Approximate Graph Edit Distance Guided by Bipartite Matching of Bags of Walks

Benoit Gaüzère [1], Sébastien Bougleux [2], Kaspar Riesen [3⋆], and Luc Brun [1]

[1] ENSICAEN, GREYC CNRS UMR 6072, France
{benoit.gauzere,luc.brun}@ensicaen.fr
[2] Université de Caen Basse-Normandie, GREYC CNRS UMR 6072, France
bougleux@unicaen.fr
[3] University of Applied Sciences and Arts Northwestern Switzerland
kaspar.riesen@fhnw.ch

Abstract. The definition of efficient similarity or dissimilarity measures between graphs is a key problem in structural pattern recognition. This problem is nicely addressed by the graph edit distance, which constitutes one of the most flexible graph dissimilarity measure in this field. Unfortunately, the computation of an exact graph edit distance is known to be exponential in the number of nodes. In the early beginning of this decade, an efficient heuristic based on a bipartite assignment algorithm has been proposed to find efficiently a suboptimal solution. This heuristic based on an optimal matching of nodes' neighborhood provides a good approximation of the exact edit distance for graphs with a large number of different labels and a high density. Unfortunately, this heuristic works poorly on unlabeled graphs or graphs with a poor diversity of neighborhoods. In this work we propose to extend this heuristic by considering a mapping of bags of walks centered on each node of both graphs.

1 Introduction

Graphs provide a generic data structure which allows to encode fine properties of a large variety of objects such as shapes or molecules. The use of a graph representation to address pattern recognition problems implies to define a similarity measure between graphs. A widely used approach consists in using the graph edit distance, which allows to measure the distortion required to transform one graph into another. The distortion between two graphs G and G' can be encoded by an edit path defined as a sequence of operations transforming G into G'. Such a sequence may include node or edge insertions, removals and substitutions. Given a non-negative cost function $c(.)$, associated to each operation, the cost of an edit path is defined as the sum of its elementary operation's costs. The optimal edit path is defined as the one associated to the minimal cost among all edit paths transforming G into G'. This minimal cost then corresponds to the edit distance between G and G'. Unfortunately, beside its appealing properties, the computational time of the graph edit distance is known to grow exponentially with the

⋆ Kaspar Riesen is supported by the Hasler Foundation Switzerland.

number of implied nodes [1, 2]. A close relationship exists between graph edit distance and morphism between graphs. Indeed, Bunke [3] has shown that under special conditions on the costs of node and edge insertions, removals and substitutions, computing the graph edit distance is equivalent to compute a maximum common subgraph of two graphs. More generally any mapping between the set of nodes and edges of two graphs induces an edit path which substitutes all mapped nodes and edges, and inserts or removes the non-mapped nodes/edges of the two graphs. Conversely, given an edit path between two graphs such that each node and each edge is substituted only once, one can define a mapping between the substituted nodes and edges of both graphs.

This close relationship between mappings and edit distance constitutes the main principle of the heuristic proposed by Riesen and Bunke [4] in order to decrease the exponential growth of the computational cost of the graph edit distance according to the number of considered nodes. This heuristic builds a mapping between the node sets of two graphs using a bipartite assignment algorithm, and deduces an edit path from this mapping. The cost of this edit path, which may not be optimal, is considered as an approximation of the exact edit distance. The optimal bipartite assignment algorithm is based on a cost function defined between the neighborhoods of each pair of nodes of the two graphs. The idea behind this heuristic being that a mapping between nodes with similar neighborhoods should induce an edit path with a low cost. However, this heuristic may work poorly on unlabeled graphs and more generally in cases where neighborhoods do not allow to easily differentiate the nodes.

In this paper we propose to extend this heuristic by considering a bipartite assignment algorithm between bags of walks incident to each node of both graphs. Hence, within this framework, a mapping of direct neighborhoods is similar to a mapping of bags of walks of length 1.

Our paper is structured as follows: Section 2 defines the bipartite assignment problem, and Section 3 defines the computation of an approximate edit distance from a bipartite assignment algorithm together with the heuristic defined in [4]. Section 4 defines an efficient computation of the bag of walks associated to each node of a graph, together with the costs of substituting, inserting or removing such a bag. Finally, Section 5 presents experiments on molecule datasets showing the accuracy gain obtained using our approach.

2 Assignment Problem

2.1 Linear Sum Assignment Problem (LSAP)

Let $\mathcal{X} = \{x_i\}_i$ and $\mathcal{Y} = \{y_i\}_i$ be two sets with $|\mathcal{X}| = |\mathcal{Y}| = n$. Assigning the n elements of $\mathcal{X}$ to the n elements of $\mathcal{Y}$ can be described by a bijective mapping $\mathcal{X} \to \mathcal{Y}$, reduced to a permutation of $\{1, \ldots, n\}$ if indices of elements are considered. Provided a matrix $\mathbf{C} \in \mathbb{R}_+^{n \times n}$ so that $C_{i,j} = c(x_i \to y_j) = c(y_j \to x_i)$ measures the cost of assigning element $x_i \in \mathcal{X}$ to element $y_j \in \mathcal{Y}$, the Linear Sum Assignment Problem (LSAP) finds an optimal permutation $\hat{\varphi} \in \underset{\varphi \in S_n}{\operatorname{argmin}} \sum_{i=1}^{n} C_{i,\varphi(i)}$,

where S_n is the set of all permutations of $\{1,\dots,n\}$. Recall that any permutation φ can be associated to a permutation matrix $\mathbf{P}\in\{0,1\}^{n\times n}$ satisfying $P_{i,j}=\delta_{i,\varphi(i)}$, where $\delta_{i,j}$ is the Kronecker delta ($\delta_{i,j}=1$ if $i=j$ and 0 else). Note that $\mathbf{P}$ is doubly stochastic (sum of rows is equal to 1 and similarly for columns). Then, the LSAP corresponds to find an optimal permutation matrix

$$\hat{\mathbf{P}}\in\underset{\mathbf{P}\in\mathcal{P}_n}{\operatorname{argmin}}\ \sum_{i=1}^{n}\sum_{j=1}^{n}C_{i,j}\,P_{i,j}, \tag{1}$$

where $\mathcal{P}_n$ denotes the set of all $n\times n$ permutation matrices.

The LSAP may also be formulated as a maximization problem, and is also known as the maximum weighted bipartite matching problem. It can be solved by the Hungarian or Kuhn-Munkres algorithm in $O(n^3)$ time complexity [5, 6], and it has been generalized in many directions, see [7] for more details.

2.2 LSAP with Insertion and Removal of Elements

Let $\mathcal{X}$ and $\mathcal{Y}$ be two sets, with $n=|\mathcal{X}|$ and $m=|\mathcal{Y}|$. As before, each element $x_i\in\mathcal{X}$ can be assigned to an element $y_j\in\mathcal{Y}$ according to a given substitution cost matrix $\mathbf{C}(\mathcal{X},\mathcal{Y})\in\mathbb{R}_+^{n\times m}$ with $[\mathbf{C}(\mathcal{X},\mathcal{Y})]_{i,j}=c(x_i\to y_j)$. Also, assume that each element of both $\mathcal{X}$ and $\mathcal{Y}$ can be deleted, or equivalently inserted, that is assigned to the null element denoted by ϵ. Removal and insertion of an element $x_i\in\mathcal{X}$ have the same cost $c(x_i\to\epsilon)=c(\epsilon\to x_i)$, and similarly for the elements of $\mathcal{Y}$. Removal-insertion costs associated to the n elements of $\mathcal{X}$ can be represented by the matrix $\mathbf{C}_\epsilon(\mathcal{X})\in\mathbb{R}^{n\times n}$, with $[\mathbf{C}_\epsilon(\mathcal{X})]_{i,j}=c(x_i\to\epsilon)$ if $i=j$ and $+\infty$ else. Similarly consider $\mathbf{C}_\epsilon(\mathcal{Y})\in\mathbb{R}^{m\times m}$. In other terms, each set is augmented with null elements, $\mathcal{X}_\epsilon=\mathcal{X}\cup\{\epsilon_i\}_{i=1,\dots,m}$ and $\mathcal{Y}_\epsilon=\mathcal{Y}\cup\{\epsilon_i\}_{i=1,\dots,n}$, such that $|\mathcal{X}_\epsilon|=|\mathcal{Y}_\epsilon|=n+m$. Following [4], the optimal linear sum assignment $\mathcal{X}_\epsilon\to\mathcal{Y}_\epsilon$, according to the cost matrix

$$\mathbf{C}_\epsilon(\mathcal{X},\mathcal{Y})=[C_{i,j}]_{i,j}=\begin{bmatrix}\mathbf{C}(\mathcal{X},\mathcal{Y}) & \mathbf{C}_\epsilon(\mathcal{X})\\ \mathbf{C}_\epsilon(\mathcal{Y}) & \mathbf{0}\end{bmatrix}\in[0,+\infty]^{(n+m)\times(n+m)}, \tag{2}$$

substitutes at most $\min(n,m)$ elements of $\mathcal{X}$ to at most $\min(n,m)$ elements of $\mathcal{Y}$, with insertion or removal of the remaining ones. Since the substitution of empty elements should not cause any cost, we always have $c(\epsilon_i\to\epsilon_j)=0$ (lower right submatrix of $\mathbf{C}_\epsilon(\mathcal{X},\mathcal{Y})$). An optimal assignment $\mathcal{X}_\epsilon\to\mathcal{Y}_\epsilon$ can then be defined as a matrix $\mathbf{P}$ minimizing the total cost functional

$$A(\mathbf{C}_\epsilon(\mathcal{X},\mathcal{Y}),\mathbf{P})=\underbrace{\sum_{i=1}^{n}\sum_{j=1}^{m}C_{i,j}\,P_{i,j}}_{\text{substitution}}+\underbrace{\sum_{i=1}^{n}C_{i,m+i}\,P_{i,m+i}+\sum_{j=1}^{m}C_{n+j,j}\,P_{n+j,j}}_{\text{removal/insertions}} \tag{3}$$

among the set $\mathcal{P}_{n,m,\epsilon}$ of all doubly substochastic matrices

$$\mathbf{P}=\begin{bmatrix}\mathbf{Q} & \mathbf{R}\\ \mathbf{S} & \mathbf{0}\end{bmatrix}\in\{0,1\}^{(n+m)\times(n+m)},$$

where $\mathbf{Q} \in \{0,1\}^{n \times m}$ represents the (partial) assignment $\mathcal{X} \to \mathcal{Y}$, and $\mathbf{R} \in \{0,1\}^{n \times n}$ and $\mathbf{S} \in \{0,1\}^{m \times m}$ are diagonal matrices representing removal and insertions. Columns and rows of $\mathbf{P}$ are constrained to satisfy

$$P_{i+j,j} + \sum_{i=1}^{n} P_{i,j} = 1, \ \forall j = 1, \ldots, m, \quad \text{and} \quad P_{i,j+i} + \sum_{j=1}^{m} P_{i,j} = 1, \ \forall i = 1, \ldots, n.$$

According to Section 2.1, the computation cost of the assignment is $O((n+m)^3)$. This assignment problem with edition is used to design approximate graph edit distances, as described in the following section.

3 Approximate Graph Edit Distance based on the LSAP

We consider simple labeled graphs denoted by $G = (V, E, \mu, \nu)$, where V is the finite set of nodes, $E \subset V \times V$ is the set of edges, $\mu : V \to \mathcal{L}_\mathcal{V}$ is the node labeling function, and $\nu : E \to \mathcal{L}_\mathcal{E}$ is the edge labeling function. $\mathcal{L}_\mathcal{V}$ and $\mathcal{L}_\mathcal{E}$ are label sets for both nodes and edges (e.g. the vector space $\mathbb{R}^n$ or a set of symbolic labels).

As mentioned in Section 1, a major drawback of graph edit distance is its computational complexity. In fact, the problem of finding the minimum cost edit path between G and G' can be reformulated as an instance of a *Quadratic Assignment Problem* (*QAP*), known to be $\mathcal{NP}$*-complete*. Hence, exact computation of the graph edit distance is limited to graphs of rather small size in practice.

3.1 Graph Edit Distance Approximation

The graph edit distance approximation framework introduced in [4] reduces the QAP of graph edit distance computation to an instance of an LSAP which can be, in contrast with QAPs, efficiently solved. The algorithmic framework mainly consists of the following three steps.

Step 1. First, the graphs to be matched are subdivided into individual nodes plus local structures whereon a cost matrix $\mathbf{C}_\epsilon$, as defined in Eq. (2), is built.

Formally, let us consider an input graph $G = (V, E, \mu, \nu)$ together with a bag of bags of structural patterns $B = \{B_i\}_{i=1,\ldots,|V|}$. Every bag B_i is associated to a node $u_i \in V$ and characterizes the local structure of G around node u_i. The target graph $G' = (V', E', \mu', \nu')$ and its corresponding bags of structural patterns $B' = \{B'_i\}_{i=1,\ldots,|V'|}$ are given analogously. We define a cost $c(B_i \to B'_j)$ for the substitution of two bags of patterns, and a cost $c(B_i \to \epsilon)$ as well as a cost $c(\epsilon \to B'_j)$ for the removal and insertion of a bag, respectively. Given the cost model and following the scheme outlined in Section 2 we build the cost matrix $\mathbf{C}_\epsilon(B, B')$, encoding the cost of substitutions, insertions, and removals of bags of structural patterns.

Step 2. In the second step of the approximation framework, an assignment algorithm is applied to the square cost matrix $\mathbf{C}_\epsilon(B, B')$ in order to find a minimum

cost assignment between both set of bags (possibly including removals and/or insertions of bags):

$$\hat{\mathbf{P}} \in \underset{\mathbf{P}\in\mathcal{P}_{|B|,|B'|,\epsilon}}{\operatorname{argmin}} A\left(\mathbf{C}_\epsilon(B,B'),\mathbf{P}\right). \tag{4}$$

Note that each bag B_i is associated to a single node u_i, and therefore, the optimal assignment defined by Eq. (4) provides an optimal assignment between the nodes of both graphs with respect to their bags of patterns. That is, the permutation $\hat{\mathbf{P}}$ provides a mapping $\psi : V \cup \{\varepsilon\} \to V' \cup \{\varepsilon\}$ of the nodes V of G to the nodes V' of G'. Due to the definition of the cost matrix, which allows both insertions and removals of elements, the mapping ψ includes node assignments of the form $u_i \to u'_j$, $u_i \to \varepsilon$, $\varepsilon \to u'_j$, and $\varepsilon \to \varepsilon$.

Step 3. Clearly, the mapping ψ can be interpreted as a partial edit path between the graphs G and G' considering edit operations on nodes only. Thus, in the last step this partial edit path is completed with respect to the edges. This can be accomplished since edit operations on edges are implied by edit operations on their nodes. That is, whether an edge is substituted, removed, or inserted, depends on the edit operations performed on its nodes. Hence, given the set of node operations in ψ, the global edge structures from G and G' can be edited accordingly. The cost of the complete edit path is finally returned as an approximate graph edit distance between graphs G and G'.

3.2 Defining Bags of Structural Patterns

Note that the edit path corresponding to the approximate edit distance value considers the edge structure of G and G' in a global and consistent way while the optimal permutation $\hat{\mathbf{P}}$ is able to consider the structural information in an isolated way only (bags of local structural patterns). This is due to the fact that during the optimization process of the specific LSAP, no information about neighboring node mappings is available. Hence, the definition of powerful structural patterns is a crucial task in this approximation framework.

In [4], every bag B_i of structural patterns represents the set of edges incident to node $v_i \in V$. Formally, assume that node v_i has incident edges E_{v_i}, then we define $B_i = \{\, (v_i, v_k) \in E_{v_i} : v_k \in V \,\}$. The present paper introduces a major generalization of this formalism. That is, rather than "the star neighborhood" of every node, bags of walks centered on each node are considered as bags of structural patterns. Both the computation of these bags of walks and the definition of an adequate cost model on them are described in the next section.

4 Walks and Approximate GED for Labeled Graphs

Recall that a walk of length k in a simple graph $G = (V, E, \mu, \nu)$, or k-walk, is a sequence $(u_i)_i$ of $(k+1)$ nodes of V such that $(u_i, u_{i+1}) \in E$ for all $i = 1, \ldots, k$. Any k-walk $(u_i)_i$, in a labeled graph, can be associated to a sequence

$$s = (s_l)_l = (\mu(u_0)\, \nu(u_0,u_1)\, \mu(u_1)\, \nu(u_1,u_2)\, \cdots\, \mu(u_{k-1})\, \nu(u_{k-1},u_k)\, \mu(u_k))$$

of $(2k+1)$ labels, alternating node and edge labels. Let B_i be the bag of sequences of $(2k+1)$ labels associated to all k-walks starting at node $v_i \in V$. Now given two graphs G and G', together with their bags $B = \{B_i\}_{i=1,\ldots,|V|}$ and $B' = \{B'_i\}_{i=1,\ldots,|V'|}$ of bags of label sequences, for each pair of bags $(B_i, B'_j) \in B \times B'$, the substitution cost $c(B_i \to B'_j)$ can be defined by comparing the label sequences. This is equivalent to the comparison of two bags of labeled k-walks, starting at nodes v_i and v'_j respectively.

By assuming that the substitution of node or edge labels does not depend on the labels themselves when they are different, the edit cost between two sequences $s \in B_i$ and $s' \in B'_j$ can simply be defined from the number of common labels at the same position in both sequences:

$$c(s \to s') = c_{\text{ns}} \sum_{l=0}^{k} \delta_{2l+1} + c_{\text{es}} \sum_{l=1}^{k} \delta_{2l}, \tag{5}$$

where $\delta_l = 0$ if $s_l = s'_l$ and 1 else, and c_{ns} and c_{es} denote node and edge substitution costs, respectively. When $s = s'$, the associated k-walks are equivalent, or similar, and $c(s \to s') = 0$. In other cases, different labels at the same position in s and s' appear at least once, the k-walks are said to be different. Since to compute this cost, k-walks needs to be explicitly extracted, it is difficult to derive a cost between bags which is computationally attractive. So we propose to restrict the knowledge of each k-walk to its terminal nodes (begin and end nodes), together with their labels, which allows to consider the cost

$$\hat{c}(s \to s') = \begin{cases} 0 & \text{if } s = s' \\ (\delta_1 + \delta_{2k+1} + k - 1)\, c_{\text{ns}} + k\, c_{\text{es}} & \text{else,} \end{cases} \tag{6}$$

so that non-terminal node labels and also edge labels are treated as if they were pairwise different when sequences are different. Obviously the cost $\hat{c}$ satisfies $c(s \to s') \leqslant \hat{c}(s \to s')$ for any s and s'.

Any optimal mapping between the walks of two bags according to Eq. (6) should include a mapping of similar walks with 0 cost. The cost of an optimal mapping between two bags of walks may thus be rewritten as:

$$[\mathbf{C}(B, B')]_{i,j} = 0 \cdot |B_i \cap B'_j| + \min_{\mathbf{P} \in \mathcal{P}_{|B_i \setminus B'_j|, |B'_j \setminus B_i|, \epsilon}} A\left(\mathbf{C}_\epsilon(B_i \setminus B'_j, B'_j \setminus B_i), \mathbf{P}\right), \tag{7}$$

which separates similar and different k-walks. Determining if k-walks (sequences) are similar can be achieved through the construction of the direct product of the two corresponding labeled graphs (Section 4.1). This also allows to derive assignment costs for the remaining different k-walks (Section 4.2).

4.1 Similar walks

The direct product of two labeled graphs $G = (V, E, \mu, \nu)$ and $G' = (V', E', \mu', \nu')$ is the graph $G \times G' = (V_\times, E_\times, \mu_\times, \nu_\times)$. The node set and the edge set are given

by $V_\times = \left\{(v_i, v'_j) \in V \times V' : \mu(v_i) = \mu'(v'_j)\right\}$ and $E_\times$, where $E_\times$ is defined by

$$\left\{\left((v_i, v'_j), (v_k, v'_l)\right) \in V_\times \times V_\times : (v_i, v_k) \in E \wedge (v'_j, v'_l) \in E' \wedge \nu(v_i, v_k) = \nu'(v'_j, v'_l)\right\}$$

such that $\mu_\times((v_i, v'_j)) = \mu(v_i) = \mu'(v'_j)$ for all node $(v_i, v'_j) \in V_\times$, and similarly $\nu_\times((v_i, v'_j), (v_k, v'_l)) = \nu(v_i, v_k) = \nu'(v'_j, v'_l)$ for all edge $((v_i, v'_j), (v_k, v'_l)) \in E_\times$. In particular, a walk from node (v_i, v'_j) to node (v_k, v'_l) in $G \times G'$ corresponds to a walk from v_i to v_k in G, and to a similar walk from v'_j to v'_l in G', both having the same sequence of node and edge labels by construction ([8] for an overview). This allows to partially match the two bags with a zero cost according to Eq. (7). Recall that the number of k-walks, between any pair of nodes of a graph, can be computed by $\mathbf{W}^k$, where $\mathbf{W}$ is the adjacency matrix of the graph. So, the number of k-walks common to the two graphs G and G' can be deduced from $\mathbf{W}^k_\times$, where $\mathbf{W}_\times$ defines the adjacency matrix of the direct product graph. Note that a walk in G similar to p walks in G' will be duplicated p times in the direct graph product.

4.2 Different walks

Given a k value, and two different k-walks s and s', $c(s, s')$ can only take four different values depending on the values of δ_1 and δ_{2k+1}. This last point drastically simplifies the optimal assignment of the bags B_i and B'_j defined by Eq. (7), which can be efficiently approximated through histograms encoding terminal node's labels of sequences.

Let $h_i : \mathcal{L}_\mathcal{V} \to \mathbb{N}$ be the histogram function which assigns to each label $l \in \mathcal{L}_\mathcal{V}$, the number of k-walks ending at a node of label l in the bag B_i. This number of k-walks can be efficiently computed using $\mathbf{W}^k$. Similarly consider histograms h'_j and $h^\times_{(i,j)}$. From the definition of the direct product, we have $h^\times_{(i,j)} = z_i\, z'_j$, where z_i (resp. z'_j) defines the number of k-walks in B_i (resp. B'_j), for each node label, which are similar to at least one k-walk in B'_j (resp. B_i). The number of k-walks, in each bag, which can be matched with 0 cost, is thus given by $\min\{z_i, z'_j\}$. The remaining k-walks in B_i is then given by $h_{i\setminus j} = h_i - \min\{z_i, z'_j\}$. Similarly we consider $h_{j\setminus i} = h_j - \min\{z_i, z'_j\}$. Since computing z_i and z'_j may be computationally costly using an implicit enumeration of walks, $h_{i\setminus j}$ is approximated by $\hat{h}_{i\setminus j} = h_i - \min\{h_i, h'_j, \lfloor (h^\times_{(i,j)})^{1/2} \rfloor\}$, and similarly for $h'_{j\setminus i}$. According to (6), the cost of assigning the bag B_i to the bag B'_j is finally given by:

$$\begin{aligned}
\left[\mathbf{C}(B, B')\right]_{i,j} = {} & \left((\delta_1 + k - 1)\, c_{\text{ns}} + k\, c_{\text{es}}\right) \sum_{l=1}^{|\mathcal{L}_\mathcal{V}|} \min\left\{\hat{h}_{i\setminus j}(l), \hat{h}'_{j\setminus i}(l)\right\} \\
& + \left((\delta_1 + k)\, c_{\text{ns}} + k\, c_{\text{es}}\right) \min\left\{r_{i,j}, r'_{j,i}\right\} \\
& + \left((\delta_1 + k)\, c_{\text{nri}} + k\, c_{\text{eri}}\right) \left|r_{i,j} - r'_{j,i}\right|,
\end{aligned} \tag{8}$$

where c_{nri} and c_{eri} denote node and edge removal/insertion costs, and $r_{i,j}$ corresponds to the k-walks of B_i not similar to a k-walk of B'_j, and whose terminal

nodes need also to be substituted:

$$r_{i,j} = \sum_{l=1}^{|\mathcal{L}_\mathcal{V}|} \hat{h}_{i\setminus j}(l) - \min\left\{\hat{h}_{i\setminus j}(l), \hat{h}'_{j\setminus i}(l)\right\}, \tag{9}$$

and similarly for $r'_{j,i}$. The first line of (8) corresponds to substituted k-walks ending with the same node label ($\delta_{2k+1}=0$ in Eq. 6), the second line corresponds to substituted k-walks ending with a different node label ($\delta_{2k+1}=1$ in Eq. (6)), and third line to the remaining k-walks to be removed/inserted. From Eq. (8) and Eq. (9), the cost of removing/inserting a bag, *i.e.* the cost of removing/inserting all its k-walks, is given by $[\mathbf{C}_\epsilon(B)]_{i,i} = ((k+1)\,c_{\text{nri}} + k\,c_{\text{eri}})\,|B_i|$, and $[\mathbf{C}_\epsilon(B')]_{i,i} = ((k+1)\,c_{\text{nri}} + k\,c_{\text{eri}})\,|B'_i|$. The costs given by (8) and this last equation allow to construct the cost matrix $\mathbf{C}_\epsilon(B, B')$ in order to build the optimal assignment of bags of walks B and B'. An efficient approximation of the GED is then deduced from this optimal assignment (Sec. 3.1).

5 Experiments

Our new heuristic to compute an approximate edit distance has been tested on 4 graph datasets[4] encoding molecular graphs. For all these experiments, insertion/removal costs have been arbitrarily set to 3 for both edges and nodes and substitution cost to 1 for edges and nodes, regardless of node's or edge's labels. Graphs included within the 4 datasets have different characteristics: Alkane and PAH are only composed of unlabeled graphs whereas MAO and Acyclic correspond to labeled graphs. In addition, Alkane and Acyclic correspond to acyclic graphs having a low number of nodes (8 to 9 nodes in average) whereas MAO and PAH correspond to larger cyclic graphs (about 20 nodes in average). Tables 1 and 2 show a comparison of the accuracy of our proposition with state of the art method [4] and exact edit distance. First, Table 1 shows the percentage of distance matrix entries corresponding to a gain (i.e. computed edit distance is lower), a loss or no changes on the accuracy of our approximation method versus the one proposed by [4]. As we can see in column "Gain", our approach provides a more accurate approximation of the edit distance for 45% to 98% of molecules' pairs while we observe a loss on the accuracy for only $< 1\%$ to 27% of computed edit distances, depending on the dataset. Same conclusions are observed in Table 2 which shows the average edit distance for each dataset and each method together with the average time required to compute the associated edit distance matrix. We can note that the time required to compute our edit distances is higher, but still comparable, than the one required by [4]. However, one can note that computation times obtained for the lines 1 and 2 have been computed using a Java implementation [9] whereas line 3 corresponds to a Matlab implementation. Finally, results for A* method for MAO and PAH datasets are not displayed since it takes too much time to compute. These first results allow us

[4] These datasets are available at `http://iapr-tc15.greyc.fr/links.html`

to highlight the gain on the accuracy induced by using our matching approach instead of the one initially proposed by [4]. In addition, we can note that taking into account a larger radius than the direct neighborhood (i.e. walk size > 1) allows us to increase the percentage of distance matrix's entries corresponding to an accuracy gain using our approximation, with maximum percentage obtained for walks of size equals to 3 or 4 (Figure 1(c)). However, we can note that the accuracy decreases when considering walks up to 5 nodes. This observation can be explained by the tottering phenomenon which induces non representative walks into the computation of the cost matrix. In addition, we can note that this observation is stronger for Acyclic and Alkane datasets which are more prone to tottering since they are both composed of smaller molecules than PAH and MAO.

In order to validate our proposition on prediction problems, we predicted the classes of PAH and MAO molecules thanks to a k-ppv algorithm, with k equals to 1, 3 and 5. Table in Figure 1(b) shows the percentage of correctly classified molecules using a 10-fold cross validation. As observed in previous experiments, the gain on the accuracy provided by our approximation (line 2, Table in Figure 1(b)) allows us to obtain significantly better classification results than the ones obtained by the approximation method proposed in [4] (line 1, Table in Figure 1(b)). This classification experiment shows thus the relevance of our contribution for prediction problems. This accuracy gain is also shown by the scatter plot of our approximation (x-axis) and the approximation of [4] (y-axis) on PAH and MAO datasets (Figure 1(a)). Points over the diagonal corresponds to a better accuracy of our approach than the one obtained by [4].

6 Conclusion

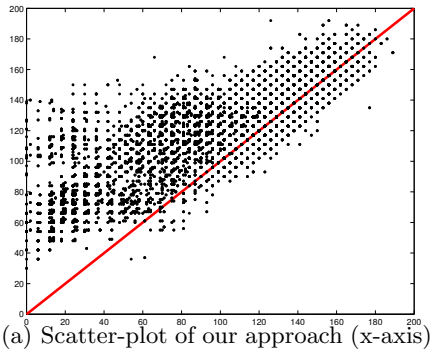
We have presented in this paper a natural extension of a well known heuristic computing an approximate graph edit distance between labeled graphs. Our heuristic is based on an assignment of bags of walks incident to each node. Experiments show that the proposed heuristic brings a significant decrease of the graph edit distance compared to the previous heuristic at a cost which remains much lower than the computational cost of the exact edit distance. Moreover,

Dataset	Gain	Loss	Equality	Size
Alkane	45%	28%	27%	3
PAH	73%	14%	13%	4
MAO	98%	2%	$< 1\%$	4
Acyclic	56%	24%	21%	4

Table 1. Accuracy comparison between our approach and [4].

	Alkane		**Acyclic**		**MAO**		**PAH**	
	$\bar{d}$	$\bar{t}$	$\bar{d}$	$\bar{t}$	$\bar{d}$	$\bar{t}$	$\bar{d}$	$\bar{t}$
$\mathbf{A}^\star$	15	28 800	17	172 800	-	-	-	-
[4]	35	20	35	22	105	20	138	40
BW	33	55	31	86	49	129	120	390

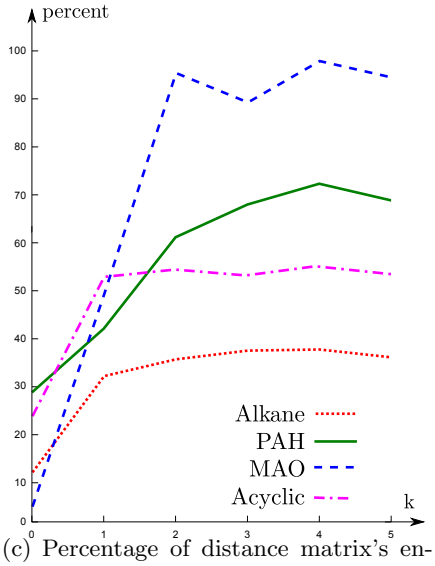
Table 2. Average edit distance ($\bar{d}$) and average time in seconds ($\bar{t}$) for each method and each dataset (BW = bags of walks).



(a) Scatter-plot of our approach (x-axis) and [4] (y-axis).

Method	MAO			PAH		
k	**1**	**3**	**5**	**1**	**3**	**5**
[4]	68%	62%	54%	59%	63%	61%
BW	93%	90%	71%	79%	78%	74%

(b) Classification results.



(c) Percentage of distance matrix's entries corresponding to an accuracy gain using our approach versus the size of considered walks (k) for each dataset.

Fig. 1. Classification of MAO and PAH using k-ppv and walk size comparisons.

according to our experiments our heuristic provides a significant gain on classification results using a kppv classifier. Our future work will consist to test other types of patterns and to compare explicit vs implicit enumeration of patterns.

References

1. Sanfeliu, A., Fu, K.: A distance measure between attributed relational graphs for pattern recognition. Systems, Man and Cybernetics **13**(3) (1983) 353–363
2. Bunke, H., Allermann, G.: Inexact graph matching for structural pattern recognition. Pattern Recognition Letters **1** (1983) 245–253
3. Bunke, H.: On a relation between graph edit distance and maximum common subgraph. Pattern Recogn. Lett. **18**(9) (August 1997) 689–694
4. Riesen, K., Bunke, H.: Approximate graph edit distance computation by means of bipartite graph matching. Image and Vision Computing **27** (2009) 950–959
5. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Research Logistics Quaterly **2** (1955) 83–97
6. Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the Society for Industrial and Applied Mathematics **5**(1) (1957) 32–38
7. Burkard, R., Dell'Amico, M., Martello, S.: Assignment Problems. SIAM (2009)
8. Hammack, R., Imrich, W., Klavžar, S.: Hanbook of Product Graphs. 2nd edn. Discrete Mathematics and its Applications. CRC Press, Taylor & Francis (2011)
9. Riesen, K., Emmenegger, S., Bunke, H.: A novel software toolkit for graph edit distance computation. In: Graph-Based Representations in Pattern Recognition. Springer (2013) 142–151