Graph Edit Distance as a Quadratic Program

Appears in 23rd International Conference on Pattern Recognition (ICPR), pp 1701–1706, IEEE, 2016
http://doi.org/10.1109/ICPR.2016.7899881

Sébastien Bougleux
Normandie Univ, UNICAEN
CNRS, GREYC, France

Benoit Gaüzère
Normandie Univ, INSA Rouen
LITIS, France

Luc Brun
Normandie Univ, ENSICAEN
CNRS, GREYC, France

***Abstract*—The graph edit distance (GED) measures the amount of distortion needed to transform a graph into another graph. Such a distance, developed in the context of error-tolerant graph matching, is one of the most flexible tool used in structural pattern recognition. However, the computation of the exact GED is NP-complete. Hence several suboptimal solutions, such as the ones based on bipartite assignments with edition, have been proposed. In this paper we propose a binary quadratic programming problem whose global minimum corresponds to the exact GED. This problem is interpreted as a quadratic assignment problem (QAP) where some constraints have been relaxed. This allows to adapt the integer projected fixed point algorithm, initially designed for the QAP, to efficiently compute an approximate GED by finding an interesting local minimum. Experiments show that our method remains quite close to the exact GED for datasets composed of small graphs, while keeping low execution times on datasets composed of larger graphs.**

I. Introduction

Computing an efficient similarity or dissimilarity measure between graphs is a major problem in structural pattern recognition. The graph edit distance (GED) [1]–[4], developed in the context of error-tolerant graph matching, provides such a measure. It may be understood as the minimal amount of distortion required to transform a graph into another, by a sequence of edit operations applied on nodes and on edges, restricted here to substitutions, insertions and removals. Such a sequence is called an edit path. Depending on the nature of the graphs and on the context, all paths may not have a same importance, in particular when nodes or edges are labeled or attributed. So each possible edit operation e is penalized by a non-negative cost $c(e)$, and the integration of these costs over an edit path γ defines the length $A(\gamma)$ of this path. An edit path having a minimal length, among all edit paths $\Gamma(G_1, G_2)$ transforming a graph G_1 into a graph G_2, defines the GED from G_1 to G_2:

$$\mathrm{GED}(G_1, G_2) = \min_{\gamma \in \Gamma(G_1, G_2)} \left\{ A(\gamma) \stackrel{\text{def.}}{=} \sum_{e \subset \gamma} c(e) \right\} \quad (1)$$

Since computing the GED is NP-complete, it is restricted to rather small graphs. So several approaches have been proposed to approximate the GED efficiently and to process larger graphs. In this paper, graphs are assumed to be simple (no loop nor multiple edge), and each element of the two graphs can be edited only once (no composition of edit operations).

Under these conditions, the GED can be expressed as a binary quadratic program (QP), which is the subject of this paper. Contrary to the graph matching problem, which can be rewritten as a quadratic assignment problem (QAP) [5], [6], an error-tolerant graph matching can also remove and insert elements. Managing these edit operations is the main difficulty of the quadratic framework. To this, a binary QP is obtained in [7] by considering an edit grid. It is transformed into a binary linear program, which can be solved exactly with classical solvers, but graphs are restricted to be undirected, edges to be unlabeled, and it is also restricted to small graphs since there are $(n+m)^2$ binary variables, with $n = |V_1|$ and $m = |V_2|$. In [8] a QP with nm real variables is defined to optimize the cost restricted to node substitutions and associated edge operations. This QP, solved by an interior point algorithm, provides a weight for each possible node substitution. From this fuzzy assignment between nodes, an approximate minimal-cost edit path is then constructed using an heuristic method. Other approximate GED are based on a similar two-step strategy.

In particular, the bipartite graph edit distance (bGED) [9], [10] approximates the GED by replacing the error-tolerant matching between two graphs by an error-tolerant bipartite matching of their nodes, from which an edit path is constructed. Such an error-tolerant set matching can be rewritten as a square linear sum assignment problem (LSAP), with $(n+m)^2$ binary variables to take into account substitutions, insertions and removals of elements. To approximate the GED, these operations are penalized by the cost of substituting, removing or inserting substructures attached to the nodes [9]–[14]. Since the LSAP can be solved in polynomial time, for instance in $O((n+m)^3)$ with the well-known Hungarian algorithm [5], [15], the bGED provides an efficient approximation of the GED. This linear framework has been extended to a quadratic one [16], where the GED is expressed as a global solution of a QAP with $(n+m)^2$ binary variables. An interesting local minimal solution is obtained by relaxing the constraints on the solutions, and by using the integer projected fixed point algorithm (IPFP) [6]. This algorithm iterates: an instance of the LSAP (projection of local continuous solution in the discrete domain), and a line search in the continuous domain to get the next local continuous solution.

While this last quadratic framework improves the approximation of the exact GED, the size of the initial data has been artificially augmented to manage removal and insertion opera-

tions more easily. This is also the case in the linear framework above, and in the one proposed by [7]. Recently, the error-tolerant set matching involved in the computation of bGEDs has been formulated as an extension of the LSAP with only $(n+1)(m+1)$ binary variables [17]. Since a solution can be computed in $O(\min\{n,m\}^2 \max\{n,m\})$ time complexity by an adapted Hungarian algorithm, the computation of the bGED is improved, in particular for large graphs.

Based on this last formulation (Sec. II) [17], we propose in this paper a new quadratic expression of the GED with only $(n+1)(m+1)$ binary variables (Sec. III). This problem is interpreted as a quadratic assignment problem (QAP) where constraints on solutions have been relaxed for removal and insertions operations. This allows to adapt the IPFP algorithm used in [16] to compute efficiently an approximate GED. Experiments (Sec. IV) show that our method remains quite close to the exact GED for datasets composed of small graphs, as previously obtained in [16]. Moreover, contrary to [16], execution times on datasets composed of larger graphs are similar to the ones obtained by bGEDs.

II. Linear sum assignment problem with edition

This section describes the problem of minimal-cost error-tolerant set matching, formalized as an extension of the LSAP. This is detailed in our technical report [17].

A. Assignment with edition

A transformation of a set V_1 into a set V_2 can be performed by applying edit operations as follows: Each element $i \in V_1$ is either substituted by an unique element $j \in V_2$ (denoted by $i \to j$), or removed ($i \to \epsilon$). Then each remaining element $j \in V_2$, not previously used for a substitution, is inserted (denoted by $\epsilon \to j$). So each element of V_1 and V_2 is involved in only one edit operation.

Consider the extended sets $V_1^\epsilon = V_1 \cup \{\epsilon\}$ and $V_2^\epsilon = V_2 \cup \{\epsilon\}$. An error-correcting set matching, or ϵ-assignment, can be represented by a mapping $\varphi : V_1^\epsilon \to \mathcal{P}(V_2^\epsilon)$ satisfying:

$$\begin{cases} \forall i \in V_1, & |\varphi(i)| = 1 \\ \forall j \in V_2, & \left|\varphi^{-1}[j]\right| = 1 \\ & \epsilon \in \varphi(\epsilon) \end{cases} \tag{2}$$

where $\mathcal{P}(\cdot)$ is the powerset, and $\varphi^{-1}[j] \stackrel{\text{def.}}{=} \varphi^{-1}[\{j\}]$ denotes the pre-image of any singleton $\{j\} \in \mathcal{P}(V_2^\epsilon)$. Remark that by definition we have $1 \le |\varphi(\epsilon)| \le m+1$ and $1 \le |\varphi^{-1}[\epsilon]| \le n+1$, with $n = |V_1|$ and $m = |V_2|$.

Let $\mathcal{A}_\epsilon(V_1, V_2)$ be the set of all ϵ-assignments from V_1 to V_2. To simplify the forthcoming expressions, we assume that $V_1 = \{1, \dots, n\}$, $V_2 = \{1, \dots, m\}$, $V_1^\epsilon = V_1 \cup \{n+1\}$ and $V_2^\epsilon = V_2 \cup \{m+1\}$. Any ϵ-assignment $\varphi \in \mathcal{A}_\epsilon(V_1, V_2)$ can be equivalently represented by a $(n+1) \times (m+1)$ binary matrix

$$\mathbf{X} = \begin{array}{c} \\ \\ \\ \\ \hline \\ \epsilon \end{array} \overset{\begin{array}{ccc} & V_2 & \end{array}}{\left(\begin{array}{ccc|c} x_{1,1} & \dots & x_{1,m} & x_{1,m+1} \\ \vdots & \ddots & \vdots & \vdots \\ x_{n,1} & \dots & x_{n,m} & x_{n+1,m+1} \\ \hline x_{n+1,1} & \dots & x_{n+1,m} & 1 \end{array}\right)} \begin{array}{c} \epsilon \\ \\ \\ \\ \hline \\ V_1 \end{array} \tag{3}$$

such that

$$\begin{cases} x_{i,j} = \delta_{\varphi(i)=\{j\}}, & \forall (i,j) \in V_1 \times V_2 \quad \text{(sub.)} \\ x_{i,m+1} = \delta_{\varphi(i)=\{\epsilon\}}, & \forall i \in V_1 \quad \text{(rem.)} \\ x_{n+1,j} = \delta_{\varphi^{-1}[j]=\{\epsilon\}}, & \forall j \in V_2 \quad \text{(ins.)} \\ x_{n+1,m+1} = 1 \end{cases} \tag{4}$$

where $\delta_r = 1$ if the relation r is true or $\delta_r = 0$ else. Due to the constraints on φ, the matrix $\mathbf{X}$ has a 1 on each of its n first rows and a 1 on each of its m first columns:

$$\begin{cases} \sum_{j=1}^{m+1} x_{i,j} = 1, & \forall i = 1, \dots, n \\ \sum_{i=1}^{n+1} x_{i,j} = 1, & \forall j = 1, \dots, m \\ x_{n+1,m+1} = 1 \end{cases} \tag{5}$$

Reciproqualy, any matrix $\mathbf{X} \in \{0,1\}^{(n+1)\times(m+1)}$ satisfying Eq. 5 represents an ϵ-assignment.

B. Problem formulation

The definition of an ϵ-assignment does not rely on the nature of the underlying data. To select a relevant ϵ-assignment in $\mathcal{A}_\epsilon(V_1, V_2)$, each edit operation o is penalized by a non-negative cost $c(o)$. Then the cost of an ϵ-assignment φ is obtained by summing the costs of its edit operations:

$$\begin{aligned} L(\varphi) &= \sum_{i \in V_1^\epsilon} \sum_{j \in \varphi(i)} c(i \to j) \\ &= \underbrace{\sum_{\substack{i \in V_1 \\ \varphi(i) = \{j\}}} c(i \to j)}_{\text{substitutions}} + \underbrace{\sum_{\substack{i \in V_1 \\ \varphi(i) = \{\epsilon\}}} c(i \to \epsilon)}_{\text{removals}} + \underbrace{\sum_{\substack{j \in V_2 \\ \varphi^{-1}[j] = \{\epsilon\}}} c(\epsilon \to j)}_{\text{insertions}} \end{aligned} \tag{6}$$

Note that the mapping $\epsilon \to \epsilon$ has a zero cost, since it does not represent any edit operation.

All possible edit costs can be collected into a matrix $\mathbf{C} \in [0, +\infty)^{(n+1)\times(m+1)}$ such that $c_{i,j} = c(i \to j)$ for all $(i,j) \in V_1^\epsilon \times V_2^\epsilon$, with $c_{n+1,m+1} = 0$.

Consider an ϵ-assignment φ and its corresponding binary matrix $\mathbf{X}$ (Eq. 4). Then the cost of φ (Eq. 6) is equal to:

$$L(\mathbf{X}) = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} c_{i,j} x_{i,j} = \text{vec}(\mathbf{C})^T \text{vec}(\mathbf{X}) = \mathbf{c}^T \mathbf{x} \tag{7}$$

where $\mathbf{x} = \text{vec}(\mathbf{X}) \in \{0,1\}^{(n+1)(m+1)}$ is the vectorization of $\mathbf{X}$ obtained by concatenating its rows, and similarly $\mathbf{c} = \text{vec}(\mathbf{C})$.

A relevant ϵ-assignment is then defined as one having a minimal cost, among all ϵ-assignments transforming V_1 into V_2, *i.e.* satisfying

$$\underset{\varphi \in \mathcal{A}_\epsilon(V_1, V_2)}{\text{argmin}} \; L(\varphi) \tag{8}$$

or equivalently satisfying the following binary linear program:

$$\underset{\mathbf{x}}{\text{argmin}} \left\{ \mathbf{c}^T \mathbf{x} \,|\, \mathbf{L}\mathbf{x} = \mathbf{1}_{n+m},\, \mathbf{x} \in \{0,1\}^p,\, x_p = 1 \right\} \tag{9}$$

where $p = (n+1)(m+1)$, and the linear system $\mathbf{L}\mathbf{x} = \mathbf{1}$ is the matrix version of the constraints given by Eq. 2 or Eq. 5. The assignment matrix $\mathbf{L} \in \{0,1\}^{(n+m)\times(n+1)(m+1)}$ is defined by:

$$\forall (i,j), \quad \begin{cases} l_{k,(i,j)} = \delta_{k=i}, & \forall k = 1, \dots, n \\ l_{n+k,(i,j)} = \delta_{k=j}, & \forall k = 1, \dots, m \end{cases} \tag{10}$$

Since $\mathbf{L}$ is totally unimodular, from standard tools in linear programming, the problem defined by Eq. 9 (or Eq. 8) has a binary solution. This problem is close to the classical linear sum assignment problem (LSAP). Compared to the LSAP, elements can also be removed or inserted, *i.e.* the constraints in Eq. 9 are relaxed for one element (ϵ) in each of the two sets. For this reason, the problem defined by Eq. 9 is called the linear sum assignment problem with edition (LSAPE) [17].

The LSAPE can be efficiently solved in polynomial time complexity by adapting algorithms that initially solve the LSAP. An adaptation of the Hungarian algorithm [5], [15] is presented in [17]. It computes a solution to the LSAPE in $O(\min\{n,m\}^2 \max\{n,m\})$ time complexity and in $O(nm)$ space complexity. This improves the results obtained by the initial approach proposed to compute the bipartite GED [4], [9], [10]. In this last approach, the problem is formalized as a LSAP with a specific $(n+m)\times(m+n)$ cost matrix. It can be solved by the classical Hungarian algorithm in $O((n+m)^3)$ time complexity and in $O((n+m)^2)$ space complexity. Time complexity has been reduced to $O(\max\{n,m\}^3)$ with some restrictions on the edit costs [18]. The Jonker-Volgenant algorithm is also explored in these works. Note that a similar encoding of the assignment matrix defined by Eq. 3 has been proposed in [19], together with an adaptation of the Jonker-Volgenant algorithm. This algorithm can be combined with the Hungarian algorithm proposed in [17].

III. A QUADRATIC EXPRESSION OF THE GED

In this section we extend the linear sum assignment problem with edition to a quadratic one. This simplifies the quadratic expression of the GED proposed in [16].

A. Simultaneous node assignment with edition and GED

Let $G_1=(V_1,E_1)$ and $G_2=(V_2,E_2)$ be two simple graphs, both directed or not. A transformation of G_1 into G_2, with edit operations applied on nodes and on edges, can be fully represented by an ϵ-assignment $\varphi: V_1^\epsilon \to \mathcal{P}(V_2^\epsilon)$. Operations on edges are induced by operations on nodes:

1) Any edge $(i,j)\in E_1$ is mapped by φ to a pair $(k,l)\in V_2^\epsilon \times V_2^\epsilon$ with $\varphi(i)=\{k\}$ and $\varphi(j)=\{l\}$. The edge (i,j) is either
   a) substituted by (k,l) if $(k,l)\in E_2$, or
   b) removed if $(k,l)\notin E_2$, in particular if i or j is removed by φ ($k=\epsilon$ or $l=\epsilon$).
2) Any edge $(k,l)\in E_2$ is inserted into E_1 if it has not been used by a substitution in Step 1, *i.e.* if there is a pair $(i,j)\in V_1^\epsilon \times V_1^\epsilon$ such that $k\in\varphi(i)$, $l\in\varphi(j)$ and $(i,j)\notin E_1$. In particular, this occurs when k or l is inserted by φ into V_1 ($i=\epsilon$ or $j=\epsilon$).

Each edge of E_1 and E_2 is thus involved in exactly one edit operation. So the transformation of G_1 into G_2, represented here by φ, corresponds to an edit path having a minimal number of operations. Such a transformation is usually called an error-correcting graph matching, see [4] for more details.

Error-correcting graph matching can be linked to the GED by penalizing operations on edges as follows:

$$\begin{aligned} Q(\varphi) = & \sum_{\substack{(i,j)\in E_1 \\ \text{with } \{k\}=\varphi(i),\{l\}=\varphi(j),(k,l)\in E_2}} \bar{c}((i,j)\to(k,l)) && \text{(sub.)} \\ & + \sum_{\substack{(i,j)\in E_1 \\ \text{with } \{k\}=\varphi(i),\{l\}=\varphi(j),(k,l)\notin E_2}} c((i,j)\to\epsilon) && \text{(rem.)} \\ & + \sum_{\substack{(k,l)\in E_2 \\ \text{with } \{i\}=\varphi^{-1}[k],\{j\}=\varphi^{-1}[l],(i,j)\notin E_1}} c(\epsilon\to(k,l)) && \text{(ins.)} \end{aligned} \tag{11}$$

where

$$\bar{c}((i,j)\to(k,l)) = \min\{c((i,j)\to(k,l)), \\ c((i,j)\to\epsilon)+c(\epsilon\to(k,l))\}$$

Indeed, from the definition of an error-correcting graph matching, when two connected nodes $i,j\in V_1$ are substituted by two connected nodes $k,l\in V_2$, the edge (i,j) is substituted by the edge (k,l), so (i,j) cannot be removed and then (k,l) inserted. But these operations are structurally equivalent, so the GED (Eq. 1) can be rewritten as follows:

$$\text{GED}(G_1,G_2) = \min_{\varphi\in\mathcal{A}_\epsilon(V_1,V_2)} \left\{ A(\varphi) \overset{\text{def.}}{=} Q(\varphi)+L(\varphi) \right\} \tag{12}$$

where $L(\varphi)$ is the linear cost of the ϵ-assignment defined by Eq. 6. This optimization problem is close to the quadratic assignment problem (QAP), see [5] for more details on QAP.

B. Matrix form

Consider an ϵ-assignment $\varphi\in\mathcal{A}_\epsilon(V_1,V_2)$. The cost of simultaneously assigning $i\in V_1^\epsilon$ to $k\in V_2^\epsilon$, and $j\in V_1^\epsilon\setminus\{i\}$ to $l\in V_2^\epsilon\setminus\{k\}$, can written as:

$$\begin{aligned} d_{ik,jl} = & \; \bar{c}\left((i,j)\to(k,l)\right)\delta_{(i,j)\in E_1}\delta_{(k,l)\in E_2} && \text{(sub.)} \\ & + c((i,j)\to\epsilon)\delta_{(i,j)\in E_1}(1-\delta_{(k,l)\in E_2}) && \text{(rem.)} \\ & + c(\epsilon\to(k,l))(1-\delta_{(i,j)\in E_1})\delta_{(k,l)\in E_2} && \text{(ins.)} \end{aligned} \tag{13}$$

and $d_{ik,jl}=0$ if $i=j$ or $k=l$ (since graphs are simple). Observe that $d_{ik,jl}$ is also null when both (i,j) and (k,l) are not edges. In this case the simultaneous assignment does not correspond to any edit operation.

Then, the cost $Q(\varphi)$ defined by Eq. 11 can be rewritten more compactly as:

$$Q(\varphi) = g \sum_{i\in V_1^\epsilon}\sum_{k\in\varphi(i)}\sum_{j\in V_1^\epsilon}\sum_{l\in\varphi(j)} d_{ik,jl} \\ \text{with } g=\begin{cases}\frac{1}{2} & \text{if } G_1 \text{ and } G_2 \text{ are undirected} \\ 1 & \text{else}\end{cases} \tag{14}$$

When graphs are both undirected, the cost $d_{ik,jl}$ is symmetric ($d_{ik,jl}=d_{jl,ik}$). So a factor $\frac{1}{2}$ is introduced to count the cost of each edit operation only once.

Consider the binary vector $\mathbf{x}\in\{0,1\}^p$ associated to φ, with $p=(n+1)(m+1)$. Nodes i and j of V_1^ϵ are simultaneously

assigned by φ to k and l of V_2^ϵ, respectively, iff $x_{ik} = x_{jl} = 1$. Then the cost $Q(\varphi)$ can be rewritten as follows:

$$Q(\mathbf{x}) = g \sum_{i=1}^{n+1} \sum_{k=1}^{m+1} \sum_{j=1}^{n+1} \sum_{l=1}^{m+1} d_{ik,jl} x_{ik} x_{jl} = g\, \mathbf{x}^T \mathbf{D} \mathbf{x} \quad (15)$$

where the matrix $\mathbf{D} \in [0, +\infty)^{p \times p}$, of general term $d_{ik,jl}$ (Eq. 13), encodes the cost of all possible simultaneous assignments. Compared to Eq. 14, the sum in Eq. 15 is also carried on simultaneous assignments which are not induced by the ϵ-assignment φ, *i.e.* $x_{ik} = 0$ or $x_{jl} = 0$. Since they do not contribute to the sum, we have $Q(\mathbf{x}) = Q(\varphi)$. Note that all diagonal elements are null ($d_{ik,ik} = 0$).

Then the cost of transforming G_1 into G_2, defined in Eq. 12, can be written as

$$A(\mathbf{x}) = g\, \mathbf{x}^T \mathbf{D} \mathbf{x} + \mathbf{c}^T \mathbf{x} \quad (16)$$

where $\mathbf{c} \in [0, +\infty)^p$ is the edit cost vector defined in Eq. 9. Since $\mathbf{x}^T \mathbf{D} \mathbf{x} = \frac{1}{2} \mathbf{x}^T (\mathbf{D} + \mathbf{D}^T) \mathbf{x}$, we have

$$A(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \Delta \mathbf{x} + \mathbf{c}^T \mathbf{x}$$
$$\text{with } \Delta = \begin{cases} \mathbf{D} & \text{if } G_1 \text{ and } G_2 \text{ are undirected} \\ \mathbf{D} + \mathbf{D}^T & \text{else} \end{cases} \quad (17)$$

The cost matrix Δ is always symmetric. The linear part of the cost can also be included in the quadratic part, leading to

$$A(\mathbf{x}) = \mathbf{x}^T \widehat{\Delta} \mathbf{x}, \text{ with } \widehat{\Delta} = \tfrac{1}{2} \Delta + \text{diag}(\mathbf{c}) \quad (18)$$

Then the GED is rewritten as the following quadratic program:

$$\text{GED}(G_1, G_2) = \min \left\{ \mathbf{x}^T \widehat{\Delta} \mathbf{x} \mid \mathbf{L}\mathbf{x} = \mathbf{1}_{n+m},\, \mathbf{x} \in \{0,1\}^p \right\} \quad (19)$$

We call it the quadratic assignment problem with edition (QAPE). The QAPE differs from the classical QAP by the constraint matrix $\mathbf{L}$ (Eq. 10). A QAP finds a bijection or an injection inducing a simultaneous assignment, as for the quadratic expression of the GED proposed in [16]. This expression describes a QAP involving assignments represented by vectors of size $(n+m)^2$. This is simplified by Eq. 19 by involving vectors of size $(n+1)(m+1)$ only.

C. Relaxation, approximation, algorithm

While exact solutions to the QAPE could be computed by adapting algorithms that initially solve the QAP [5], this would not be competitive with bipartite GEDs in terms of computational time. So we propose to approximate the GED by adapting the integer projected fixed point (IPFP) algorithm [6], designed to approximate the solution to the QAP, and it is also used in [16] to approximate the GED. As several other algorithms, the IPFP algorithm tries to find a solution to the relaxed quadratic problem given here by

$$\text{argmin} \left\{ \mathbf{x}^T \widehat{\Delta} \mathbf{x} \mid \mathbf{L}\mathbf{x} = \mathbf{1}_{n+m},\, \mathbf{x} \in [0,1]^p \right\} \quad (20)$$

where the space of solutions is now continuous.

Given an initial candidate solution $\mathbf{x}$, the algorithm iterates the three following steps

$$\mathbf{b}^\star \leftarrow \text{argmin} \left\{ (\mathbf{x}^T \widehat{\Delta}) \mathbf{b} \mid \mathbf{L}\mathbf{b} = \mathbf{1}, \mathbf{b} \in \{0,1\}^p, b_p = 1 \right\} \quad (21)$$

$$\alpha^\star \leftarrow \underset{\alpha \in [0,1]}{\text{argmin}}\ A(\mathbf{x} + \alpha(\mathbf{b}^\star - \mathbf{x})) \quad (22)$$

$$\mathbf{x} \leftarrow \mathbf{x} + \alpha^\star (\mathbf{b}^\star - \mathbf{x}) \quad (23)$$

until convergence (when $\mathbf{x}$ becomes discrete) or after a given number of iterations.

The first step (Eq. 21) consists in computing a linear approximation of A around the current solution $\mathbf{x}$ in the discrete domain. This can be realized by solving a LSAPE with $\mathbf{x}^T \widehat{\Delta}$ as an edit cost matrix (Sec. II). Indeed, the first-order Taylor expansion of A around $\mathbf{x}$ is given by

$$A(\mathbf{y}) \approx A(\mathbf{x}) + (\mathbf{x}^T \widehat{\Delta})(\mathbf{y} - \mathbf{x}),\ \mathbf{y} \in [0,1]^p \quad (24)$$

so minimizing $A(\mathbf{x})$ locally is approximatively equivalent to minimizing $\mathbf{x}^T \widehat{\Delta} \mathbf{y}$ with $\mathbf{x}$ fixed. By standard tools in linear programming, this linear program has a same binary solution $\mathbf{b}^\star$ as the problem defined by Eq. 21, here a LSAPE. It can be solved in $O(\min\{n,m\}^2 \max\{n,m\})$ time complexity and in $O(nm)$ space complexity (Sec. II-B). Compared to the IPFP algorithm proposed in [16], where Eq. 21 corresponds to a square LSAP, computational time is reduced at each iteration. This is confirmed experimentally (Section IV).

The second step consists in computing the local minimum of A, by a line search, between the current solution $\mathbf{x}$ and the candidate binary solution $\mathbf{b}^\star$ included. As detailed in [16] this can be solved analytically, here in $O(mn)$. The vector realizing this minimum is the new candidate solution.

The iterative process generally converges to a local minimum of the relaxed problem defined by Eq. 20. When $\mathbf{x}$ is continuous, an additional projection (Eq. 21) is performed to get the final binary vector. The quality of the approximation depends mainly on the initialization. A random vector satisfying the constraints could be used. Instead, it seems to be more natural to compute a bipartite GED (bGED), *i.e.* to solve a LSAPE [17], and consider the corresponding optimal ϵ-assignment as initial vector in IPFP algorithm. In this way, results obtained by a bGED should be improved. This is analyzed experimentally in the following section.

IV. Experiments

A. Chemical datasets

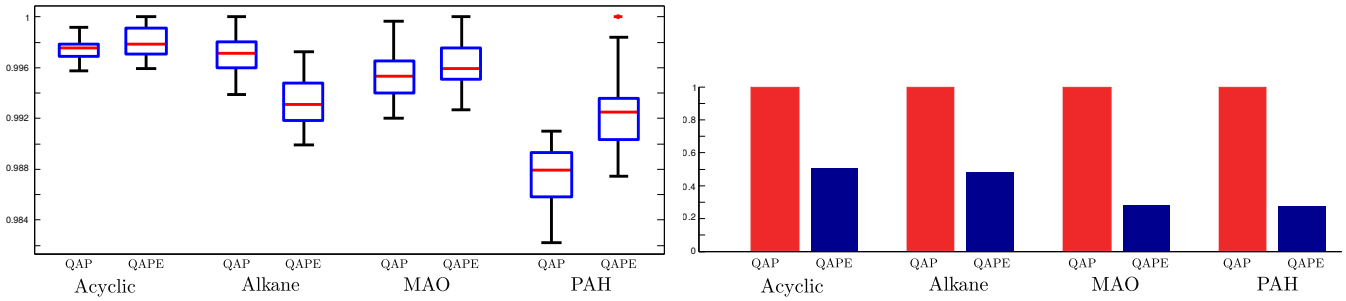
Table I shows results obtained by different state of the art methods to compute an estimation of the GED. These methods have been tested on four chemoinformatics datasets [1], in the same way as in previous papers using these datasets [12], [13]. Each dataset is composed of molecular graphs. Costs have been set to 1 for any substitution operation, and 3 for insertions and deletions. It is important to notice that each graph adjacency matrix is randomly permuted in order

[1] Datasets are available at https://iapr-tc15.greyc.fr/links.html

TABLE I

ACCURACY AND COMPUTATIONAL SCORES. d IS THE AVERAGE EDIT DISTANCE, e THE AVERAGE ERROR AND t THE AVERAGE COMPUTATIONAL TIME.

	Algorithm	Alkane			Acyclic			MAO		PAH	
		d	e	t	d	e	t	d	t	d	t
1	A^*	15		1.29	17		6.02				
2	[9], [10]	35	18	$\simeq 10^{-3}$	35	18	$\simeq 10^{-3}$	105	$\simeq 10^{-3}$	138	$\simeq 10^{-3}$
3	[12]	33	18	$\simeq 10^{-3}$	31	14	0.002	49	$\simeq 10^{-2}$	120	$\simeq 10^{-2}$
4	[13]	26	11	2.27	28	9	0.73	44	6.16	129	2.01
5	QAP	20.2	4.7	0.002	20.8	3.4	0.004	33.9	0.019	53.9	0.038
6	(this paper) QAPE	20.1	4.6	$\simeq 10^{-3}$	20.8	3.5	0.002	33.9	0.005	53.6	0.010



(a) Repartition of normalized average mean edit distance for each repetition.

(b) Computational times normalized over the 4 datasets

Fig. 1. Comparison of IPFP algorithm for the QAP and for the QAPE.

to avoid some alignment bias which may provide optimistic approximations of the GED. The tested methods are:

- Exact GED based on $A^\star$ algorithm (line 1), restricted to very small graphs (not adapted to MAO and PAH).
- Bipartite GED based on the Hungarian algorithm adapted to the LSAPE [17], with the substructure attached to each node defined by: incident edges and adjacent nodes (line 2) [9], [10], a bag of random walks up to 3 edges (line 3) [12], a subgraph up to a radius 3 (line 4) [13].
- Approximate GED formalized as a QAP (line 5) [16] and as a QAPE (line 6), both solved by the IPFP algorithm with the same initialization provided by [12], since initialization influences the approximation of the GED.

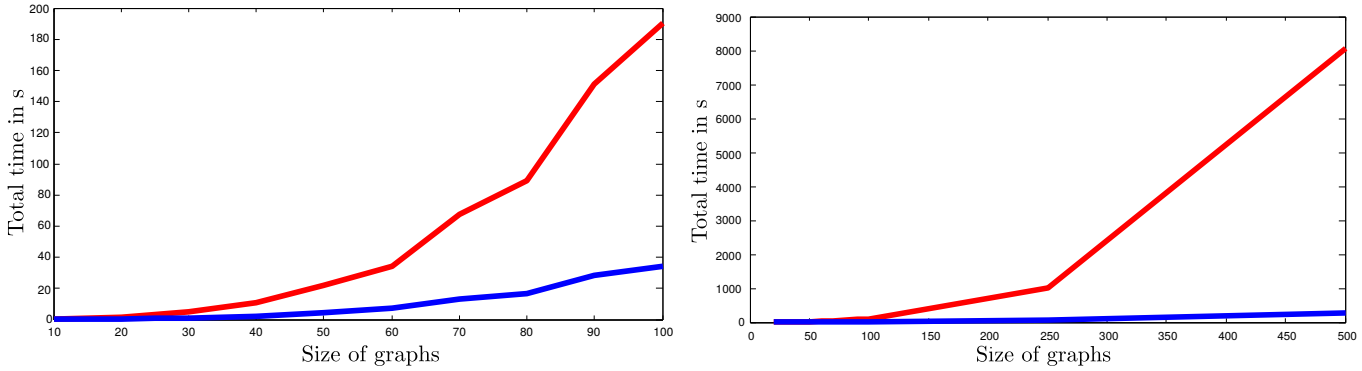
Observe that these two last approaches obtain the best approximation results. Considering Alkane and Acyclic datasets, this is trivially inferred from average errors. For the two remaining datasets, we observe lower average edit distances which may correspond to better approximations since the exact GED is always over-estimated. In addition, we can note that computational times of quadratic approaches remain globally acceptable, especially compared to the bGED proposed in [13] where most of the time is spent to compute the costs.

Moreover, quadratic approaches seem to have similar accuracies, while the approach presented in this paper improves clearly the computational time. This is analyzed through a second experiment consisting in 30 repetitions of each approach, for each dataset. As before, graph adjacency matrices are randomly permuted in order to avoid some alignment bias. Figure 1a shows the boxplot of the normalized average approximate GEDs. For each dataset, these average values are normalized by the maximal mean approximate GED given by $\max\{\max_i\{\bar{d}_i^{\text{QAPE}}\}, \max_i\{\bar{d}_i^{\text{QAP}}\}\}$ where $\bar{d}_i$ is the mean approximate GED obtained for the repetition $i \in \{1, \ldots, 30\}$. Considering Acyclic and MAO datasets, the distribution are statistically similar according to a paired Student's test with a p-value equals to 0.01. Considering Alkane and PAH datasets, the difference is statistically significant but not so important for the approximation of GEDs (about 0.4%). From a computational time point of view, Figure 1b shows the normalized mean times according to a same initialization. We can clearly see that the proposed approach is faster than previous one, with a ratio around 2 for smaller graphs and 4 for larger graphs.

B. Synthetic datasets

To explore the behavior of the quadratic approaches when the number of nodes increases, we generate synthetic datasets having the same characteristics as MAO dataset but generalized to different graph sizes. These characteristics include node's and edge's labels distribution and ratio between number of edges and number of nodes. For a given number of nodes, each new synthetic graph is created according to these characteristics. This allows to retrieve characteristics of common molecular graphs. For each tested size, up to 100 nodes, a dataset composed of 100 synthetic graphs has been synthesized. Each of this graph, called source graph, is then associated to a target graph which has been obtained by removing one node and by substituting another one from the associated source graph. The exact GED between a source and a target is thus known by construction and is around 10.

Figure 2a shows the behavior of the quadratic approaches when the size increases. As we can see, the proposed approach



(a) Both graphs have the same size (up to one node)

(b) 1st graph has 20 nodes, 2nd second one has 20 to 500 nodes

Fig. 2. Total computational time in seconds taken by IPFP algorithm for the QAP (upper red line) and for the QAPE (bottom blue).

based on a QAPE scales better in terms of computational times. Indeed, for graphs having 100 nodes, the computational ratio is over 5. Moreover, the average error is the same for both methods and remains quite stable on the set of sizes. It is worth noticing that the most important part of time is dedicated here to the initialization, realized as in the previous section. Figure 2b shows a similar result but with a fixed size for the first graph (20 nodes) and a varying size for the second one (from 20 to 500 nodes). The gain in computational times increases as the size of graphs also increases. Considering a second graph with 500 nodes, this gain corresponds to a computational time reduced by a factor 29. This is explained by the reduction of variable sizes from $(n+m)^2$ to $(m+1)(n+1)$, and by the use of a LSAPE instead of a LSAP, at each iteration of IPFP algorithm (Sec. III-C).

V. Conclusion

Based on the notion of assignment with edition, this paper presents a binary quadratic programming problem whose global optimum is equal to the GED when graphs are simple, and when edit operations are restricted to substitutions, removals and insertions. The proposed quadratic program reformulates a previous expression of the GED as a QAP, by reducing the number of binary variables to $(n+1)(m+1)$ and by relaxing the constraints associated to removals and insertions. This allows us to adapt an iterative algorithm, that initially finds an approximate solution to the QAP, to approximate the GED and reduce both space and time complexities. This is confirmed experimentally. Experiments also show that computational times are close to those of bGEDs, while providing distances closer to exact GEDs, on tested datasets.

References

[1] W.-H. Tsai and K.-S. Fu, "Error-correcting isomorphisms of attributed relational graphs for pattern analysis," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 9, no. 12, pp. 757–768, 1979.

[2] H. Bunke and G. Allermann, "Inexact graph matching for structural pattern recognition," *Pattern Recognition Letters*, vol. 1, no. 4, pp. 245–253, 1983.

[3] A. Sanfeliu and K.-S. Fu, "A distance measure between attributed relational graphs for pattern recognition," *IEEE Transactions on Systems, Man. and Cybernetics*, vol. 13, no. 3, pp. 353–362, 1983.

[4] K. Riesen, *Structural Pattern Recognition with Graph Edit Distance*, ser. Advances in Computer Vision and Pattern Recognition. Springer International Publishing, 2015.

[5] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*. SIAM, 2009.

[6] M. Leordeanu, M. Hebert, and R. Sukthankar, "An integer projected fixed point method for graph matching and map inference," in *Advances in Neural Information Processing Systems*, 2009, vol. 22, pp. 1114–1122.

[7] 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.

[8] M. Neuhaus and H. Bunke, "A quadratic programming approach to the graph edit distance problem," in *Graph-Based Representations in Pattern Recognition*, ser. LNCS, 2007, vol. 4538, pp. 92–102.

[9] K. Riesen, M. Neuhaus, and H. Bunke, "Bipartite graph matching for computing the edit distance of graphs," in *Graph-Based Representations in Pattern Recognition*, ser. LNCS, 2007, vol. 4538, pp. 1–12.

[10] K. Riesen and H. Bunke, "Approximate graph edit distance computation by means of bipartite graph matching," *Image and Vision Comp.*, vol. 27, pp. 950–959, 2009.

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

[12] B. Gaüzère, S. Bougleux, K. Riesen, and L. Brun, "Approximate Graph Edit Distance Guided by Bipartite Matching of Bags of Walks," in *Structural, Syntactic and Statistical Pattern Recognition*, ser. LNCS, 2014, vol. 8621, pp. 73–82.

[13] V. Carletti, B. Gaüzère, L. Brun, and M. Vento, "Approximate graph edit distance computation combining bipartite matching and exact neighborhood substructure distance." in *Graph-Based Representations in Pattern Recognition*, ser. LNCS, 2015, vol. 9069, pp. 168–177.

[14] K. Riesen and H. Bunke, "Improving bipartite graph edit distance approximation using various search strategies," *Pattern Recognition*, vol. 28, no. 4, pp. 1349–1363, 2015.

[15] E. Lawler, *Combinatorial Optimization: Networks and Matroids*. New York: Holt, Rinehart and Winston, 1976.

[16] S. Bougleux, L. Brun, V. Carletti, P. Foggia, B. Gaüzère, and M. Vento, "A quadratic assignment formulation of the graph edit distance," Normandie Univ, NormaSTIC FR 3638, France, Tech. Rep., 2015. [Online]. Available: http://arxiv.org/abs/1512.07494

[17] S. Bougleux and L. Brun, "Linear sum assignment with edition," Normandie Univ, GREYC UMR 6072, Tech. Rep., 2016.

[18] F. Serratosa, "Computation of graph edit distance: Reasoning about optimality and speed-up," *Image and Vision Computing*, vol. 40, pp. 38–48, 2015.

[19] W. Jones, A. Chawdhary, and A. King, "Revisiting Volgenant-Jonker for approximating graph edit distance," in *Graph-Based Representations in Pattern Recognition*, ser. LNCS, vol. 9069, 2015, pp. 98–107.