Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematics and Computation 182 (2006) 447–459
APPLIED MATHEMATICS AND COMPUTATION
www.elsevier.com/locate/amc

A similarity measure for graphs with low computational complexity

Matthias Dehmer [a,*], Frank Emmert-Streib [b], Jürgen Kilian [a]

[a] Technische Universität Darmstadt, 64289 Darmstadt, Germany
[b] Stowers Institute for Medical Research, 1000 E. 50th Street, Kansas City, MO 64110, USA

Abstract

We present and analyze an algorithm to measure the structural similarity of generalized trees, a new graph class which includes rooted trees. For this, we represent structural properties of graphs as strings and define the similarity of two graphs as optimal alignments of the corresponding property stings. We prove that the obtained graph similarity measures are so called Backward similarity measures. From this we find that the time complexity of our algorithm is polynomial and, hence, significantly better than the time complexity of classical graph similarity methods based on isomorphic relations.

Keywords: Graph theory; Graph similarity; Dynamic programming; Computational complexity

1. Introduction

In this paper, we examine the computational complexity of a novel algorithm to measure the structural similarity of graphs. Thereby, we deal with a special graph class: directed and hierarchical graphs which we call *generalized trees*. There are classical approaches to measure the structural similarity of graphs, e.g., [4,11–13,20,25]. The typical and classical approach to measure the structural similarity of graphs is based on isomorphic relations [11–13,20,25]. For example, Zelinka [25] was the first who measured the distance between isomorphism classes of unlabeled graphs. The graph similarity measure of Zelinka is based on the principle that two graphs are more similar the bigger the common induced isomorphic subgraph is. Furthermore Sobik [20] and Kaden [12] generalized the measure of Zelinka for arbitrary and labeled graphs of different orders.[1] The main disadvantage of these graph measures is that the computational complexity is $\mathcal{NP}$-complete [20,23]. Our approach to measure the structural similarity of generalized trees is completely different from similarity or distance measures which are based on isomorphic relations mentioned above. Our novel similarity measure is

* Corresponding author.
E-mail addresses: dehmer@informatik.tu-darmstadt.de (M. Dehmer), fes@stowers-institute.org (F. Emmert-Streib), kilian@noteserver.org (J. Kilian).

[1] If we denote the given graph as $G = (V, E)$, then the order of these graphs is defined by $|V|$.

0096-3003/$ - see front matter
doi:10.1016/j.amc.2006.04.006

mainly based on a new representation of graphs as strings, whose components represent structural properties of the graph. We call these strings *property strings* of a generalized tree. In Section 3, the structural similarity of two graphs will be defined as the optimal alignment of the underlying property strings. In the present paper we examine the computational complexity of our algorithm. We want to emphasize, that generalized trees are more complex than normal directed rooted trees. Therefore this graph class can represent more complex problems than ordinary trees as demonstrated, e.g., in [26]. In [6], Emmert-Streib et al. showed that our novel method is applicable to measure the structural similarity of a more general graph type (unweighted and undirected graphs) by a local decomposition of the graph in generalized trees. Due to [6], considering generalized trees is not necessarily a restriction in terms of generality.

This paper is organized as follows: In Section 2 we give a review of classical graph and tree similarity methods to distinguish these methods better from our novel method. We repeat in Section 3 the construction of our graph similarity measures and prove for the first time that there are *Backward similarity measures*. The examination of the computational complexity of our method will be presented in Section 4. Section 5 finishes this paper with a summary.

2. Measuring the structural similarity of graphs

2.1. Preliminaries

In this section we give first some fundamental definitions of graph theory and then review some approaches to measure the structural similarity of trees and graphs. In order to distinguish trees from generalized trees we need the following definition.

Definition 2.1. Let $\mathscr{H} = (\hat{V}, E_1)$ be an directed rooted tree. The vertex set $\hat{V}$ is defined by

$$\hat{V} := \{v_{0,1}, v_{1,1}, v_{1,2}, \ldots, v_{1,\sigma_1}, v_{2,1}, v_{2,2}, \ldots, v_{2,\sigma_2}, \ldots, v_{h,1}, v_{h,2}, \ldots, v_{h,\sigma_h}\},$$

where $v_{i,j}$ denotes the jth vertex on the ith level, $0 \leqslant i \leqslant h$, $1 \leqslant j \leqslant \sigma_i$. h denotes the depth of $\mathscr{H}$ and σ_i is the number of vertices on level i. The edge set $\hat{E} := \hat{E}_1 \cup \hat{E}_2 \cup \hat{E}_3 \cup \hat{E}_4$ is defined as [16]

- $(\hat{E}_1)$ forms the edge set of the underlying directed rooted tree $\mathscr{H}$.
- $(\hat{E}_2)$ *Up-edges* associate analogously vertices of the tree hierarchy with one of their (dominating) predecessor vertices.
- $(\hat{E}_3)$ *Down-edges* associate vertices of the tree hierarchy with one of their (dominated) successor vertices in terms of that tree hierarchy.
- $(\hat{E}_4)$ *Cross-edges* associate vertices of the tree hierarchy, none of which is an (immediate) predecessor of the other in terms of the tree hierarchy.

If $\hat{E}_2$, $\hat{E}_3$ or $\hat{E}_4 \neq \emptyset$ then $\hat{\mathscr{H}} = (\hat{V}, \hat{E})$ is called a generalized tree (Fig. 1).

Definition 2.2. Let $\mathscr{H} := (V, E), E \subseteq V \times V$ be an unlabeled, directed graph. $\mathscr{G} := (\hat{V}, \hat{E})$ with $\hat{V} \subseteq V$ and $\hat{E} \subseteq E$ denotes the partial graph of $\mathscr{H}$, $\mathscr{G} \subseteq \mathscr{H}$.

Definition 2.3. Let $\mathscr{H} := (V, E)$ be a directed graph and $\mathscr{G} := (\hat{V}, \hat{E})$ be a partial graph of $\mathscr{H}$. Moreover, it holds $\hat{E} = E \cap (\hat{V} \times \hat{V})$. Then we call $\mathscr{G}$ the induced subgraph of $\mathscr{H}$.

In the following, we give the definition of graph isomorphism that provides the structural equivalence of two graphs [21].

Definition 2.4. Let $\mathscr{H} := (V, E)$ and $\mathscr{G} := (\hat{V}, \hat{E})$ be directed graphs. We call $\mathscr{H}$ and $\mathscr{G}$ isomorphic ($\mathscr{H} \cong \mathscr{G}$) if it exists a bijective mapping $\phi : V \to \hat{V}$ such that

$$E \ni (v_i, v_i) : \iff (\phi(v_i), \phi(v_j)) \in \hat{E}.$$

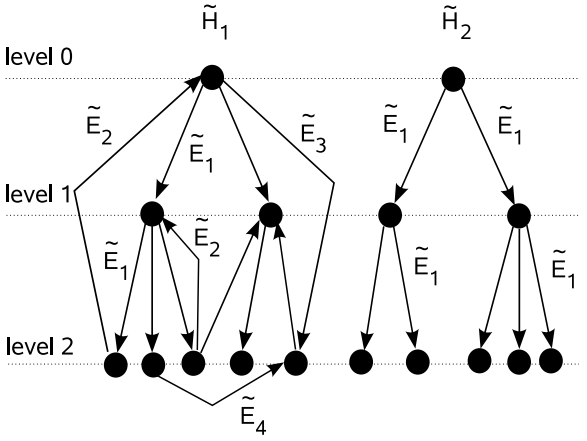

Fig. 1. $\hat{\mathscr{H}}_1$ shows a generalized tree and his edge types fulfills Definition 2.1. In contrast $\hat{\mathscr{H}}_2$ represents an ordinary directed rooted tree, which consists only of edges $e \in \hat{E}_1$. An edge $e \in \hat{E}_1$ over jumps always just one level, $e \in \hat{E}_3$ over jumps at least one level, $e \in \hat{E}_4$ does not necessarily over jump a level.

The mapping ϕ is called the isomorphism of $\mathscr{H}$ on $\mathscr{G}$. Explaining informally two graphs are isomorphic iff, we obtain the first graph by renaming vertex labels of the second graph.

For completing this Section we state the definition of directed rooted trees.

Definition 2.5. We call $T = (V_T, E_T)$ a directed rooted tree iff E_T consists only of edges from E_1 (see Definition 2.1).

2.2. Classical tree similarity measures

There are known approaches [17,18] to measure the *edit distance* between strings $s_1, s_2 \in A^\star$ on the basis of string alignments, where $A^\star$ is an arbitrary alphabet. Since one can consider special types of graphs as a generalization of the string concept, Tai [22] and Selkow [19] extended the string alignment problem to undirected and *ordered* trees. Thereby, we call a tree ordered tree, if the sequence of child vertices $v_1, v_2, \ldots, v_{\delta(v)}$ of $v \in V_T$ is significant, where $\delta(v)$ denotes the degree of v. Höchstmann et al. [9] provide a method for measuring the structural similarity of RNA *secondary structures* representing a *forest* [9] of ordered trees. In [9] a forest is defined by a finite sequence of ordered trees. The approach of Höchstmann et al. is based on the tree alignment technique of Jiang et al. [10]. Further developments and novel applications of tree similarity measures can be found in [7,15,24].

2.3. Computational complexity of tree similarity measures

Assuming the vertex labeled, ordered trees $T_1 = (V_1, E_1)$, $T_2 = (V_2, E_2)$, the basic *tree edit problem* [19,22] can be solved in time $\mathcal{O}(|V_1| \cdot |V_2| \cdot D_1^2 \cdot D_2^2)$, where D_1, D_2 are the associated maximum depths of T_1, T_2. An alternative to tree edit has been stated by Jiang et al. [10]. In [10], Jiang et al. introduced an algorithm to measure the structural similarity of ordered and undirected trees with time complexity $\mathcal{O}(|V_1| \cdot |V_2| \cdot \deg(T_1) \cdot \deg(T_2))$, where $\deg(T_k) := \max(\{\delta(v_i^k) | v_i^k \in V_k, k = 1, 2\})$. The algorithm of Höchstmann et al. [9] has the time complexity

$$\mathcal{O}(|F_1| \cdot |F_2| \cdot \deg(F_1) \cdot \deg(F_2) \cdot (\deg(F_1) + \deg(F_2))).$$

2.4. Classical graph similarity measures

In this section we do not give a complete review of graph similarity measuring, but we repeat only results relevant for this paper. First, we give a well-known key result due to Zelinka [25].

Theorem 2.1. *Let $\mathscr{H}$, $\widetilde{\mathscr{H}}$ be unlabeled graphs without reflexive*[2] *and multiple edges. Furthermore let $|V| = |\widetilde{V}| = n$. $\overline{SUB}_m(\mathscr{H})$ denotes the set of induced subgraphs of order m. $\mathscr{H}^\star$ denotes the isomorphism classes of such graphs in which lie $\mathscr{H}$ and let*

$$SUB_m(\mathscr{H}) := \{\mathscr{H}^\star | \mathscr{H} \in \overline{SUB}_m(\mathscr{H})\}. \tag{1}$$

$SUB_m(\mathscr{H})$ is just the set of isomorphism classes in which lie the induced subgraphs of $\mathscr{H}$ with order m. Then,

$$d_Z(\mathscr{H}, \widetilde{\mathscr{H}}) := n - \mathrm{SIM}(\mathscr{H}, \widetilde{\mathscr{H}}), \tag{2}$$

is a graph metric, where

$$\mathrm{SIM}(\mathscr{H}, \widetilde{\mathscr{H}}) := \max\{m | SUB_m(\mathscr{H}) \cap SUB_m(\widetilde{\mathscr{H}}) \neq \emptyset\}. \tag{3}$$

Sobik [20] and Kaden [12] generalized this measure for arbitrary (also labeled) graphs of different orders.

Theorem 2.2. *Let $\mathscr{H} := (V, E, f_V, f_E, A_V, A_E)$ be a finite, labeled and directed graph. A_V, A_E denote finite, non-empty vertex and edge alphabets and $f_V : V \to A_V, f_E : E \to A_E$ the associated vertex and edge labeling functions. Now, let $\mathscr{H}$ and $\widetilde{\mathscr{H}}$ be finite, labeled graphs of arbitrary orders. Then,*

$$d_S(\mathscr{H}, \widetilde{\mathscr{H}}) := \max\{|\mathscr{H}|, |\widetilde{\mathscr{H}}|\} - \mathrm{SIM}(\mathscr{H}, \widetilde{\mathscr{H}})\} \tag{4}$$

is a graph metric.

Fig. 2 shows the situation for determining subgraph isomorphism. Fig. 3 shows the application of Theorem 2.2. A direct application of the similarity measure of Zelinka provides Kaden [12]. Kaden transforms graphs in *line graphs* [1] and applies the theorem of Zelinka to the transformed graphs. For example the line graph of an undirected graph $G = (V, E)$ is defined by $\overline{G} = (E, \overline{E})$, $\overline{E} := \{e, \hat{e} | e, \hat{e} \in E$ and $e, \hat{e}$ are incident in $G\}$. By iterating the application of the line graph definition, Kaden obtains the graphs $\overline{G}^m$. As a result, it holds

Theorem 2.3. *Let Λ_n be the set of connected graphs of order n. For given $0 \leqslant m < n$ it holds:*

$$d_K^m(\mathscr{H}, \widetilde{\mathscr{H}}) = d_S(\mathscr{H}^m, \widetilde{\mathscr{H}}^m) = \max\{|\mathscr{H}^m|, |\widetilde{\mathscr{H}}^m|\} - \mathrm{SIM}(\mathscr{H}^m, \widetilde{\mathscr{H}}^m)\}, \tag{5}$$

is a graph metric on Λ_n.

Another well-known approach to measure the structural similarity of graphs is due to Bunke et al. [3].

Definition 2.6. We denote as an Optimal Inexact Match a sequence of transformations (insertions, deletions and substitutions of vertices) which transform $\mathscr{H}_1$ to $\mathscr{H}_2$ by producing minimal transformation costs. Assuming $m_1, m_2, \ldots, m_n$ are all possible Inexact Matches between $\mathscr{H}_1$ and $\mathscr{H}_2$. Then, the Optimal Inexact Match m' is defined by $c(m') = \min\{c(m_i) | 1 \leqslant i \leqslant n\}$, where $c(m_i)$ denotes the costs of c_i.

Theorem 2.4. *Let $d(\mathscr{H}_1, \mathscr{H}_2)$ be the costs for determining the Optimal Inexact Match between $\mathscr{H}_1$ and $\mathscr{H}_2$. $d(\mathscr{H}_1, \mathscr{H}_2)$ is a graph metric.*

A thorough overview of graph similarity measures can be found in [4,5].

2.5. Computational complexity of classical graph similarity measures

From the algorithm of Ullman [23] follows that the subgraph isomorphism problem is $\mathscr{NP}$-complete for general graphs. Then, we receive that the computation of Zelinka's measure and the generalizations of Sobik and Kaden are $\mathscr{NP}$-complete. The theorem of Bunke et al. leads us to the same result.

[2] That means edges of the type $(v, v) \in E$.

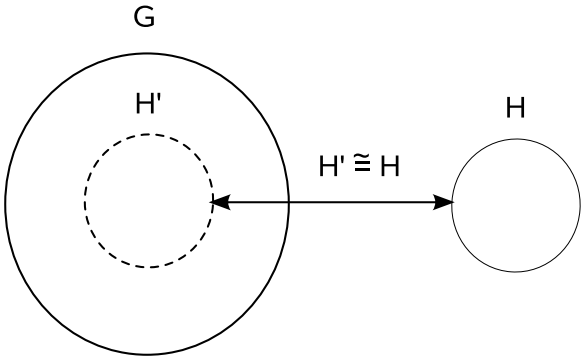

Fig. 2. The graphs G, H, $|G| > |H|$ and H' are illustrated as sets.

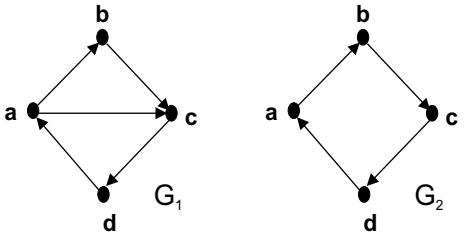

Fig. 3. Two vertex labeled directed graphs. The application of Theorem 2.2 yields $d_S(G_1, G_2) = 4 - 3 = 1$.

3. Novel approach to measure the similarity of graphs

In this section we repeat our novel approach to measure the structural similarity of generalized trees [6]. Our novel approach is based on the following idea: Transform the underlying graphs into property strings and then align these strings with a dynamic programming [2] algorithm. Hence, we reduce a graph similarity problem into a string similarity problem. From the used string alignment technique we obtain similarity measures which can be computed polynomially in time.

For explaining the construction of our novel method we note that the graphs under consideration – generalized trees – can be transformed into property strings in a level oriented way. The property strings are the out-degree and in-degree sequences on each level of the generalized trees. In order to solve the graph similarity problem for our graph class we consider the transformations

$$
\begin{aligned}
&S_0^{\hat{\mathscr{H}}_1} := r_1^{\hat{\mathscr{H}}_1},\\
&S_1^{\hat{\mathscr{H}}_1} := v_{1,1}^{\hat{\mathscr{H}}_1} \circ v_{1,2}^{\hat{\mathscr{H}}_1} \circ \cdots \circ v_{i,\delta_{\text{out}}(r_1^{\hat{\mathscr{H}}_1})}^{\hat{\mathscr{H}}_1},\\
&\quad\vdots\\
&S_{h_1}^{\hat{\mathscr{H}}_1} := v_{h_1,1}^{\hat{\mathscr{H}}_1} \circ v_{h_1,2}^{\hat{\mathscr{H}}_1} \circ \cdots \circ v_{h_1,\sigma_{h_1}}^{\hat{\mathscr{H}}_1},\\
&S_0^{\hat{\mathscr{H}}_2} := r_2^{\hat{\mathscr{H}}_2},\\
&S_1^{\hat{\mathscr{H}}_2} := v_{1,1}^{\hat{\mathscr{H}}_2} \circ v_{1,2}^{\hat{\mathscr{H}}_2} \circ \cdots \circ v_{1,\delta_{\text{out}}(r_2^{\hat{\mathscr{H}}_2})}^{\hat{\mathscr{H}}_2},\\
&\quad\vdots\\
&S_{h_2}^{\hat{\mathscr{H}}_2} := v_{h_2,1}^{\hat{\mathscr{H}}_2} \circ v_{h_2,2}^{\hat{\mathscr{H}}_2} \circ \cdots \circ v_{h_2,\sigma_{h_2}}^{\hat{\mathscr{H}}_2},
\end{aligned}
$$

representing generalized trees $\hat{\mathscr{H}}_1, \hat{\mathscr{H}}_2$ together with their induced out-degree and in-degree sequences on a level i (see also Definition 2.1). It holds $r_k^{\hat{\mathscr{H}}_k} := v_{0,1}^{\hat{\mathscr{H}}_k}$, $k \in \{0,1\}$. The problem of determining the structural similarity between $\hat{\mathscr{H}}_1$ and $\hat{\mathscr{H}}_2$ is now defined as alignment of the sequences above according to a cost function α.

In other words: the more similar the out-degree and in-degree alignments on a level i, $0 \leqslant i \leqslant h$ are, the more similar is the common structure of the graphs, and vice versa.

Fig. 4 shows two generalized trees together with their property strings on the graph levels. Now, we can write the sequences $S_0^{\hat{\mathscr{H}}_1}, S_1^{\hat{\mathscr{H}}_1}, \ldots, S_k^{\hat{\mathscr{H}}_1}, k \in \{1,2\}$ in a more compact form

$$S_1 := r_1^{\hat{\mathscr{H}}_1} \circ v_{1,1}^{\hat{\mathscr{H}}_1} \circ v_{1,2}^{\hat{\mathscr{H}}_1} \circ \cdots \circ v_{h_1,\sigma_{h_1}}^{\hat{\mathscr{H}}_1}, \tag{6}$$

$$S_2 := r_2^{\hat{\mathscr{H}}_2} \circ v_{1,1}^{\hat{\mathscr{H}}_2} \circ v_{1,2}^{\hat{\mathscr{H}}_2} \circ \cdots \circ v_{h_2,\sigma_{h_2}}^{\hat{\mathscr{H}}_2}. \tag{7}$$

That means we transform the generalized trees $\hat{\mathscr{H}}_1$, $\hat{\mathscr{H}}_2$ in sequences S_1, S_2 which have not nessecarily the same length. Here, we assume $S_k[i]$ as the ith position of the sequences S_k and it holds $S_1[n] = v_{h_1,\sigma_{h_1}}^{\hat{\mathscr{H}}_1}$, $S_2[m] = v_{h_2,\sigma_{h_2}}^{\hat{\mathscr{H}}_2}$, $\mathbb{N} \ni n, m \geqslant 1$, $S_k[1] = r_k^{\hat{\mathscr{H}}_k}, k \in \{1,2\}$.

For each possible aligned pair $[a,b]$ a cost function $\alpha([a,b]) \in \mathbb{R}_+$ is assigned, where a, b are sequence entries of S_1 and S_2 or the gap symbol $-$. The algorithm for finding the optimal alignment of the sequences (6) and (7) produces a matrix $(\mathscr{M}(i,j))_{ij}, 0 \leqslant i \leqslant n, 0 \leqslant j \leqslant m$, where $\mathscr{M}(i,j)$ is equivalent to the minimal edit distance between the sequences $\widetilde{S}_1, \widetilde{S}_2$. $\widetilde{S}_1$ and $\widetilde{S}_2$ consist of the first ith, jth characters of S_1 and S_2. Finally, we are looking for $\mathscr{M}(n,m)$ because $\mathscr{M}(n,m)$ is the minimal edit distance

$$d(S_1,S_2) := \min_{S_1 \to S_2} \sum \alpha([a,b]) \tag{8}$$

due to Levenstein [14]. This leads us to the following definition.

Definition 3.1. The recursive algorithm is defined by the following equations:

$$\mathscr{M}(0,0) := 0, \tag{9}$$

$$\mathscr{M}(i,0) := \mathscr{M}(i-1,0) + \alpha(S_1[i],-), \quad 1 \leqslant i \leqslant n, \tag{10}$$

$$\mathscr{M}(0,j) := \mathscr{M}(0,j-1) + \alpha(-,S_2[j]), \quad 1 \leqslant j \leqslant m, \tag{11}$$

$$\mathscr{M}(i,j) := \min \begin{cases} \mathscr{M}(i-1,j) + \alpha(S_1[i],-), \\ \mathscr{M}(i,j-1) + \alpha(-,S_2[j]), \\ \mathscr{M}(i-1,j-1) + \alpha(S_1[i],S_2[j]), \end{cases} \quad i \in [1,n],\ j \in [1,m]. \tag{12}$$

By tracing back along the minimal values from $\mathscr{M}(n,m)$ to $\mathscr{M}(0,0)$ we find the optimal alignment between S_1 and S_2. In Section 4 we will analyze the complexity of the graph similarity problem of generalized trees. Starting from this algorithm we construct now similarity functions α^{out}, α^{in} in order to define the final graph similarity measures $(d_i(\hat{\mathscr{H}}_1, \hat{\mathscr{H}}_2))_{1 \leqslant i \leqslant k}$. We define:

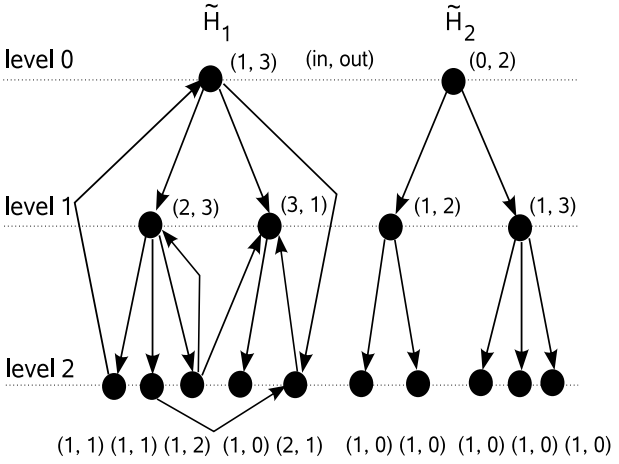

Fig. 4. Property strings in tuple-form: out-degree and in-degree sequences on the generalized tree levels.

$$\alpha^{\text{out}}\left(v_{i_1,j_1}^{\hat{\mathscr{H}}_1}, v_{i_2,j_2}^{\hat{\mathscr{H}}_2}\right) := \begin{cases} \beta^{\text{out}}\left(\delta_{\text{out}}\left(v_{i_1,j_1}^{\hat{\mathscr{H}}_1}\right), \delta_{\text{out}}\left(v_{i_2,j_2}^{\hat{\mathscr{H}}_2}\right), \sigma_{\text{out}}^1\right), & i_1 = i_2, \\ +\infty, & \text{else}, \end{cases} \tag{13}$$

$0 \leqslant i_k \leqslant h_k, 1 \leqslant j_k \leqslant \sigma_{i_k}$. Thereby, it holds $\beta^{\text{out}}(x,y,\sigma_{\text{out}}^k) := 1 - \mathrm{e}^{-\frac{1}{2}\frac{(x-y)^2}{\left(\sigma_{\text{out}}^k\right)^2}}$, $x,y,\sigma_{\text{out}}^k \in \mathbb{R}$, $k \in \{1,2\}$, and

$$\alpha^{\text{out}}\left(v_{i,j_1}^{\hat{\mathscr{H}}_1}, -\right) := \beta^{\text{out}}\left(\delta_{\text{out}}\left(v_{i,j_1}^{\hat{\mathscr{H}}_1}\right), \xi, \sigma_{\text{out}}^2\right), \tag{14}$$

$$\alpha^{\text{out}}\left(-, v_{i,j_2}^{\hat{\mathscr{H}}_2}\right) := \beta^{\text{out}}\left(\xi, \delta_{\text{out}}\left(v_{i,j_2}^{\hat{\mathscr{H}}_2}\right), \sigma_{\text{out}}^2\right). \tag{15}$$

By defining $\beta^{\text{in}}(x,y,\sigma_{\text{in}}^k) := 1 - \mathrm{e}^{-\frac{1}{2}\frac{(x-y)^2}{\left(\sigma_{\text{in}}^k\right)^2}}$ we obtain in the same way

$$\alpha^{\text{in}}\left(v_{i_1,j_1}^{\hat{\mathscr{H}}_1}, v_{i_2,j_2}^{\hat{\mathscr{H}}_2}\right) := \begin{cases} \beta^{\text{in}}\left(\delta_{\text{in}}\left(v_{i_1,j_1}^{\hat{\mathscr{H}}_1}\right), \delta_{\text{in}}\left(v_{i_2,j_2}^{\hat{\mathscr{H}}_2}\right), \sigma_{\text{in}}^1\right), & i_1 = i_2, \\ +\infty, & \text{else}, \end{cases} \tag{16}$$

$$\alpha^{\text{in}}\left(v_{i,j_1}^{\hat{\mathscr{H}}_1}, -\right) := \beta^{\text{in}}\left(\delta_{\text{in}}\left(v_{i,j_1}^{\hat{\mathscr{H}}_1}\right), \xi, \sigma_{\text{in}}^2\right), \tag{17}$$

$$\alpha^{\text{in}}\left(-, v_{i,j_2}^{\hat{\mathscr{H}}_2}\right) := \beta^{\text{in}}\left(\xi, \delta_{\text{in}}\left(v_{i,j_2}^{\hat{\mathscr{H}}_2}\right), \sigma_{\text{in}}^2\right), \tag{18}$$

$\xi > 0$. The choice $\xi > 0$ in Eqs. (14), (15), (17) and (18) prevents an alignment between two leaves being better evaluated as an alignment between a leaf and a gap. The definitions (13) and (16) of the functions α^{out}, α^{in} state that we do not align vertices on different levels. To prevent this, we set the gap penalty to $+\infty$, whereby our dynamic programming algorithm will never choose this cost-intensive path. In order to evaluate the alignments on each level of the given graphs $\hat{\mathscr{H}}_1$, $\hat{\mathscr{H}}_2$, we define the functions

$$\gamma^{\text{out}}(\hat{\mathscr{H}}_k, i) := \frac{\sum_{j=1}^{\sigma_i^k} \hat{\alpha}_{\text{out}}\left(v_{i,j}^{\hat{\mathscr{H}}_k}, \text{align}\left(v_{i,j}^{\hat{\mathscr{H}}_k}\right)\right)}{\sigma_i^k}, \tag{19}$$

$$\gamma^{\text{in}}(\hat{\mathscr{H}}_k, i) := \frac{\sum_{j=1}^{\sigma_i^k} \hat{\alpha}_{\text{in}}\left(v_{i,j}^{\hat{\mathscr{H}}_k}, \text{align}\left(v_{i,j}^{\hat{\mathscr{H}}_k}\right)\right)}{\sigma_i^k}, \tag{20}$$

$k \in \{1,2\}$. Once again, σ_i^k denotes the upper index of a vertex on level i related to $\hat{\mathscr{H}}_k$. For completing the definitions (19), (20), we define the mapping align and $\hat{\alpha}_{\text{out}}$, $\hat{\alpha}_{\text{in}}$ as follows:

$$\text{align}\left(v_{i,j_1}^{\hat{\mathscr{H}}_1}\right) := \begin{cases} v_{i,j_2}^{\hat{\mathscr{H}}_2}, & \text{align}^{-1}\left(v_{i,j_2}^{\hat{\mathscr{H}}_2}\right) = v_{i,j_1}^{\hat{\mathscr{H}}_1}, \\ -, & \text{else}, \end{cases} \tag{21}$$

$$\hat{\alpha}^{\text{out}}\left(v_{i,j_1}^{\hat{\mathscr{H}}_1}, -\right) := \beta^{\text{out}}\left(\delta_{\text{out}}\left(v_{i,j_1}^{\hat{\mathscr{H}}_1}\right), \xi, \hat{\sigma}_{\text{out}}^1\right), \tag{22}$$

$$\hat{\alpha}^{\text{out}}\left(-, v_{i,j_2}^{\hat{\mathscr{H}}_2}\right) := \beta^{\text{out}}\left(\xi, \delta_{\text{out}}\left(v_{i,j_2}^{\hat{\mathscr{H}}_2}\right), \hat{\sigma}_{\text{out}}^1\right), \tag{23}$$

$$\hat{\alpha}^{\text{out}}\left(v_{i,j_1}^{\hat{\mathscr{H}}_1}, v_{i,j_2}^{\hat{\mathscr{H}}_2}\right) := \beta^{\text{out}}\left(\delta_{\text{out}}\left(v_{i,j_1}^{\hat{\mathscr{H}}_1}\right), \delta_{\text{out}}\left(v_{i,j_2}^{\hat{\mathscr{H}}_2}\right), \hat{\sigma}_{\text{out}}^2\right), \tag{24}$$

$$\hat{\alpha}^{\text{in}}\left(v_{i,j_1}^{\hat{\mathscr{H}}_1}, -\right) := \beta^{\text{in}}\left(\delta_{\text{in}}(v_{i,j_1}^{\hat{\mathscr{H}}_1}), \xi, \hat{\sigma}_{\text{in}}^1\right), \tag{25}$$

$$\hat{\alpha}^{\text{in}}\left(-, v_{i,j_2}^{\hat{\mathscr{H}}_2}\right) := \beta^{\text{in}}\left(\xi, \delta_{\text{in}}\left(v_{i,j_2}^{\hat{\mathscr{H}}_2}\right), \hat{\sigma}_{\text{in}}^1\right), \tag{26}$$

$$\hat{\alpha}^{\text{in}}\left(v_{i,j_1}^{\hat{\mathscr{H}}_1}, v_{i,j_2}^{\hat{\mathscr{H}}_2}\right) := \beta^{\text{in}}\left(\delta_{\text{in}}\left(v_{i,j_1}^{\hat{\mathscr{H}}_1}\right), \delta_{\text{out}}\left(v_{i,j_2}^{\hat{\mathscr{H}}_2}\right), \hat{\sigma}_{\text{in}}^2\right). \tag{27}$$

In order to define local similarity functions which weighs out-degree and in-degree sequences on a generalized tree level, we set

$$\gamma^{\text{out}}(i) := 1 - \frac{1}{\sigma_i^1 + \sigma_i^2} \cdot \left\{ \sum_{j=1}^{\sigma_i^1} \hat{\alpha}^{\text{out}}\left(v_{i,j}^{\hat{\mathscr{H}}_1}, \text{align}\left(v_{i,j}^{\hat{\mathscr{H}}_1}\right)\right) \right\} + \frac{1}{\sigma_i^1 + \sigma_i^2} \cdot \left\{ \sum_{j=1}^{\sigma_i^2} \hat{\alpha}^{\text{out}}\left(v_{i,j}^{\hat{\mathscr{H}}_2}, \text{align}\left(v_{i,j}^{\hat{\mathscr{H}}_2}\right)\right) \right\}, \tag{28}$$

and

$$\gamma^{\text{in}}(i) := 1 - \frac{1}{\sigma_i^1 + \sigma_i^2} \cdot \left\{ \sum_{j=1}^{\sigma_i^1} \hat{\alpha}^{\text{in}}\left(v_{i,j}^{\hat{\mathscr{H}}_1}, \text{align}\left(v_{i,j}^{\hat{\mathscr{H}}_1}\right)\right) \right\} + \frac{1}{\sigma_i^1 + \sigma_i^2} \cdot \left\{ \sum_{j=1}^{\sigma_i^2} \hat{\alpha}^{\text{in}}\left(v_{i,j}^{\hat{\mathscr{H}}_2}, \text{align}\left(v_{i,j}^{\hat{\mathscr{H}}_2}\right)\right) \right\}. \tag{29}$$

Hence, we obtain measures which indicate how similar the out-degree and in-degree alignments of two sequences on a level i are. Now, we get

Corollary 3.1

$\gamma^{\text{out}}(i), \gamma^{\text{in}}(i) \in [0,1]$.

In order to classify our final graph similarity measures we first express

Definition 3.2. Let U be a set of units and a mapping $\phi : U \times U \to [0,1]$. ϕ is called a Backward similarity measure if

$$\phi(u,v) = \phi(v,u) \quad \forall u, v \in U \quad \text{(Symmetry)} \tag{30}$$

and

$$\phi(u,u) \geqslant \phi(u,v) \quad \forall u, v \in U \quad \text{(Backward)}. \tag{31}$$

Finally, we prove the main result for measuring the structural similarity of generalized trees.

Theorem 3.2. *Let $\hat{\mathscr{H}}_1$, $\hat{\mathscr{H}}_2$ be generalized trees, $0 \leqslant i \leqslant \rho$, $\rho := \max(h_1, h_2)$.*

$$d_1(\hat{\mathscr{H}}_1, \hat{\mathscr{H}}_2) := \frac{\sum_{i=0}^{\rho} \lambda_i \cdot \gamma^{\text{fin}}(i)}{\sum_{i=0}^{\rho} \lambda_i}, \tag{32}$$

$$d_2(\hat{\mathscr{H}}_1, \hat{\mathscr{H}}_2) := \frac{\sum_{i=0}^{\rho} \gamma^{\text{fin}}(i)}{\rho + 1}, \tag{33}$$

$$d_3(\hat{\mathscr{H}}_1, \hat{\mathscr{H}}_2) := \frac{\prod_{i=0}^{\rho} \gamma^{\text{fin}}(i)}{d_2(\hat{\mathscr{H}}_1, \hat{\mathscr{H}}_2)}, \tag{34}$$

are a family $(d_i(\hat{\mathscr{H}}_1, \hat{\mathscr{H}}_2))_{1 \leqslant i \leqslant 3}$ of Backward similarity measures, where $\gamma^{\text{fin}}(i)$ is defined as

$$\gamma^{\text{fin}}(i) := \zeta \cdot \gamma^{\text{out}}(i) + (1 - \zeta) \cdot \gamma^{\text{in}}(i). \tag{35}$$

Proposition 3.3. *It holds $(d_i(\hat{\mathscr{H}}_1, \hat{\mathscr{H}}_2))_{1 \leqslant i \leqslant 3} \in [0,1]$.*

Proof. Starting from Eqs. (32)–(34), Proposition 3.3 follows immediately from Corollary 3.1. □

Proof of Theorem 3.2. To prove we first consider the function

$$\gamma^{\text{out}}(i) := 1 - \frac{1}{\sigma_i^1 + \sigma_i^2} \cdot \left\{ \sum_{j=1}^{\sigma_i^1} \hat{\alpha}^{\text{out}}\left(v_{i,j}^{\hat{\mathscr{H}}_1}, \text{align}\left(v_{i,j}^{\hat{\mathscr{H}}_1}\right)\right) \right\} + \frac{1}{\sigma_i^1 + \sigma_i^2} \cdot \left\{ \sum_{j=1}^{\sigma_i^2} \hat{\alpha}^{\text{out}}\left(v_{i,j}^{\hat{\mathscr{H}}_2}, \text{align}\left(v_{i,j}^{\hat{\mathscr{H}}_2}\right)\right) \right\}.$$

With Eq. (21), it follows that we have to distinguish the three cases for the function $\hat{\alpha}^{\text{out}} : \hat{\alpha}^{\text{out}}\left(v_{i,j_1}^{\hat{\mathscr{H}}_1}, -\right)$, $\hat{\alpha}^{\text{out}}\left(-, v_{i,j_2}^{\hat{\mathscr{H}}_2}\right)$, $\hat{\alpha}^{\text{out}}\left(v_{i,j_1}^{\hat{\mathscr{H}}_1}, v_{i,j_2}^{\hat{\mathscr{H}}_2}\right)$. We infer by Eqs. (22)–(24) that

$$\hat{\alpha}^{\text{out}}\left(v_{i,j_1}^{\hat{\mathscr{H}}_1}, \text{align}\left(v_{i,j_1}^{\hat{\mathscr{H}}_1}\right)\right) \leqslant 1 \quad \text{and} \quad \hat{\alpha}^{\text{out}}\left(v_{i,j_2}^{\hat{\mathscr{H}}_2}, \text{align}\left(v_{i,j_2}^{\hat{\mathscr{H}}_2}\right)\right) \leqslant 1.$$

Hence, by definition (28) we obtain $\gamma^{\text{out}}(i) \leqslant 1$. The proof $\gamma^{\text{in}}(i) \leqslant 1$ is identical. Since

$$\gamma^{\text{fin}}(i) \leqslant \zeta + (1 - \zeta) = 1,$$

we obtain

$$d_1(\hat{\mathscr{H}}_1, \hat{\mathscr{H}}_2) \leqslant \frac{\sum_{i=0}^{\rho} \lambda_i}{\sum_{i=0}^{\rho} \lambda_i} = 1. \tag{36}$$

To prove that $d_1(\hat{\mathscr{H}}_1, \hat{\mathscr{H}}_2)$ is symmetric, we see that

$$\begin{aligned}\gamma^{\text{out}}(i) &:= 1 - \frac{1}{\sigma_i^1 + \sigma_i^2} \cdot \left\{ \sum_{j=1}^{\sigma_i^1} \hat{\alpha}^{\text{out}}\left(v_{i,j}^{\hat{\mathscr{H}}_1}, \text{align}\left(v_{i,j}^{\hat{\mathscr{H}}_1}\right)\right) \right\} + \frac{1}{\sigma_i^1 + \sigma_i^2} \cdot \left\{ \sum_{j=1}^{\sigma_i^2} \hat{\alpha}^{\text{out}}\left(v_{i,j}^{\hat{\mathscr{H}}_2}, \text{align}\left(v_{i,j}^{\hat{\mathscr{H}}_2}\right)\right) \right\} \\ &= 1 - \frac{1}{\sigma_i^2 + \sigma_i^1} \cdot \left\{ \sum_{j=1}^{\sigma_i^1} \hat{\alpha}^{\text{out}}\left(v_{i,j}^{\hat{\mathscr{H}}_2}, \text{align}\left(v_{i,j}^{\hat{\mathscr{H}}_2}\right)\right) \right\} + \frac{1}{\sigma_i^2 + \sigma_i^1} \cdot \left\{ \sum_{j=1}^{\sigma_i^2} \hat{\alpha}^{\text{out}}\left(v_{i,j}^{\hat{\mathscr{H}}_1}, \text{align}\left(v_{i,j}^{\hat{\mathscr{H}}_1}\right)\right) \right\},\end{aligned}$$

and

$$\begin{aligned}\gamma^{\text{in}}(i) &:= 1 - \frac{1}{\sigma_i^1 + \sigma_i^2} \cdot \left\{ \sum_{j=1}^{\sigma_i^1} \hat{\alpha}^{\text{in}}\left(v_{i,j}^{\hat{\mathscr{H}}_1}, \text{align}\left(v_{i,j}^{\hat{\mathscr{H}}_1}\right)\right) \right\} + \frac{1}{\sigma_i^1 + \sigma_i^2} \cdot \left\{ \sum_{j=1}^{\sigma_i^2} \hat{\alpha}^{\text{in}}\left(v_{i,j}^{\hat{\mathscr{H}}_2}, \text{align}\left(v_{i,j}^{\hat{\mathscr{H}}_2}\right)\right) \right\} \\ &= 1 - \frac{1}{\sigma_i^2 + \sigma_i^1} \cdot \left\{ \sum_{j=1}^{\sigma_i^1} \hat{\alpha}^{\text{in}}\left(v_{i,j}^{\hat{\mathscr{H}}_2}, \text{align}\left(v_{i,j}^{\hat{\mathscr{H}}_2}\right)\right) \right\} + \frac{1}{\sigma_i^2 + \sigma_i^1} \cdot \left\{ \sum_{j=1}^{\sigma_i^2} \hat{\alpha}^{\text{in}}\left(v_{i,j}^{\hat{\mathscr{H}}_1}, \text{align}\left(v_{i,j}^{\hat{\mathscr{H}}_1}\right)\right) \right\}.\end{aligned}$$

Therefore, we conclude from Eq. (32) that

$$d_1(\hat{\mathscr{H}}_1, \hat{\mathscr{H}}_2) = d_1(\hat{\mathscr{H}}_2, \hat{\mathscr{H}}_1).$$

For finalizing the proof for the similarity measure (32), we have to show that

$$d_1(\hat{\mathscr{H}}_1, \hat{\mathscr{H}}_1) \geqslant d_1(\hat{\mathscr{H}}_1, \hat{\mathscr{H}}_2).$$

If $\hat{\mathscr{H}}_1 = \hat{\mathscr{H}}_2$ then, $\gamma^{\text{out}}(i) = 1$, $\gamma^{\text{in}}(i) = 1$ and $\gamma^{\text{fin}}(i) = 1$. Therefore we infer from Eq. (32) that $d_1(\hat{\mathscr{H}}_1, \hat{\mathscr{H}}_1) = 1$ and see

$$d_1(\hat{\mathscr{H}}_1, \hat{\mathscr{H}}_1) = 1 \geqslant \frac{\sum_{i=0}^{\rho} \lambda_i \cdot \gamma^{\text{fin}}(i)}{\sum_{i=0}^{\rho} \lambda_i} = d_1(\hat{\mathscr{H}}_1, \hat{\mathscr{H}}_2),$$

the Backward property. In the case of similarity measure (33), we have nothing to prove, because if we set $1 = \lambda_0 = \lambda_1 = \cdots = \lambda_\rho$ in Eq. (32), we obtain Eq. (33). To prove the assertion of the theorem for Eq. (34), we consider the well-known inequality

$$(p_1 \cdot p_2 \cdots p_n)^{\frac{1}{n}} \leqslant \frac{p_1 + p_2 + \cdots + p_n}{n}, \quad p_i > 0, \ \ 1 \leqslant i \leqslant n. \tag{37}$$

Since $\gamma^{\text{fin}}(i) \leqslant 1$, we can apply inequality (37) and get

$$\gamma^{\text{fin}}(0) \cdot \gamma^{\text{fin}}(1) \cdots \gamma^{\text{fin}}(\rho) \leqslant [\gamma^{\text{fin}}(0) \cdot \gamma^{\text{fin}}(1) \cdots \gamma^{\text{fin}}(\rho)]^{\frac{1}{\rho+1}} \leqslant \frac{\gamma^{\text{fin}}(0) + \gamma^{\text{fin}}(1) + \cdots + \gamma^{\text{fin}}(\rho)}{\rho + 1}.$$

Especially, we have

$$1 \geqslant (\rho + 1) \frac{\gamma^{\text{fin}}(0) \cdot \gamma^{\text{fin}}(1) \cdots \gamma^{\text{fin}}(\rho)}{\gamma^{\text{fin}}(0) + \gamma^{\text{fin}}(1) + \cdots + \gamma^{\text{fin}}(\rho)}. \tag{38}$$

The symmetry condition is clear, because the expression in the denominator of Eq. (34) is a special case of d_1. The Backward condition follows immediately from Inequality Eq. (38),

$$1 = d_3(\hat{\mathscr{H}}_1, \hat{\mathscr{H}}_1) \geqslant \frac{\prod_{i=0}^{\rho} \gamma^{\text{fin}}(i)}{d_2(\hat{\mathscr{H}}_1, \hat{\mathscr{H}}_2)}. \qquad \square$$

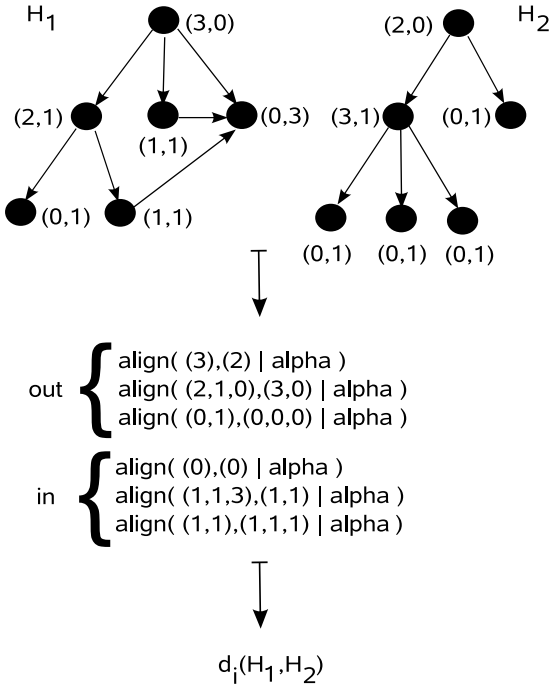

Fig. 5. Steps of similarity measuring for generalized trees.

Fig. 5 shows the overall approach exemplarily. In the following Section 4 we analyze the computational complexity of our novel graph similarity measure.

4. Efficiency analysis

To determine the computational complexity, first, we need the following definition.

Definition 4.1. Let $V_{S_1,S_2} := \{(i,j)|0 \leqslant i \leqslant n, 0 \leqslant j \leqslant m\}$ be the vertex set and the edge types are $e_{\text{Del}} := (i-1,j) \to (i,j)$, $e_{\text{Ins}} := (i,j-1) \to (i,j)$, $e_{\text{Subst}} := (i-1,j-1) \to (i,j)$. $f_{E_{S_1,S_2}} : E_{S_1,S_2} \to \mathbb{R}_+$ denotes the edge labeling function. The edge set is defined by

$$E_{S_1,S_2} := \begin{cases} e_{\text{Del}}, & i \in [1,n], f_{E_{S_1,S_2}}(e_{\text{Del}}) = [S_1[i],-], \\ e_{\text{Ins}}, & j \in [1,m], f_{E_{S_1,S_2}}(e_{\text{Ins}}) = [-,S_2[j]], \\ e_{\text{Subst}}, & i \in [1,n], j \in [1,m], f_{E_{S_1,S_2}}(e_{\text{Subst}}) = [S_1[i],S_2[j]], \end{cases}$$

$G_{S_1,S_2} := (V_{S_1,S_2}, E_{S_1,S_2}, f_{E_{S_1,S_2}})$ is called the alignment graph of the sequences S_1 and S_2 representing generalized trees.

The edge types of these graphs reflect computational meanings in terms of transforming S_1 to S_2.

Definition 4.2. $(i-1,j) \to (i,j)$ equals the deletion of $S_1[i]$ in S_1, $(i,j-1) \to (i,j)$ equals the insertion of $S_2[j]$ in S_1 at the ith position, and $(i-1,j-1) \to (i,j)$ equals the substitution of $S_1[i]$ by $S_2[j]$.

Definition 4.3. Let A be an arbitrary alphabet and $w_1 \in A^{\bigstar}$. Let I, D, R, M be a sequence of edit operations. I (insert) denotes the insertion, D (deletion) denotes the deletion, R (replace) denotes the substitution and M (match) denotes the correspondence. A transformation of w_1 into w_2 based on these edit operations is called edit transcript.

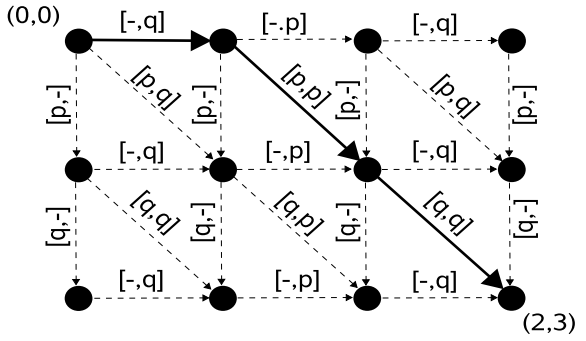

Fig. 6. The alignment graph G_{S_1,S_2} of S_1 and S_2.

Definition 4.4. Let A be an arbitrary alphabet and $\{S_i | S_i \in A^{\star}\}$ be the set of finite sequences of $A^{\star}$. Then, $l : S \to \mathbb{N}$ denotes the function that determines the length of $S_i \in S$.

In order to state the time complexity of the recursive algorithm defined by Definition 3.1 we need a theorem of Gusfield [8] that states the relationship between alignments and paths of the associated alignment graph.

Theorem 4.1. *Let S_1, $S_2 \in S$ be sequences and it holds $l(S_1) = n$, $l(S_2) = m$. The following assertions are equivalent*:

- *An edit transcript of S_1 and S_2 possesses a minimal number of edit operations.*
- *It corresponds a path in the associated alignment graph from vertex position $(0,0)$ to (n,m) with total minimal costs.*

It follows straightforward

Corollary 4.2. *The set of paths with total minimal costs from vertex position $(0,0)$ to (n,m) in the associated alignment graph specifies the set of optimal edit transcripts of S_1 and S_2. The set of optimal edit transcripts describes the set of all optimal alignments from S_1 and S_2.*

Fig. 6 shows the alignment graph G_{S_1,S_2} of a simple example. If we set $S_1 = p \circ q$ and $S_2 = q \circ p \circ q$ the alignment is

$$\begin{array}{cccc} - & p & q \\ q & p & q \end{array}$$

Now, we obtain a theorem which states the computational complexity of the recursive algorithm defined by Definition 3.1.

Theorem 4.3. *Let $\hat{\mathscr{H}}_1$, $\hat{\mathscr{H}}_2$ be generalized trees and S_1, S_2 be the associated vertex sequences, based on $\hat{V}_1$, $\hat{V}_2$. The recursive algorithm for computing the optimal alignment between S_1 and S_2 given by Definition 3.1 possesses the computational complexity $\mathscr{O}(|\hat{V}_1| \cdot |\hat{V}_2| \cdot \mathscr{O}(1))$.*

Proof. Starting from Definition 3.1 and especially with Eq. (12) we yield that the underlying algorithm needs time complexity $\mathscr{O}(|\hat{V}_1| \cdot |\hat{V}_2| \cdot \mathscr{O}(1))$ for creating the matrix $(\mathscr{M}(i,j))_{ij}, 0 \leqslant i \leqslant |\hat{V}_1|, 0 \leqslant j \leqslant |\hat{V}_2|$. Furthermore, at each matrix position we compute the cost function α with time complexity $\mathscr{O}(1)$. It follows that the edit distance $\mathscr{M}(|\hat{V}_1|, |\hat{V}_2|)$ needs complexity $\mathscr{O}(|\hat{V}_1| \cdot |\hat{V}_2| \cdot \mathscr{O}(1))$. But Corollary 4.2 states that we have also found the optimal alignment. This completes the proof of the theorem. □

Finally, we get

Corollary 4.4. *The structural similarity between two generalized trees $\hat{\mathscr{H}}_1$ and $\hat{\mathscr{H}}_2$ holds the time complexity* $\mathscr{O}(|\hat{V}_1| \cdot |\hat{V}_2| \cdot \mathscr{O}(1))$.

5. Conclusions

In the present paper we presented an algorithm with polynomial computational complexity to measure the structural similarity of a new graph class – generalized trees. The procedure to measure the structural similarity of graphs is completely different from known methods. First, we transformed the graphs in linear structures which contain structural information. Second, we determined the optimal alignment between the linear structures (out-degree and in-degree sequences on each level) and then we defined a family $(d_i)_{1 \leqslant i \leqslant 3}$ of graph similarity measures based on the obtained similarity scores (from $\gamma^{\text{out}}(i)$, $\gamma^{\text{out}}(i)$).

The efficiency analysis of the underlying algorithm was performed in Section 4. From this analysis, we found an efficient approach for graph similarity measuring in contrast to classical approaches [12,20,25] which are $\mathscr{NP}$-complete (see Section 2.5). Compared to known methods to measure the structural similarity of trees stated in Section 2.2 we found that our algorithm is comparable efficient despite its more general character.

References

[1] J. Bang-Jensen, G. Gutin, Digraphs. Theory, Algorithms and Applications, Springer-Verlag, London–Berlin–Heidelberg, 2000.
[2] R. Bellman, Dynamic Programming, Princeton University Press, 1957.
[3] H. Bunke, G. Allermann, A metric on graphs for structural pattern recognition, in: H.W. Schussler (Ed.), SIGNAL PROCESSING II: Theory and Applications, 1983, pp. 257–260.
[4] H. Bunke, Graph matching: theoretical foundations, algorithms, and applications, in: Proceedings of the Vision Interface 2000, Montreal/Canada, 2000, pp. 82–88.
[5] H. Bunke, Recent developments in graph matching, in: Proceedings of the 15th International Conference on Pattern Recognition, Barcelona/Spain, 2000, vol. 2, pp. 117–124.
[6] F. Emmert-Streib, M. Dehmer, J. Kilian, Classification of large graphs by a local tree decomposition, in: H.R. Arabnia, A. Scime (Eds.), Proceedings of the 2005 International Conference on Data Mining (DMIN'05), 2005, pp. 200–207.
[7] R. Gleim, A. Mehler, M. Dehmer, Web corpus mining by instance of Wikipedia, Appears in Proceedings of the EACL 2006 Workshop on Web as Corpus, Trento, Italy, 3–7 April 2006.
[8] D. Gusfield, Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology, Cambridge University Press, 1997.
[9] M. Höchstmann, T. Töller, R. Giegerich, S. Kurtz, Local similarity in RNA secondary structures, in: Proceedings of the IEEE Computational Systems Bioinformatics Conference (CSB'03), 2003, pp. 159–168.
[10] T. Jiang, L. Wang, K. Zhang, Alignment of Trees – An Alternative to Tree Edit, Theoretical Computer Science, vol. 143, Elsevier, 1995, pp. 137–148.
[11] F. Kaden, Graph metrics and distance-graphs, in: M. Fiedler (Ed.), Graphs and other Combinatorial Topics Teubner Texte zur Math., Leipzig, 1983, vol. 59, pp. 145–158.
[12] F. Kaden, Graphmetriken und Distanzgraphen, ZKI-Informationen, Akad. Wiss. DDR (1982) 1–63.
[13] I. Koch, T. Lengauer, E. Wanke, An algorithm for finding maximal common subtopologies in a set of protein structures, J. Comput. Biol. 3 (2) (1996) 289–306.
[14] V.I. Levenstein, Binary codes capable of correcting deletions insertions and reversals, Soviet Phys. Dokl. 10 (1966) 707–710.
[15] A. Mehler, Hierarchical orderings of textual units, in: Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002), Taipei, Taiwan, Morgan Kaufmann, San Francisco, 2002, pp. 646–652.
[16] A. Mehler, M. Dehmer, R. Gleim, Towards logical hypertext structure. A graph-theoretic perspective, in: Proceedings of I2CS'04, Guadalajara/Mexico, Lecture Notes in Computer Science, Springer, Berlin–New York, 2004.
[17] D. Sankoff, J.B. Kruskal, Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison, Addison-Wesley, 1983.
[18] D. Sankoff, J.B. Kruskal, S. Mainville, R.J. Cedergren, Fast algorithms to determine RNA secondary structures containing multiple loops, in: D. Sankoff, J.B. Kruskal (Eds.), Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison, Addison-Wesley, 1983, pp. 93–120.
[19] S.M. Selkow, The tree-to-tree editing problem, Inform. Process. Lett. 6 (6) (1977) 184–186.
[20] F. Sobik, Graphmetriken und klassifikation strukturierter objekte, ZKI-Informationen Akad. Wiss. DDR 2 (82) (1982) 63–122.
[21] F. Sobik, Modellierung von Vergleichsprozessen auf der Grundlage von Ähnlichkeitsmasen für Graphen, ZKI-Informationen, Akad. Wiss. DDR 4 (1986) 104–144.

[22] K. Tai, The tree-to-tree correction problem, J. ACM 26 (1979) 422–433.
[23] J.R. Ullman, An algorithm for subgraph isomorphism, J. ACM 23 (1) (1976) 31–42.
[24] L. Wang, J. Zhao, Parametric alignments of ordered trees, Bioinformatics 19 (17) (2003) 2237–2245.
[25] B. Zelinka, On a certain distance between isomorphism classes of graphs, Časopis pro pěst. Mathematiky 100 (1975) 371–373.
[26] K. Zhang, R. Statman, D. Shasha, On the editing distance between unordered labeled trees, Int. Inform. Process. Lett. 42 (3) (1992) 133–139.