Ring Based Approximation of Graph Edit Distance

David B. Blumenthal[1], Sébastien Bougleux[2], Johann Gamper[1], and Luc Brun[2]

[1] Faculty of Computer Science, Free University of Bozen-Bolzano, Bolzano, Italy
`david.blumenthal@inf.unibz.it, gamper@inf.unibz.it`
[2] Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, Caen, France
`bougleux@unicaen.fr, luc.brun@ensicaen.fr`

Abstract. The graph edit distance (GED) is a flexible graph dissimilarity measure widely used within the structural pattern recognition field. A widely used paradigm for approximating GED is to define local structures rooted at the nodes of the input graphs and use these structures to transform the problem of computing GED into a linear sum assignment problem with error correction (LSAPE). In the literature, different local structures such as incident edges, walks of fixed length, and induced subgraphs of fixed radius have been proposed. In this paper, we propose to use rings as local structure, which are defined as collections of nodes and edges at fixed distances from the root node. We empirically show that this allows us to quickly compute a tight approximation of GED.

Keywords: Graph Edit Distance · Graph Matching · Upper Bounds.

1 Introduction

Due to the flexibility and expressiveness of labeled graphs, graph representations of objects such as molecules and shapes are widely used for addressing pattern recognition problems. For this, a graph (dis-)similarity measure has to be defined. A widely used measure is the graph edit distance (GED), which equals the minimum cost of a sequence of edit operations transforming one graph into another. As exactly computing GED is *NP*-hard [17], research has mainly focused on the design of approximative heuristics that quickly compute upper bounds for GED. The development of such heuristics was particularly triggered by the introduction of the paradigm `LSAPE-GED`, which transforms GED to the linear sum assignment problem with error correction (LSAPE) [10,17]. LSAPE extends the linear sum assignment problem by allowing rows and columns to be not only substituted, but also deleted and inserted. `LSAPE-GED` works as follows: In a first step, the graphs G and H are decomposed into local structures rooted at their nodes. Next, a distance measure between these local structures is defined. This measure is used to populate an instance of LSAPE, whose rows and columns correspond to the nodes of G and H, respectively. Finally, the constructed LSAPE

instance is solved. The computed solution is interpreted as a sequence of edit operations, whose cost is returned as an upper bound for $\mathrm{GED}(G, H)$.

The original instantiations BP [10] and STAR [17] of LSAPE-GED define the local structure of a node as, respectively, the set of its incident edges and the set of its incident edges together with the terminal nodes. Since then, further instantiations have been proposed. Like BP, the algorithms BRANCH-UNI [18], BRANCH, and BRANCH-FAST [2] use the incident edges as local structures. They differ from BP in that they use distance measures for the local structures that also allow to derive lower bounds for GED. In contrast to that, the algorithms SUBGRAPH [6] and WALKS [8] define larger local structures. Given a constant L, SUBGRAPH defines the local structure of a node u as the subgraph which is induced by the set of nodes that are within distance L from u, while WALKS defines it as the set of walks of length L starting at u. SUBGRAPH uses GED as the distance measure between its local structures and hence runs in polynomial time only if the input graphs have constantly bounded maximum degrees. Not all instantiations of LSAPE-GED are designed for general edit costs: STAR and BRANCH-UNI expect the edit costs to be uniform, and WALKS assumes that the costs of all edit operation types are constant. As an extension of LSAPE-GED, it has been suggested to define node centrality measures, transform the LSAPE instance constructed by any instantiation of LSAPE-GED such that assigning central to non-central nodes is penalized, and return the minimum of the edit costs induced by solutions to the original and the transformed instances as an upper bound for GED [12,16].

Not all heuristics for GED follow the paradigm LSAPE-GED. Most notably, some methods use variants of local search to improve a previously computed upper bound [4,7,11,14]. These methods yield tighter upper bounds than LSAPE-GED instantiations at the price of a significantly increased runtime, and use LSAPE-GED instantiations for initialization. They are thus no competitors of LSAPE-GED instantiations and will hence not be considered any further in this paper.

In this paper, we propose a new instantiation RING of LSAPE-GED that is similar to SUBGRAPH and WALKS in that it also uses local structures whose sizes are bounded by a constant L — namely, rings. Intuitively, the ring rooted at a node u is a collection of disjoint sets of nodes and edges which are within distances $l < L$ from u. Experiments show that RING yields the tightest upper bound of all instantiations of LSAPE-GED. The advantage of rings w. r. t. subgraphs is that ring distances can be computed in polynomially. The advantage w. r. t. walks is that rings can model general edit costs, avoid redundancies due to multiple node or edges inclusions, and allow to define a fine-grained distance measure between the local structures. The rest of the paper is organized as follows: In Section 2, important concepts are introduced. In Section 3, RING is presented. In Section 4, the experimental results are summarized. Section 5 concludes the paper.

2   Preliminaries

In this paper, we consider undirected labeled graphs $G = (V^G, E^G, \ell_V^G, \ell_E^G)$ where V^G and E^G are sets of nodes and edges, and $\ell_V^G : V^G \to \Sigma_V$, $\ell_E^G : E^G \to \Sigma_E$ are

Table 1. Edit operations and edit costs for transforming a graph G into a graph H.

Edit Operation	Edit Cost	Short Notation
substitute node $u \in V^G$ by node $v \in V^H$	$c_V(\ell_V^G(u), \ell_V^H(u))$	$c_V(u, v)$
delete isolated node $u \in V^G$ from V^G	$c_V(\ell_V^G(u), \epsilon)$	$c_V(u, \epsilon)$
insert isolated node v into V^H	$c_V(\epsilon, \ell_V^H(u))$	$c_V(\epsilon, v)$
substitute edge $e \in E^G$ by edge $f \in E^H$	$c_E(\ell_E^G(e), \ell_E^H(f))$	$c_E(e, f)$
delete edge $e \in E^G$ from E_G	$c_E(\ell_E^G(e), \epsilon)$	$c_E(e, \epsilon)$
insert edge f into E^H	$c_E(\epsilon, \ell_E^H(f))$	$c_E(\epsilon, f)$

labeling functions. Furthermore, we are given non-negative edit cost functions $c_V : \Sigma_V \cup \{\epsilon\} \times \Sigma_V \cup \{\epsilon\} \to \mathbb{R}_{\geq 0}$ and $c_E : \Sigma_E \cup \{\epsilon\} \times \Sigma_E \cup \{\epsilon\} \to \mathbb{R}_{\geq 0}$, where ϵ is a special label reserved for dummy nodes and edges, and the equations $c_V(\alpha, \alpha) = 0$ and $c_E(\beta, \beta) = 0$ hold for all $\alpha \in \Sigma_V \cup \{\epsilon\}$ and all $\beta \in \Sigma_E \cup \{\epsilon\}$. An edit path P between graphs G and H is a sequence of edit operations with non-negative edit costs defined in terms of c_V and c_E (Table 1) that transform G into H. Its cost $c(P)$ is defined as the sum over the costs of its edit operations.

Definition 1 (GED). *The graph edit distance between graphs G and H is defined as* $\text{GED}(G, H) = \min_{P \in \Psi(G,H)} c(P)$*, where $\Psi(G, H)$ is the set of all edit paths between G and H.*

The key insight behind the paradigm `LSAPE-GED` is that a complete set of node edit operations—i.e., a set of node edit operations that specifies for each node of the input graphs whether is has to be substituted, inserted, or deleted—can be extended to an edit path, whose edit cost is an upper bound for GED [3, 4, 17]. For constructing a set of node operations that induces a cheap edit path, a suitably defined instance of LSAPE is solved. LSAPE is defined as follows [5]:

Definition 2 (LSAPE). *Given a matrix* $\mathbf{C} = (c_{i,k}) \in \mathbb{R}_{\geq 0}^{(n+1) \times (m+1)}$ *with* $c_{n+1,m+1} = 0$*, LSAPE consists in the task to compute an assignment* $\pi^\star \in \arg\min_{\pi \in \Pi_{n,m}} \mathbf{C}(\pi)$*.* $\Pi_{n,m}$ *is the set of assignments of rows of* $\mathbf{C}$ *to columns of* $\mathbf{C}$ *such that each row except for* $n + 1$ *and each column except for* $m + 1$ *is covered exactly once, and* $\mathbf{C}(\pi) = \sum_{i=1}^{n+1} \sum_{k \in \pi[i]} c_{i,k}$*.*

Instantiations of `LSAPE-GED` construct a LSAPE instance $\mathbf{C}$ of size $(|V^G| + 1) \times (|V^H| + 1)$, such that the rows and columns of $\mathbf{C}$ correspond to the nodes of G and H plus one dummy node used for representing insertions and deletions. A feasible solution for $\mathbf{C}$ can hence be interpreted as a complete set of node edit operations, which induces an upper bound for GED. An optimal solution for $\mathbf{C}$ can be found in $O(\min\{n, m\}^2 \max\{n, m\})$ time [5]; greedy suboptimal solvers run in in $O(nm)$ time [13]. For populating $\mathbf{C}$, instantiations of `LSAPE-GED` associate the nodes $u_i \in V^G$ and $v_k \in V^H$ with local structures $\mathcal{S}^G(u_i)$ and $\mathcal{S}^H(v_k)$, and then construct $\mathbf{C}$ by setting $c_{i,k} = d_{\mathcal{S}}(\mathcal{S}^G(u_i), \mathcal{S}^H(v_k))$, $c_{i,|V^H|+1} = d_{\mathcal{S}}(\mathcal{S}^G(u_i), \mathcal{S}(\epsilon))$, and $c_{|V^G|+1,k} = d_{\mathcal{S}}(\mathcal{S}(\epsilon), \mathcal{S}^H(v_k))$, where $d_{\mathcal{S}}$ is a distance measure for the local structures and $\mathcal{S}(\epsilon)$ is a special local structure assigned to dummy nodes.

3   Ring Based Upper Bounds for GED

3.1   Definition of Ring Structures and Ring Distances

Let $u_i, u_j \in V^G$ be two nodes in G. The distance $d_V^G(u_i, u_j)$ between the nodes u_i and u_j is defined as the number of edges of a shortest path connecting them or as ∞ if they are in different connected components of G. The eccentricitiy of a node $u_i \in V^G$ and the diameter of a graph G are defined as $e_V^G(u_i) = \max_{u_j \in V^G} d_V^G(u_i, u_j)$ and $\mathrm{diam}(G) = \max_{u \in V^G} e_V^G(u)$, respectively.

Definition 3 (Ring, Layer, Outer Edges, Inner Edges). *Given a constant $L \in \mathbb{N}_{>0}$ and a node $u_i \in V^G$, we define the ring rooted at u_i in G as the sequence of disjoint layers $\mathcal{R}_L^G(u_i) = (\mathcal{L}_l^G(u_i))_{l=0}^{L-1}$ (Figure 1). The lth layer rooted at u_i is defined as $\mathcal{L}_l^G(u_i) = (V_l^G(u_i), OE_l^G(u_i), IE_l^G(u_i))$ where:*

- $V_l^G(u_i) = \{u_j \in V^G \mid d_V^G(u_i, u_j) = l\}$ *is the set of nodes at distance l of u_i,*
- $IE_l^G(u_i) = E^G \cap (V_l^G(u_i) \times V_l^G(u_i))$ *is the set of inner edges connecting two nodes in the lth layer, and*
- $OE_l^G(u_i) = E^G \cap (V_l^G(u_i) \times V_{l+1}^G(u_i))$ *is the set of outer edges connecting a node in the lth layer to a node in the $(l+1)^{th}$ layer.*

For the dummy node ϵ, we define $\mathcal{R}_L(\epsilon) = ((\emptyset, \emptyset, \emptyset)_l)_{l=0}^{L-1}$.
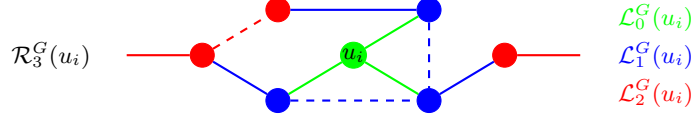

Fig. 1. Visualization of Definition 3. Inner edges are dashed, outer edges are solid.

Remark 1 (Properties of Rings and Layers). The first layer $\mathcal{L}_0^G(u_i)$ of a node u_i corresponds to u_i's local structure as defined by BP, BRANCH, BRANCH-FAST, and BRANCH-UNI. We have $OE_l^G(u_i) = \emptyset$ just in case $l > e_V^G(u_i) - 1$ and $\mathcal{L}_l^G(u_i) = (\emptyset, \emptyset, \emptyset)$ just in case $l > e_V^G(u_i)$. Moreover, the identities $E^G = \bigcup_{l=0}^{L-1}(OE_l^G(u_i) \cup IE_l^G(u_i))$ and $V^G = \bigcup_{l=0}^{L-1} V_l^G(u_i)$ hold for all $u_i \in V^G$ just in case $L > \mathrm{diam}(G)$.

In our instantiation RING of LSAPE-GED, we use rings as local structures, i.e., define $\mathcal{S}^G(u_i) = \mathcal{R}_L^G(u_i)$. The next step is to define a distance measure $d_{\mathcal{R}}$ that maps two rings to a non-negative real number. For doing so, we first define a measure $d_{\mathcal{L}}$ that returns the distance between two layers. So let $\mathcal{L}_l^G(u)$ and $\mathcal{L}_l^H(v)$ be the lth layers rooted at nodes $u \in V^G \cup \{\epsilon\}$ and $v \in V^H \cup \{\epsilon\}$, respectively. Then $d_{\mathcal{L}}$ is defined as

$$d_{\mathcal{L}}\left(\mathcal{L}_l^G(u), \mathcal{L}_l^H(v)\right) = \alpha_0 \phi_V\left(V_l^G(u), V_l^H(v)\right) + \alpha_1 \phi_E\left(OE_l^G(u), OE_l^H(v)\right) \\ + \alpha_2 \phi_E\left(IE_l^G(u), IE_l^H(v)\right),$$

where $\phi_V : \mathcal{P}(V^G) \times \mathcal{P}(V^H) \to \mathbb{R}_{\geq 0}$ and $\phi_E : \mathcal{P}(E^G) \times \mathcal{P}(E^H) \to \mathbb{R}_{\geq 0}$ are functions that measures the dissimilarity between two sets of nodes and edges, respectively, and $\alpha_0, \alpha_1, \alpha_2 \in \mathbb{R}_{\geq 0}$ are weights assigned to the dissimilarities between the nodes, the outer edges, and the inner edges. We now define $d_{\mathcal{R}}$ as

$$d_{\mathcal{R}}\left(\mathcal{R}_L^G(u), \mathcal{R}_L^H(v)\right) = \sum_{l=0}^{L-1} \lambda_l d_{\mathcal{L}}\left(\mathcal{L}_l^G(u), \mathcal{L}_l^H(v)\right), \tag{1}$$

where $\lambda_l \in \mathbb{R}_{\geq 0}$ are weights assigned to the distances between the layers.

Recall that we are defining $d_{\mathcal{R}}$ to the purpose of populating a LSAPE instance $\mathbf{C}$ which is then used to derive an upper bound for GED. Since we want this upper bound to be as tight as possible, we want $d_{\mathcal{R}}\left(\mathcal{R}_L^G(u), \mathcal{R}_L^H(v)\right)$ to be small if and only if we have good reasons to assume that substituting u by v leads to a small overall edit cost. This can be achieved by defining the functions ϕ_V and ϕ_E in a way that makes crucial use of the edit cost functions c_V and c_E:

LSAPE Based Definition of ϕ_V and ϕ_E. Let $U = \{u_1, \ldots, u_r\} \subseteq V^G$ and $V = \{v_1, \ldots, u_s\} \subseteq V^H$ be two node sets. Then a LSAPE instance $\mathbf{C} = (c_{i,k}) \in \mathbb{R}^{(r+1)\times(s+1)}$ is defined by setting $c_{i,k} = c_V(u_i, v_k)$, $c_{i,s+1} = c_V(i, \epsilon)$, and $c_{r+1,k} = c_V(\epsilon, v_k)$ for all $i \in \{1, \ldots, r\}$ and all $k \in \{1, \ldots, s\}$. This instance is solved — either optimally in $O(\min\{r, s\}^2 \max\{r, s\})$ time or greedily in $O(rs)$ time — and ϕ_V is defined to return $\mathbf{C}(\pi^\star)/\max\{|U|, |V|, 1\}$, where $\mathbf{C}(\pi^\star)$ is the cost of the computed solution $\pi^\star$. We normalize by the sizes of U and V in order not to overrepresent large layers. The function ϕ_E can be defined analogously.

Multiset Intersection Based Definition of ϕ_V and ϕ_E. Alternatively, we suggest to define ϕ_V as

$$\phi_V(U, V) = \left[\overline{c_V^{U,\epsilon}}\delta_{|U|\geq|V|}(|U| - |V|) + \overline{c_V^{\epsilon,V}}(1 - \delta_{|U|\geq|V|})(|V| - |U|)\right.$$
$$\left. + \overline{c_V^{U,V}}\left(\min\{|U|, |V|\} - |\ell_V^G[[U]] \cap \ell_V^H[[V]]|\right)\right]/\max\{|U|, |V|, 1\},$$

where $\delta_{|U|\geq|V|}$ equals 1 if $|U| \geq |V|$ and 0 otherwise, $\overline{c_V^{U,\epsilon}}$, $\overline{c_V^{\epsilon,V}}$, and $\overline{c_V^{U,V}}$ are the average costs of deleting a node in U, inserting a node in V, and substituting a node in U by a differently labeled node in V, and $\ell_V^G[[U]]$ and $\ell_V^H[[V]]$ are the multiset images of U and V under the labelling functions ℓ_V^G and ℓ_V^H. Again, ϕ_E can be defined analogously. Note that, if the edit costs are quasimetric, then the LSAPE based definition of ϕ_V and ϕ_E given above leads to the same number of node or edge substitutions, insertions, or deletions as the multiset intersection based definition; and if all substitution, insertion, and deletion costs are the same, then the two definitions are equivalent (cf. Proposition 1). Therefore, the multiset intersection based approach for defining ϕ_V and ϕ_E can be seen as a proxy for the one based on LSAPE. The advantage of using multiset intersection is that it allows for a very quick evaluation of ϕ_V and ϕ_E. In fact, since multiset intersections can be computed in quasilinear time [17], the dominant operation is the computation of the average substitution cost, which requires quadratic time. The drawback is that we loose some of the information encoded in the layers.

Proposition 1. *If all node substitution costs are equal to a constant c_V^S, all node removal costs to c_V^R, and all node insertion costs to c_V^I with $c_V^S \leq c_V^R + c_V^I$, then both definitions of ϕ_V coincide. For ϕ_E, an analogous proposition holds.*

Proof. We assume w. l. o. g. that $|U| \leq |V|$. Then, from $c_V^S \leq c_V^R + c_V^I$ and by [5], Proposition 1, the optimal solution π^* does not contain removals and contains exactly $|V| - |U|$ insertions. The optimal cost $\mathbf{C}(\pi^*)$ is thus reduced to the cost of $|V| - |U|$ insertions plus c_V^S times the number of non identical substitutions. This last quantity is provided by $\min\{|U|, |V|\} - l_V^G[[U]] \cap l_V^H[[V]]$. We thus have:

$$\mathbf{C}(\pi^*) = c_V^I(|V| - |U|) + c_V^S \left(\min\{|U|, |V|\} - l_V^G[[U]] \cap l_V^H[[V]]\right)$$

Since costs are constant, we have $\overline{c_V^{U,\epsilon}} = c_V^R, \overline{c_V^{U,V}} = c_V^S$, and $\overline{c_V^{\epsilon,V}} = c_V^I$, which provides the expected result. The proof for ϕ_E is analogous.   □

3.2   Algorithms and Choice of Meta-Parameters

Construction of the Rings and Overall Runtime Complexity. Figure 2 shows how to build the rings via breadth-first search. Clearly, constructing all rings of a graph G requires $O(|V^G|(|V^G| + |E^G|))$ time. After constructing the rings, the LSAPE instance $\mathbf{C}$ must be populated. Depending on the choice of ϕ_V and ϕ_E, this requires $O(|\text{supp}(\boldsymbol{\lambda})||V^G||V^H|\Omega^3)$ or $O(|\text{supp}(\boldsymbol{\lambda})||V^G||V^H|\Omega^2)$ time, where Ω is the size of the largest set contained in one of the rings of G and H, and $\text{supp}(\boldsymbol{\lambda})$ is the support of $\boldsymbol{\lambda}$. Finally, $\mathbf{C}$ is solved optimally in $O(\min\{|V^G|, |V^H|\}^2 \max\{|V^G|, |V^H|\})$ time or greedily in $O(|V^G||V^H|)$ time.

Choice of the Meta-Parameters $\boldsymbol{\alpha}$, $\boldsymbol{\lambda}$, and L. When introducing $d_{\mathcal{L}}$ and $d_{\mathcal{R}}$ in Section 3.1, we allowed $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$ to be arbitrary vectors from $\mathbb{R}_{\geq 0}^3$ and $\mathbb{R}_{\geq 0}^L$. However, we can be more restrictive: Since LSAPE does not care about scaling, we can assume w. l. o. g. that $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$ are simplex vectors, i. e., that we have $\sum_{s=0}^2 \alpha_s = \sum_{l=0}^{L-1} \lambda_l = 1$. This reduces the search space for $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$ but still leaves us with too many degrees of freedom for choosing them via grid search. We hence suggest to learn $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$ with the help of a blackbox optimizer [15]. For a training set of graphs $\mathcal{T}$ and a fixed $L \in \mathbb{N}_{>0}$, the optimizer should minimize

$$obj(\boldsymbol{\alpha}, \boldsymbol{\lambda}) = \left[\mu + (1 - \mu)\left(\frac{|\text{supp}(\boldsymbol{\lambda})| - 1}{\max\{1, L - 1\}}\right)\right] \sum_{(G,H) \in \mathcal{T}^2} \text{RING}_{\boldsymbol{\alpha}, \boldsymbol{\lambda}}^{\phi_V; \phi_E}(G, H)$$

and respect the constraints that $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$ are simplex vectors. $\text{RING}_{\boldsymbol{\alpha}, \boldsymbol{\lambda}}^{\phi_V; \phi_E}(G, H)$ is the upper bound for $\text{GED}(G, H)$ returned by RING given fixed $\boldsymbol{\alpha}$, $\boldsymbol{\lambda}$, ϕ_V, and ϕ_E, and $\mu \in [0, 1]$ is a tuning parameter that should be close to 1 if one wants to optimize for tightness and close to 0 if one wants to optimize for runtime. We include $|\text{supp}(\boldsymbol{\lambda})| - 1$ in the objective, because if $\boldsymbol{\lambda}$'s support is small, only few layer distances have to be computed (cf. Equation (1)). In particular, $|\text{supp}(\boldsymbol{\lambda})| = 1$ means that RING's runtime cannot be decreased any further via modification of $\boldsymbol{\lambda}$, which is why, in this case, the $(1 - \mu)$-part of the objective is set to 0.

Input: A graph G, a node $u \in V^G$, and a constant $L \in \mathbb{N}_{>0}$.
Output: The ring $\mathcal{R}_L^G(u)$ rooted at u.

$l \leftarrow 0$; $V \leftarrow \emptyset$; $OE \leftarrow \emptyset$; $IE \leftarrow \emptyset$; $\mathcal{R}_L^G(u) \leftarrow ((\emptyset,\emptyset,\emptyset)_l)_{l=0}^{L-1}$;   // *initialize ring*
$\mathtt{d}[u] \leftarrow 0$; **for** $u' \in V^G \setminus \{u\}$ **do** $\mathtt{d}[u'] \leftarrow \infty$;   // *initialize distances to root*
for $e \in E^G$ **do** $\mathtt{discovered}[e] \leftarrow \mathtt{false}$;   // *mark all edges as undiscovered*
$\mathtt{open} \leftarrow \{u\}$;   // *initialize FIFO queue*
while $\mathtt{open} \neq \emptyset$ **do**   // *main loop*
   $u' \leftarrow \mathtt{open.pop}()$;   // *pop node from queue*
   if $\mathtt{d}[u'] > l$ **then**   // *the lth layer is complete*
      $\mathcal{R}_L^G(u)_l = (V, OE, IE)$; $l \leftarrow l+1$;   // *store l^{th} layer and increment l*
      $V \leftarrow \emptyset$; $OE \leftarrow \emptyset$; $IE \leftarrow \emptyset$;   // *reset nodes, inner, and outer edges*
   $V \leftarrow V \cup \{u'\}$;   // *$u'$ is node at l^{th} layer*
   for $u'u'' \in E^G$ **do**   // *iterate through neighbours of u'*
      if $\mathtt{discovered}[u'u'']$ **then continue**;   // *skip discovered edges*
      if $\mathtt{d}[u''] = \infty$ **then**   // *found new node*
         $\mathtt{d}[u''] \leftarrow l+1$;   // *set distance of new node*
         if $\mathtt{d}[u''] < L$ **then** $\mathtt{open.push}(u'')$;   // *add close new node to queue*
      if $\mathtt{d}[u''] = l$ **then** $IE \leftarrow IE \cup \{u'u''\}$;   // *$u'u''$ is inner edge at l^{th} layer*
      else $OE \leftarrow OE \cup \{u'u''\}$;   // *$u'u''$ is outer edge at l^{th} layer*
      $\mathtt{discovered}[u'u''] \leftarrow \mathtt{true}$;   // *mark $u'u''$ as discovered*
$\mathcal{R}_L^G(u)_l = (V, OE, IE)$; **return** $\mathcal{R}_L^G(u)$;   // *store last layer and return ring*

Fig. 2. Construction of rings via Breadth-First Search.

Before building the rings for the graphs contained in the training set, L should be set to an upper bound for their diameters, e. g., to $L = 1 + \max_{G \in \mathcal{T}} |V^G|$. After the rings have been build, L can be lowered to $L = 1 + \max\{l \mid \exists G \in \mathcal{T}, u \in V^G : \mathcal{R}_L^G(u)_l \neq (\emptyset, \emptyset, \emptyset)\} = 1 + \max_{G \in \mathcal{T}} \mathrm{diam}(G)$ (cf. Remark 1). In the next step, the blackbox optimizer should be run, which returns an optimized pair of parameter vectors $(\boldsymbol{\alpha}^\star, \boldsymbol{\lambda}^\star)$. As the lth layers contribute to $d_{\mathcal{R}}$ only if $l \in \mathrm{supp}(\boldsymbol{\lambda}^\star)$ (cf. equation (1)), L can then be further lowered to $L = 1 + \max_{l \in \mathrm{supp}(\boldsymbol{\lambda}^\star)} l$.

4   Empirical Evaluation

We tested on the datasets MAO, PAH, ALKANE, and ACYCLIC, which contain graphs representing chemical compounds. For all datasets, we used the (non-uniform) edit costs 1 defined in [1], Table 2. We tested three variants of our method: $\mathtt{RING}^{\mathtt{OPT}}$ uses optimal LSAPE for defining the distance functions ϕ_V and ϕ_E, $\mathtt{RING}^{\mathtt{GD}}$ uses greedy LSAPE, and $\mathtt{RING}^{\mathtt{MS}}$ uses the multiset intersection based approach. We compared them to instantiations of $\mathtt{LSAPE\text{-}GED}$ that can cope with non-uniform edit costs: $\mathtt{BP}$, $\mathtt{BRANCH}$, $\mathtt{BRANCH\text{-}FAST}$, $\mathtt{SUBGRAPH}$, and $\mathtt{WALKS}$. As $\mathtt{WALKS}$ assumes that the costs of all edit operation types are constant, we slightly extended it by averaging the costs before each run. In order to handle the exponential complexity of $\mathtt{SUBGRAPH}$, we enforced a time limit of 1 ms for

computing a cell $c_{i,k}$ of its LSAPE instance. All methods were run with and without pagerank centralities with the meta-parameter β set to 0.3, which, in [12], is reported to be the setting that yields the tightest average upper bound.

For learning the meta-parameters of RING$^{\text{OPT}}$, RING$^{\text{GD}}$, RING$^{\text{MS}}$, SUBGRAPH, and WALKS, we picked a training set $\mathcal{T} \subset \mathcal{D}$ with $|\mathcal{T}| = 50$ for each dataset $\mathcal{D}$. As suggested in [6, 8], we learned the parameter L of the methods SUBGRAPH and WALKS by picking the $L \in \{1, 2, 3, 4, 5\}$ which yielded the tightest average upper bound on $\mathcal{T}$. For choosing the meta-parameters of the variants of RING, we proceeded as suggested in Section 3.2: We set the tuning parameter μ to 1 and used NOMAD [9] as our blackbox optimizer, which we initalized with 100 randomly constructed simplex vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$. All methods are implemented in C++ and use the same implementation of the LSAPE solver proposed in [5]. Except for WALKS, all methods allow to populate the LSAPE instance $\mathbf{C}$ in parallel and were set up to run in five threads. Tests were run on a machine with two Intel Xeon E5-2667 v3 processors with 8 cores each and 98 GB of main memory.[3]

For each dataset $\mathcal{D}$, we ran each method with and without pagerank centralities on each pair $(G, H) \in \mathcal{D} \times \mathcal{D}$ with $G \neq H$. We recorded the runtime and the value of the returned upper bound for GED. Figure 3 shows the results of our experiments. The first column shows the average runtimes and upper bounds of the tested methods without centralities. The second column shows the effect of including centralities. On all datasets, RING$^{\text{OPT}}$ yielded the tightest upper bound. Also RING$^{\text{MS}}$ performed excellently, as its upper bound deviated from the one produced by RING$^{\text{OPT}}$ by at most $4.15\,\%$ (on ALKANE). At the same time, on the datasets ACYCLIC, PAH, and MAO, RING$^{\text{MS}}$ was around two times faster than RING$^{\text{OPT}}$. On the contrary, RING$^{\text{GD}}$ was not significantly faster than RING$^{\text{OPT}}$ and, on ACYCLIC, produced a $16.18\,\%$ looser upper bound.

All competitors produced significantly looser upper bounds than our algorithms. In terms of runtime, our algorithms were outperformed by BRANCH, BRANCH-FAST, and BP, performed similarly to WALKS, and were much faster than SUBGRAPH. Adding pagerank centralities did not improve the overall performance of the tested methods: It lead to a maximal tightness gain of $4.90\,\%$ (WALKS on ALKANE) and dramatically increased the runtimes of some algorithms.

5   Conclusions and Future Work

In this paper, we have presented RING, a new instantiation of the paradigm LSAPE-GED which defines the local structure of a node u as a collection of node and edge sets at fixed distances from u. An empirical evaluation has shown that RING produces the tightest upper bound among all instantiations of LSAPE-GED. In the future, we will use ring structures for defining feature vectors of node assignments to be used in a machine learning based approach for approximating GED. Furthermore, we will examine how using RING for initialization affects the performance of the local search methods suggested in [4, 7, 11, 14].

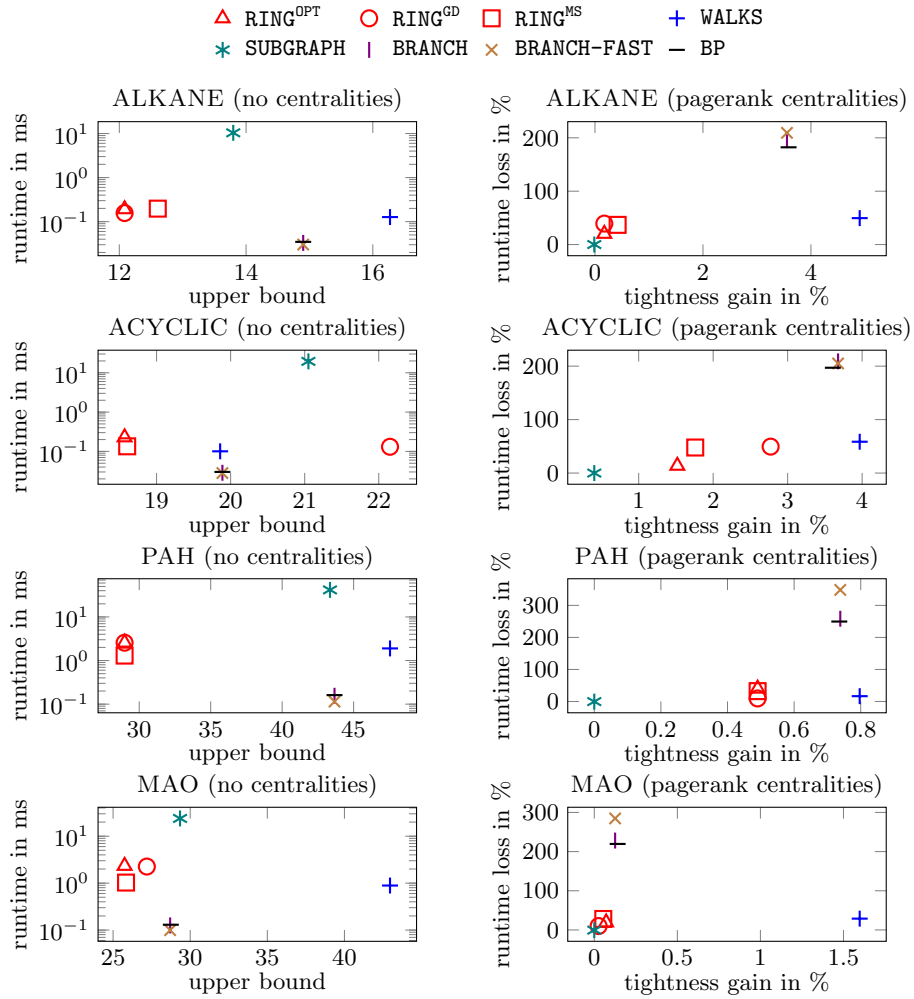[3] Source code and datasets: `http://www.inf.unibz.it/~blumenthal/gedlib.html`.

Fig. 3. Results of the experiments.

References

1. Abu-Aisheh, Z., Gaüzere, B., Bougleux, S., Ramel, J.Y., Brun, L., Raveaux, R., Héroux, P., Adam, S.: Graph edit distance contest 2016: Results and future challenges. Pattern Recogn. Lett. **100**, 96–103 (2017). https://doi.org/10.1016/j.patrec.2017.10.007
2. Blumenthal, D.B., Gamper, J.: Improved lower bounds for graph edit distance. IEEE Trans. Knowl. Data Eng. **30**(3), 503–516 (2018). https://doi.org/10.1109/TKDE.2017.2772243
3. Blumenthal, D.B., Gamper, J.: On the exact computation of the graph edit distance. Pattern Recogn. Lett. (2018). https://doi.org/10.1016/j.patrec.2018.05.002

4. Bougleux, S., Brun, L., Carletti, V., Foggia, P., Gaüzère, B., Vento, M.: Graph edit distance as a quadratic assignment problem. Pattern Recogn. Lett. **87**, 38–46 (2017). https://doi.org/10.1016/j.patrec.2016.10.001

5. Bougleux, S., Gaüzère, B., Blumenthal, D.B., Brun, L.: Fast linear sum assignment with error-correction and no cost constraints. Pattern Recogn. Lett. (2018). https://doi.org/10.1016/j.patrec.2018.03.032

6. Carletti, V., Gaüzère, B., Brun, L., Vento, M.: Approximate graph edit distance computation combining bipartite matching and exact neighborhood substructure distance. In: Liu, C., Luo, B., Kropatsch, W.G., Cheng, J. (eds.) GbRPR 2015. LNCS, vol. 9069, pp. 188–197. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18224-7_19

7. Ferrer, M., Serratosa, F., Riesen, K.: A first step towards exact graph edit distance using bipartite graph matching. In: Liu, C., Luo, B., Kropatsch, W.G., Cheng, J. (eds.) GbRPR 2015. LNCS, vol. 9069, pp. 77–86. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18224-7_8

8. Gauzère, B., Bougleux, S., Riesen, K., Brun, L.: Approximate graph edit distance guided by bipartite matching of bags of walks. In: Fränti, P., Brown, G., Loog, M., Escolano, F., Pelillo, M. (eds.) S+SSPR 2014. LNCS, vol. 8621, pp. 73–82. Cham (2014). https://doi.org/10.1007/978-3-662-44415-3_8

9. Le Digabel, S.: Algorithm 909: NOMAD: Nonlinear optimization with the MADS algorithm. ACM Trans. Math. Softw. **37**(4), 44:1–44:15 (2011). https://doi.org/10.1145/1916461.1916468

10. Riesen, K., Bunke, H.: Approximate graph edit distance computation by means of bipartite graph matching. Image Vis. Comput. **27**(7), 950–959 (2009). https://doi.org/10.1016/j.imavis.2008.04.004

11. Riesen, K., Bunke, H.: Improving bipartite graph edit distance approximation using various search strategies. Pattern Recogn. **48**(4), 1349–1363 (2015). https://doi.org/10.1016/j.patcog.2014.11.002

12. Riesen, K., Bunke, H., Fischer, A.: Improving graph edit distance approximation by centrality measures. In: ICPR 2014. pp. 3910–3914. IEEE Computer Society (2014). https://doi.org/10.1109/ICPR.2014.671

13. Riesen, K., Ferrer, M., Fischer, A., Bunke, H.: Approximation of graph edit distance in quadratic time. In: Liu, C., Luo, B., Kropatsch, W.G., Cheng, J. (eds.) GbRPR 2015. LNCS, vol. 9069, pp. 3–12. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18224-7_1

14. Riesen, K., Fischer, A., Bunke, H.: Improved graph edit distance approximation with simulated annealing. In: Foggia, P., Liu, C., Vento, M. (eds.) GbRPR 2017. LNCS, vol. 10310, pp. 222–231. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58961-9_20

15. Rios, L.M., Sahinidis, N.V.: Derivative-free optimization: a review of algorithms and comparison of software implementations. J. Global Optim. **56**(3), 1247–1293 (2013). https://doi.org/10.1007/s10898-012-9951-y

16. Serratosa, F., Cortés, X.: Graph edit distance: Moving from global to local structure to solve the graph-matching problem. Pattern Recogn. Lett. **65**, 204–210 (2015). https://doi.org/10.1016/j.patrec.2015.08.003

17. Zeng, Z., Tung, A.K.H., Wang, J., Feng, J., Zhou, L.: Comparing stars: On approximating graph edit distance. PVLDB **2**(1), 25–36 (2009). https://doi.org/10.14778/1687627.1687631

18. Zheng, W., Zou, L., Lian, X., Wang, D., Zhao, D.: Efficient graph similarity search over large graph databases. IEEE Trans. Knowl. Data Eng. **27**(4), 964–978 (2015). https://doi.org/10.1109/TKDE.2014.2349924