# Computing Upper and Lower Bounds of Graph Edit Distance in Cubic Time

Kaspar Riesen[1], Andreas Fischer[2], and Horst Bunke[3]

[1] Institute for Information Systems, University of Applied Sciences and Arts Northwestern Switzerland, Riggenbachstrasse 16, 4600 Olten, Switzerland
kaspar.riesen@fhnw.ch
[2] Biomedical Science and Technologies Research Centre, Polytechnique Montreal 2500 Chemin de Polytechnique, Montreal H3T 1J4, Canada
andreas.fischer@polymtl.ca
[3] Institute of Computer Science and Applied Mathematics, University of Bern, Neubrückstrasse 10, 3012 Bern, Switzerland
bunke@iam.ch

**Abstract.** Exact computation of graph edit distance (GED) can be solved in exponential time complexity only. A previously introduced approximation framework reduces the computation of GED to an instance of a linear sum assignment problem. Major benefit of this reduction is that an optimal assignment of nodes (including local structures) can be computed in polynomial time. Given this assignment an approximate value of GED can be immediately derived. Yet, since this approach considers local – rather than the global – structural properties of the graphs only, the GED derived from the optimal assignment is suboptimal. The contribution of the present paper is twofold. First, we give a formal proof that this approximation builds an upper bound of the true graph edit distance. Second, we show how the existing approximation framework can be reformulated such that a lower bound of the edit distance can be additionally derived. Both bounds are simultaneously computed in cubic time.

## 1 Introduction

Graph-based representations, which are used in the field of structural pattern recognition, have found widespread applications in the last decades [1,2]. In fact, graphs offer two major advantages over feature vectors. First, in contrast with vectors graphs provide a direct possibility to describe structural relations in the patterns under consideration. Second, while the size of a graph can be adapted to the size and complexity of a given pattern, vectors are constrained to a predefined length, which has to be preserved for all patterns encountered in a particular application.

Graph matching refers to the task of evaluating the similarity of graphs. A huge amount of graph matching methodologies have been developed in the last four decades [1]. They include methods stemming from *spectral graph theory* [3], *relaxation labeling* [4], or *graph kernel theory* [5], to name just a few.

N. El Gayar et al. (Eds.): ANNPR 2014, LNAI 8774, pp. 129–140, 2014.

Among the vast number of graph matching methods available, the concept of *graph edit distance* [6] is in particular interesting because it is able to cope with directed and undirected, as well as with labeled and unlabeled graphs. If there are labels on nodes, edges, or both, no constraints on the respective label alphabets have to be considered. Moreover, through the use of a cost function graph edit distance can be adapted and tailored to various problem specifications.

A major drawback of graph edit distance is its computational complexity. In fact, the problem of graph edit distance can be reformulated as an instance of a *Quadratic Assignment Problem* (*QAP*). QAPs belong to the most difficult combinatorial optimization problems for which only exponential run time algorithms are available to date.[1]

In recent years, a number of methods addressing the high complexity of graph edit distance computation have been proposed. In [7], for instance, an efficient algorithm for edit distance computation of planar graphs has been proposed. Another approach described in [8] formulates the graph edit distance problem as a binary linear programming problem. This reformulation is applicable to graphs with unlabeled and undirected edges only, and determines lower and upper bounds of graph edit distance in $O(n^7)$ and $O(n^3)$ time, respectively ($n$ refers to the number of nodes in the graphs). The authors of [9] propose the use of continuous-time quantum walks for graph edit distance computation without explicitly determining the underlying node correspondences.

Most of the approximation methods for graph edit distance restrict their applicability to special classes of graphs. In [10] the authors of the present paper introduced an algorithmic framework for the approximation of graph edit distance which is applicable to any kind of graphs. The basic idea of this approach is to reduce the difficult QAP of graph edit distance computation to a *linear sum assignment problem* (LSAP) which can be efficiently solved. This approximation framework builds the basis for the present work. In [10] the result of an initial node assignment is used to derive a valid, yet suboptimal, edit path between the graphs. In the present paper we give a formal prove that this approximation builds an upper bound of the true edit distance. Moreover, we show how the same approximation framework can be exploited to instantly derive a lower bound of the graph edit distance. Both bounds can be simultaneously computed in $O((n+m)^3)$ time, where $n$ and $m$ refers to the number of nodes in the graphs under consideration.

## 2 Exact Graph Edit Distance Computation

### 2.1 Graph Edit Distance

Let $L_V$ and $L_E$ be finite or infinite label sets for nodes and edges, respectively. A *graph* $g$ is a four-tuple $g = (V, E, \mu, \nu)$, where $V$ is the finite set of nodes, $E \subseteq V \times V$ is the set of edges, $\mu : V \to L_V$ is the node labeling function, and $\nu : E \to L_E$ is the edge labeling function.

[1] Note that QAPs are known to be $\mathcal{NP}$*-complete*, and therefore, an exact and efficient algorithm for the graph edit distance problem can not be developed unless $\mathcal{P} = \mathcal{NP}$.

Given two graphs, the source graph $g_1 = (V_1, E_1, \mu_1, \nu_1)$ and the target graph $g_2 = (V_2, E_2, \mu_2, \nu_2)$, the basic idea of graph edit distance [6] is to transform $g_1$ into $g_2$ using some edit operations. A standard set of edit operations is given by *insertions*, *deletions*, and *substitutions* of both nodes and edges. We denote the substitution of two nodes $u \in V_1$ and $v \in V_2$ by $(u \to v)$, the deletion of node $u \in V_1$ by $(u \to \varepsilon)$, and the insertion of node $v \in V_2$ by $(\varepsilon \to v)$, where $\varepsilon$ refers to the empty "node". For edge edit operations we use a similar notation.

**Definition 1.** *A sequence $(e_1, \ldots, e_k)$ of $k$ edit operations $e_i$ that transform $g_1$ completely into $g_2$ is called a (complete) edit path $\lambda(g_1, g_2)$ between $g_1$ and $g_2$. A partial edit path, i.e. a subsequence of $(e_1, \ldots, e_k)$, edits proper subsets of nodes and/or edges of the underlying graphs.*

Note that in an edit path $\lambda(g_1, g_2)$ each node of $g_1$ is either deleted or uniquely substituted with a node in $g_2$, and analogously, each node in $g_2$ is either inserted or matched with a unique node in $g_1$. The same applies for the edges. Yet, edit operations on edges are always defined by the edit operations on their adjacent nodes. That is, whether an edge $(u, v)$ is substituted, deleted, or inserted, depends on the edit operations actually performed on both adjacent nodes $u$ and $v$.

Since edge edit operations are uniquely defined via node edit operations, it is sufficient that edit operations $e_i \in \lambda(g_1, g_2)$ only cover the nodes from $V_1$ and $V_2$. That is, an edit path $\lambda(g_1, g_2)$ explicitly describes the correspondences found between the graphs' nodes $V_1$ and $V_2$, while the edge edit operations are implicitly given by these node correspondences.

Let $\Upsilon(g_1, g_2)$ denote the set of all admissible and complete edit paths between two graphs $g_1$ and $g_2$. To find the most suitable edit path out of $\Upsilon(g_1, g_2)$, one introduces a cost $c(e)$ for every edit operation $e$, measuring the strength of the corresponding operation. The idea of such a cost is to define whether or not an edit operation $e$ represents a strong modification of the graph. By means of cost functions for elementary edit operations, graph edit distance allows the integration of domain specific knowledge about object similarity. Furthermore, if in a particular case prior knowledge about the labels and their meaning is not available, automatic procedures for learning the edit costs from a set of sample graphs are available as well [11].

Clearly, between two similar graphs, there should exist an inexpensive edit path, representing low cost operations, while for dissimilar graphs an edit path with high cost is needed. Consequently, the edit distance of two graphs is defined as follows.

**Definition 2.** *Let $g_1 = (V_1, E_1, \mu_1, \nu_1)$ be the source and $g_2 = (V_2, E_2, \mu_2, \nu_2)$ the target graph. The graph edit distance $d_{\lambda_{\min}}(g_1, g_2)$, or $d_{\lambda_{\min}}$ for short, between $g_1$ and $g_2$ is defined by*

$$d_{\lambda_{\min}}(g_1, g_2) = \min_{\lambda \in \Upsilon(g_1, g_2)} \sum_{e_i \in \lambda} c(e_i) \quad , \tag{1}$$

*where $\Upsilon(g_1, g_2)$ denotes the set of all complete edit paths transforming $g_1$ into $g_2$, $c$ denotes the cost function measuring the strength $c(e_i)$ of edit operation $e_i$*

*(including the cost of the implied edge edit operations), and* $\lambda_{\min}$ *refers to the minimal cost edit path found in* $\Upsilon(g_1, g_2)$.

For our further investigations it will be necessary to subdivide any graph distance value $d_\lambda(g_1, g_2)$ corresponding to a (not necessarily minimal) edit path $\lambda \in \Upsilon(g_1, g_2)$ into the sum of costs $C_\lambda^{\langle V \rangle}$ for all node edit operations $e_i \in \lambda$ and the sum of costs $C_\lambda^{\langle E \rangle}$ for all edge edit operations implied by the node operations $e_j \in \lambda$. That is,

$$d_\lambda(g_1, g_2) = C_\lambda^{\langle V \rangle} + C_\lambda^{\langle E \rangle} \tag{2}$$

### 2.2 Exact Computation of Graph Edit Distance

Optimal algorithms for computing the edit distance $d_{\lambda_{\min}}(g_1, g_2)$ are typically based on combinatorial search procedures that explore the space of all possible mappings of the nodes and edges of $g_1$ to the nodes and edges of $g_2$ (i.e. the search space corresponds to $\Upsilon(g_1, g_2)$). Such an exploration is often conducted by means of A* based search techniques [12].

The basic idea of A* based search methods is to organize the underlying search space as an ordered tree. The root node of the search tree represents the starting point of our search procedure, inner nodes of the search tree correspond to partial edit paths, and leaf nodes represent complete – not necessarily optimal – edit paths. Such a search tree is dynamically constructed at runtime by iteratively creating successor nodes linked by edges to the currently considered node in the search tree.

The search tree nodes, i.e. (partial or complete) edit paths $\lambda$, to be processed in the next steps are typically contained in a set *OPEN*. In order to determine the most promising (partial) edit path $\lambda \in$ *OPEN*, i.e. the edit path to be used for further expansion in the next iteration, an assessment function $f(\lambda) = g(\lambda) + h(\lambda)$ is usually used, which includes the accumulated cost $g(\lambda)$ of the edit operations $e_i \in \lambda$ plus a heuristic estimation $h(\lambda)$ of the future cost to complete $\lambda$. One can show that, given that the estimation of the future cost is lower than, or equal to, the real cost, the algorithm is admissible. Hence, this procedure guarantees that a complete edit path $\lambda_{\min}$ found by the algorithm first is always optimal in the sense of providing minimal cost among all possible competing paths.

Note that the edge operations implied by the node edit operations can be derived from every partial or complete edit path $\lambda$ during the search procedure. The cost of these implied edge operations are dynamically added to the corresponding paths $\lambda \in$ *OPEN* and are thus considered in the edit path assessment $f(\lambda)$.

## 3 Bipartite Graph Matching

The computational complexity of exact graph edit distance is exponential in the number of nodes of the involved graphs. That is considering $m$ nodes in $g_1$ and $n$ nodes in $g_2$, $\Upsilon(g_1, g_2)$ contains $O(m^n)$ edit paths to be explored. This means that for large graphs the computation of edit distance is intractable. The graph

edit distance approximation framework introduced in [10] reduces the difficult *Quadratic Assignment Problem* (*QAP*) of graph edit distance computation to an instance of a *Linear Sum Assignment Problem* (*LSAP*). For solving LSAPs a large number of algorithms exist [13]. The time complexity of the best performing exact algorithms for LSAPs is cubic in the size of the problem. The LSAP is defined as follows.

**Definition 3.** *Given two disjoint sets* $S = \{s_1, \ldots, s_n\}$ *and* $Q = \{q_1, \ldots, q_n\}$ *and an* $n \times n$ *cost matrix* $\mathbf{C} = (c_{ij})$*, where* $c_{ij}$ *measures the suitability of assigning the* $i$*-th element of the first set to the* $j$*-th element of the second set. The Linear Sum Assignment Problem (LSAP) is given by finding the minimum cost permutation*

$$(\varphi_1, \ldots, \varphi_n) = \underset{(\varphi_1,\ldots,\varphi_n)\in\mathcal{S}_n}{\arg\min} \sum_{i=1}^{n} c_{i\varphi_i} \quad ,$$

*where* $\mathcal{S}_n$ *refers to the set of all* $n!$ *possible permutations of* $n$ *integers, and permutation* $(\varphi_1, \ldots, \varphi_n)$ *refers to the assignment where the first entity* $s_1 \in S$ *is mapped to entity* $q_{\varphi_1} \in Q$*, the second entity* $s_2 \in S$ *is assigned to entity* $q_{\varphi_2} \in Q$*, and so on.*

By reformulating the graph edit distance problem to an instance of an LSAP, three major issues have to be resolved. First, LSAPs are generally stated on independent sets with equal cardinality. Yet, in our case the elements to be assigned to each other are given by the sets of nodes (and edges) with unequal cardinality in general. Second, solutions to LSAPs refer to assignments of elements in which every element of the first set is assigned to exactly one element of the second set and vice versa (i.e. a solution to an LSAP corresponds to a bijective assignment of the the underlying entities). Yet, graph edit distance is a more general assignment problem as it explicitly allows both deletions and insertions to occur on the basic entities (rather than only substitutions). Third, graphs do not only consist of independent sets of entities (i.e. nodes) but also of structural relationships between these entities (i.e. edges that connect pairs of nodes). LSAPs are not able to consider these relationships in a global and consistent way. The first two issues are perfectly – and the third issue partially – resolvable by means of the following definition of a square cost matrix whereon the LSAP is eventually solved.

**Definition 4.** *Based on the node sets* $V_1 = \{u_1, \ldots, u_n\}$ *and* $V_2 = \{v_1, \ldots, v_m\}$ *of* $g_1$ *and* $g_2$*, respectively, a cost matrix* $\mathbf{C}$ *is established as follows.*

$$\mathbf{C} = \left[\begin{array}{cccc|cccc} c_{11} & c_{12} & \cdots & c_{1m} & c_{1\varepsilon} & \infty & \cdots & \infty \\ c_{21} & c_{22} & \cdots & c_{2m} & \infty & c_{2\varepsilon} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \infty \\ c_{n1} & c_{n2} & \cdots & c_{nm} & \infty & \cdots & \infty & c_{n\varepsilon} \\ \hline c_{\varepsilon 1} & \infty & \cdots & \infty & 0 & 0 & \cdots & 0 \\ \infty & c_{\varepsilon 2} & \ddots & \vdots & 0 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \infty & \vdots & \ddots & \ddots & 0 \\ \infty & \cdots & \infty & c_{\varepsilon m} & 0 & \cdots & 0 & 0 \end{array}\right] \tag{3}$$

*Entry $c_{ij}$ thereby denotes the cost of a node substitution $(u_i \rightarrow v_j)$, $c_{i\varepsilon}$ denotes the cost of a node deletion $(u_i \rightarrow \varepsilon)$, and $c_{\varepsilon j}$ denotes the cost of a node insertion $(\varepsilon \rightarrow v_j)$.*

Note that matrix $\mathbf{C} = (c_{ij})$ is by definition quadratic. Hence, the first issue (sets of unequal size) is instantly eliminated. Obviously, the left upper corner of the cost matrix $\mathbf{C} = (c_{ij})$ represents the costs of all possible node substitutions, the diagonal of the right upper corner the costs of all possible node deletions, and the diagonal of the bottom left corner the costs of all possible node insertions. Note that every node can be deleted or inserted at most once. Therefore any non-diagonal element of the right-upper and left-lower part is set to $\infty$. The bottom right corner of the cost matrix is set to zero since substitutions of the form $(\varepsilon \rightarrow \varepsilon)$ should not cause any cost.

Given the cost matrix $\mathbf{C} = (c_{ij})$, the LSAP optimization consists in finding a permutation $(\varphi_1, \ldots, \varphi_{n+m})$ of the integers $(1, 2, \ldots, (n+m))$ that minimizes the overall assignment cost $\sum_{i=1}^{(n+m)} c_{i\varphi_i}$. This permutation corresponds to the assignment

$$\psi = ((u_1 \rightarrow v_{\varphi_1}), (u_2 \rightarrow v_{\varphi_2}), \ldots, (u_{m+n} \rightarrow v_{\varphi_{m+n}}))$$

of the nodes of $g_1$ to the nodes of $g_2$. Note that assignment $\psi$ includes node assignments of the form $(u_i \rightarrow v_j)$, $(u_i \rightarrow \varepsilon)$, $(\varepsilon \rightarrow v_j)$, and $(\varepsilon \rightarrow \varepsilon)$ (the latter can be dismissed, of course). Hence, the definition of the cost matrix in Eq. 3 also resolves the second issue stated above and allows insertions and/or deletions to occur in an optimal assignment.

The third issue is about the edge structure of both graphs which cannot be entirely considered by LSAPs. In fact, so far the the cost matrix $\mathbf{C} = (c_{ij})$ considers the nodes of both graphs only, and thus mapping $\psi$ does not take any structural constraints into account. In order to integrate knowledge about the graph structure, to each entry $c_{ij}$, i.e. to each cost of a node edit operation $(u_i \rightarrow v_j)$, the minimum sum of edge edit operation costs, implied by the corresponding node operation, is added. That is, we encode the minimum matching cost arising from the local edge structure in the individual entries $c_{ij} \in \mathbf{C}$.

Formally, assume that node $u_i$ has adjacent edges $E_{u_i}$ and node $v_j$ has adjacent edges $E_{v_j}$. With these two sets of edges, $E_{u_i}$ and $E_{v_j}$, an individual cost matrix similarly to Eq. 3 can be established and an optimal assignment of the elements $E_{u_i}$ to the elements $E_{v_j}$ using an LSAP solving algorithm can be computed. Following this procedure, the assignment of adjacent edges is not constrained by an assignment of adjacent nodes other than $u_i$ and $v_j$. Therefore, the estimated edge edit costs implied by $(u_i \rightarrow v_j)$ are less than, or equal to, the costs implied by a complete edit path. These minimum edge edit costs are eventually added to the entry $c_{ij}$. To entry $c_{i\varepsilon}$, which denotes the cost of a node deletion, the cost of the deletion of all adjacent edges of $u_i$ is added, and to the entry $c_{\varepsilon j}$, which denotes the cost of a node insertion, the cost of all insertions of the adjacent edges of $v_j$ is added. This particular encoding of the minimal edge edit operation cost enables the LSAP to consider information about the

local, yet not global, edge structure of a graph. Hence, this heuristic procedure partially resolves the third issue.

## 4 Upper and Lower Bounds of Graph Edit Distance

### 4.1 Upper Bound $d_\psi$

Given the node assignment $\psi$ two different distance values approximating the exact graph edit distance $d_{\lambda_{\min}}(g_1, g_2)$ can be inferred. As stated above, the LSAP optimization finds an assignment $\psi$ in which every node of $g_1$ is either assigned to a unique node of $g_2$ or deleted. Likewise, every node of $g_2$ is either assigned to a unique node of $g_1$ or inserted. Hence, mapping $\psi$ refers to an admissible and complete edit path between the graphs under consideration, i.e. $\psi \in \Upsilon(g_1, g_2)$. Therefore, the edge operations, which are implied by edit operations on their adjacent nodes, can be completely inferred from $\psi$. This gives us a first approximation value $d_\psi(g_1, g_2)$, or $d_\psi$ for short, defined by (cf. Eq. 2)

$$d_\psi(g_1, g_2) = C_\psi^{\langle V \rangle} + C_\psi^{\langle E \rangle} \quad . \tag{4}$$

Note that in case of $d_{\lambda_{\min}}$ the sum of edge edit cost $C_{\lambda_{\min}}^{\langle E \rangle}$ is dynamically built while the search tree is constructed and eventually added to every partial edit path $\lambda \in \mathit{OPEN}$. Yet, the sum of edge costs $C_\psi^{\langle E \rangle}$ is added to the cost of the complete edit path $\psi$ only after the optimization process has been terminated. This is because LSAP solving algorithms are not able to take information about assignments of adjacent nodes into account during run time. In other words, for finding the edit path $\psi \in \Upsilon(g_1, g_2)$ based on the cost matrix $\mathbf{C} = (c_{ij})$ the structural information of the graphs is considered in an isolated way only (single nodes and their adjacent edges). This observation brings us to the following Lemma.

**Lemma 1.** *The distance $d_\psi(g_1, g_2)$ derived from the node assignment $\psi$ constitutes an upper bound of the true graph edit distance $d_{\lambda_{\min}}(g_1, g_2)$. That is,*

$$d_\psi(g_1, g_2) \geq d_{\lambda_{\min}}(g_1, g_2)$$

*holds for every pair of graphs $g_1, g_2$.*

*Proof.* We distinguish two cases.

1. $\psi = \lambda_{\min}$: That is, the edit path $\psi$ returned by our approximation framework is identical with the edit path $\lambda_{\min}$ computed by an exact algorithm. It follows that $d_\psi = d_{\lambda_{\min}}$.
2. $\psi \neq \lambda_{\min}$: In this case the approximate edit distance $d_\psi$ cannot be smaller than $d_{\lambda_{\min}}$. Otherwise an exact algorithm for graph edit distance computation, which exhaustively explores $\Upsilon(g_1, g_2)$, would return $\psi$ as edit path with minimal cost, i.e. $\psi = \lambda_{\min}$. Yet, this is a contradiction to our initial assumption that $\psi \neq \lambda_{\min}$.

### 4.2 Lower Bound $d'_\psi$

The distance value $d_\psi(g_1, g_2)$ is directly used as an approximate graph edit distance between graphs $g_1$ and $g_2$ in previous publications (e.g. in [10]). We now define another approximation of the true graph edit distance based on mapping $\psi$. As we will see below, this additional approximation builds a lower bound of the true graph edit distance $d_{\lambda_{\min}}(g_1, g_2)$.

First, we consider the the minimal sum of assignment costs $\sum_{i=1}^{(n+m)} c_{i\varphi_i}$ returned by our LSAP solving algorithm. Remember that every entry $c_{i\varphi_i}$ reflects the cost of the corresponding node edit operation $(u_i \to v_{\varphi_i})$ plus the minimal cost of editing the incident edges of $u_i$ to the incident edges of $v_{\varphi_i}$. Hence, the sum $\sum_{i=1}^{(n+m)} c_{i\varphi_i}$ can be – similarly to Eq. 4 – subdivided into costs for node edit operations and costs for edge edit operations. That is,

$$\sum_{i=1}^{(n+m)} c_{i\varphi_i} = C_\psi^{\langle V \rangle} + C_\varphi^{\langle E \rangle} \quad . \tag{5}$$

Analogously to Eq. 4, $C_\psi^{\langle V \rangle}$ corresponds to the sum of costs for node edit operations $e_i \in \psi$. Yet, note the difference between $C_\varphi^{\langle E \rangle}$ and $C_\psi^{\langle E \rangle}$. While $C_\psi^{\langle E \rangle}$ reflects the costs of editing the edge structure from $g_1$ to the edge structure of $g_2$ in a globally consistent way (with respect to all edit operations in $\psi$ applied on both adjacent nodes of every edge), the sum $C_\varphi^{\langle E \rangle}$ is based on the optimal permutation $(\varphi_1, \ldots, \varphi_{(n+m)})$ and in particular on the limited, because local, information about the edge structure integrated in the cost matrix $\mathbf{C} = (c_{ij})$. Moreover, note that every edge $(u, v)$ is adjacent with two individual nodes $u, v$ and thus the sum of edge edit costs $C_\varphi^{\langle E \rangle}$ considers every edge twice in two independent edit operations. Therefore, we define our second approximation value $d'_\psi(g_1, g_2)$, or $d'_\psi$ for short, by

$$d'_\psi(g_1, g_2) = C_\psi^{\langle V \rangle} + \frac{C_\varphi^{\langle E \rangle}}{2} \tag{6}$$

Clearly, Eq. 6 can be reformulated as

$$d'_\psi(g_1, g_2) = C_\psi^{\langle V \rangle} + \frac{\sum_{i=1}^{(n+m)} c_{i\varphi_i} - C_\psi^{\langle V \rangle}}{2} \tag{7}$$

and thus, $d'_\psi(g_1, g_2)$ only depends on quantities $C_\psi^{\langle V \rangle}$ and $\sum_{i=1}^{(n+m)} c_{i\varphi_i}$, which are already computed for $d_\psi$. Therefore, $d'_\psi$ can be derived without any additional computations from the established approximation $d_\psi$.

Note that the approximation $d_\psi$ corresponds to an admissible and complete edit path with respect to the nodes and edges of the underlying graphs. Yet, the second approximation $d'_\psi$ is not related to a valid edit path since the edges of both graphs are not uniquely assigned to each other (or deleted/inserted at most once). The following Lemma shows an ordering relationship between $d_\psi$ and $d'_\psi$.

**Lemma 2.** *For the graph edit distance approximations $d_\psi(g_1, g_2)$ (Eq. 4) and $d'_\psi(g_1, g_2)$ (Eq. 6) the inequality*

$$d'_\psi(g_1, g_2) \leq d_\psi(g_1, g_2)$$

*holds for every pair of graphs $g_1, g_2$ and every complete node assignment $\psi$.*

*Proof.* According to Eq. 4 and Eq. 6 we have to show that

$$\frac{C_\varphi^{\langle E \rangle}}{2} \leq C_\psi^{\langle E \rangle} \quad .$$

Assume that the node edit operation $(u_i \to v_j)$ is performed in $\psi$. Therefore, the edges $E_{u_i}$ incident to node $u_i$ are edited to the edges $E_{v_j}$ of $v_j$ in $C_\varphi^{\langle E \rangle}$ as well as in $C_\psi^{\langle E \rangle}$.

The sum of edge costs $C_\varphi^{\langle E \rangle}$ considers the minimal cost edit path between the edges $E_{u_i}$ to the edges of $E_{v_j}$ with respect to $(u_i \to v_j)$ only. In the case of $C_\psi^{\langle E \rangle}$, however, every edge in $E_{u_i}$ and $E_{v_j}$ is edited with respect to the node operations actually carried out on *both* adjacent nodes of every edge (rather than considering $(u_i \to v_j)$ only). Hence, $C_\varphi^{\langle E \rangle}$ is restricted to the best case, while $C_\psi^{\langle E \rangle}$ considers the consistent case of editing the edge sets.

Note that $C_\varphi^{\langle E \rangle}$ is built on the minimized sum $\sum_{i=1}^{(n+m)} c_{i\varphi_i}$. Yet, the cost sequence $c_{1\varphi_1}, \ldots, c_{(n+m)\varphi_{(n+m)}}$ considers every edge $(u_l, u_k) \in E_1$ twice, viz. once in entry $c_{l\varphi_l}$ and once in entry $c_{k\varphi_k}$. The same accounts for the edges in $E_2$. The cost of edge operations considered in $c_{l\varphi_l}$ as well as in $c_{k\varphi_k}$ refers to the best possible case of editing the respective edge sets. The sum of cost of these two best cases considered in $C_\varphi^{\langle E \rangle}$ are clearly smaller than, or equal to, twice the actual cost considered in $C_\psi^{\langle E \rangle}$.

We can now show that $d'_\psi$ constitutes a lower bound for $d_{\lambda_{\min}}$.

**Lemma 3.** *The distance $d'_\psi(g_1, g_2)$ derived from the node assignment $\psi$ constitutes a lower bound of the true graph edit distance $d_{\lambda_{\min}(g_1,g_2)}$. That is,*

$$d'_\psi(g_1, g_2) \leq d_{\lambda_{\min}}(g_1, g_2)$$

*holds for every pair of graphs $g_1, g_2$.*

*Proof.* We distinguish two cases.

1. $\psi = \lambda_{\min}$: An optimal algorithm would return $\psi$ as optimal solution and thus $d_\psi = d_{\lambda_{\min}}$. From Lemma 2 we know that $d'_\psi \leq d_\psi$ and thus $d'_\psi \leq d_{\lambda_{\min}}$.
2. $\psi \neq \lambda_{\min}$: In this case $\psi$ corresponds to a suboptimal edit path with cost $d_\psi$ greater than (or possibly equal to) $d_{\lambda_{min}}$. The question remains whether or not $d_{\lambda_{\min}} < d'_\psi$ might hold in this case. According to Lemma 2 we know that $d'_{\lambda_{\min}} \leq d_{\lambda_{\min}}$ and thus assuming that $d_{\lambda_{\min}} < d'_\psi$ holds, it follows that $d'_{\lambda_{\min}} < d'_\psi$. Yet, this is contradictory to the optimality of the LSAP solving algorithm that guarantees to find the assignment $\psi$ with lowest cost $d'_\psi$.

We can now conclude this section with the following theorem.

**Theorem 1.**

$$d'_\psi(g_1, g_2) \leq d_{\lambda_{\min}}(g_1, g_2) \leq d_\psi(g_1, g_2) \quad \forall g_1, g_2$$

*Proof.* See Lemmas 1, 2, and 3.

## 5 Experimental Evaluation

In Table 1 the achieved results on three data sets from the IAM graph database repository[2] are shown. The graph data sets involve graphs that represent molecular compounds (AIDS), fingerprint images (FP), and symbols from architectural and electronic drawings (GREC). On each data set and for both bounds two characteristic numbers are computed, viz. the mean relative deviation of $d_\psi$ and $d'_\psi$ from the exact graph edit distance $d_{\lambda_{\min}}$ ($\varnothing e$) and the mean run time to carry out one graph matching ($\varnothing t$).

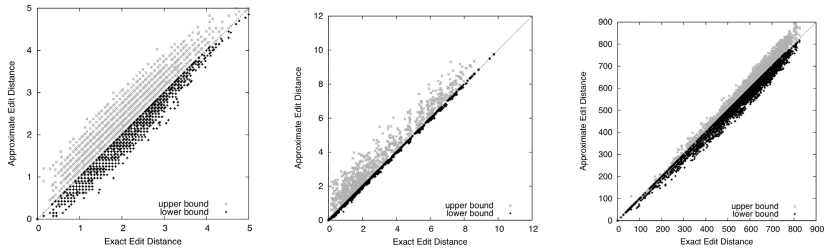
**Table 1.** The mean relative error of the exact graph edit distance ($\varnothing e$) in percentage and the mean run time for one matching ($\varnothing t$ in ms)

| Distance | Data Set | | | | | |
|---|---|---|---|---|---|---|
| | AIDS | | FP | | GREC | |
| | $\varnothing\ e$ | $\varnothing\ t$ | $\varnothing\ e$ | $\varnothing\ t$ | $\varnothing\ e$ | $\varnothing\ t$ |
| $d_{\lambda_{\min}}$ | - | 5629.53 | - | 5000.85 | - | 3103.76 |
| $d_\psi$ | +12.68 | 0.44 | +6.38 | 0.56 | +2.98 | 0.43 |
| $d'_\psi$ | -7.01 | 0.44 | -0.38 | 0.56 | -3.67 | 0.43 |

First we focus on the exact distances $d_{\lambda_{\min}}$ provided by A*. As $d_{\lambda_{\min}}$ refers to the exact edit distance, the mean relative error $\varnothing e$ is zero on all data sets. We observe that the mean run time for the computation of $d_{\lambda_{\min}}$ lies between 3.1s and 5.6s per matching. Using the approximation framework, a massive speed-up of computation time can be observed. That is, on all data sets the the computation of both distance approximations $d_\psi$ and $d'_\psi$ is possible in less than or approximately 0.5ms on average (note that both distance measures are simultaneously computed and thus offer the same matching time).

Regarding the overestimation of $d_\psi$ and the underestimation of $d'_\psi$ we observe the following. The original framework, providing the upper bound $d_\psi$, overestimates the graph distance by 12.68% on average on the AIDS data, while on the Fingerprint and GREC data the overestimations of the true distances amount to 6.38% and 2.98%, respectively. On the GREC data, the upper bound $d_\psi$ is a more accurate approximation than the lower bound $d'_\psi$, where the underestimation amounts to 3.67%. Yet, the deviations of $d_\psi$ are substantially reduced on

[2] `www.iam.unibe.ch/fki/databases/iam-graph-database`




(a) AIDS (b) FP) (c) GREC)

**Fig. 1.** Exact graph edit distance $d_{\lambda_{\min}}$ vs. upper bound $d_\psi$ (gray points) and lower bound $d'_\psi$ (black points) of the graph edit distance

the other two data sets by using the lower rather than the upper bound. That is, using $d'_\psi$ rather than $d_\psi$ the deviations can be reduced by 5.67% and 6.00% on the AIDS and FP data set, respectively.

Note the remarkable improvement of the approximation accuracy on the FP data set which can also be observed in the scatter plot in Fig. 1 (b). These scatter plots give us a visual representation of the accuracy of the suboptimal methods on all data sets. We plot for each pair of graphs their exact distance $d_{\lambda_{\min}}$ and approximate distance values $d_\psi$ and $d'_\psi$ (shown with gray and black points, respectively).

## 6 Conclusions

The main focus of the present paper is on theoretical issues. First, we give a formal prove that the existing approximation returns an upper bound of the true edit distance. Second, we show how the same approximation scheme can be used to derive a lower bound of the true edit distance. Both bounds are simultaneously computed in $O((n+m)^3)$, where $n$ and $m$ refer to the number of nodes of the graphs. In an experimental evaluation we empirically confirm our theoretical investigations and show that the lower bound leads to more accurate graph edit distance approximations on two out of three data sets.

In future work we aim at exploiting the additional lower bound in our approximation framework. For instance, a prediction of the true edit distance $d_{\lambda_{\min}}$ based on $d_\psi$ and $d'_\psi$ by means of regression analysis could be a rewarding avenue to be pursued. Moreover, we aim at using both bounds in a pattern recognition application (e.g. in database retrieval where both bounds can be beneficially employed).

**Acknowledgements.** This work has been supported by the *Hasler Foundation* Switzerland and the *Swiss National Science Foundation* project P300P2-151279.

## References

1. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in pattern recognition. Int. Journal of Pattern Recognition and Artificial Intelligence 18(3), 265–298 (2004)
2. Foggia, P., Percannella, G., Vento, M.: Graph matching and learning in pattern recognition in the last 10 years. Int. Journal of Pattern Recognition and Art. Intelligence (2014)
3. Luo, B., Wilson, R., Hancock, E.: Spectral embedding of graphs. Pattern Recognition 36(10), 2213–2223 (2003)
4. Torsello, A., Hancock, E.: Computing approximate tree edit distance using relaxation labeling. Pattern Recognition Letters 24(8), 1089–1097 (2003)
5. Gärtner, T.: A survey of kernels for structured data. SIGKDD Explorations 5(1), 49–58 (2003)
6. Bunke, H., Allermann, G.: Inexact graph matching for structural pattern recognition. Pattern Recognition Letters 1, 245–253 (1983)
7. Neuhaus, M., Bunke, H.: An error-tolerant approximate matching algorithm for attributed planar graphs and its application to fingerprint classification. In: Fred, A., Caelli, T.M., Duin, R.P.W., Campilho, A.C., de Ridder, D. (eds.) SSPR&SPR 2004. LNCS, vol. 3138, pp. 180–189. Springer, Heidelberg (2004)
8. Justice, D., Hero, A.: A binary linear programming formulation of the graph edit distance. IEEE Trans. on Pattern Analysis ans Machine Intelligence 28(8), 1200–1214 (2006)
9. Emms, D., Wilson, R.C., Hancock, E.R.: Graph edit distance without correspondence from continuous-time quantum walks. 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. 5–14. Springer, Heidelberg (2008)
10. Riesen, K., Bunke, H.: Approximate graph edit distance computation by means of bipartite graph matching. Image and Vision Computing 27(4), 950–959 (2009)
11. Caetano, T.S., McAuley, J.J., Cheng, L., Le, Q.V., Smola, A.J.: Learning graph matching. IEEE Trans. on Pattern Analysis and Machine Intelligence 31(6), 1048–1058 (2009)
12. Hart, P., Nilsson, N., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions of Systems, Science, and Cybernetics 4(2), 100–107 (1968)
13. Burkard, R., Dell'Amico, M., Martello, S.: Assignment Problems. Society for Industrial and Applied Mathematics, Philadelphia (2009)