Exact Computation of Graph Edit Distance for Uniform and Non-uniform Metric Edit Costs

David B. Blumenthal(✉) and Johann Gamper

Faculty of Computer Science, Free University of Bolzano,
Piazza Dominicani 3, 39100 Bolzano, Italy
{david.blumenthal,gamper}@inf.unibz.it

Abstract. The graph edit distance is a well-established and widely used distance measure for labelled, undirected graphs. However, since its exact computation is *NP*-hard, research has mainly focused on devising approximative heuristics and only few exact algorithms have been proposed. The standard approach A*-GED, a node-based best-first search that works for both uniform and non-uniform metric edit costs, suffers from huge runtime and memory requirements. Recently, two better performing algorithms have been proposed: DF-GED, a node-based depth-first search that works for uniform and non-uniform metric edit costs, and CSI_GED, an edge-based depth-first search that works only for uniform edit costs. Our paper contains two contributions: First, we propose a speed-up DF-GED$^{\text{u}}$ of DF-GED for uniform edit costs. Second, we develop a generalisation CSI_GED$^{\text{nu}}$ of CSI_GED that also covers non-uniform metric edit cost. We empirically evaluate the proposed algorithms. The experiments show, i.a., that our speed-up DF-GED$^{\text{u}}$ clearly outperforms DF-GED and that our generalisation CSI_GED$^{\text{nu}}$ is the most versatile algorithm.

Keywords: Graph matching · Graph similarity · Graph edit distance · Branch and bound

1 Introduction

Labelled, undirected graphs can be used for modelling various kinds of objects, such as social networks, molecular structures, and many more. Because of this, labelled graphs have received increasing attention over the past years. One task researchers have focused on is the following: Given a database $\mathcal{G}$ that contains labelled graphs, find all graphs $G \in \mathcal{G}$ that are sufficiently similar to a query graph H or to find the k graphs from $\mathcal{G}$ that are most similar to H. For approaching this task, a distance measure between undirected, labelled graphs G and H has to be defined. One of the most commonly used measures is the graph edit distance. Formally, a *labelled, undirected graph* G is a 4-tuple $G = \langle V^G, E^G, \ell_V^G, \ell_E^G \rangle$, where V^G is a set of nodes, E^G is a set of undirected edges, and $\ell_V^G : V^G \to \Sigma_V$ and $\ell_E^G : E^G \to \Sigma_E$ are labelling functions that assign nodes an edges to labels from alphabets Σ_V and Σ_E. Both Σ_V and Σ_E contain a special label ε reserved for dummy nodes and dummy edges. The *graph edit distance* $\lambda(G, H)$ between

P. Foggia et al. (Eds.): GbRPR 2017, LNCS 10310, pp. 211–221, 2017.
DOI: 10.1007/978-3-319-58961-9_19

graphs G and H on common label alphabets Σ_V and Σ_E is defined as the minimum cost of an edit path between G and H. An *edit path* is a sequence of labelled graphs starting with G and ending at a graph that is isomorphic to H. Each graph along the path can be obtained from its predecessor by applying one of the following *edit operations*: Deleting or inserting an α-labelled edge, deleting or inserting an isolated α-labelled node, changing a node's or an edge's label from α to $\beta \neq \alpha$. Edit operations on nodes and edges come with associated *edit costs* $c_V : \Sigma_V \times \Sigma_V \rightarrow \mathbb{R}$ and $c_E : \Sigma_E \times \Sigma_E \rightarrow \mathbb{R}$, respectively. The cost of an edit path is defined as the sum of the costs of its edit operations. If the cost of each edit operation equals 1, we say that the edit costs are *uniform*. In many scenarios, it is natural to consider *non-uniform metric edit costs*. For instance, if the graphs model spacial objects and the node labels are Euclidean coordinates, the cost $c_V(\alpha, \beta)$ one has to pay for changing a node's label from α to β should probably be defined as the Euclidean distance between α and β.

It has been shown that, even for uniform edit costs, it is *NP*-hard to exactly compute the graph edit distance [14]. Exact algorithms that, if applied to large graphs, terminate within an acceptable amount of time are hence out of reach. Consequently, a substantial part of research on both uniform [14–16] and non-uniform [2–4,6,10,12,13] graph edit distance has focused on the task of devising heuristics that compute lower and/or upper bounds for $\lambda(G, H)$. Nonetheless, efficient exact algorithms are still important. This is because some of the objects that are readily modelled by labelled, undirected graphs — for instance, some molecular compounds — induce graphs with very few nodes [9]. For these graphs, queries of the kind "find all $G \in \mathcal{G}$ with $\lambda(G, H) \leq \tau$" can in principle be answered. Of course, one would first use efficiently computable upper and lower bounds in order to filter out candidates from $\mathcal{G}$. However, for the surviving candidates, $\lambda(G, H) \leq \tau$ has to be verified by means of an exact algorithm.

The standard approach `A*-GED` [11] for exactly computing $\lambda(G, H)$ carries out a node-based best-first search in order to find the optimal edit path. It is very slow and has huge memory requirements. Recently, three better performing algorithms `BLP-GED` [8], `DF-GED` [1], and `CSI_GED` [5] have been proposed. `BLP-GED` formulates the problem of computing $\lambda(G, H)$ as a binary linear program which is solved by calling the commercial solver CPLEX. It has been found to be faster and more memory-efficient than `A*-GED`. `DF-GED` carries out a node-based depth-first search for finding the cheapest edit path. It has been found to be much more memory-efficient and slightly faster than `A*-GED`. In contrast, `CSI_GED` carries out an edge-based depth-first search. It also has been found to be both faster and much more memory-efficient than `A*-GED`. While `A*-GED`, `BLP-GED`, and `DF-GED` cover non-uniform metric edit costs, `CSI_GED` only works for uniform edit costs. A direct comparison between `BLP-GED`, `DF-GED`, and `CSI_GED` is lacking.

Our paper contains the following contributions: In Sect. 2, we present a speed-up `DF-GED`$^{\mathtt{u}}$ of `DF-GED`for uniform edit costs. `DF-GED`$^{\mathtt{u}}$ exploits the fact that, in the uniform case, a subroutine that `DF-GED` employs at each node of its search tree can be implemented to run in linear rather than cubic time. In Sect. 3, we propose a generalisation `CSI_GED`$^{\mathtt{nu}}$ of `CSI_GED` that also covers non-uniform metric

edit costs. This generalisation comes at the price of a slightly increased runtime. However, this increase is very moderate, as the computational complexity is increased only at the initialisation of CSI_GED$^{\text{nu}}$ and at the leafs of its search tree. In Sect. 4, we experimentally evaluate the performance of the newly proposed algorithms. The experiments show that, for uniform edit costs, our speed-up DF-GED$^{\text{u}}$ clearly outperforms DF-GED, while CSI_GED and our generalisation CSI_GED$^{\text{nu}}$ perform similarly. They also indicate that, neither for uniform nor for non-uniform edit costs, there is a clear winner between DF-GED$^{\text{u}}$ and DF-GED, on the one side, and CSI_GED and CSI_GED$^{\text{nu}}$, on the other side. Finally, the experiments suggest that CSI_GED$^{\text{nu}}$ is the most versatile algorithm: It covers both uniform and non-uniform edit costs and runs very stable even on datasets where other algorithms perform better. Section 5 concludes the paper.

2 DF-GED$^{\text{u}}$: Fast DF-GED for Uniform Edit Costs

In this section, we show how to speed-up the node-based depth-first search DF-GED for uniform edit costs. We first summarise DF-GED and then describe our speed-up DF-GED$^{\text{u}}$.

The Baseline Approach. DF-GED builds upon the following observation: If edit costs are metric, then $\lambda(G, H)$ can be defined equivalently as the minimum cost of an edit path that is induced by a node map [6]. Let $V^{G+|H|}$ and $V^{G+|H|}$ be the sets that are obtained from V^G and V^H by adding $|V^H|$ respectively $|V^G|$ isolated dummy nodes. A *node map* is an injective partial function $\pi : V^{G+|H|} \to V^{H+|G|}$, whose domain contains V^G and whose image contains V^H. For a given node map π, its *induced edit path* is defined as follows: If π maps a real node $i \in V^G$ to a dummy node j_ε, i is deleted. Conversely, if a dummy node i_ε is mapped to a real node $k \in V^H$, k is inserted. If a real node $i \in V^G$ is mapped to a real node $k \in V^H$, i's label is changed from $\ell_V^G(i)$ to $\ell_V^H(k)$. If $ij \in E^G$ but $\pi(i)\pi(j) \notin E^H$, the edge ij is deleted. If $kl \in E^H$ but $\pi^{-1}(k)\pi^{-1}(l) \notin E^G$, the edge kl is inserted. Finally, if an edge $ij \in E^G$ is mapped to an edge $kl \in E^H$, ij's label is changed from $\ell_E^G(ij)$ to $\ell_E^H(kl)$. The cost of the edit path induced by π is denoted by $g(\pi)$.

DF-GED performs a depth-first search on the set of all partial node maps between $V^{G+|H|}$ and $V^{H+|G|}$ starting with the empty node map. The tree's leafs correspond to complete node maps and its inner nodes correspond to incomplete node maps. DF-GED starts with sorting the nodes of V^G such that evident nodes will be processed first [3]. It also initialises an upper bound UB for $\lambda(G, H)$, using a fast sub-optimal heuristic [10]. For each visited node π of the search tree, values $g(\pi)$ and $h(\pi)$ are maintained. The value $g(\pi)$ denotes the cost of the corresponding incomplete induced edit path, and $h(\pi)$ is a lower bound for the cost from π to a leaf, i.e., complete node map, in π's down-shadow. Assume that all nodes in V^G up to node i have already been assigned by π. If i is the last node in V^G, π is extended to a complete node map by assigning a dummy node to each of the yet unassigned nodes $j \in V^G$, and UB is updated to $g(\pi)$ if

$g(\pi) < UB$. Otherwise, π's children $\pi' \in \{\pi \cup (i+1, j) : j \in V^H \text{ unassigned by } \pi\} \cup \{\pi \cup (i+1, j_\varepsilon)\}$ are considered in order of non-decreasing $g(\pi') + h(\pi')$. If $g(\pi') + h(\pi') < UB$, π is updated to π' and the process iterates. Otherwise, the branch rooted at π' is pruned. At termination, UB is returned.

Note that, for each visited partial node map π, the lower bound $h(\pi)$ has to be recomputed. `DF-GED` computes $h(\pi)$ as follows: For a given partial node map π, let $V^{G+|H|-\pi}$ and $V^{H+|G|-\pi}$ be the sets of unassigned nodes and $E^{G-\pi}$ and $E^{H-\pi}$ be the sets of unassigned edges filled up with dummy edges to ensure $|E^{G-\pi}| = |E^{H-\pi}|$. Furthermore, let $\ell_V^G(V^{G+|H|-\pi})$, $\ell_V^H(V^{H+|G|-\pi})$, $\ell_E^G(E^{G-\pi})$, and $\ell_E^H(E^{H-\pi})$ denote the multisets of labels of the unassigned nodes or edges contained in these sets. Then $h(\pi)$ is defined as $h(\pi) = h_V(\pi) + h_E(\pi)$, where $h_V(\pi)$ is the minimum cost of a linear assignment between $\ell_V^G(V^{G+|H|-\pi})$ and $\ell_V^H(V^{H+|G|-\pi})$ with assignment costs c_V, and $h_E(\pi)$ is the minimum cost of a linear assignment between $\ell_E^G(E^{G-\pi})$ and $\ell_E^H(E^{H-\pi})$ with assignment costs c_E. Since a minimum linear assignment can be computed in cubic time, e.g., by using the Hungarian Algorithm [7], the runtime complexity of computing $h(\pi)$ is thus cubic in n and m, where $n = |V^G| + |V^H|$ and $m = \max\{|E^G|, |E^H|\}$.

Our Speed-Up for Uniform Edit Costs. Our speed-up `DF-GED`$^{\mathtt{u}}$ builds upon the observation that, for uniform edit cost, $h(\pi)$ can be computed in linear time. For showing this, we need the following lemma:

Lemma 1. *Let A and B be two equally sized multisets and $c : A \times B \to \mathbb{R}$ be uniform in the sense that $c(a, b)$ equals 1 if $a \neq b$ and 0 otherwise. Then the cost of a minimum linear assignment between A and B for the assignment cost c equals $|A| - |A \cap B|$.*

Proof. Let $(a_i)_{i=1}^{|A|}$ and $(b_i)_{i=1}^{|B|}$ be orderings of A and B such that, for all $i \leq |A \cap B|$, it holds that $a_i = b_i$. Note that this implies $a_i \neq b_i$ for all $i > |A \cap B|$. We define $f : A \to B$ as $f(a_i) = b_i$. It is easy to see that f is a minimum linear assignment between A and B for the uniform assignment cost c. Its cost is $\sum_{a \in A} c(a, f(a)) = \sum_{i=1}^{|A \cap B|} c(a_i, b_i) + \sum_{i=|A \cap B|+1}^{|A|} c(a_i, b_i) = |A| - |A \cap B|$. □

It has been shown that, if A and B are *sorted* multisets, the size of their intersection can be computed in linear time [14]. Together with Lemma 1, this immediately implies that, if c_V and c_E are uniform, $h_V(\pi)$ and $h_E(\pi)$ can be computed in $\mathcal{O}(n \log n)$ and $\mathcal{O}(m \log m)$ time, respectively: We first sort the labels of the nodes and the edges that have not been assigned by π in $\mathcal{O}(n \log n)$ and $\mathcal{O}(m \log m)$ time, respectively. Then, we compute the intersection sizes of the resulting sorted multisets in linear time. In order to further reduce the complexity of the computation of $h_V(\pi)$ and $h_E(\pi)$, we proceed as follows. When initialising `DF-GED`, we *once* sort $\ell_V^G(V^{G+|H|})$, $\ell_V^H(V^{H+|G|})$, $\ell_E^G(E^G)$, and $\ell_E^H(E^H)$, i.e., the multisets containing the labels of *all* nodes and edges. For each L of the resulting sorted multisets and each partial node map π, we maintain a boolean vector that indicates if the node or edge with label L_i is still unassigned by π. This vector can be updated in constant additional time when updating the cost $g(\pi)$ of the

partial edit path induced by π. For each partial node map π, $h_V(\pi)$ and $h_E(\pi)$ can then be computed in linear time by using a variation of the algorithm for multiset intersection presented in [14].

3 CSI_GED$^{\text{nu}}$: CSI_GED for Non-uniform Metric Edit Costs

In this section, we show how to generalise the edge-based depth-first search CSI_GED to non-uniform metric edit costs. We first summarise CSI_GED and then describe our generalisation CSI_GED$^{\text{nu}}$.

The Baseline Approach. While DF-GED enumerates the space of all node maps, CSI_GED considers *valid edge maps* $\phi : \overrightarrow{E^G} \to \overleftrightarrow{E^H} \cup \{e_\varepsilon\}$. The set $\overrightarrow{E^G}$ contains one arbitrarily oriented edge (i, j) for each undirected edge $ij \in E^G$, $\overleftrightarrow{E^H}$ contains two directed edges (k, l) and (l, k) for each $kl \in E^H$, and e_ε denotes a dummy edge. An edge map ϕ induces a relation π_ϕ on $V^G \times V^H$: If $\phi(i, j) = (k, l)$, then $(i, k) \in \pi_\phi$ and $(j, l) \in \pi_\phi$. Since nodes cannot be assigned twice, ϕ is called *valid* if and only if π_ϕ is a partial injective function. A valid edge map ϕ also induces a partial edit path between G and H: If $\phi(i, j) = (k, l)$, ij's label is changed from $\ell_E^G(ij)$ to $\ell_E^H(kl)$. If $\phi(i, j) = e_\varepsilon$, the edge ij is deleted. If $\phi^{-1}[\{(k, l), (l, k)\}] = \emptyset$ holds for an edge $kl \in E^H$, kl is inserted. And if $\pi_\phi(i) = k$, i's label is changed from $\ell_V^G(i)$ to $\ell_V^H(k)$. The cost of the partial edit path induced by ϕ is denoted by $g(\phi)$. In general, ϕ's induced edit path is incomplete, since the sets $V^{G-\pi_\phi} \subseteq V^G$ and $V^{H-\pi_\phi} \subseteq V^H$ containing the nodes that are left unassigned by π_ϕ are in general non-empty. The following theorem constitutes the backbone of CSI_GED:

Theorem 1 (Cf. Theorem 1 in [5]). *If the edit costs c_V and c_E are uniform, then, for each node map $\pi : V^{G+|H|} \to V^{H+|G|}$, there is a valid edge map $\phi : \overrightarrow{E^G} \to \overleftrightarrow{E^H} \cup \{e_\varepsilon\}$ with $g(\pi) \geq g(\phi) + \Gamma(V^{G-\pi_\phi}, V^{H-\pi_\phi})$, where $\Gamma(V^{G-\pi_\phi}, V^{H-\pi_\phi}) = \max\{|V^{G-\pi_\phi}|, |V^{H-\pi_\phi}|\} - |\ell_V^G(V^{G-\pi_\phi}) \cap \ell_V^H(V^{H-\pi_\phi})|$. Moreover, $g(\phi) + \Gamma(V^{G-\pi_\phi}, V^{H-\pi_\phi}) \geq \lambda(G, H)$ holds for each valid each map ϕ.*

Theorem 1 implies that, for uniform edit costs, one can compute the graph edit distance by enumerating the space of all valid edge maps. To this purpose, CSI_GED carries out a depth-first search on the set of all valid partial edge maps starting with the empty edge map. CSI_GED maintains an upper bound for the graph edit distance, which is initialised as $UB = \infty$, and considers the edges $e_r \in \overrightarrow{E^G}$ in an arbitrary but fixed order. For each visited incomplete edge map ϕ, the current induced cost $g(\phi)$ and a lower bound $g'(\phi)$ for the induced cost of a complete edge map in ϕ's down-shadow are maintained. Assume that all edges in $\overrightarrow{E^G}$ up to e_r have already been assigned by ϕ. If e_r is the last edge in $\overrightarrow{E^G}$, ϕ is a complete valid edge map, and UB is updated to $g(\phi) + \Gamma(V^{G-\pi_\phi}, V^{H-\pi_\phi})$ if $g(\phi) + \Gamma(V^{G-\pi_\phi}, V^{H-\pi_\phi}) < UB$. Otherwise, ϕ's children $\phi' \in \{\phi \cup (e_{r+1}, e) : e \in \overleftrightarrow{E^H}$ unassigned by ϕ and $\phi \cup (e_{r+1}, e)$ valid$\} \cup \{\phi \cup (e_{r+1}, e_\varepsilon)\}$ are considered in order of non-decreasing $\mathcal{C}(e_{r+1}, e)$. $\mathcal{C}(e_{r+1}, e)$ is an estimate of the graph edit

distance under the constraint that the edge e_{r+1} is mapped to e. Note that the estimated cost matrix $\mathcal{C}$ only has to be computed once at initialisation. If $g'(\phi') < UB$, ϕ is updated to ϕ' and the process iterates. Otherwise, the branch rooted at ϕ' is pruned. At termination, UB is returned.

Our Generalisation to Non-uniform Metric Edit Costs. The key-ingredient of our extension CSI_GED$^{\mathtt{nu}}$ is the following generalised version of Theorem 1:

Theorem 2. *If the edit costs c_V and c_E are metric, then, for each node map $\pi : V^{G+|H|} \to V^{H+|G|}$, there is a valid edge map $\phi : \overrightarrow{E^G} \to \overleftrightarrow{E^H} \cup \{e_\varepsilon\}$ with $g(\pi) \geq g(\phi) + \Gamma^{\mathtt{nu}}(V^{G-\pi_\phi}, V^{H-\pi_\phi})$, where $\Gamma^{\mathtt{nu}}(V^{G-\pi_\phi}, V^{H-\pi_\phi})$ is defined as the cost of a minimum linear assignment between $\ell_V^G(V^{G+|H|-\pi_\phi})$ and $\ell_V^H(V^{H+|G|-\pi_\phi})$ for the assignment cost c_V. Moreover, $g(\phi) + \Gamma^{\mathtt{nu}}(V^{G-\pi_\phi}, V^{H-\pi_\phi}) \geq \lambda(G, H)$ holds for each valid edge map ϕ.*

Proof. Given a node map π, we construct a valid edge map ϕ as follows: Let $(i, j) \in \overrightarrow{E^G}$. If the corresponding undirected edge ij is preserved under π, i.e., if $\pi(i)\pi(j) \in E^H$, we define $\phi(i, j) = (\pi(i), \pi(j))$. Otherwise, we set $\phi(i, j) = e_\varepsilon$. By construction, π_ϕ equals the restriction of π to those real nodes $i \in V^G$ that are incident with an edge that is preserved under π. This implies that ϕ is valid. Next, we compare the complete edit path P_π that is induced by π and the partial edit path P_ϕ that is induced by ϕ. We observe that P_ϕ contains all edge-deletions, -insertions, and -relabelings that appear in P_π, as well as all relabelings of nodes that are incident with a preserved edge. Apart from these edit operations, P_π also contains deletions and relabelings of nodes that are not incident with a preserved edge, as well as node-insertions. These latter operations can be viewed as a linear assignment between $\ell_V^G(V^{G+|H|-\pi_\phi})$ and $\ell_V^H(V^{H+|G|-\pi_\phi})$ for the assignment cost c_V, which, together with the observation above, implies $g(\pi) \geq g(\phi) + \Gamma^{\mathtt{nu}}(V^{G-\pi_\phi}, V^{H-\pi_\phi})$. For showing the second part of the theorem, we fix a valid edge map ϕ. Let π'_ϕ be a minimum linear assignment between $\ell_V^G(V^{G+|H|-\pi_\phi})$ and $\ell_V^H(V^{H+|G|-\pi_\phi})$ for the assignment cost c_V. Then $\pi = \pi_\phi \cup \pi'_\phi$ is a complete node map. By construction, we have $g(\phi) + \Gamma^{\mathtt{nu}}(V^{G-\pi_\phi}, V^{H-\pi_\phi}) \geq g(\pi) \geq \lambda(G, H)$, where the last inequality follows from the fact that, for metric edit costs, the graph edit distance can be defined as the minimum cost of an edit path that is induced by a node map. □

Theorem 2 indicates how to extend CSI_GED to non-uniform metric edit costs: We just have to replace all occurrences of Γ by $\Gamma^{\mathtt{nu}}$. As presented above, during the depth-first search carried out by CSI_GED, Γ has to be computed at the leafs of the search tree. At initialisation, further computations of Γ are required for computing the estimated cost matrix $\mathcal{C}$ and a constant that is required for the computation of g' (cf. [5] for these details of CSI_GED). Note that computing Γ requires linear time, whereas computing $\Gamma^{\mathtt{nu}}$ needs cubic time (cf. Sect. 2). This implies that our generalisation leads to an increased runtime of CSI_GED.

However, the increase is very moderate, as $\Gamma^{\mathtt{nu}}$ does not need to be computed at the inner nodes of the (exponentially large) search tree.

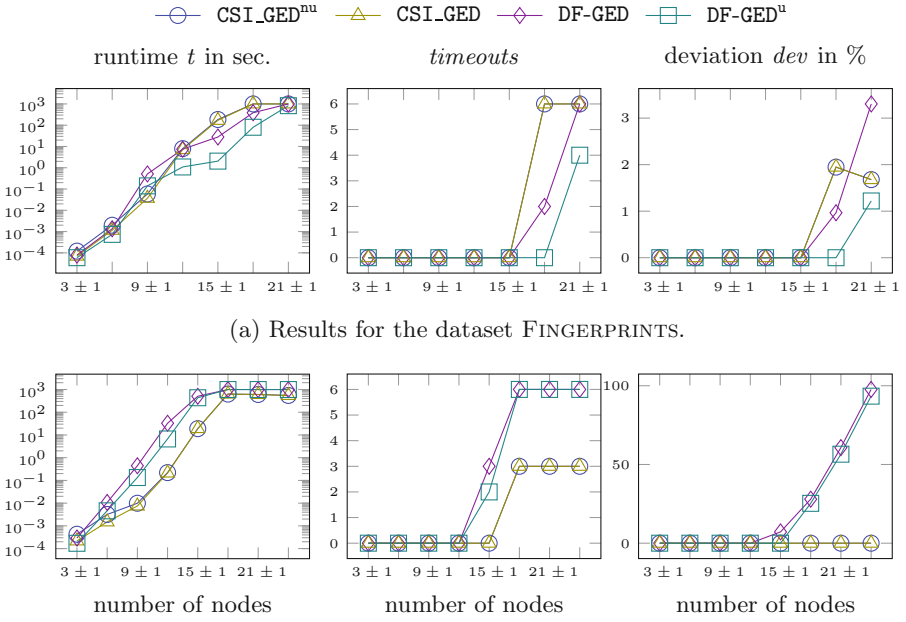
4 Empirical Evaluation

The aim of our experiments is to compare the performance of the algorithms CSI_GED, CSI_GED$^{\mathtt{nu}}$, DF-GED, and DF-GED$^{\mathtt{u}}$ for both uniform and non-uniform metric edit costs. We implemented all algorithms in C++ making them employ the same data structures and subroutines. All tests were carried out on a machine with two Intel Xeon E5-2667 v3 processors with 8 cores each and 98 GB of main memory running GNU/Linux. We conducted tests on the datasets AIDS and FINGERPRINTS [9], which are widely used in the research community [5,10–16]. Both datasets contain graphs with both node and edge labels for which non-uniform metric relabelling costs c_V and c_E are naturally induced by the domain [10]. For defining non-uniform metric edit costs, we thus only had to specify the deletion/insertion costs $c_V(\alpha, \varepsilon)$ and $c_E(\alpha, \varepsilon)$. This was done by setting $c_V(\alpha, \varepsilon) = \max\{c_V(\beta, \gamma) \mid \beta, \gamma \in \Sigma_V\}$ for all $\alpha \in \Sigma_V \setminus \{\varepsilon\}$, and $c_E(\alpha, \varepsilon) = \max\{c_E(\beta, \gamma) \mid \beta, \gamma \in \Sigma_E\}$ for all $\alpha \in \Sigma_E \setminus \{\varepsilon\}$, i.e., deleting and inserting nodes and edges was defined to be as expensive as the most expensive relabelling operations. Since both DF-GED and CSI_GED fail to compute the exact graph edit distance for graphs with more than 25 nodes within reasonable time [1,5], we excluded larger graphs from AIDS. FINGERPRINTS only contains small graphs, anyway. We then used the experimental setup suggested in [5]: For both considered datasets $\mathcal{D}$ and all $i \in \{3, 6, \ldots, \max_{G \in \mathcal{D}} |G|\}$, we defined a size-constrained test-group $\mathcal{G}_i$ that contains four randomly selected graphs $G \in \mathcal{D}$ satisfying $|V^G| = i \pm 1$. For each tested algorithm ALGand each test-group $\mathcal{G}_i$, all six pairwise comparisons between graphs contained in $\mathcal{G}_i$ were carried out. We set a time limit of 1000 s and recorded the metrics *timeouts*, t, and *dev*. Since pretesting showed that the main memory demand of all tested algorithms is negligible, we did not record memory usage.

- *timeouts*(ALG, i): The number of timeouts on $\mathcal{G}_i$, i.e., of pairwise comparisons between graphs in $\mathcal{G}_i$ where ALGdid not finish within 1000 s.
- t(ALG, i): ALG's average runtime across all six pairwise comparisons between graphs in $\mathcal{G}_i$.
- *dev*(ALG, i): ALG's average percentual deviation from the best tested algorithm as introduced in [1], i.e., the average of $100 \cdot [UB(\mathtt{ALG}) - UB^\star]/UB^\star$ across all six pairwise comparisons between graphs in $\mathcal{G}_i$. UB(ALG) denotes the value of the upper bound UB maintained by ALGafter 1000 s and $UB^\star$ is defined as $UB^\star = \min\{UB(\mathtt{ALG}') \mid \mathtt{ALG}' \text{ is tested algorithm}\}$.

Figure 1 shows the outcomes of our experiments for uniform edit costs. We observe that, on both datasets, our speed-up DF-GED$^{\mathtt{u}}$ outperforms DF-GED in terms of all recorded metrics, while CSI_GED and our generalisation CSI_GED$^{\mathtt{nu}}$ perform similarly. For instance, on FINGERPRINTS, DF-GED$^{\mathtt{u}}$ is on average 4.75 times faster than DF-GED, while $\text{avg}_i\, t(\mathtt{CSI\_GED}^{\mathtt{nu}}, i)/t(\mathtt{CSI\_GED}, i) \approx 1.32$. On AIDS, we have $\text{avg}_i\, t(\mathtt{DF\text{-}GED}, i)/t(\mathtt{DF\text{-}GED}^{\mathtt{u}}, i) \approx 2.05$ and $\text{avg}_i\, t(\mathtt{CSI\_GED}^{\mathtt{nu}}, i)/$

$t(\texttt{CSI\_GED}, i) \approx 1.31$. These results are readily explained by the fact that DF-GED has to carry out the cubic computation of the lower bound h at each node of its search tree, whereas $\texttt{CSI\_GED}^{\texttt{nu}}$ has to carry out the cubic computation of $\Gamma^{\texttt{nu}}$ only at the leafs and at initialisation. Secondly, we see that, on FINGERPRINTS (cf. Fig. 1a), the node-based approaches $\texttt{DF-GED}^{\texttt{u}}$ and DF-GED outperform the edge-based algorithms CSI_GED and $\texttt{CSI\_GED}^{\texttt{nu}}$, while, on AIDS (cf. Fig. 1b), the opposite is the case. Finally, we note that the edge-based algorithms are more stable: While their deviation never exceeds 2%, the node-based approaches' deviation explodes on comparisons between large graphs contained in AIDS.

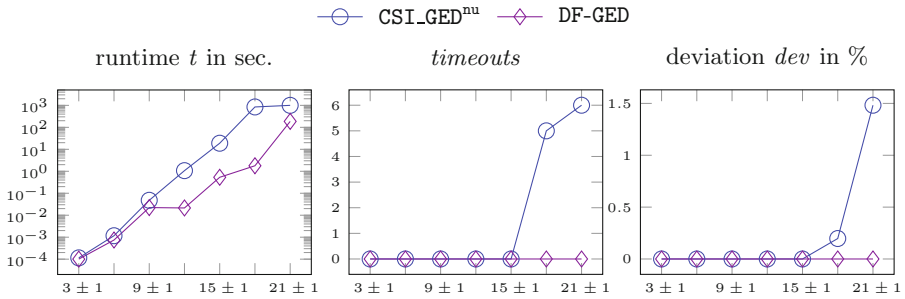


(a) Results for the dataset FINGERPRINTS.

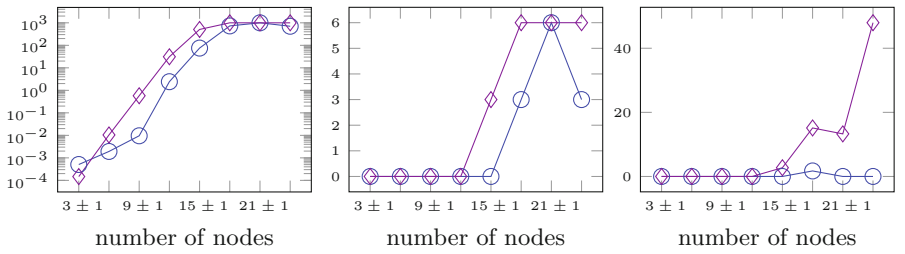
(b) Results for the dataset AIDS.

Fig. 1. Results for uniform edit costs.

The results for non-uniform edit metric costs are displayed in Fig. 2. Note that the algorithms $\texttt{DF-GED}^{\texttt{u}}$ and CSI_GED do not appear in the evaluation, as they are designed only for uniform edit costs. The first observation is that, just like for uniform edit costs, the node-based approach DF-GED performs better on FINGERPRINTS (cf. Fig. 2a), while our edge-based generalisation $\texttt{CSI\_GED}^{\texttt{nu}}$ performs better on AIDS (cf. Fig. 2b). Secondly, we again note that the edge-based algorithm runs much more stable than the node-based approach: On FINGERPRINTS, i.e., the dataset where DF-GED performs better, we have $\max_i dev(\texttt{CSI\_GED}^{\texttt{nu}}, i) \approx 1.48$; whereas on AIDS, i.e., the dataset where $\texttt{CSI\_GED}^{\texttt{nu}}$ performs better, we observe $\max_i dev(\texttt{DF-GED}, i) \approx 47.97$.



(a) Results for the dataset FINGERPRINTS.



(b) Results for the dataset AIDS.

Fig. 2. Results for non-uniform metric edit costs.

5 Conclusions and Future Work

Our experiments show that, for uniform edit costs, our speed-up DF-GED^{u} always outperforms DF-GED, while CSI_GED and our generalisation CSI_GED^{nu} perform similarly. We also observed that, neither for uniform nor for non-uniform metric edit costs, there is a clear winner between the node-based approaches DF-GED^{u} and DF-GED, on the one side, and the edge-based algorithms CSI_GED and CSI_GED^{nu}, on the other side. On FINGERPRINTS, the former two algorithms outperformed the latter in terms of runtime and timeouts, while on AIDS, the opposite outcome was observed. However, CSI_GED^{nu} and CSI_GED turned out to be more stable than DF-GED and DF-GED^{u}: While CSI_GED^{nu}'s and CSI_GED's deviation is small across all test-runs, DF-GED's and DF-GED^{u}'s deviation explodes for comparisons between large graphs contained in the AIDS dataset. A global assessment of these observations indicates that, if there is no prior knowledge about the dataset and the graph edit distance has to be computed for both uniform and non-uniform metric edit costs, our generalisation CSI_GED^{nu} is the algorithm of choice. For future research, it might be interesting to individuate graph-properties that indicate if the node-based approaches DF-GED^{u} and DF-GED or the edge-based algorithms CSI_GED and CSI_GED^{nu} perform better. A meta-algorithm could then first compute these properties and select node-based or edge-based algorithms accordingly.

References

1. Abu-Aisheh, Z., Raveaux, R., Ramel, J.Y., Martineau, P.: An exact graph edit distance algorithm for solving pattern recognition problems. In: Marsico, M.D., Figueiredo, M.A.T., Fred, A.L.N. (eds.) ICPRAM 2015, vol. 1, pp. 271–278. SciTePress, Setúbal (2015)
2. Carletti, V., Gaüzère, B., Brun, L., Vento, M.: Approximate graph edit distance computation combining bipartite matching and exact neighborhood substructure distance. In: Liu, C.-L., Luo, B., Kropatsch, W.G., Cheng, J. (eds.) GbRPR 2015. LNCS, vol. 9069, pp. 188–197. Springer, Cham (2015). doi:10.1007/978-3-319-18224-7_19
3. Ferrer, M., Serratosa, F., Riesen, K.: Learning heuristics to reduce the overestimation of bipartite graph edit distance approximation. In: Perner, P. (ed.) MLDM 2015. LNCS (LNAI), vol. 9166, pp. 17–31. Springer, Cham (2015). doi:10.1007/978-3-319-21024-7_2
4. Gaüzère, 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. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44415-3_8
5. Gouda, K., Hassaan, M.: CSI_GED: an efficient approach for graph edit similarity computation. In: ICDE 2016, pp. 265–276. IEEE Computer Society (2016)
6. Justice, D., Hero, A.: A binary linear programming formulation of the graph edit distance. IEEE Trans. Pattern Anal. Mach. Intell. **28**(8), 1200–1214 (2006)
7. Kuhn, H.W.: The hungarian method for the assignment problem. Nav. Res. Logist. Q. **2**(1–2), 83–97 (1955)
8. Lerouge, J., Abu-Aisheh, Z., Raveaux, R., Héroux, P., Adam, S.: Exact graph edit distance computation using a binary linear program. In: Robles-Kelly, A., Loog, M., Biggio, B., Escolano, F., Wilson, R. (eds.) S+SSPR 2016. LNCS, vol. 10029, pp. 485–495. Springer, Cham (2016). doi:10.1007/978-3-319-49055-7_43
9. Riesen, K., Bunke, H.: IAM graph database repository for graph based pattern recognition and machine learning. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) S+SSPR 2008. LNCS, vol. 5342, pp. 287–297. Springer, Heidelberg (2008)
10. Riesen, K., Bunke, H.: Approximate graph edit distance computation by means of bipartite graph matching. Image Vis. Comput. **27**(7), 950–959 (2009)
11. Fankhauser, S., Riesen, K., Bunke, H.: Speeding up graph edit distance computation through fast bipartite matching. In: Jiang, X., Ferrer, M., Torsello, A. (eds.) GbRPR 2011. LNCS, vol. 6658, pp. 102–111. Springer, Heidelberg (2011). doi:10.1007/978-3-642-20844-7_11
12. Riesen, K., Fischer, A., Bunke, H.: Computing upper and lower bounds of graph edit distance in cubic time. In: Gayar, N., Schwenker, F., Suen, C. (eds.) ANNPR 2014. LNCS (LNAI), vol. 8774, pp. 129–140. Springer, Cham (2014). doi:10.1007/978-3-319-11656-3_12
13. Serratosa, F.: Fast computation of bipartite graph matching. Pattern Recogn. Lett. **45**, 244–250 (2014)
14. 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)
15. Zhao, X., Xiao, C., Lin, X., Wang, W.: Efficient graph similarity joins with edit distance constraints. In: Kementsietsidis, A., Salles, M.A.V. (eds.) ICDE 2012, pp. 834–845. IEEE Computer Society (2012)

16. Zheng, W., Zou, L., Lian, X., Wang, D., Zhao, D.: Graph similarity search with edit distance constraint in large graph databases. In: He, Q., Iyengar, A., Nejdl, W., Pei, J., Rastogi, R. (eds.) CIKM 2013, pp. 1595–1600. ACM (2013)