Accepted manuscript
Pattern Recognition Letters, Special Issue on Advances in Graph-based Pattern Recognition

Journal version available at http://dx.doi.org/10.1016/j.patrec.2016.10.001
Research report associated to this publication: http://arxiv.org/abs/1512.07494

Graph Edit Distance as a Quadratic Assignment Problem

Sébastien Bougleux[a], Luc Brun[a], Vincenzo Carletti[a,b], Pasquale Foggia[b], Benoit Gaüzère[c], Mario Vento[b]

[a]*Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC UMR 6072, 6 Bd Maréchal Juin, 14050 Caen, France*
[b]*MIVIA, Dept. of Information, Electrical Engineering and Applied Mathematics, University of Salerno, via Giovanni Paolo II, 84084 Fisciano (SA), Italy*
[c]*Normandie Univ, INSA de Rouen, LITIS (EA 4108), Avenue de l'Université, 76801 Saint-Étienne du Rouvray, France*

ABSTRACT

The Graph Edit Distance (GED) is a flexible measure of dissimilarity between graphs which arises in error-correcting graph matching. It is defined from an optimal sequence of edit operations (edit path) transforming one graph into another. Unfortunately, the exact computation of this measure is NP-hard. In the last decade, several approaches were proposed to approximate the GED in polynomial time, mainly by solving linear programming problems. Among them, the bipartite GED received much attention. It is deduced from a linear sum assignment of the nodes of the two graphs, which can be efficiently computed by Hungarian-type algorithms. However, edit operations on nodes and edges are not handled simultaneously, which limits the accuracy of the approximation. To overcome this limitation, we propose to extend the linear assignment model to a quadratic one. This is achieved through the definition of a family of edit paths induced by assignments between nodes. We formally show that the GED, restricted to the paths in this family, is equivalent to a quadratic assignment problem. Since this problem is NP-hard, we propose to compute an approximate solution by adapting two algorithms: Integer Projected Fixed Point method and Graduated Non Convexity and Concavity Procedure. Experiments show that the proposed approach is generally able to reach a more accurate approximation of the exact GED than the bipartite GED, with a computational cost that is still affordable for graphs of non trivial sizes.

1. Introduction

The definition of efficient and general similarity or dissimilarity measures between attributed graphs is a key problem in structural pattern recognition. This problem is nicely addressed by the graph edit distance (GED), which constitutes one of the most flexible dissimilarity measures between attributed graphs (Tsai and Fu, 1979; Bunke and Allermann, 1983; Sanfeliu and Fu, 1983; Bunke, 1999; Neuhaus and Bunke, 2007a; Solé-Ribalta et al., 2012; Riesen, 2015). The GED is based on the notion of edit path, which is defined as a sequence of edit operations transforming a graph into another. In this paper, graphs are assumed to be simple and attributed, and edit operations are restricted to be elementary: insertion or removal of a node or an arc, or substitution of a label. Each edit operation o_i is penalized by a real non-negative cost $c_e(o_i)$. An edit path $P = (o_1, \ldots, o_k)$ is then associated to an amount of distortion required to transform a graph G_1 into a graph G_2. This amount is defined as the sum of the costs of its edit operations:

$$\gamma(P) = \sum_{i=1}^{k} c_e(o_i). \tag{1}$$

An edit path having a minimal cost, among all edit paths transforming G_1 into G_2, is called a minimal edit path. Its cost represents the minimal amount of distortion required to transform G_1 into G_2, and is called the graph edit distance from G_1 to G_2.

Computing the GED is thus a minimal path problem. This problem is NP-complete. It is classically solved by retrieving a minimal edit path with a tree search algorithm, for instance the A^*-based algorithm described by Neuhaus et al. (2006). Such algorithms have an exponential time complexity, so they can be applied on very small graphs only, generally composed of no more than 12 nodes. Several heuristics have been proposed to improve the execution time of the A^*-based algorithm (Neuhaus et al., 2006; Riesen et al., 2007a; Fischer et al., 2014). These heuristics generally lead to suboptimal estimations of the GED.

Another class of methods consists in rewriting the GED as a quadratic or a linear program. Justice and Hero (2006) proposed to model the GED as a binary linear program (BLP), restricted to undirected graphs with labeled nodes and unlabeled edges. The relaxation of this BLP provides a lower bound of the GED which cannot be readily associated to an edit path. An upper bound is also obtained by restricting edit operations to nodes only. It is expressed as a square linear sum assignment problem (LSAP), *i.e.* a minimal-cost bipartite graph matching problem, where the matching of two nodes corresponds to a substitution, a removal or an insertion. Such a bipartite matching should be called an error-correcting set matching. While this problem can be efficiently solved in polynomial time complexity, for instance with the Hungarian algorithm, it does not include any structural information. This linear approximation of the GED has been greatly improved by Riesen et al. (2007b) and Riesen and Bunke (2009) by considering graphs with attributed nodes and edges, and by replacing the cost of editing a node by the cost of editing the 1-star graph centered at this node. In particular, the cost of substituting a star graph by another one is also expressed as the solution of a LSAP. The resulting approximation, called the bipartite GED, has been extended and improved in several ways (Fankhauser et al., 2011; Serratosa, 2014; Riesen et al., 2014a; Riesen and Bunke, 2015; Riesen et al., 2015; Ferrer et al., 2015). For instance by considering larger substructures or bags of substructures (Zeng et al., 2009), (Gaüzère et al., 2014), Carletti et al. (2015), or by improving the resulting edit path by genetic algorithms (Riesen et al., 2014b).

Since the bipartite GED is based on a linear approximation of the GED, the quality of the approximation is inherently limited. To fully address the GED problem both node and arc assignments should be considered simultaneously. Indeed, operations performed on nodes may induce operations on arcs. For example, a node removal induces the removal of all its incident arcs. Our optimization process should thus not take only into account the individual costs of node operations but also the relationships between these node operations and the operations applied on the pairs of nodes encoding arcs. This pairwise constraint on nodes is closely related to the one involved in the graph matching problem. It is known that graph matching problems, and more generally problems that incorporate pairwise constraints, can be cast as a quadratic assignment problem (QAP) (Koopmans and Beckmann, 1957; Lawler, 1963, 1976; Loiola et al., 2007; Burkard et al., 2009). These problems are usually NP-hard and different relaxation algorithms have been proposed to find approximate solutions. We can quote among many examples, the Integer Projected Fixed Point (IPFP) (Leordeanu et al., 2009), the Graduated Non Convexity and Concavity Procedure (GNCCP) (Liu and Qiao, 2014) or the Factorized Graph Matching (Zhou and De la Torre, 2012).

The GED has been approximated as a binary quadratic problem by Neuhaus and Bunke (2007b) which restricts edit operations to node substitutions and edge operations induced by these substitutions. Node removals and insertions are thus not taken into account in the optimization process. Moreover, as the problem is relaxed and the result of the minimization process does not provide an edit path with a minimal cost, but a set of fuzzy edit paths. A greedy procedure is then proposed to obtain an edit path, the cost of which provides an approximation of the GED.

In this paper, we extend the linear framework proposed by Riesen et al. (2007b); Riesen and Bunke (2009) to a quadratic one. First, preliminary results concerning edit paths are established (Sec. 2). These results allow us to formalize the relationships between LSAP (Sec. 3) or QAP (Sec. 4) and such paths. In particular, we show that the GED is a QAP when graphs are simple. Then, we propose to adapt IPFP and GNCCP algorithms to the approximation of the GED (Section 5). Experimental results (Sec. 6) show that the proposed approach generally provides more accurate approximations than bipartite GED and than the quadratic approach of (Neuhaus and Bunke, 2007b), with a computational cost still affordable for graphs of non trivial sizes.

2. Restricted edit paths

We assume that graphs are simple, directed or undirected, and attributed. In order not to overload our notations, attributes are implicitly taken into account by cost functions defining the graph edit distance. A graph G is thus simply defined by a pair $G = (V, E)$, assuming the implicit definition of attributes on the set of nodes V and the set of arcs E.

2.1. Edit operations, edit paths and graph edit distance

A graph $G_1 = (V_1, E_1)$ can be transformed into another graph by elementary edit operations consisting of removals, substitutions and insertions of both nodes and arcs. However, in order to insure that any sequence of edit operations produces a valid graph without dangling arcs we have to introduce the two following constraints:

- C1 A node can be removed only if its incident arcs have been previously removed.
- C2 An edge can be inserted only if its terminal nodes exist or have been previously inserted.

A sequence of edit operations fulfilling the above constraints is called an edit path. Let $\mathcal{P}(G_1, G_2)$ be the set of edit paths transforming G_1 into G_2. As mentioned in Section 1, each elementary edit operation is associated to a cost. The cost of an edit path $P \in \mathcal{P}(G_1, G_2)$ is simply defined as:

$$\gamma_c(P) = \sum_{o \in P} c(o), \tag{2}$$

Intuitively $\gamma_c(P)$ measures the amount of distortions required by P in order to transform G_1 into G_2. A path having a minimal cost is called a minimal edit path. This last cost which corresponds to the minimal amount of distortion required to transform G_1 into G_2 is called the graph edit edit distance (GED) from G_1 to G_2:

$$\mathrm{GED}(G_1, G_2, c) = \min_{P \in \mathcal{P}(G_1, G_2)} \gamma_c(P). \tag{3}$$

Using positive costs, edit paths only constrained by C1 and C2 may contain many sequences of edit operations which can

not lead to a minimal edit path. In order to avoid such sub sequences we additionally introduce the following constraints:

C3. A node or an arc cannot be substituted and then removed.
C4. A node or an arc cannot be inserted and then substituted.
C5. A node or an arc cannot be inserted and then removed.
C6. A node or an arc is substituted at most once.

Constraints C3 to C6 forbid to apply an operation the effect of which will be partially or completely removed by an ulterior operation. For example, constraint C4 forbids to insert an element with a wrong label and then to correct it by a substitution. Let $\mathcal{P}_m(G_1, G_2)$ be the set of edit paths transforming G_1 into G_2, and satisfying constraints C1 to C6. This set of edit paths still includes paths having a minimal cost. Given an edit path P, constraints C3 to C6 allow to reorder the sequence of edit operations in P into an equivalent sequence (R, S, I) which first performs all removals (R), then all substitutions (S) and finally all insertions (I) (Bougleux et al. (2015)). The order in each sub sequences are deduced from the one of P.

$$G_1 \xrightarrow{R} \hat{G}_1 \xrightarrow{S} \hat{G}_2 \xrightarrow{I} G_2 \tag{4}$$

By construction, $\hat{G}_1 = (\hat{V}_1, \hat{E}_1)$ is a subgraph of G_1, while $\hat{G}_2 = (\hat{V}_2, \hat{E}_2)$ is a subgraph of G_2. The set of substitutions S induces a structural isomorphism between $\hat{G}_1$ and $\hat{G}_2$. We have thus in particular a bijective mapping $\varphi : \hat{V}_1 \to \hat{V}_2$ establishing the correspondence between nodes of both subgraphs. Moreover, the nodes removed by P correspond to $V_1 \setminus \hat{V}_1$, and the nodes inserted by P correspond to $V_2 \setminus \hat{V}_2$. Therefore, all the operations applied on nodes are encoded by φ, this last property being independent from edit costs.

However, such a correspondence between node operations and the mapping function φ does not work for arcs. Indeed, let us consider two substituted nodes u_i and u_j belonging to $\hat{V}_1$ such that $\varphi(u_i) = v_k$ and $\varphi(u_j) = v_l$. Let us further suppose that $e_{ij} = (u_i, u_j) \in E_1$ and $e_{kl} = (v_k, v_l) \in E_2$. We have then two solutions to transform e_{ij} into e_{kl} while still fulfilling constraints C_1 to $C6$:

- $e_{i,j}$ is substituted by $e_{k,l}$, or remains unchanged, or
- $e_{i,j}$ is removed and then $e_{k,l}$ is inserted.

So several edit paths of $\mathcal{P}_m(G_1, G_2)$ are induced by φ. In order to remove such ambiguities, the set of all possible edit paths must be further restricted.

2.2. Restricted edit paths

Let G_1 and G_2 be two graphs, and let P be an edit path of $\mathcal{P}_m(G_1, G_2)$ (satisfying constraints C1 to C6). Path P is *restricted* if it also matches the following constraint:

C7. Any arc between two substituted nodes cannot be removed and then inserted.

Such an additional constraint removes the previous ambiguity by forcing the substitution of an edge when an ambiguity occurs. Let $\mathcal{P}_r(G_1, G_2)$ be the set of all restricted edit paths from G_1 to G_2. Then the following property (Bougleux et al., 2015) shows the main interest in using the additional constraint $C7$:

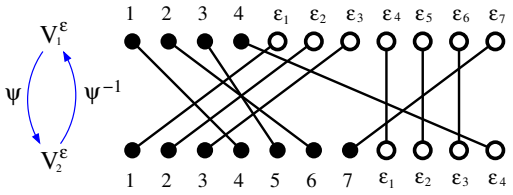


Fig. 1. A node assignment taking into account removals/insertions.

Proposition 1. *There is a one-to-one correspondence between the set of restricted edit paths $\mathcal{P}_r(G_1, G_2)$ and the set of injective functions from a subset of V_1 to V_2.*

Note that such an injective function φ defines a bijection from from a subset $\hat{V}_1$ of V_1 onto a subset $\hat{V}_2 = \varphi[V_1]$ of V_2.

Constraint C7 restricts the set of edit paths by forcing the substitution of some edges instead of their removal followed by their reinsertions. However, using some specific cost functions, the removal of an edge followed by its re insertion may be cheaper than its substitution. In such a case the computation of the minimal restricted edit path will not coincide with the same computation on the whole set of edit paths. However, this last drawback may be easily avoided by setting a new edge substitution cost equal to the minimum between the initial edge substitution cost and the sum of the costs of an edge removal followed by an edge insertion. In the rest of the paper, we assume that this initial setting is always performed when needed. Note that this operation is only applied on edge costs.

The two following sections show the relation between restricted edit paths and assignment problems.

3. ϵ-assignments, LSAP, bipartite GED and edit paths

As introduced in Section 1, the GED can be approximated by solving LSAPs (Riesen et al., 2007b; Riesen and Bunke, 2009, 2015). Since this framework is extended to quadratic assignments in the following section, it is first detailed and linked to restricted edit paths.

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs, with $V_1 = \{1, \ldots, n\}$ and $V_2 = \{1, \ldots, m\}$ to simplify the forthcoming expressions. Following Riesen and Bunke (2009) and in order to apply removal or insertion operation on nodes, node sets are augmented by dummy elements. The removal of each node $i \in V_1$ is modeled as a mapping $i \to \epsilon_i$ where ϵ_i is the dummy element associated with i. As a consequence, the set V_2 is increased by n dummy elements $\mathcal{E}_2 = \{\epsilon_1, \ldots, \epsilon_n\}$ to form the set $V_2^\epsilon = V_2 \cup \mathcal{E}_2$. The node set V_1 is increased similarly by m dummy elements $\mathcal{E}_1 = \{\epsilon_1, \ldots, \epsilon_m\}$ to represent the insertion of each node $j \in V_2$ as a mapping $\epsilon_j \to j$. This defines the set $V_1^\epsilon = V_1 \cup \mathcal{E}_1$. Note that these two sets have the same cardinality $n + m$ (Fig. 1). We assume, without loss of generality, that symbols ϵ_i and ϵ_j represent integers, *i.e.* $\mathcal{E}_1 = \{n + 1, \ldots, n + m\}$ and $\mathcal{E}_2 = \{m + 1, \ldots, m + n\}$.

It is now possible to define a bijective mapping $\psi : V_1^\epsilon \to V_2^\epsilon$, here a permutation, such that for each element of V_1^ϵ one of the four following cases occurs:

1. Substitutions: $\psi(i) = j$ with $(i, j) \in V_1 \times V_2$.

2. Removals: $\psi(i) = \epsilon_i$ with $i \in V_1$.
3. Insertions: $\psi(\epsilon_j) = j$ with $j \in V_2$.
4. Finally $\psi(\epsilon_j) = \epsilon_i$ allow to complete the bijective property of ψ, and thus should be ignored. This occurs when $i \in V_1$ and $j \in V_2$ are both substituted.

Note that mappings $i \to \epsilon_k$ with $k \neq i$, and mappings $\epsilon_j \to l$ with $l \neq j$, are forbidden. We call such a bijective mapping an ϵ-assignment, and the set of all ϵ-assignments from V_1^ϵ onto V_2^ϵ is denoted by $\Psi_\epsilon(V_1, V_2)$.

The selection of a relevant ϵ-assignment is achieved through the design of a pairwise cost function adapted to edit operations on nodes. To this, each possible mapping of an element $i \in V_1^\epsilon$ to an element $j \in V_2^\epsilon$ is penalized by a non-negative cost $c_{i,j}$. All the costs can be encoded by a $(n+m) \times (m+n)$ matrix

$$\mathbf{C} = \left(\begin{array}{c|c|c} & 1 \cdots m & \epsilon_1 \cdots \epsilon_n \\ \hline 1 & & \\ \vdots & \mathbf{C}^{\text{sub}} & \mathbf{C}^{\text{rem}} \\ n & & \\ \hline \epsilon_1 & & \\ \vdots & \mathbf{C}^{\text{ins}} & \mathbf{0}_{m,n} \\ \epsilon_m & & \end{array}\right) \tag{5}$$

where the matrix $\mathbf{C}^{\text{sub}} \in [0, +\infty)^{n \times m}$ encodes substitution costs, $\mathbf{C}^{\text{rem}} \in [0, +\infty)^{n \times n}$ encodes removal costs, and $\mathbf{C}^{\text{ins}} \in [0, +\infty)^{m \times m}$ encodes insertion costs. According to cases 2 and 3 above, off-diagonal values of $\mathbf{C}^{\text{rem}}$ and $\mathbf{C}^{\text{ins}}$ are typically set to a large value ω satisfying $\max\{c_{i,\psi(i)} \mid \forall i, \forall \psi \in \Psi_\epsilon(V_1, V_2)\} \ll \omega < +\infty$, in order to avoid forbidden mappings. Finally, according to case 4, the mapping of any ϵ_i to an ϵ_j should not induce any cost, so the last block of $\mathbf{C}$ is set to the null matrix $\mathbf{0}_{m,n}$.

The cost of an ϵ-assignment ψ is defined as the sum of all pairwise costs $c_{i,\psi(i)}$. The bipartite GED is then defined in two main steps (Riesen et al., 2007b; Riesen and Bunke, 2009). First, an ϵ-assignment having a minimal cost, among all possible ϵ-assignments, is determined:

$$\hat{\psi} \in \underset{\psi}{\text{argmin}} \left\{ \sum_{i=1}^{n+m} c_{i,\psi(i)} \mid \psi \in \Psi_\epsilon(V_1, V_2) \right\} \tag{6}$$

In a second step, an edit path is defined from the optimal ϵ-assignment $\hat{\psi}$. The cost of this path defines the bipartite GED from G_1 to G_2. This second step being automatic, the cost of the final edit path and thus the relevance of the bipartite GED is determined by the definition of the pairwise costs encoded by $\mathbf{C}$. As introduced in Section 1, a common solution to guide the LSAP step towards the construction of a low cost edit path consists in attaching to each node information about its neighborhood through the construction of bags of patterns attached to each node. The matrix $\mathbf{C}$ is then designed to encode the costs related to the substitution, removal or insertion of these bags of patterns.

Eq. 6 corresponds to a LSAP which can be efficiently solved in polynomial time complexity by using Hungarian-type algorithms (see (Burkard et al., 2009) for more details). In order to apply such algorithms we first need to write ϵ-assignments in matrix form. Since an ϵ-assignment ψ corresponds to a bijection from V_1^ϵ to V_2^ϵ, it can be represented by a permutation matrix $\mathbf{X} \in \{0,1\}^{(n+m)\times(n+m)}$, such that $x_{i,j} = 1$ iff $j = \psi(j)$, and 0 otherwise. Remember that a $(n+m) \times (n+m)$ permutation matrix $\mathbf{X}$ satisfies the doubly stochastic and binary constraints:

$$(\mathbf{X}\mathbf{1}_{n+m} = \mathbf{1}_{n+m}) \wedge \left(\mathbf{X}^T\mathbf{1}_{n+m} = \mathbf{1}_{n+m}\right) \wedge \left(\mathbf{X} \in \{0,1\}^{(n+m)\times(n+m)}\right). \tag{7}$$

The permutation matrix associated to an ϵ-assignment has the following form:

$$\mathbf{X} = \begin{pmatrix} \mathbf{X}^{\text{sub}} & \mathbf{X}^{\text{rem}} \\ \mathbf{X}^{\text{ins}} & \mathbf{X}^{\epsilon} \end{pmatrix} \in \{0,1\}^{(n+m)\times(n+m)}, \tag{8}$$

where matrix $\mathbf{X}^{\text{sub}} \in \{0,1\}^{n \times m}$ encodes node substitutions, $\mathbf{X}^{\text{rem}} \in \{0,1\}^{n \times n}$ encodes node removals, and $\mathbf{X}^{\text{ins}} \in \{0,1\}^{m \times m}$ encodes node insertions. Matrix $\mathbf{X}^{\epsilon} \in \{0,1\}^{m \times n}$ is an auxiliary matrix (case 4 above), it ensures that $\mathbf{X}$ is a permutation matrix. Due to the constraints on dummy nodes (cases 2 and 3 above) matrices $\mathbf{X}^{\text{rem}}$ and $\mathbf{X}^{\text{ins}}$ always satisfy:

$$\begin{aligned} &\forall (i,j) \in \{1,\ldots,n\}^2, i \neq j, \quad x^{\text{rem}}_{i,j} = 0 \\ &\forall (i,j) \in \{1,\ldots,m\}^2, i \neq j, \quad x^{\text{ins}}_{i,j} = 0 \end{aligned} \tag{9}$$

In this paper, a $(n+m) \times (m+n)$ matrix satisfying equations 7, 8 and 9 is called an ϵ-assignment matrix. The set of all ϵ-assignment matrices is denoted by $\mathcal{A}_{n,m}$.

Auxiliary matrix $\mathbf{X}^\epsilon$ suggests the definition of an equivalence relation between ϵ-assignment matrices. Two ϵ-assignment matrices $\mathbf{X}_1$ and $\mathbf{X}_2$, defined by the two sequences of block matrices $(\mathbf{X}_1^{\text{sub}}, \mathbf{X}_1^{\text{rem}}, \mathbf{X}_1^{\text{ins}}, \mathbf{X}_1^{\epsilon})$ and $(\mathbf{X}_2^{\text{sub}}, \mathbf{X}_2^{\text{rem}}, \mathbf{X}_2^{\text{ins}}, \mathbf{X}_2^{\epsilon})$, are equivalent if and only if

$$(\mathbf{X}_1^{\text{sub}} = \mathbf{X}_2^{\text{sub}}) \wedge (\mathbf{X}_1^{\text{rem}} = \mathbf{X}_2^{\text{rem}}) \wedge (\mathbf{X}_1^{\text{ins}} = \mathbf{X}_2^{\text{ins}}). \tag{10}$$

The set of ϵ-assignment matrices up to this equivalence relation is denoted by $\tilde{\mathcal{A}}_{n,m}$ ($\mathbf{X}^\epsilon$ is not considered). Note that this relation does not depend on edit costs.

The relation between ϵ-assignment matrices and mapping functions between two node sets V_1 and V_2 is clarified as follows. The proof can be found in (Bougleux et al., 2015).

Proposition 2. *There is a one-to-one relation between $\tilde{\mathcal{A}}_{n,m}$ and the set of injective functions from a subset of V_1 to V_2.*

By using Proposition 1, we can connect such a mapping to a restricted edit path from G_1 to G_2. Up to the equivalence relation (Eq. 10), there is thus a one-to-one correspondence between ϵ-assignment matrices and restricted edit paths.

Also, note that two equivalent ϵ-assignment matrices have a same cost since the block associated to $\mathbf{X}^\epsilon$ in $\mathbf{C}$ is equal to $\mathbf{0}_{m,n}$. We consider the vectorized version of the LSAP given by:

$$\underset{\mathbf{x}}{\text{argmin}} \left\{ \mathbf{c}^T\mathbf{x} \mid \mathbf{x} \in \text{vec}[\mathcal{A}_{n,m}] \right\} \tag{11}$$

where vector $\mathbf{c} = \text{vec}(\mathbf{C}) \in [0, +\infty]^{(n+m)^2}$ is the vectorization of cost matrix $\mathbf{C}$ obtained by concatenating its rows, similarly $\mathbf{x} = \text{vec}(\mathbf{X}) \in \{0,1\}^{(n+m)^2}$, and $\text{vec}[\mathcal{A}_{n,m}] \subset \{0,1\}^{(n+m)^2}$ is the set of all vectorized ϵ-assignment matrices.

In our experiments, we have used a $O(n^3)$ version of the Hungarian algorithm that explictly takes into account Eq. 9, instead of imposing large ω values.

4. GED as a quadratic assignment problem

As mentioned in Section 3 bipartite GED methods deduce the final edit path and thus the set of arc edit operations from an LSAP defining an assignment between the nodes of both graphs. However, the simultaneous assignment of two adjacent nodes should obviously have an influence on the operation applied on the arc connecting them. In return, the cost of this arc operation should influence the type of operation applied on its incident nodes. This is not the case in bipartite GED methods which thus find an optimal solution of an approximation of the initial problem. Taking into account simultaneously node and arc assignments can be formalized as a quadratic assignment problem. In this section, we propose to extend the linear model to a quadratic one based on ϵ-assignments, and we show that this model corresponds to the cost of a restricted edit path.

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs, and let $\psi \in \Psi_\epsilon(V_1, V_2)$ be an ϵ-assignment as defined in Section 3. When a pair $(i, j) \in V_1^\epsilon \times V_1^\epsilon$ is assigned by ψ to a pair $(\psi(i), \psi(j)) \in V_2^\epsilon \times V_2^\epsilon$, one of the following cases occurs:

1. Arc substitution: $(\psi(i), \psi(j)) \in E_2$ with $(i, j) \in E_1$.
2. Arc removal: $(\psi(i), \psi(j)) \notin E_2$ with $(i, j) \in E_1$.
3. Arc insertion: $(\psi(i), \psi(j)) \in E_2$ with $(i, j) \notin E_1$.
4. Finally $(\psi(i), \psi(j)) \notin E_2$ with $(i, j) \notin E_1$ allows to complete the bijection property.

Note that by definition any vertex of $\mathcal{E}_1$ (resp. $\mathcal{E}_2$) is not adjacent to any vertex of V_1 (resp. V_2).

Each possible simultaneous mapping of nodes $i, j \in V_1^\epsilon$ onto respectively nodes k and l in V_2^ϵ, is penalized by a non-negative cost $d_{ik,jl}$ which depends on the underlying edit operation described by one of the cases above. The overall arc cost associated to a simultaneous node assignment is then measured by:

$$d(\psi) = \sum_{i=1}^{n+m} \sum_{j=1}^{n+m} d_{i\psi(i),j\psi(j)}, \tag{12}$$

where cost values are defined as follows.

Remember that all mappings from a node of V_1^ϵ to a node of V_2^ϵ are not allowed. Indeed (Section 3), $i \to \epsilon_j$ with $i \in V_1$ and $j \neq i$, and reciprocally $\epsilon_k \to l$ with $l \in V_2$ and $k \neq l$ are forbidden. Then, a simultaneous node mapping involving at least one of these two cases is also forbidden. We denote by $\nrightarrow$ a forbidden mapping. As in Section 3, the cost is set to a (large) value ω in this case.

For any other simultaneous node mapping $(i \to k, j \to l)$, with $i, j \in V_1^\epsilon$ and $k, l \in V_2^\epsilon$, its cost depends on the presence or the absence of arcs $(i, j) \in E_1$ and $(k, l) \in E_2$:

- If $(i, j) \in E_1$ and $(k, l) \in E_2$ then $d_{ik,jl}$ is the cost of the arc assignment $(i, j) \to (k, l)$, *i.e.* arc substitution.
- If $(i, j) \in E_1$ and $(k, l) \notin E_2$ then $d_{ik,jl}$ is the cost of removing the arc (i, j).
- If $(i, j) \notin E_1$ and $(k, l) \in E_2$ then $d_{ik,jl}$ is the cost of inserting the arc (k, l).
- Else, the simultaneous mapping must not influence the overall cost and so its cost is set to 0.

By using the edit cost function c_e, the cost of an allowed simultaneous node mapping is then defined by

$$\begin{aligned} c_e(i \to k, j \to l) = {} & c_e\left((i, j) \to (k, l)\right) \delta_{(i,j)\in E_1} \delta_{(k,l)\in E_2} \\ & + c_e\left((i, j) \to \epsilon\right) \delta_{(i,j)\in E_1} (1 - \delta_{(k,l)\in E_2}) \\ & + c_e\left(\epsilon \to (k, l)\right) (1 - \delta_{(i,j)\in E_1}) \delta_{(k,l)\in E_2} \\ & \text{with } (i \neq j) \wedge (k \neq l) \end{aligned} \tag{13}$$

where $\delta_{e\in E} = 1$ if $e \in E$ and 0 otherwise, $(i, j) \to \epsilon$ denotes arc removal and $\epsilon \to (k, l)$ denotes arc insertion. When $i = k$ or $j = l$, then $d_{ik,ik} = 0$. Note also that the potential symmetry of $c_e(i \to k, j \to l)$ depends both on the one of edit operations and on the symmetry of graphs when they are directed.

Finally, the cost of a simultaneous node mapping is given by

$$d_{ik,jl} = \begin{cases} \omega & \text{if } (i \nrightarrow k) \vee (j \nrightarrow l) \\ c_e(i \to k, j \to l) & \text{else} \end{cases} \tag{14}$$

Let $\mathbf{x} \in \text{vec}[\mathcal{A}_{n,m}] \subset \{0, 1\}^{(n+m)^2}$ be the vectorization of the ϵ-assignment matrix associated to ψ. All costs can be represented by a $(n+m)^2 \times (n+m)^2$ matrix $\mathbf{D} = (d_{ik,jl})_{i,k,j,l}$ with $d_{ik,jl} x_{ik} x_{jl} = d_{i\psi(i),j\psi(j)}$ if $x_{ik} = x_{jl} = 1$, and 0 else. So each row and each column of $\mathbf{D}$, and $\mathbf{x}$, have the same organization of pairwise indices, and then the total cost of the simultaneous node assignment can be written in quadratic form as:

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

To fully represent edit operations, the cost $\mathbf{c}^T\mathbf{x}$ of the mapping between nodes (Sec. 3) is also considered. Here costs are defined directly from the initial edit cost function:

$$\begin{aligned} c_{i,k}^{\text{sub}} &= c_e(i \to k) \\ c_{i,k}^{\text{rem}} &= \begin{cases} c_e(i \to \epsilon_i) & \text{if } k = i \\ \omega & \text{else} \end{cases} \\ c_{i,k}^{\text{ins}} &= \begin{cases} c_e(\epsilon_k \to k) & \text{if } i = k \\ \omega & \text{else} \end{cases} \end{aligned} \tag{15}$$

According to the following result, summing the quadratic and the linear costs defined above leads to the cost of a restricted edit path, see (Bougleux et al., 2015) for a proof.

Proposition 3. *Given an ϵ-assignment $\mathbf{x} \in vec[\mathcal{A}_{n,m}]$, any non-infinite value of $\frac{1}{2}\mathbf{x}^T\Delta\mathbf{x} + \mathbf{c}^T\mathbf{x}$ is equal to the cost of a restricted edit path, and conversely, where $\Delta = \mathbf{D} + \mathbf{D}^T$ if graphs are directed, or $\Delta = \mathbf{D}$ if graphs are undirected.*

Hence, the determination of a restricted edit path with a minimal cost is equivalent to find an ϵ-assignment having a minimal quadratic cost. In other words, for the class of graphs under consideration, *i.e.* simple graphs, we have

$$\text{GED}(G_1, G_2) = \min_{\mathbf{x}} \left\{ \frac{1}{2}\mathbf{x}^T\Delta\mathbf{x} + \mathbf{c}^T\mathbf{x} \mid \mathbf{x} \in \text{vec}[\mathcal{A}_{n,m}] \right\} \tag{16}$$

This is a QAP, see (Lawler, 1963; Burkard et al., 2009) for more details on QAPs. In particular, QAPs are NP-hard and exact algorithms can solve QAPs of small size only. So, many heuristics enable to find suboptimal solutions in short computing time have been explored.

Algorithm 1 IPFP($\mathbf{x}, \mathbf{c}, \Delta$)

1: **while** a fixed point is not reached **do**
2: $\quad \mathbf{b}^* \leftarrow \text{argmin}\{(\mathbf{x}^T\Delta + \mathbf{c}^T)\mathbf{b} \mid \mathbf{Ab} = \mathbf{1},\ \mathbf{b} \in \{0,1\}^{(n+m)^2}\}$
3: $\quad t^* \leftarrow \text{argmin}\{S(\mathbf{x} + t(\mathbf{b}^* - \mathbf{x})) \mid t \in [0,1]\}$
4: $\quad \mathbf{x} \leftarrow \mathbf{x} + t^*(\mathbf{b}^* - \mathbf{x})$
5: **end while**

Let's recall that graphs G_1 and G_2 may be both directed or both undirected. When graphs are directed, Note that Δ is symmetric by definition (Proposition 3). When graphs are undirected, as long as the edit cost function is symmetric, $\mathbf{D}$ and thus Δ are always symmetric (Bougleux et al., 2015). So, the same algorithms can be used to approximate the GED of undirected graphs and the GED of directed ones.

5. Algorithms approximating the GED

5.1. IPFP algorithm

Integer Projected Fixed Point (IPFP) is an algorithm initially proposed by Leordeanu et al. (2009) to find a solution to the quadratic assignment problem in the context of weighted graph matching and MAP inference (maximization problems). The adaptation of this algorithm to the GED consists in finding a permutation (from Eq. 16):

$$\underset{\mathbf{x}}{\text{argmin}} \left\{ S(\mathbf{x}) \overset{\text{def.}}{=} \frac{1}{2}\mathbf{x}^T\Delta\mathbf{x} + \mathbf{c}^T\mathbf{x} \mid \mathbf{Ax} = \mathbf{1},\ \mathbf{x} \in \{0,1\}^{(n+m)^2} \right\} \tag{17}$$

where $\mathbf{Ax} = \mathbf{1}_{2(n+m)}$, with $\mathbf{A} \in \{0,1\}^{2(n+m)\times(n+m)^2}$, is the matrix version of the bijectivity constraints given by Eq. 7.

A common approach to find an approximate solution to a QAP consists in relaxing the constraints by searching for a continuous solution instead of a discrete one:

$$\underset{\mathbf{x}}{\text{argmin}} \left\{ S(\mathbf{x}) \mid \mathbf{Ax} = \mathbf{1},\ \mathbf{x} \geq \mathbf{0}_{(n+m)^2} \right\}. \tag{18}$$

The algorithm proposed by Leordeanu et al. (2009) tries to find a solution to both problems. Given an initial continuous or binary candidate solution $\mathbf{x}_0$, it improves (here reduces) iteratively the corresponding quadratic cost S in two steps, adapted here to the GED (Alg. 1).

In the first step (line 2), the quadratic function S is linearly approximated by its 1st-order expansion around the current solution $\mathbf{x}$: $S(\mathbf{b}) \approx S(\mathbf{x}) + (\mathbf{x}^T\Delta + \mathbf{c}^T)(\mathbf{b} - \mathbf{x})$, with $\mathbf{b} \geq \mathbf{0}$. Keeping $\mathbf{x}$ fixed, the minimization of $S(\mathbf{b})$ is approximatively equivalent to the minimization of $(\mathbf{x}^T\Delta + \mathbf{c}^T)\mathbf{b}$, which is a relaxed LSAP ($\mathbf{b} \geq \mathbf{0}$). As any solution to the relaxed LSAP is always a solution to the LSAP, *i.e.* a permutation, the next candidate solution is thus computed by solving a LSAP (line 2). The resulting optimal assignment $\mathbf{b}^*$ determines a direction of largest possible decrease of S in the 1st-order approximation.

The second step (line 3 and line 4) consists in minimizing the quadratic function S in the continuous domain along the direction given by $\mathbf{b}^*$, which reduces to compute the extremum of S on the segment joining $\mathbf{x}$ to $\mathbf{b}^*$. This can be done analytically, see Bougleux et al. (2015) for more details.

The iteration of these two steps converges to a local minimum of the relaxed problem (Eq. 18), which can be continuous or discrete. When the solution $\mathbf{x}$ is continuous (bi-stochastic), it is projected to the closest permutation by solving the LSAP $\text{argmin}\{\mathbf{x}^T\mathbf{b} \mid \mathbf{b} \in \text{vec}[\mathcal{A}_{n,m}]\}$. The final assignment allows to reconstruct an edit path and to compute the associated approximated GED, which is thus suboptimal.

Two important remarks should be put forward concerning the complexity of this algorithm. Concerning the LSAP solved in the 1st step (line 2), $\mathbf{x}^T\Delta$ can be computed in $O((n+m)^4)$ time complexity. However, this complexity reduces to $O((n+m)^2)$ if $\mathbf{x}$ is a permutation instead of being bi-stochastic. Using the Hungarian algorithm to solve the LSAP, the time complexity of step 2 is thus in $O((n+m)^3)$, like the computation of the bipartite GED with the same algorithm. Concerning the second step, it can be computed in $O((n+m)^2)$. Also, the cost matrix Δ is not stored but computed online using Eq. 13 and Eq. 14. The memory complexity of the algorithm is thus reduced to $O((n+m)^2)$ (the size of $\mathbf{x}^T\Delta$ and $\mathbf{c}^T$).

The efficiency of IPFP is closely related to the choice of the initial solution, which also influences the number of iterations required to reach the convergence. From a quadratic optimization point of view, bipartite graph edit distances (Section 3) approximate the quadratic problem by including information about arcs into the cost matrix encoding edit operations on nodes. Solutions provided by such methods, while not being optimal according to the quadratic problem, should be close to a low local minimum of the objective function S. Permutations computed by such methods can thus be refined by IPFP algorithm by using the full expression of the quadratic problem. The choice of a particular bipartite graph edit distance algorithm is discussed in Section 6.

5.2. GNCCP

Graduated NonConvexity and Concativity Procedure (GNCCP) (Liu and Qiao, 2014; Zaslavskiy et al., 2009) is a path following algorithm which aims at approximating the solution of a QAP by considering a convex-concave relaxation through the modified quadratic function:

$$S_\zeta(\mathbf{x}) = (1 - |\zeta|)S(\mathbf{x}) + \zeta\mathbf{x}^T\mathbf{x} \tag{19}$$

where $\zeta \in [-1, 1]$. When $\zeta = 1$, $S_\zeta(\mathbf{x}) = \mathbf{x}^T\mathbf{x}$ is fully convex, and when $\zeta = -1$, $S_\zeta(\mathbf{x}) = -\mathbf{x}^T\mathbf{x}$ is concave.

GNCCP algorithm (Alg. 2) starts with $\zeta = 1$, and since the minimization of $\mathbf{x}^T\mathbf{x}$ corresponds to a convex problem, the initialization step has no influence on the result of this first iteration. So, unlike IPFP algorithm, no initial mapping is required. This avoids accuracy variations induced by initialization, as observed experimentally with IPFP (Section 6).

Then GNCCP algorithm smoothly interpolates convex and concave relaxations by iteratively decreasing ζ from 1 to -1 with step size d (equal to 0.1 in all our experiments). Each iteration, corresponding to a ζ, minimizes S_ζ (Eq. 19). This is initially achieved by a Frank-Wolfe-like algorithm in (Leordeanu et al., 2009). Here we use IPFP algorithm to perform this step.

GNCCP always converges to a permutation (Liu and Qiao, 2014). Convergence is reached when $\zeta = -1$, or before, when

Algorithm 2 GNCCP($\Delta, k_{\max}$)

$\zeta = 1, d = 0.1, \mathbf{x} = \mathbf{0}$
while $(\zeta > -1) \wedge (\mathbf{x} \notin \mathcal{A}_{n,m})$ **do**
  $\Delta_\zeta \leftarrow \frac{1}{2}(1 - |\zeta|)\Delta + \zeta\mathbf{I}$
  $\mathbf{x} \leftarrow \text{IPFP}(\mathbf{x}, (1 - |\zeta|)\mathbf{c}, \Delta_\zeta)$
  $\zeta \leftarrow \zeta - d$
end while

Table 1. Characteristics of the four GREYC's chemistry datasets.

Dataset	*Number of graphs*	*Avg Size*	*Avg Degree*
Alkane	150	8.9	1.8
Acyclic	183	8.2	1.8
MAO	68	18.4	2.1
PAH	94	20.7	2.4
MUTAG	8×10	40	2

$\mathbf{x}$ is a permutation. Note that the minimum of the concave relaxation is a permutation (Zaslavskiy et al., 2009). So here, no projection step is required at the end of IPFP algorithm (see previous section).

A closely related approach has been proposed by Zhou and De la Torre (2012). However, it is based on a spectral decomposition of the quadratic matrix Δ, which increases both the execution time and the amount of memory required to store Δ (Δ is not stored as described in Section 5.1).

6. Experiments

In this section, we evaluate the two suggested methods (Section 5) based on IPFP and GNCCP algorithms through several experiments. So, we compare our approach to an exact graph edit distance method based on the A^* algorithm, three methods based on LSAP, and one method based on quadratic programming (Neuhaus and Bunke (2007b)). The exact GED is used as a baseline to measure for each method its distances to the optimum results. However, A^* is restricted to small graphs and we did not succeed to use it on all datasets due to its intractable computational time. The three LSAP-based methods, *i.e.* bipartite GEDs, are the ones proposed in Riesen and Bunke (2009); Gaüzère et al. (2014); Carletti et al. (2015). As already discussed, the definition of the costs encoding substitutions, removals and insertions of nodes constitutes the main difference between these methods.

Our experiments have been performed on four chemoinformatics datasets[1] plus one additional chemical dataset: MUTAG (Riesen and Bunke, 2008; Abu-Aisheh et al., 2015) (see Table 1 for their characteristics). The variety of these datasets allows to evaluate the different methods on four types of graphs: acyclic labeled (Acyclic), acyclic unlabeled (Alkane), cyclic labeled (MAO), cyclic unlabeled (PAH). In addition, in order to check the performances of the compared methods on larger graphs, we have performed several experiments on synthetic graphs having the same characteristics as graphs in MAO but with a size extended up to 500 nodes (details below) and on MUTAG dataset, which is divided into 7 different subsets composed of graphs having from 10 to 70 nodes, plus one subset composed of graphs having different sizes (Abu-Aisheh et al., 2015).

Table 2 shows the results of our experiments on the four GREYC's datasets. Note that, in order to avoid any bias, all the results have been computed using random permutations of adjacency matrices before computing graph edit distances. For each method and on each dataset we record the average edit distance (d), the average approximation error (e) with respect to the exact graph edit distance when available and the average computational time (t) required to get the graph edit distance for a pair of graphs. For all these measures, lower values correspond to better results. Due to the computational complexity required by A* algorithm, the exact graph edit distance has not been computed on PAH and MAO datasets which are composed of larger graphs than the ones in Acyclic and Alkane datasets.

IPFP approach allows to drastically improve the accuracy of the approximation with respect to LSAP approaches while keeping a reasonable computational time. This observation can be made on the four datasets, hence showing the consistency of this QAP approach. The results shown in Table 2 also highlight the importance of the initialization step. Indeed, as stated in Section 5, IPFP iteratively improves an initial solution. Hence, a solution close to a low minima induces a low number of iterations and a low value of the final solution. This phenomenon is observed in Table 2 where the best results of IPFP methods are obtained by using Gaüzère et al. (2014) which provides better results than Riesen and Bunke (2009). Moreover, less iterations are required to reach convergence since the algorithm is initialized close to a minima. This phenomenon explains the low differences of computational time between the different approaches. Note that we didn't combine the LSAP method presented in Carletti et al. (2015) with IPFP due to its high computational time. Compared to Neuhaus and Bunke (2007b), the IPFP method obtains better or equivalent results in much less times. Indeed, the ratio between the execution times of Neuhaus and Bunke (2007b) and the ones of IPFP initialized with Gaüzère et al. (2014) is greater than 63 on all datasets. This last point may be explained by the use of the LSAP method within IPFP method which provides a faster convergence than classical interior point methods used by Neuhaus and Bunke (2007b). The greater errors obtained by Neuhaus and Bunke (2007b) are related to the fact that this method does not incorporate node insertion and deletion operations within the optimization process. This last drawback has a large influence on the accuracy especially when graphs have different sizes. Finally, the path following encoded by the GNCCP method obtains significantly better results on three datasets over four with execution times slightly lower than the method of Neuhaus and Bunke (2007b). IPFP method initialized with Gaüzère et al. (2014) obtains better results than GNCCP on MAO dataset. We interpret this last point as follows: On this dataset and for several couples of graphs, the LSAP method Gaüzère et al. (2014) provides to IPFP an initialization close to a lowest minima than the one found by GNCCP method, which is insensitive to the

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

Table 2. Accuracy and complexity 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
A^*	15		1.29	17		6.02				
Riesen and Bunke (2009)	35	18	$\simeq 10^{-3}$	35	18	$\simeq 10^{-3}$	105	$\simeq 10^{-3}$	138	$\simeq 10^{-3}$
Gaüzère et al. (2014)	33	18	$\simeq 10^{-3}$	31	14	$\simeq 10^{-2}$	49	$\simeq 10^{-2}$	120	$\simeq 10^{-2}$
Carletti et al. (2015)	26	11	2.27	28	9	0.73	44	6.16	129	2.01
$\text{IPFP}_{\text{Random init}}$	22.6	7.1	0.007	23.4	6.1	0.006	65.2	0.031	63	0.04
$\text{IPFP}_{\text{Init Riesen and Bunke (2009)}}$	22.4	7.0	0.007	22.6	5.3	0.006	59	0.031	62.2	0.04
$\text{IPFP}_{\text{Init Gaüzère et al. (2014)}}$	20.5	5	0.006	20.7	3.4	0.005	33.6	0.016	52.5	0.037
Neuhaus and Bunke (2007b)	20.5	4.9	0.38	25.7	7.6	0.42	59.1	7	52.9	8.20
GNCCP	16.7	1.2	0.46	18.8	1.5	0.33	40.3	4.2	41.8	6.24

Table 3. Average edit distances obtained on MUTAG datasets.

Algorithm	$c_{sub} < c_{del} + c_{ins}$			$c_{sub} > c_{del} + c_{ins}$		
	MUTAG 20	MUTAG 50	MUTAG mixed	MUTAG 20	MUTAG 50	MUTAG mixed
$\text{IPFP}_{\text{Init Gaüzère et al. (2014)}}$	40.68	109.66	195.71	26.57	53.87	78.10
Neuhaus and Bunke (2007b)	33.18	89.68	231.78	38.81	90.26	122.25
GNCCP	38.08	111.34	192.90	23.82	46.55	73.73

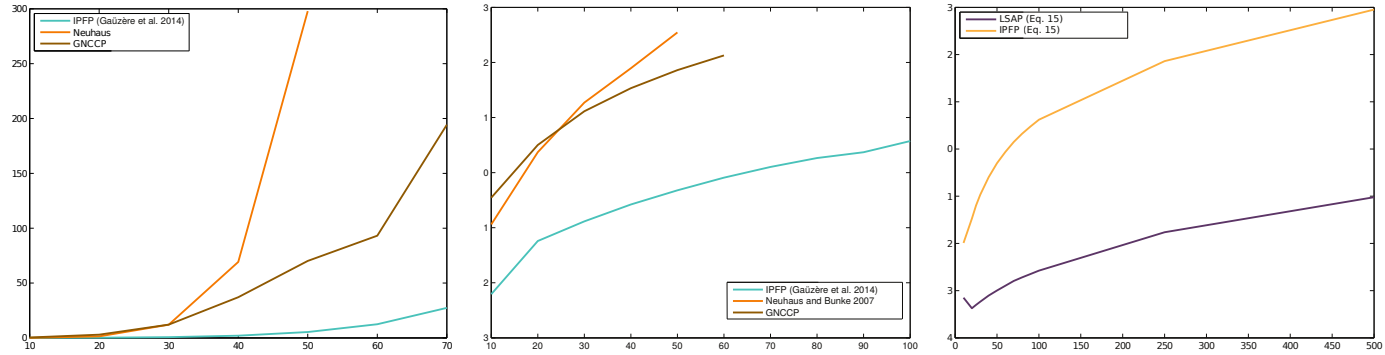
choice of a starting point.

In conclusion, these results show that the IPFP approach combined with a relevant choice of the initial solution outperforms methods based on LSAP formulation and Neuhaus and Bunke (2007b) while keeping an interesting computational time with respect to the one required to compute an exact graph edit distance. The path following approach provided by the GNCCP method allows to further improve the results provided by the IPFP method but at the cost of higher execution times since GNCCP basically iterates the IPFP method.

These conclusions are confirmed by the results obtained on MUTAG dataset. Experiments reported in Table 3 are obtained on three MUTAG databases : graphs of size 20, 50 and mixed graphs of various sizes. Neuhaus and Bunke (2007b) obtains the best results on the first two columns. However, these results are somehow biased by the fact that these two columns correspond to graph edit distances computed between graphs having a same size. Moreover, considering a substitution cost lower than the sum of insertion and deletion costs also reduces the need to consider node insertion and deletion operations. These two last points explain the good results obtained by Neuhaus and Bunke (2007b) on these columns. The remaining columns confirm this hypothesis. When considering the mixed dataset, Neuhaus method reaches a lower accuracy than the two other methods which include insertion and deletion operations of nodes within their optimization process. The second part of Table 3 shows results on the same datasets but with a substitution cost greater than the sum of the costs of insertion and deletion operations. This setting of costs increases the relevance of node insertion and removal operations and as shown in Table 3 Neuhaus and Bunke (2007b) does not provide a good approximation in this case.

Figures 2(a) and 2(b) report execution times of three methods based on a QAP approach. Namely: IPFP initialized with Gaüzère et al. (2014); Neuhaus and Bunke (2007b) and the GNCCP (Liu and Qiao (2014)). These results have been computed on a decomposition of the MUTAG dataset onto 7 subsets composed of graphs of a same size from 10 to 70 nodes (Figure 2(a)) and on synthetic datasets composed of graphs from 10 to 100 nodes (Figure 2(b)). The construction of this synthetic dataset is detailed in the next paragraph. As already observed on Table 2, IPFP approach allows to keep a reasonable computational time whereas Neuhaus and GNCCP methods require prohibitive execution times on large graphs. In particular we had to stop the execution of Neuhaus and Bunke (2007b) on graphs of size larger than 50 on MUTAG datasets due to non reasonable execution times, i.e. more than 24 hours for a single subset of 10 graphs. Figure 2(c) shows the behavior of IPFP and LSAP methods when graphs become larger. To this purpose, we computed approximate graph edit distances using IPFP and LSAP approaches on graphs up to 500 nodes. Since we are only interested in the computational times required by IPFP and LSAP methods, we used the simple cost matrices defined by Eq. 15. This cost matrix is similar to the one used by Justice and Hero (2006). This last point explains the slight difference in computational times observed between Figure 2(b) and Figure 2(c). Neuhaus and Bunke (2007b) and GNCCP approaches have been excluded from this comparison since they do not scale with graphs having more than 100 nodes. As we can see in Figure 2(c), the execution times of our IPFP method is about 900 seconds when applied on graphs of 500 nodes while it remains equal to 0.1 seconds for LSAP.

Figure 3 reports the results obtained by 4 methods: two based on LSAP : Riesen and Bunke (2009) and Gaüzère et al. (2014) and two based on IPFP using the same couple of initializations. In order to deeply analyze the behavior of these approaches, we performed our experiments on increasing graph sizes (from 10 to 100 nodes) and different amount of distortions between the pairs of graphs being compared. These results have been computed on synthetic datasets having the same node and arc

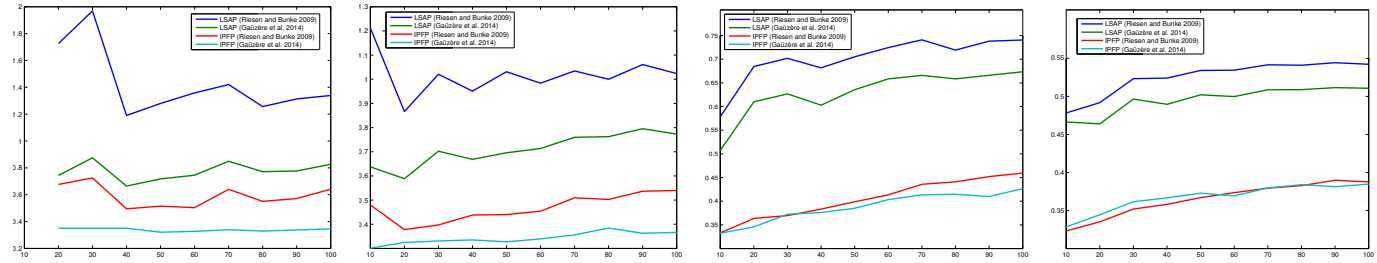


(a) Average computational times in seconds for each MUTAG dataset.

(b) Log 10 of average computationnal times on synthetic dataset (50% of alterations).

(c) Log 10 of average computationnal times on synthetic dataset up to 500 nodes.

Fig. 2. Analysis of computationnal times on MUTAG and synthetic datasets



(a) 5% of nodes altered.

(b) 10% of nodes altered.

(c) 25% of nodes altered.

(d) 50% of nodes altered.

Fig. 3. Log 10 of the ratios between the computed edit distances and an estimation of the optimal GED on synthetic graphs from 10 to 100 nodes.

labels distribution and the same ratio between the number of arcs and the number of nodes as MAO dataset but generalized to different graph sizes. For a given number of nodes, a synthetic dataset is composed of 400 pairs of source and target graphs. For each source graph we generate 4 target graphs by altering 5, 10, 25 and 50% of nodes' source graph. Half the altered nodes are removed and the other half are substituted. The overall edit distance between source and target graphs is then estimated by the costs associated to edit operations performed on nodes and the ones induced on arcs. Note that this edit distance may not be optimal.

Given this protocol, we generated 10 synthetic datasets composed of 400 pairs of graphs, based on source graphs having a same number of nodes. This last number ranges from 10 to 100 nodes. For each method and each pair of graphs, we define the error as the ratio between the GED returned by the method and the one computed from the set of distortions applied on the source graph. Curves displayed in Fig. 3 show the log_{10} of these ratios. A value equal to 1 indicates that the approximated GED is ten times higher than the exact one, and a value equal to 0 indicates a perfect approximation.

Concerning LSAP methods we can note that Gaüzère et al. (2014) work better than Riesen and Bunke (2009) for all percentages of distortion and all graph sizes. However, the gap between the curves corresponding to both LSAP methods tends to decrease as the percentage of distortions increases. We can additionally note that the mean value of both curves decreases as the distortion increases.

Concerning IPFP based methods, in all experiments reported in Figure 3 these methods work better than the ones based on LSAP. We can also note that the curve corresponding to IPFP initialized with Gaüzère et al. (2014) remains approximately constant for low distortions (5% and 10%) and increases according to the size of graphs at higher distortion rates. Finally, IPFP initialized with Gaüzère et al. (2014) obtains better results than the one initialized with Riesen and Bunke (2009) for small distortion rates, results obtained by both initialization becoming similar for high distortion rates (25% and 50%). This last point is coherent with the previous discussion on the decrease of the gap between both LSAP methods at high distortion rates. Computational times corresponding to these synthetic datasets lead to the same observations made for MUTAG and GREYC's chemical datasets (Figure 2(c)).

7. Conclusion

In this paper, we have pointed out the relation between constrained edit paths and the solutions of specific linear and quadratic assignment problems. This characterization allows to express the graph edit distance as a quadratic assignment problem with a clear definition of the family of edit paths implied in the minimization process. This expression extends the linear approximation of the graph edit distance provided by the bipartite graph edit distance. We have proposed to optimize the quadratic problem by the IPFP and the GNCCP algorithms, which provide two different and interesting ratio between time

complexity and approximation quality. These algorithms can be seen as a refinement of the bipartite graph edit distance. Through experiments, we show that these algorithms find values close to the optimal solution with computational times still affordable for graphs composed of 500 nodes.

References

Abu-Aisheh, Z., Raveaux, R., Ramel, J., 2015. A graph database repository and performance evaluation metrics for graph edit distance, in: Graph-Based Representations in Pattern Recognition, pp. 138–147.

Bougleux, S., Brun, L., Carletti, V., Foggia, P., Gaüzère, B., Vento, M., 2015. A Quadratic Assignment Formulation of the Graph Edit Distance. Technical Report. Normandie Univ, NormaSTIC, FR 3638 CNRS. France.

Bunke, H., 1999. Error correcting graph matching: on the influence of the underlying cost function. IEEE Trans. on Pattern Analysis and Machine Intelligence 21, 917–922.

Bunke, H., Allermann, G., 1983. Inexact graph matching for structural pattern recognition. Pattern Recognition Letters 1, 245–253.

Burkard, R., Dell'Amico, M., Martello, S., 2009. Assignment Problems. SIAM.

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

Fankhauser, S., Riesen, K., Bunke, H., 2011. Speeding up graph edit distance computation through fast bipartite matching, in: Graph-based Representations in Pattern Recognition. volume 6658 of *LNCS*, pp. 102–111.

Ferrer, M., Serratosa, F., Riesen, K., 2015. A first step towards exact graph edit distance using bipartite graph matching, in: Graph Based Representations in Pattern Recognition. volume 9069 of *LNCS*, pp. 77–86.

Fischer, A., Plamondon, R., Savaria, Y., Riesen, K., Bunke, H., 2014. A Hausdorff Heuristic for Efficient Computation of Graph Edit Distance. Springer Berlin Heidelberg. pp. 83–92.

Gaüzère, B., Bougleux, S., Riesen, K., Brun, L., 2014. Approximate graph edit distance guided by bipartite matching of bags of walks, in: Structural, Syntactic and Statistical Pattern Recognition. volume 8621 of *LNCS*, pp. 73–82.

Justice, D., Hero, A., 2006. A binary linear programming formulation of the graph edit distance. IEEE Trans. Pattern Anal. Mach. Intell. 28, 1200–1214.

Koopmans, T., Beckmann, M., 1957. Assignment Problems and the Location of Economic Activities. Econometrica 25, 53–76.

Lawler, E., 1963. The quadratic assignment problem. Management Sci. 9, 586–599.

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

Leordeanu, M., Hebert, M., Sukthankar, R., 2009. An integer projected fixed point method for graph matching and map inference, in: Advances in Neural Information Processing Systems. volume 22, pp. 1114–1122.

Liu, Z.Y., Qiao, H., 2014. GNCCP–Graduated NonConvexity and Concavity Procedure. IEEE Trans. on Pattern Analysis and Machine Intelligence 36, 1258–1267.

Loiola, E., Abreu, N.D., Boaventura, P., Hahn, P., Querido, T., 2007. A survey for the quadratic assignment problem. European Journal of Operational Research 176, 657–690.

Neuhaus, M., Bunke, H., 2007a. Bridging the Gap Between Graph Edit Distance and Kernel Machines. volume 68 of *Machine Perception and Artificial Intelligence*. World Scientific Publishing Co., Inc.

Neuhaus, M., Bunke, H., 2007b. A quadratic programming approach to the graph edit distance problem, in: Graph-Based Representations in Pattern Recognition. volume 4538 of *LNCS*, pp. 92–102.

Neuhaus, M., Riesen, K., Bunke, H., 2006. Fast suboptimal algorithms for the computation of graph edit distance, in: Structural, Syntactic and Statistical Pattern Recognition. Springer, pp. 163–172.

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

Riesen, K., Bunke, H., 2008. Iam graph database repository for graph based pattern recognition and machine learning., in: da Vitora Lobo, N.e.a. (Ed.), SSPR&SPR 2008, pp. 287–297.

Riesen, K., Bunke, H., 2009. Approximate graph edit distance computation by means of bipartite graph matching. Image and Vision Comp. 27, 950–959.

Riesen, K., Bunke, H., 2015. Improving bipartite graph edit distance approximation using various search strategies. Pattern Recognition 28, 1349–1363.

Riesen, K., Fankhauser, S., Bunke, H., 2007a. Speeding up graph edit distance computation with a bipartite heuristic, in: Mining and Learning with Graphs.

Riesen, K., Fischer, A., Bunke, H., 2014a. Combining Bipartite Graph Matching and Beam Search for Graph Edit Distance Approximation, in: Artificial Neural Networks in Pattern Recognition. volume 8774 of *LNCS*, pp. 117–128.

Riesen, K., Fischer, A., Bunke, H., 2014b. Improving approximate graph edit distance using genetic algorithms, in: Structural, Syntactic and Statistical Pattern Recognition. volume 8621 of *LNCS*, pp. 63–72.

Riesen, K., Fischer, A., Bunke, H., 2015. Estimating graph edit distance using lower and upper bounds of bipartite approximations. Int. J. Patt. Recogn. Artif. Intell. 29.

Riesen, K., Neuhaus, M., Bunke, H., 2007b. Bipartite graph matching for computing the edit distance of graphs, in: Graph-Based Representations in Pattern Recognition. volume 4538 of *LNCS*, pp. 1–12.

Sanfeliu, A., Fu, K.S., 1983. A distance measure between attributed relational graphs for pattern recognition. IEEE Transactions on Systems, Man. and Cybernetics 13, 353–362.

Serratosa, F., 2014. Fast computation of bipartite graph matching. Pattern Recognition Letters 45, 244–250.

Solé-Ribalta, A., Serratosa, F., Sanfeliu, A., 2012. On the graph edit distance cost: Properties and applications. Int. J. Patt. Recogn. Artif. Intell. 26.

Tsai, W.H., Fu, K.S., 1979. Error-correcting isomorphisms of attributed relational graphs for pattern analysis. IEEE Trans. on Systems, Man and Cybernetics 9, 757–768.

Zaslavskiy, M., Bach, F., Vert, J.P., 2009. A path following algorithm for the graph matching problem. Pattern Anal. Mach. Intell. 31, 2227–2242.

Zeng, Z., Tung, A.K.H., Wang, J., Feng, J., Zhou, L., 2009. Comparing stars: On approximating graph edit distance. Proceedings of the VLDB Endowment 2, 25–36.

Zhou, F., De la Torre, F., 2012. Factorized graph matching, in: IEEE Conference on Computer Vision and Pattern Recognition, pp. 127–134.