applying the vectorization operator vec, the original QAP-formultation $\mathbf{C}$ can be rewritten in the following way:

$$\mathbf{C} = \begin{array}{c} \\ V^G\times V^H \\ \mathcal{E}^G\times V^H \\ V^G\times \mathcal{E}^H \\ \mathcal{E}^G\times \mathcal{E}^H \end{array} \begin{array}{c} \begin{array}{cccc} V^G\times V^H & \mathcal{E}^G\times V^H & V^G\times \mathcal{E}^H & \mathcal{E}^G\times \mathcal{E}^H \end{array} \\ \begin{bmatrix} \mathbf{C}^{\mathrm{ss}} & \mathbf{C}^{\mathrm{si}} & \mathbf{C}^{\mathrm{sr}} & \mathbf{C}^{\mathrm{s0}} \\ \mathbf{C}^{\mathrm{is}} & \mathbf{C}^{\mathrm{ii}} & \mathbf{C}^{\mathrm{ir}} & \mathbf{C}^{\mathrm{i0}} \\ \mathbf{C}^{\mathrm{rs}} & \mathbf{C}^{\mathrm{ri}} & \mathbf{C}^{\mathrm{rr}} & \mathbf{C}^{\mathrm{r0}} \\ \mathbf{C}^{\mathrm{0s}} & \mathbf{C}^{\mathrm{0i}} & \mathbf{C}^{\mathrm{0r}} & \mathbf{C}^{\mathrm{00}} \end{bmatrix} \end{array}$$

The first lemma tells us that, w. l. o. g., we can focus on permutation matrices without forbidden assignments.

Lemma 1: Let $\mathbf{X}^\star = (x^\star_{i,k}) \in \Pi_{(n+m),(m+n)}$ be optimal for $\mathbf{C}$. Then $x^\star_{i,k} = 0$ for all $i \nrightarrow k$, i. e., $\mathbf{X}^\star$ does not contain any forbidden assignments.

Proof: Let $\mathbf{X} = (x_{i,k}) \in \{0,1\}^{(n+m)\times(n+m)}$ be defined as $x_{i,k} = \delta_{k=\epsilon_i} + \delta_{i=\epsilon_k}$. It is straightforward to see that we have $\mathbf{X} \in \Pi_{(n+m),(m+n)}$ and that $Q(\mathbf{X},\mathbf{C}) = \omega - 1$. Assume now that there is a forbidden assignment $i \nrightarrow k$ such that $x^\star_{i,k} = 1$. Then we have, $Q(\mathbf{X}^\star,\mathbf{C}) \geq c_{(i,k),(i,k)} x^\star_{i,k} x^\star_{i,k} = \omega$. ∎

Next, we observe that some parts of $\mathbf{C}$ can be ignored when computing the quadratic cost of permutation matrices without forbidden assignments.

Lemma 2: Let $\mathbf{X} \in \Pi_{(n+m),(m+n)}$ be a permutation matrix without forbidden assignments. Then its quadratic cost $Q(\mathbf{X},\mathbf{C})$ can be rewritten as follows:

$$\begin{aligned}Q(\mathbf{X},\mathbf{C}) = &\operatorname{vec}(\mathbf{X}^{\mathrm{s}})^T \mathbf{C}^{\mathrm{ss}} \operatorname{vec}(\mathbf{X}^{\mathrm{s}}) \\ &+ \operatorname{vec}(\mathbf{X}^{\mathrm{s}})^T \mathbf{C}^{\mathrm{si}} \operatorname{vec}(\mathbf{X}^{\mathrm{i}}) + \operatorname{vec}(\mathbf{X}^{\mathrm{s}})^T \mathbf{C}^{\mathrm{sr}} \operatorname{vec}(\mathbf{X}^{\mathrm{r}}) \\ &+ \operatorname{vec}(\mathbf{X}^{\mathrm{i}})^T \mathbf{C}^{\mathrm{is}} \operatorname{vec}(\mathbf{X}^{\mathrm{s}}) + \operatorname{vec}(\mathbf{X}^{\mathrm{i}})^T \mathbf{C}^{\mathrm{ii}} \operatorname{vec}(\mathbf{X}^{\mathrm{i}}) \\ &+ \operatorname{vec}(\mathbf{X}^{\mathrm{r}})^T \mathbf{C}^{\mathrm{rs}} \operatorname{vec}(\mathbf{X}^{\mathrm{s}}) + \operatorname{vec}(\mathbf{X}^{\mathrm{r}})^T \mathbf{C}^{\mathrm{rr}} \operatorname{vec}(\mathbf{X}^{\mathrm{r}})\end{aligned}$$

Proof: The lemma immediately follows from the construction of $\mathbf{C}$. ∎

We now construct a function f that maps a partial permutation matrix $\widehat{\mathbf{X}} = (\widehat{x}_{i,k}) \in \Pi_{n,m}$ for $\widehat{\mathbf{C}}$ to a permutation matrix for $\mathbf{C}$ without node removals. For dummy nodes $(\epsilon_k,\epsilon_i) \in \mathcal{E}^G \times \mathcal{E}^H$, we introduce the notation $\epsilon_k \xrightarrow{\widehat{\mathbf{X}}} \epsilon_i$ to denote the condition that k is the i^{th} node in V^H to which $\widehat{\mathbf{X}}$ assigns a node from V^G. The mapping f is defined as follows:

$$f(\widehat{\mathbf{X}})_{i,k} = \begin{cases} \widehat{x}_{i,k} & \text{if } (i,k) \in V^G \times V^H \\ 1 - \sum_{j\in V^G} \widehat{x}_{j,k} & \text{if } i = \epsilon_k \\ 1 & \text{if } (i,k) \in \mathcal{E}^G \times \mathcal{E}^H \wedge i \xrightarrow{\widehat{\mathbf{X}}} k \\ 0 & \text{otherwise} \end{cases}$$

Lemma 3: For each $\widehat{\mathbf{X}} \in \Pi_{n,m}$, it holds that $f(\widehat{\mathbf{X}}) \in \Pi_{n+m,m+n}$, that $f(\widehat{\mathbf{X}})^{\mathrm{r}} = \mathbf{0}_{n\times n}$, and that $f(\widehat{\mathbf{X}})_{i,k} = 0$ for all $i \nrightarrow k$.

Proof: For proving the first part of the lemma, note that, since $\widehat{\mathbf{X}} \in \Pi_{n,m}$ and $n \leq m$, the first two lines in the definition of f ensure that $f(\widehat{\mathbf{X}})$ covers all rows in V^G, all columns in V^H, and leaves exactly n rows in $\mathcal{E}^G$ uncovered. These rows as well as all columns in $\mathcal{E}^H$ are covered by the third line of the definition of f. The second and the third part of the lemma immediately follow from the definition of f. ∎

The next lemma shows that restricting to permutation matrices without node removals and restricting to permutation matrices that are contained in $\mathrm{img}(f)$ is equivalent.

Lemma 4: Let $\mathbf{X} \in \Pi_{(n+m),(m+n)}$ be a permutation matrix without forbidden assignments that satisfies $\mathbf{X}^{\mathrm{r}} = \mathbf{0}_{n\times n}$. Then there is a permutation matrix $\mathbf{X}' \in \mathrm{img}(f)$ such that $Q(\mathbf{X},\mathbf{C}) = Q(\mathbf{X}',\mathbf{C})$.

Proof: Since $\mathbf{X}^{\mathrm{r}} = \mathbf{0}_{n\times n}$ and $\mathbf{X} \in \Pi_{(n+m),(m+n)}$, we know that $\mathbf{X}^{\mathrm{s}} \in \Pi_{n,m}$. This allows us to define $\mathbf{X}' = f(\mathbf{X}^{\mathrm{s}})$. Since $\mathbf{X}$ and $\mathbf{X}'$ do not contain forbidden assignments, we know from Lemma 2 that $\mathbf{X}^0$ and $\mathbf{X}'^0$ do not contribute to $Q(\mathbf{X},\mathbf{C})$ and $Q(\mathbf{X}',\mathbf{C})$, respectively. Furthermore, we have $\mathbf{X}^{\mathrm{s}} = \mathbf{X}'^{\mathrm{s}}$ and $\mathbf{X}^{\mathrm{r}} = \mathbf{0}_{n\times n} = \mathbf{X}'^{\mathrm{r}}$ by construction. The lemma thus follows if we can show that $\mathbf{X}^{\mathrm{i}} = \mathbf{X}'^{\mathrm{i}}$. So let $(i,k) \in \mathcal{E}^G \times V^H$. If $i \neq \epsilon_k$, we have $i \nrightarrow k$ and hence $x'_{i,k} = x_{i,k} = 0$. Otherwise, it holds that $x'_{i,k} = 1 - \sum_{j\in V^G} x_{i,j} = x_{i,k}$, where the last equality follows from $\sum_{j\in V^{G+\epsilon}} x_{j,k} = 1$ and $x_{j,k} = 0$ for all $j \in \mathcal{E}^G$ with $j \neq \epsilon_k$. ∎

Next, we show that, for quasimetric edit costs, it suffices to optimize over the permutation matrices contained in $\mathrm{img}(f)$.

Lemma 5: If the edit costs c_V and c_E are quasimetric, then it holds that $\mathrm{QAP}(\mathbf{C}) = \min_{\mathbf{X}\in\mathrm{img}(f)} Q(\mathbf{X},\mathbf{C})$.

Proof: Because of Lemma 1, Lemma 3, and Lemma 4, it suffices to show that, for each permutation matrix $\mathbf{X} \in \Pi_{(n+m),(m+n)}$ without forbidden assignments and $r > 0$ node removals, i. e., $|\operatorname{supp}(\mathbf{X}^{\mathrm{r}})| = r$, there is a permutation matrix $\mathbf{X}' \in \Pi_{(n+m),(m+n)}$ without forbidden assignments and $r-1$ node removals such that $Q(\mathbf{X}',\mathbf{C}) \leq Q(\mathbf{X},\mathbf{C})$. So assume that $\mathbf{X} \in \Pi_{(n+m),(m+n)}$ contains no forbidden assignments and the node removal $x_{i,\epsilon_i} = 1$ for some node $i \in V^G$. Since $\mathbf{X}$ is a permutation matrix without forbidden assignment and $n \leq m$, we then know that $\mathbf{X}$ also contains a node insertion $x_{\epsilon_k,k} = 1$ for some node $k \in V^H$.

We now define the matrix $\mathbf{X}' = (x'_{j,l}) \in \Pi_{(n+m),(m+n)}$ as

$$x'_{j,l} = \begin{cases} 1 & \text{if } (j,l) = (i,k) \vee (j,l) = (\epsilon_k,\epsilon_i) \\ 0 & \text{if } (j,l) = (i,\epsilon_i) \vee (j,l) = (\epsilon_k,k) \\ x_{j,l} & \text{otherwise,} \end{cases}$$

and introduce $\Delta = Q(\mathbf{X},\mathbf{C}) - Q(\mathbf{X}',\mathbf{C})$. Since $\mathbf{X}' \in \Pi_{(n+m),(m+n)}$ is immediately implied by $\mathbf{X} \in \Pi_{(n+m),(m+n)}$, the lemma follows if we can show that $\Delta \geq 0$.

Let $I = \{(i,k),(i,\epsilon_i),(\epsilon_k,k),(\epsilon_k,\epsilon_i)\}$ be the set of all indices $(j,l) \in V^{G+\epsilon} \times V^{H+\epsilon}$ with $x_{j,l} \neq x'_{j,l}$. It is easy to see that we have $c_{(j,l),(j',l')}x_{j,l}x_{j',l'} = c_{(j,l),(j',l')}x'_{j,l}x'_{j',l'} = 0$ for all $((j,l),(j',l')) \in I \times I \setminus \{((i,k),(i,k)),((i,\epsilon_i),(i,\epsilon_i)),((\epsilon_k,k),(\epsilon_k,k))\}$. For this reason, since $\mathbf{C}$ is symmetric, and $\mathbf{X}$ does not contain forbidden assignments, we can write Δ as

$$\Delta = \Delta_V + \sum_{(j,l)\in(V^{G+\epsilon}\times V^{H+\epsilon})\setminus(I\cup F)} \Delta_{j,l}x_{j,l},$$

where $F = \{(j,l) \in V^{G+\epsilon} \times V^{H+\epsilon} \mid j \nrightarrow l\}$ is the set of all forbidden assignments and

$$\Delta_V = c_{(i,\epsilon_i),(i,\epsilon_i)} + c_{(\epsilon_k,k),(\epsilon_k,k)} - c_{(i,k),(i,k)}$$
$$\Delta_{j,l} = 2(c_{(i,\epsilon_i),(j,l)} + c_{(\epsilon_k,k),(j,l)} - c_{(i,k),(j,l)} - c_{(\epsilon_k,\epsilon_i),(j,l)}).$$

By definition of $\mathbf{C}$ and c'_V given in (4), we have $\Delta_V = c_V(i,\epsilon)+c_V(\epsilon,k)-c_V(i,k) \geq 0$, where the inequality follows from the fact that c_V is quasimetric. Similarly, the definition of $\mathbf{C}$ and c'_E given in (5) and the fact that $(j,l) \notin F$ gives us $\Delta_{j,l} = \delta_{(i,j)\in E^G}\delta_{(k,l)\in E^H}[c_E((i,j),\epsilon) + c_E(\epsilon,(k,l)) - c_E((i,j),(k,l))] \geq 0$ for all $(j,l) \in (V^{G+\epsilon}\times V^{H+\epsilon})\setminus(I\cup F)$, where the inequality follows from c_E being quasimetric. ∎

The next lemma simplifies the quadratic cost $Q(\mathbf{X},\mathbf{C})$ for permutation matrices $\mathbf{X} \in \mathrm{img}(f)$.

Lemma 6: The quadratic cost $Q(\mathbf{X},\mathbf{C})$ of a permutation matrix $\mathbf{X} \in \mathrm{img}(f)$ can be written as follows:

$$\begin{aligned} &Q(\mathbf{X},\mathbf{C}) \\ &= \mathrm{vec}(\mathbf{X}^{\mathrm{s}})^T\mathbf{C}^{\mathrm{ss}}\,\mathrm{vec}(\mathbf{X}^{\mathrm{s}}) \\ &\quad + \sum_{i\in V^G}\sum_{k\in V^H}\sum_{l\in V^H}\frac{\delta_{(k,l)\in E^H}}{2}c_E(\epsilon,(k,l))x_{i,k}x_{\epsilon_l,l} && (9) \\ &\quad + \sum_{k\in V^H}\sum_{j\in V^G}\sum_{l\in V^H}\frac{\delta_{(k,l)\in E^H}}{2}c_E(\epsilon,(k,l))x_{\epsilon_k,k}x_{j,l} && (10) \\ &\quad + \sum_{k\in V^H}\sum_{l\in V^H}\frac{\delta_{(k,l)\in E^H}}{2}c_E(\epsilon,(k,l))x_{\epsilon_k,k}x_{\epsilon_l,l} && (11) \\ &\quad + \sum_{k\in V^H} c_V(\epsilon,k)x_{\epsilon_k,k} && (12) \end{aligned}$$

Proof: By Lemma 3, $\mathbf{X}$ does not contain forbidden assignments, which implies $\mathrm{vec}(\mathbf{X}^{\mathrm{s}})^T\mathbf{C}^{\mathrm{si}}\,\mathrm{vec}(\mathbf{X}^{\mathrm{i}}) =$ (9), $\mathrm{vec}(\mathbf{X}^{\mathrm{i}})^T\mathbf{C}^{\mathrm{is}}\,\mathrm{vec}(\mathbf{X}^{\mathrm{s}}) =$ (10), and $\mathrm{vec}(\mathbf{X}^{\mathrm{i}})^T\mathbf{C}^{\mathrm{ii}}\,\mathrm{vec}(\mathbf{X}^{\mathrm{i}}) =$ (11)+(12). Furthermore, we have $\mathbf{X}^{\mathrm{r}} = \mathbf{0}_{n\times n}$ from Lemma 3. Therefore, the lemma follows from Lemma 2. ∎

The next step is to relate the cost of a partial permutation matrix $\widehat{\mathbf{X}} \in \Pi_{n,m}$ to the cost of its image under f.

Lemma 7: For each $\widehat{\mathbf{X}} \in \Pi_{n,m}$, it holds that $Q(f(\widehat{\mathbf{X}}),\mathbf{C}) = Q(\widehat{\mathbf{X}},\widehat{\mathbf{C}}) + \delta_{n<m}[\sum_{(k,l)\in E^H} c_E(\epsilon,(k,l)) + \sum_{k\in V^H} c_V(\epsilon,k)]$.

Proof: Let $\mathbf{X} = f(\widehat{\mathbf{X}})$ and $c_{k,l} = \frac{\delta_{(k,l)\in E^H}}{2}c_E(\epsilon,(k,l))$. If $n = m$, we obtain $\mathbf{X}^{\mathrm{i}} = \mathbf{0}_{m\times m}$ from Lemma 3 and the definition of $\Pi_{n+m,m+n}$. Since $\widehat{\mathbf{C}} = \mathbf{C}^{\mathrm{ss}}$, this implies the statement of the lemma. So we can focus on the case $n < m$. By definition of $\widehat{\mathbf{C}}$, $Q(\widehat{\mathbf{X}},\widehat{\mathbf{C}})$ can be written as follows:

$$\begin{aligned} &Q(\widehat{\mathbf{X}},\widehat{\mathbf{C}}) \\ &= \sum_{i\in V^G}\sum_{k\in V^H}\sum_{j\in V^G}\sum_{l\in V^H} c_{(i,k),(j,l)}\widehat{x}_{i,k}\widehat{x}_{j,l} && (13) \\ &\quad - \sum_{i\in V^G}\sum_{k\in V^H}\sum_{l\in V^H} c_{k,l}\widehat{x}_{i,k}\sum_{j\in V^G}\widehat{x}_{j,l} && (14) \\ &\quad - \sum_{k\in V^H}\sum_{j\in V^G}\sum_{l\in V^H} c_{k,l}\widehat{x}_{j,l}\sum_{i\in V^G}\widehat{x}_{i,k} && (15) \\ &\quad - \sum_{k\in V^H}\sum_{l\in V^H} c_{k,l}\sum_{i\in V^G}\widehat{x}_{i,k}\sum_{j\in V^G}\widehat{x}_{j,l} && (16) \\ &\quad - \sum_{k\in V^H} c_V(\epsilon,k)\sum_{i\in V^G}\widehat{x}_{i,k} && (17) \end{aligned}$$

Since $\widehat{x}_{i,k} = x_{i,k}$ for all $(i,k) \in V^G \times V^H$, it holds that:

$$(13) = \mathrm{vec}(\mathbf{X}^{\mathrm{s}})^T\mathbf{C}^{\mathrm{ss}}\,\mathrm{vec}(\mathbf{X}^{\mathrm{s}}) \quad (18)$$

Furthermore, by substituting $-\sum_{j\in V^G}\widehat{x}_{j,l}$ with $1 - \sum_{j\in V^G}\widehat{x}_{j,l} + 1 = x_{\epsilon_l,l} - 1$ in (14) and (16) and substituting $-\sum_{i\in V^G}\widehat{x}_{i,k}$ with $1 - \sum_{i\in V^G}\widehat{x}_{i,k} + 1 = x_{\epsilon_k,k} - 1$ in (15), (16), and (17), we obtain the following equalities:

$$\begin{aligned} (14) &= (9) - \sum_{i\in V^G}\sum_{k\in V^H}\sum_{l\in V^H} c_{k,l}\widehat{x}_{i,k} && (19) \\ (15) &= (10) - \sum_{k\in V^H}\sum_{j\in V^G}\sum_{l\in V^H} c_{k,l}\widehat{x}_{j,l} && (20) \\ (16) &= (11) + \sum_{i\in V^G}\sum_{k\in V^H}\sum_{l\in V^H} c_{k,l}\widehat{x}_{i,k} && (21) \\ &\quad + \sum_{k\in V^H}\sum_{j\in V^G}\sum_{l\in V^H} c_{k,l}\widehat{x}_{j,l} - \sum_{k\in V^H}\sum_{l\in V^H} c_{k,l} \\ (17) &= (12) - \sum_{k\in V^H} c_V(\epsilon,k) && (22) \end{aligned}$$

Since $\sum_{k\in V^H}\sum_{l\in V^H} c_{k,l} = \sum_{(k,l)\in E^H} c_E(\epsilon,(k,l))$, the lemma follows from summing the equalities (18), (20), (19), (21), and (22) and applying Lemma 6. ∎

We are now in the position to prove the main theorem.

Proof of Theorem 2: Let $\widehat{\mathbf{X}}^\star \in \Pi_{n,m}$ be optimal for $\widehat{\mathbf{C}}$, i. e., $Q(\widehat{\mathbf{X}}^\star,\widehat{\mathbf{C}}) = \mathrm{QAP}(\widehat{\mathbf{C}})$. Then we know from Lemma 7 that $Q(f(\widehat{\mathbf{X}}^\star),\mathbf{C}) = \mathrm{QAP}(\widehat{\mathbf{C}}) + \sum_{(k,l)\in E^H} c_E(\epsilon,(k,l)) + \sum_{k\in V^H} c_V(\epsilon,k)$, which implies $\mathrm{QAP}(\mathbf{C}) \leq \mathrm{QAP}(\widehat{\mathbf{C}}) + \sum_{(k,l)\in E^H} c_E(\epsilon,(k,l)) + \sum_{k\in V^H} c_V(\epsilon,k)$. On the other hand, we know from Lemma 5 that there is a permutation matrix $\mathbf{X}^\star \in \mathrm{img}(f)$ such that $Q(\mathbf{X}^\star,\mathbf{C}) = \mathrm{QAP}(\mathbf{C})$. Let $\widehat{\mathbf{X}} \in \Pi_{n,m}$ such that $f(\widehat{\mathbf{X}}) = \mathbf{X}^\star$. Then Lemma 7 tells us that $\mathrm{QAP}(\mathbf{C}) = Q(\widehat{\mathbf{X}},\widehat{\mathbf{C}}) + \delta_{n<m}[\sum_{(k,l)\in E^H} c_E(\epsilon,(k,l)) + \sum_{k\in V^H} c_V(\epsilon,k)]$. Since $n \leq m$, this proves the theorem. ∎

B. Turning the Reduction Principle into a GED-Paradigm

Given the definition of $\widehat{\mathbf{C}}$ in Theorem 2, it might well happen that $\widehat{c}_{(i,k),(j,l)} < 0$ for some $(i,j) \in V^G \times V^G$, $(k,l) \in V^H \times V^H$. However, some QAP-methods only work on non-negative cost matrices. In order to render these methods applicable to $\widehat{\mathbf{C}}$, $\widehat{\mathbf{C}}$ can be transformed into a non-negative cost matrix $\widehat{\mathbf{C}}^+ = (\widehat{c}^+_{(i,k),(j,l)}) \in \mathbb{R}^{(n\cdot m)\times(n\cdot m)}$ defined as

$$\widehat{c}^+_{(i,k),(j,l)} = \widehat{c}_{(i,k),(j,l)} + c, \tag{23}$$

where $c = \min\{\widehat{c}_{(i,k),(j,l)} \mid (i,j) \in V^G \times V^G \wedge (k,l) \in V^H \times V^H\}$. Clearly a partial permutation matrix $\mathbf{X} \in \Pi_{n,m}$ is optimal for $\widehat{\mathbf{C}}$ just in case it is optimal for $\widehat{\mathbf{C}}^+$, as

$$Q(\widehat{\mathbf{C}},\mathbf{X}) = Q(\widehat{\mathbf{C}}^+,\mathbf{X}) - \min\{n,m\}^2 c \tag{24}$$

holds for all $\mathbf{X} \in \Pi_{n,m}$.

We are now in the position to state how to use Theorem 2 as a paradigm for (suboptimally) computing GED:

1) Construct compact QAP-instance $\widehat{\mathbf{C}}$ according to equation (7).
2) Select QAP-method `M` for (suboptimally) solving $\widehat{\mathbf{C}}$.
3) If required by `M`, render $\widehat{\mathbf{C}}$ non-negative by applying equation (23).
4) Run `M` on $\widehat{\mathbf{C}}$ to obtain feasible solution and/or (suboptimal) assignment cost for $\widehat{\mathbf{C}}$.
5) If `M` yields feasible solution, apply the function f to obtain feasible edit path between G and H.
6) If `M` yields assignment cost, apply equation (8) (and equation (24), if $\widehat{\mathbf{C}}$ was positivized in step 3) to obtain (suboptimal) edit distance for G and H.

IV. Empirical Evaluation

A. Compared Methods

To empirically test the proposed compact QAP-formulation $\widehat{\mathbf{C}}$, we evaluated how employing $\widehat{\mathbf{C}}$ instead of the baseline formulation $\mathbf{C}$ [22], [23] and the QAPE-based formulation $\mathbf{C}'$ [25] affects the performance of `IPFP` with multistart (`mIPFP`) proposed in [24], which is the best performing currently available QAP-based method for upper-bounding GED. The algorithm `mIPFP` builds upon the algorithm `IPFP` [22], [23], [25], which is an adaptation of the Frank-Wolfe algorithm to the context of GED.

Given a cost matrix $\overline{\mathbf{C}}$, a set of feasible integral solutions Π with continuous relaxation Π', and a (possibly continuous) initial feasible solution $\mathbf{X} \in \Pi'$, `IPFP` starts by solving the following linear minimization problem:

$$\mathbf{B}^\star = \arg\min_{\mathbf{B}\in\Pi} \mathrm{vec}(\mathbf{X})^T \overline{\mathbf{C}}\,\mathrm{vec}(\mathbf{B}) \tag{25}$$

If $\overline{\mathbf{C}} = \mathbf{C}$ or $\overline{\mathbf{C}} = \widehat{\mathbf{C}}$, solving (25) amounts to solving a linear sum assignment problem, whereas, if $\overline{\mathbf{C}} = \mathbf{C}'$, it amounts to solving a linear sum assignment problem with error-correction. In the next step, the optimal step-width $\alpha^\star = \arg\min_{\alpha\in[0,1]} Q(\mathbf{X} + \alpha(\mathbf{B}^\star - \mathbf{X}))$ is determined analytically, and $\mathbf{X}$ is updated to $\mathbf{X} + \alpha^\star(\mathbf{B}^\star - \mathbf{X}) \in \Pi'$. The process terminates once $|(Q(\mathbf{X}^{\mathrm{old}},\overline{\mathbf{C}}) - \mathrm{vec}(\mathbf{X}^{\mathrm{old}})^T\overline{\mathbf{C}}\,\mathrm{vec}(\mathbf{B}^\star))/Q(\mathbf{X}^{\mathrm{old}},\overline{\mathbf{C}})| < \beta$ for a given convergence threshold $\beta \in [0,1]$ or a maximum number of iterations I is reached. Upon termination, $\mathbf{X}$ is projected to an integral solution $\mathbf{B}^\star \in \Pi$ by once more solving the linear problem (25), and $Q(\mathbf{B}^\star,\overline{\mathbf{C}})$ is returned as an upper bound for GED. The algorithm `mIPFP` extends `IPFP` by starting with k heuristically computed initial feasible solutions, running `IPFP` on each of them (possibly in parallel), and returning the best upper bound. In the following, we use the expressions `B-QAP-mIPFP`, `QAPE-mIPFP`, and `C-QAP-mIPFP` to denote `mIPFP` with $\overline{\mathbf{C}}$ set to the baseline QAP-formulation $\mathbf{C}$, the QAPE-formulation $\mathbf{C}'$, and the compact QAP-formulation $\widehat{\mathbf{C}}$ proposed in this paper, respectively.

B. Experimental Setup and Datasets

For our experiments, we set up `mIPFP` with $I = 100$, $\beta = 10^{-5}$, and randomly drew $k = 40$ initial feasible integral solutions from Π. Experiments were run on the chemoinformatics datasets Alkane, Acyclic, MAO, and PAH used in [22]–[25],[1] which naturally induce quasimetric edit costs (cf. Table 2, Setting 1 in [28]). We recorded the mean upper bound for GED (d), the mean runtime in seconds (t), and the mean error w. r. t. the exact GED (e), which we computed with the standard exact algorithm `A*` [2]. All methods were implemented in C++ and experiments were executed on a Linux Ubuntu machine with 512 GB of main memory and four AMD Opteron processors with 2.6 GHz and 64 cores, four of which where used to run `mIPFP` in parallel.

C. Results of the Experiments

Table IV-C shows the results of our experiments. Note that the graphs contained in MAO and PAH are too large to allow for an exact computation of GED, and so, for these datasets, no mean errors are reported. We see that, on all datasets expect for MAO, `C-QAP-mIPFP` is significantly faster than both `B-QAP-mIPFP` and `QAPE-mIPFP`. On MAO, `C-QAP-mIPFP` is still faster than `B-QAP-mIPFP` but slower than `QAPE-mIPFP`.

In terms of accuracy, on the datasets Alkane, Acyclic, and PAH, `C-QAP-mIPFP` performs only marginally worse than

[1] Available at https://iapr-tc15.greyc.fr/links.html.

TABLE II
EFFECT OF DIFFERENT QAP- AND QAPE-FORMULATIONS ON PERFORMANCE OF mIPFP

algorithm	d	e	t	d	e	t	d	e	t	d	e	t
	Alkane			Acyclic			MAO			PAH		
B-QAP-mIPFP [22], [23]	15.37	0.023	0.41	16.77	0.035	0.24	33.4	–	2.9	36.7	–	3.14
QAPE-mIPFP [25]	15.34	0.009	0.22	16.73	0.0076	0.13	33.3	–	0.8	36.6	–	1.17
C-QAP-mIPFP [this paper]	15.39	0.062	0.15	16.81	0.079	0.06	39.7	–	1.5	36.7	–	0.89

B-QAP-mIPFP and QAPE-mIPFP. Furthermore, we see from the results for Alkane and Acyclic that all three algorithms return an upper bound which is very close to the exact GED. The only exception is again MAO, where the upper bound returned by C-QAP-mIPFP is around 20 % looser than the ones returned by B-QAP-mIPFP and QAPE-mIPFP. The slight accuracy loss of C-QAP-mIPFP w. r. t. B-QAP-mIPFP and QAPE-mIPFP can be explained by the fact that, since $\widehat{\mathbf{C}}$ is denser that $\mathbf{C}$ and $\mathbf{C}'$, C-QAP-mIPFP reached the maximum number of iterations I before reaching the convergence threshold β more often than the other two algorithms.

V. CONCLUSIONS AND FUTURE WORK

In this paper, a compact QAP-formulation $\widehat{\mathbf{C}}$ of GED with quasimetric edit costs is proposed. Experiments show that running the state-of-the-art algorithm mIPFP with $\widehat{\mathbf{C}}$ instead of the baseline formulation $\mathbf{C}$ leads to a speed-up by a factor between 2 and 4, while the accuracy loss is negligible on most datasets. In comparison to the QAPE-formulation $\mathbf{C}'$, the speed-up obtained by using $\widehat{\mathbf{C}}$ is smaller. However, implementing mIPFP with $\widehat{\mathbf{C}}$ is much easier than implementing it with $\mathbf{C}'$: For implementing mIPFP with $\widehat{\mathbf{C}}$, one can use an off-the-shelf solver for the linear sum assignment problem; for implementing it with $\mathbf{C}'$, one has to implement a solver for the linear sum assignment problem with error-correction.

In a future work, we will evaluate more extensively how using the different formulations of GED affects the performance of QAP-based methods. For instance, we will systematically vary the parameters I, β, and k and test how B-QAP-mIPFP, QAPE-mIPFP, and C-QAP-mIPFP behave if different initialization techniques are employed.

REFERENCES

[1] Z. Zeng, A. K. H. Tung, J. Wang, J. Feng, and L. Zhou, "Comparing stars: On approximating graph edit distance," *PVLDB*, vol. 2, no. 1, pp. 25–36, 2009.

[2] K. Riesen, S. Fankhauser, and H. Bunke, "Speeding up graph edit distance computation with a bipartite heuristic," in *MLG*, 2007, pp. 21–24.

[3] Z. Abu-Aisheh, R. Raveaux, J.-Y. Ramel, and P. Martineau, "An exact graph edit distance algorithm for solving pattern recognition problems," in *ICPRAM*, 2015, pp. 271–278.

[4] D. B. Blumenthal and J. Gamper, "Exact computation of graph edit distance for uniform and non-uniform metric edit costs," in *GbRPR*, 2017, pp. 211–221.

[5] K. Gouda and M. Hassaan, "CSI_GED: An efficient approach for graph edit similarity computation," in *ICDE*, 2016, pp. 265–276.

[6] J. Lerouge, Z. Abu-Aisheh, R. Raveaux, P. Héroux, and S. Adam, "New binary linear programming formulation to compute the graph edit distance," *Pattern Recogn.*, vol. 72, pp. 254–265, 2017.

[7] M. Neuhaus, K. Riesen, and H. Bunke, "Fast suboptimal algorithms for the computation of graph edit distance," in *S+SSPR*, 2006, pp. 163–172.

[8] K. Riesen, A. Fischer, and H. Bunke, "Improving approximate graph edit distance using genetic algorithms," in *S+SSPR*, 2014, pp. 63–72.

[9] A. Fischer, C. Y. Suen, V. Frinken, K. Riesen, and H. Bunke, "Approximation of graph edit distance based on Hausdorff matching," *Pattern Recogn.*, vol. 48, no. 2, pp. 331–343, 2015.

[10] K. Riesen, A. Fischer, and H. Bunke, "Combining bipartite graph matching and beam search for graph edit distance approximation," in *ANNPR*, 2014, pp. 117–128.

[11] M. Ferrer, F. Serratosa, and K. Riesen, "Learning heuristics to reduce the overestimation of bipartite graph edit distance approximation," in *MLDM*, 2015, pp. 17–31.

[12] D. Justice and A. Hero, "A binary linear programming formulation of the graph edit distance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 8, pp. 1200–1214, 2006.

[13] S. Bougleux, B. Gaüzère, D. B. Blumenthal, and L. Brun, "Fast linear sum assignment with error-correction and no cost constraints," *Pattern Recogn. Lett.*, 2018, in press.

[14] K. Riesen and H. Bunke, "Approximate graph edit distance computation by means of bipartite graph matching," *Image Vis. Comput.*, vol. 27, no. 7, pp. 950–959, 2009.

[15] D. B. Blumenthal and J. Gamper, "Correcting and speeding-up bounds for non-uniform graph edit distance," in *ICDE*, 2017, pp. 131–134.

[16] ——, "Improved lower bounds for graph edit distance," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 3, pp. 503–516, 2018.

[17] W. Zheng, L. Zou, X. Lian, D. Wang, and D. Zhao, "Efficient Graph Similarity Search Over Large Graph Databases," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 4, pp. 964–978, 2015.

[18] B. Gauzère, S. Bougleux, K. Riesen, and L. Brun, "Approximate graph edit distance guided by bipartite matching of bags of walks," in *S+SSPR*, 2014, pp. 73–82.

[19] V. Carletti, B. Gaüzère, L. Brun, and M. Vento, "Approximate graph edit distance computation combining bipartite matching and exact neighborhood substructure distance," in *GbRPR*, 2015, pp. 188–197.

[20] F. Serratosa, "Fast computation of bipartite graph matching," *Pattern Recogn. Lett.*, vol. 45, pp. 244–250, 2014.

[21] ——, "Speeding up fast bipartite graph matching through a new cost matrix," *Int. J. Pattern Recogn.*, vol. 29, no. 2, 2015.

[22] B. Gaüzère, S. Bougleux, and L. Brun, "Approximating graph edit distance using GNCCP," in *S+SSPR*, 2016, pp. 496–506.

[23] S. Bougleux, L. Brun, V. Carletti, P. Foggia, B. Gaüzère, and M. Vento, "Graph edit distance as a quadratic assignment problem," *Pattern Recogn. Lett.*, vol. 87, pp. 38–46, 2017.

[24] É. Daller, S. Bougleux, B. Gaüzère, and L. Brun, "Approximate graph edit distance by several local searches in parallel," in *ICPRAM*, 2018, pp. 149–158.

[25] S. Bougleux, B. Gaüzère, and L. Brun, "Graph edit distance as a quadratic program," in *ICPR*, 2016, pp. 1701–1706.

[26] E. Ozdemir and C. Gunduz-Demir, "A hybrid classification model for digital pathology using structural and statistical pattern recognition," *IEEE Trans. Med. Imaging*, vol. 32, no. 2, pp. 474–483, 2013.

[27] M. Stauffer, T. Tschachtli, A. Fischer, and K. Riesen, "A survey on applications of bipartite graph edit distance," in *GbRPR*, 2017, pp. 242–252.

[28] Z. Abu-Aisheh, B. Gaüzere, S. Bougleux, J.-Y. Ramel, L. Brun, R. Raveaux, P. Héroux, and S. Adam, "Graph edit distance contest 2016: Results and future challenges," *Pattern Recogn. Lett.*, vol. 100, pp. 96–103, 2017.