Faster Algorithms for the Maximum Common Subtree Isomorphism Problem*

Andre Droschinsky[1], Nils M. Kriege[2], and Petra Mutzel[3]

1 **Dept. of Computer Science, Technische Universität Dortmund, Germany**
andre.droschinsky@tu-dortmund.de
2 **Dept. of Computer Science, Technische Universität Dortmund, Germany**
nils.kriege@tu-dortmund.de
3 **Dept. of Computer Science, Technische Universität Dortmund, Germany**
petra.mutzel@tu-dortmund.de

Abstract

The maximum common subtree isomorphism problem asks for the largest possible isomorphism between subtrees of two given input trees. This problem is a natural restriction of the maximum common subgraph problem, which is NP-hard in general graphs. Confining to trees renders polynomial time algorithms possible and is of fundamental importance for approaches on more general graph classes. Various variants of this problem in trees have been intensively studied. We consider the general case, where trees are neither rooted nor ordered and the isomorphism is maximum w.r.t. a weight function on the mapped vertices and edges. For trees of order n and maximum degree Δ our algorithm achieves a running time of $\mathcal{O}(n^2\Delta)$ by exploiting the structure of the matching instances arising as subproblems. Thus our algorithm outperforms the best previously known approaches. No faster algorithm is possible for trees of bounded degree and for trees of unbounded degree we show that a further reduction of the running time would directly improve the best known approach to the assignment problem. Combining a polynomial-delay algorithm for the enumeration of all maximum common subtree isomorphisms with central ideas of our new algorithm leads to an improvement of its running time from $\mathcal{O}(n^6 + Tn^2)$ to $\mathcal{O}(n^3 + Tn\Delta)$, where n is the order of the larger tree, T is the number of different solutions, and Δ is the minimum of the maximum degrees of the input trees. Our theoretical results are supplemented by an experimental evaluation on synthetic and real-world instances.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases MCS, maximum common subtree, enumeration algorithms, maximum weight bipartite matchings

Digital Object Identifier 10.4230/LIPIcs.MFCS.2016.33

1 Introduction

The maximum common subgraph isomorphism problem (MCS) asks for an isomorphism between induced subgraphs of two given graphs that is of maximum weight w.r.t. a weight function on the mapped vertices and edges. The problem is of fundamental importance in applications like pattern recognition [5] or bio- and cheminformatics [9, 18]. MCS naturally generalizes the subgraph isomorphism problem (SI), where the task is to decide if one graph

* This work was supported by the German Research Foundation (DFG), priority programme "Algorithms for Big Data" (SPP 1736).

is isomorphic to a subgraph of another graph. Both problems are known to be NP-hard for general graphs.

It is not astonishing that these problems have been extensively studied for restricted graph classes. Polynomial time algorithms for SI and MCS in trees have been pioneered by Edmonds and Matula in the 1960s. They rely on solving a series of maximum bipartite matching instances, see [15]. These early results focused on the polynomial time complexity of the problem; since then considerable progress has been achieved in improving the running time of SI algorithms (also see [1] and references therein): Reyner [17, 23] and Matula [15] both showed a running time of $\mathcal{O}(n^{2.5})$ for rooted trees. Chung [4] later obtained the same bound for unrooted trees. Further improvements were made by Shamir and Tsur [19] who obtained time $\mathcal{O}(n^{2.5}/\log n)$ and $\mathcal{O}(n^{\omega})$, where ω is the exponent of matrix multiplication.

MCS on trees seems to be harder. For two rooted trees of size n, it is known that the problem can be solved, roughly speaking, in the same time as the associated maximum weight matching problem in a bipartite graph on n vertices: Gupta and Nishimura [11] presented an $\mathcal{O}(n^{2.5}\log n)$ algorithm for MCS in rooted trees by assuming weights to be in $\mathcal{O}(n)$, which allows to employ a scaling approach to the matching problem [10]. The running time can be improved to $\mathcal{O}(\sqrt{\Delta}n^2\log\frac{2n}{\Delta})$, where Δ denotes the maximum degree [12]. Allowing a real weight function to determine the similarity of mapped vertices gives rise to bipartite matching instances with unrestricted weights. Solving these with the Hungarian method leads to cubic running time, see e.g. [22]. Since the the size of the matching instances is bounded by the maximum degree Δ, the result can be improved to $\mathcal{O}(n^2\Delta)$ time [20]. Various related concepts for the comparison of rooted trees, either ordered or unordered, have been proposed and were studied in detail, see [22] and references therein, where the *tree edit distance* is a prominent example [3, 6].

In this article we consider the problem of finding a common subtree isomorphism in unrooted, unordered trees that is maximum w.r.t. a weight function on the mapped vertices and edges. This problem is directly relevant in various applications, where real-world objects like molecules or shapes are represented by (attributed) trees [16, 20]. Moreover, it forms the basis for several recent approaches to solve MCS in more general graph classes, see [2, 13, 14, 18]. Methods directly based on algorithms for rooted trees result in time $\mathcal{O}(n^4\Delta)$ by considering all pairs of possible roots. An improvement to $\mathcal{O}(n^3\Delta)$ has been reported in [20], which is limited to non-negative weight functions. Schietgat, Ramon and Bruynooghe [18] suggested an approach for MCS in outerplanar graphs, which solves the considered problem when applied to trees. The approach is stated to have a running time of $\mathcal{O}(n^{2.5})$, but in fact leads to a running time of $\Omega(n^4)$ in the worst case.[1]

Our contribution. We show that for arbitrary weights a maximum common subtree isomorphism between two trees G and H of order n with $\Delta(G) \leq \Delta(H)$ can be computed in time $\mathcal{O}(n^2(\Delta(G) + \log\Delta(H))$. We obtain the improvement by (i) considering only a specific subset of subproblems that we show to be sufficient to guarantee an optimal solution; (ii) exploiting the close relation between the emerging matching instances. We show that for general trees any further improvement of this time bound would allow to solve the assignment problem in $o(n^3)$, and hence improve over the best known approach to this famous problem for more than 30 years. For trees of bounded degree the running time bound of $\mathcal{O}(n^2)$ is

[1] The analysis of the algorithm appears to be flawed. An Erratum to [18] has been submitted to the *Annals of Mathematics and Artificial Intelligence*, see Appendix of `https://arxiv.org/abs/1602.07210`. Our experimental study of their implementation actually suggests a time bound of $\Omega(n^5)$.

tight. We apply our new techniques to the problem of enumerating all maximum common subtree isomorphisms, thus improving the state-of-the-art running times. Finally, we present an experimental evaluation on synthetic and real-world instances showing that our new algorithm is faster than existing approaches.

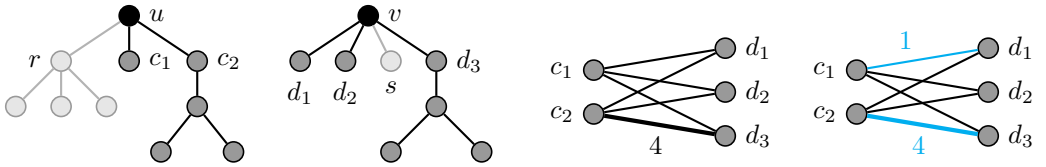
2 Preliminaries

In this paper, $G = (V, E)$ is a *simple undirected graph*. We call $v \in V$ a *vertex* and $uv = vu \in E$ an *edge* of G. For a graph $G = (V, E)$ we define $V(G) := V_G := V$, $E(G) := E_G := E$ and $|G| := |V(G)|$. For a subset of vertices $V' \subseteq V$ the graph $G[V'] := (V', E')$, $E' := \{uv \in E \mid u, v \in V'\}$, is called *induced* subgraph. A connected graph with a unique path between any two vertices is a *tree*. A tree G with an explicit root vertex $r \in V_G$ is called *rooted* tree, denoted by G^r. In a rooted tree G^r we denote the children of a vertex v by $C(v)$ and its parent by $p(v)$, where $p(r) = r$. The *depth* $\operatorname{depth}(v)$ of a vertex v is the number of edges on the path from v to r. The *neighbors* of a vertex v are defined as $N(v) := \{u \in V_G \mid vu \in E_G\}$. The *degree* of a vertex $v \in V_G$ is $\delta(v) := |N(v)|$, the *degree* $\Delta(G)$ of a graph G is the maximum degree of its vertices.

For a graph $G = (V, E)$ a *matching* $M \subseteq E$ is a set of edges, such that no two edges share a vertex. A matching M of G is said to be *perfect*, if $2|M| = |V|$. A *weighted graph* is a graph endowed with a function $w : E \to \mathbb{R}$. The weight of a matching M in a weighted graph is $W(M) := \sum_{e \in M} w(e)$. We call a matching M of a weighted bipartite graph G a *maximum weight matching* (MWM) if there is no other matching M' of G with $W(M') > W(M)$. The *assignment problem* asks for a matching with maximum weight among all perfect matchings and we refer to a solution by MWPM.

An *isomorphism* between two graphs G and H is a bijective function $\phi : V_G \to V_H$ such that $uv \in E_G \Leftrightarrow \phi(u)\phi(v) \in E_H$; if such an isomorphism exists, G and H are said to be *isomorphic*. We call a graph G *subgraph isomorphic* to a graph H, if there is an induced subgraph $H' \subseteq H$ isomorphic to G. In this case, we write $G \preceq_\phi H$, where $\phi : V_G \to V_{H'}$ is an isomorphism between G and H'. A *common subgraph* of G and H is a graph I, such that $I \preceq_\phi G$ and $I \preceq_{\phi'} H$. The isomorphism $\varphi := \phi' \circ \phi^{-1}$ is called *common subgraph isomorphism* (CSI). For a function $f : X \to Y$ let $\operatorname{dom}(f) := X$ be the domain of f. If there is no other CSI φ' with $|\operatorname{dom}(\varphi')| > |\operatorname{dom}(\varphi)|$, we call φ *maximum* common subgraph isomorphism (MCSI).

We generalize the above definitions to a pair of graphs G, H under a commutative weight-function $\omega : (V_G \times V_H) \cup (E_G \times E_H) \to \mathbb{R} \cup \{-\infty\}$. The weight $\mathcal{W}(\phi)$ of an isomorphism ϕ between G and H under ω is the sum of the weights $\omega(v, \phi(v))$ and $\omega(vu, \phi(v)\phi(u))$ of all vertices and edges mapped by ϕ. A maximum common subgraph isomorphism ϕ under a weight function is one of maximum weight $\mathcal{W}(\phi)$ instead of maximum size $|\operatorname{dom}(\phi)|$. Note, this is less restrictive than a common approach for isomorphisms on labeled graphs, where the labels must match. By defining ω such that mapped vertices and mapped edges add 1 and 0, respectively, to the weight, we obtain an isomorphism of maximum size. Therefore, in the following we consider graphs under a weight function unless stated otherwise and refer to the corresponding solution as MCSI. For convenience we replace the word *graph* by *tree* in the above definitions when appropriate. The *maximum common subtree isomorphism problem* (MCST) is to determine the weight of an MCSI, where the input graphs and the common subgraph are trees. We further define $[1..k] := \{1, \ldots, k\}$ for $k \in \mathbb{N}$.



(a) Rooted subtree G_u^r. **(b)** Rooted subtree H_v^s. **(c)** Matching problem. **(d)** MWM.

Figure 1 Two rooted subtrees (a) and (b), the associated weighted matching instance (c), and an MWM on that instance (d). Light gray vertices and edges are not part of the rooted subtrees, root vertices are shown in solid black. The maximum weight matching is shown in blue. We assume a weight function ω with $\omega(u, v) = 1$ for all $(u, v) \in V_G \times V_H$ and $\omega(e, f) = 0$ for all $(e, f) \in E_G \times E_H$. The edges without label in (c) have weight 1.

3 Problem Decomposition and Fundamental Algorithms

We introduce the basic techniques for solving MCST following the ideas of Edmonds and Matula [15]. The approach requires to compute MWMs in bipartite graphs as a subroutine. We discuss the occurring matching instances in detail in Section 4.

By fixing the roots of both trees we can develop an algorithm solving MCST on this restricted setting. It is easy to generalize this solution by considering all possible pairs of roots. We then show that it is sufficient to fix the root of one tree while still obtaining a maximum solution.

Rooted trees. We first consider the problem restricted to rooted trees under the assumption that the roots of the two trees must be mapped to each other. For a rooted tree G^r we define the *rooted subtree* G_u^r as the subtree induced by u and all its descendants in G^r that is rooted at u, cf. Figures 1a and b. Note that $G_r^r = G^r$ and that G_u^r and G_u^s both refer to the same subtree unless s is contained in G_u^r. The key to solving MCST for two rooted trees G^r and H^s is the following recursive formulation:

$$\mathrm{MCS}_{\mathrm{root}}(G^r, H^s) = \omega(r, s) + W(M), \tag{1}$$

where M is an MWM of the complete bipartite graph on the vertex set $C(r) \uplus C(s)$ with weights $w(uv) = \omega(ru, sv) + \mathrm{MCS}_{\mathrm{root}}(G_u^r, H_v^s)$ for all $u \in C(r)$ and $v \in C(s)$. Hence, each edge weight corresponds to the solution of a problem of the same type for a pair of smaller rooted subtrees and the recursion naturally stops at the leaves. Each subproblem, the initial one as well as those arising in recursive calls, is uniquely defined by a pair of rooted subtrees and essentially consists of solving a matching instance.

Figure 1 illustrates the two rooted subtrees G_u^r and H_v^s and the corresponding matching problem under the weight function as given in the figure. For rooted trees G^r and H^s this problem arises on the second level in the recursion of Eq. (1). We obtain $\mathrm{MCS}_{\mathrm{root}}(G_u^r, H_v^s) = \omega(u, v) + W(M) = 1 + 5 = 6$, where M is an MWM of Figure 1c, depicted in Figure 1d.

In order to compute Eq. (1) the subproblems defined by the pairs of rooted subtrees $\mathcal{S}_{\mathrm{root}}(G^r, H^s) := \{(G_u^r, H_v^s) \mid \mathrm{depth}(u) = \mathrm{depth}(v)\}$ have to be solved.

▶ **Proposition 1.** *A maximum common subtree isomorphism for two rooted trees G^r and H^s can be computed in time $\mathcal{O}(n^3)$, where $n = |G| + |H|$.*

Proof. The bipartite graph for the subproblem (G_u^r, H_v^s) contains $k_u + l_v$ vertices, where $k_u := |C(u)|$ and $l_v := |C(v)|$. For the total running time we distinguish the cases $k_u \leq l_v$

and $k_u > l_v$. For the first case we obtain a MWM in time $\mathcal{O}(k_u l_v(k_u + \log l_v))$ according to Lemma 5. The second case is analog. Since $\mathcal{S}_{\text{root}}(G^r, H^s) \subseteq \{(G^r_u, H^s_v) \mid u \in V_G, v \in V_H\}$ the total running time is bounded by $\mathcal{O}(n^3)$ as

$$\sum_{u \in V_G} \sum_{v \in V_H, k_u \leq l_v} k_u l_v(k_u + \log l_v) \leq \sum_{u \in V_G} k_u \sum_{v \in V_H} l_v(l_v + \log l_v) \leq n \cdot 2n^2 \in \mathcal{O}(n^3)$$

◀

Unrooted trees. We now consider the problem for unrooted trees. An immediate solution is to solve the rooted problem variant for all possible pairs of roots, i.e., by computing

$$\text{MCS}(G, H) := \max \{\text{MCS}_{\text{root}}(G^r, H^s) \mid r \in V(G), s \in V(H)\}. \tag{2}$$

Clearly, this yields the optimal solution in time $\mathcal{O}(n^5)$ with Proposition 1. Note that several recursive calls involve solving the same subproblem. Repeated computation can easily be avoided by means of a lookup table. Let $\text{RT}(G^r) := \{G^r_u \mid u \in V(G)\}$ and $\text{RT}(G) := \bigcup_{r \in V(G)} \text{RT}(G^r)$. Note that we may uniquely associate the subtree G^r_u with G^p_u, where p is the parent of u in G^r. Hence, each rooted subtree $G^u_v \in \text{RT}(G)$ either is the whole tree G with root $u = v$ or is the subtree rooted at v of some edge $uv \in E(G)$, where u is not contained in the subtree. Thus, $\text{RT}(G) = \{G^u_v \mid v \in V(G) \wedge u \in N(v) \cup \{v\}\}$ is the set of all rooted subtrees of G. In total the subproblems defined by $\mathcal{S}(G, H) := \text{RT}(G) \times \text{RT}(H)$ have to be solved.

However, ensuring that each subproblem is solved only once does not allow to improve the bound on the running time, since $\mathcal{S}(G, H)$ still may contain a quadratic number of subproblems of linear size: Let G and H be two star graphs on n vertices, i.e., trees with all but one vertex of degree one. Each of the $(n-1)^2$ pairs of leaves can be selected as root pair and leads to a different subproblem of size $n - 1$.

We show that it is sufficient to consider only a subset of the subproblems to guarantee that an optimal solution is found. Let

$$\text{MCS}_{\text{fast}}(G^r, H) := \max \{\text{MCS}_{\text{root}}(G^r_u, H^s) \mid u \in V(G), s \in V(H)\}, \tag{3}$$

where $r \in V(G)$ is an arbitrary but fixed root of G. To compute Eq. (3), only the subproblems $\mathcal{S}_{\text{fast}}(G^r, H) := \text{RT}(G^r) \times \text{RT}(H) \subseteq \mathcal{S}(G, H)$ need to be solved.

▶ **Lemma 2.** *Let* MCS_{fast} *and* MCS *be defined as above and* $r \in V(G)$ *arbitrary but fixed, then* $\text{MCS}_{\text{fast}}(G^r, H) = \text{MCS}(G, H)$ *for all trees* G, H.

Proof. Let ϕ be an MCSI. If r is in the domain of ϕ, then $\omega(\phi) = \text{MCS}_{\text{root}}(G^r, H^{\phi(r)}) = \text{MCS}_{\text{fast}}(G^r, H)$. Otherwise the domain of ϕ is contained in the subtree rooted at one child of r. Let u be the unique vertex that is closest to r and mapped by ϕ. Then $\omega(\phi) = \text{MCS}_{\text{root}}(G^r_u, H^{\phi(u)}) = \text{MCS}_{\text{fast}}(G^r, H)$. ◀

Algorithm 1 implements this strategy, where the postorder traversal on G^r (line 2) ensures that the solutions to smaller subproblems are always available when required (line 9). The lookup table contains one entry for each subproblem in $\mathcal{S}_{\text{fast}}(G^r, H)$ and hence requires space $\mathcal{O}(n^2)$. Note that it is also possible to compute a concrete isomorphism from the MWMs associated with the computed optimal solution. The restriction of the considered subproblems allows to improve the bound on the running time.

▶ **Proposition 3.** *Algorithm 1 solves the maximum common subtree isomorphism problem for two trees* G *and* H *in time* $\mathcal{O}(n^4)$, *where* $n = |G| + |H|$.

Algorithm 1: Maximum Common Subtree Isomorphism

Input : Trees G and H under a weight function ω
Output : Weight of an MCSI between G and H.
Data : Table $D(u, s, v)$ storing solutions $\mathrm{MCS}_{\mathrm{root}}(G_u^r, H_v^s)$ of subproblems.

```
1  Select an arbitrary root vertex r ∈ V_G.
2  foreach u ∈ V_G in postorder traversal on G^r do        ▷ All possible G_u^r ∈ Rt(G^r)
3      U ← C(u) in G^r
4      foreach v ∈ V_H do
5          foreach s ∈ N(v) ∪ {v} do                       ▷ All possible H_v^s ∈ Rt(H)
6              V ← C(v) in H^s
7              if ω(u, v) ≠ −∞ then
8                  foreach pair (u', v') ∈ U × V do
9                      w(u'v') ← ω(uu', vv') + D(u', v, v')
10                 M ← MWM of the complete graph on U ⊎ V with weights w.
11                 D(u, s, v) ← ω(u, v) + W(M)
12             else D(u, s, v) ← −∞
13 return the maximum entry in D
```

Proof. According to Lemma 2 computing Eq. (3) yields the optimal solution and it suffices to solve the subproblems $\mathcal{S}_{\mathrm{fast}}(G^r, H)$ as realized by Algorithm 1. Let k_u be the number of children of u in G_u^r, l_v^s the number of children of v in $H^s, s \in V(H)$, and $l_v = |N(v)|$. For all s we have $l_v^s \leq l_v$. Similar to Proposition 1 the subproblems $\mathcal{S}_{\mathrm{fast}}(G, H)$ can be solved in a total time of

$$\mathcal{O}\left(\sum_{u\in V_G}\sum_{s\in V_H}\sum_{v\in V_H}(k_u l_v^s)(\min\{k_u, l_v^s\} + \log\max\{k_u, l_v^s\})\right) \subseteq \mathcal{O}\left(\sum_{s\in V_H} n^3\right) \subseteq \mathcal{O}\left(n^4\right).$$

◀

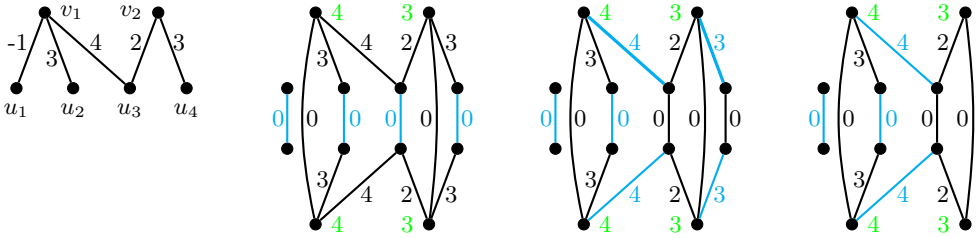
Further improvement of the running time is possible by no longer considering the MWM subroutine as a black box. We pursue this direction in the next section. Our findings there yield the following theorem.

▶ **Theorem 4.** *An MCSI between two unrooted trees G and H can be computed in time $\mathcal{O}(|G||H|(\min\{\Delta(G), \Delta(H)\} + \log\max\{\Delta(G), \Delta(H)\}))$.*

Proof. The MWM computations in Algorithm 1 are dominating, thus we obtain the above running time directly from Theorem 7 of the following section. ◀

4 Computing All Maximum Weight Matchings

In this section we improve the total time bound for solving all the matching instances arising in Algorithm 1. First, we provide a time bound to compute an MWM in a single bipartite graph $(V \uplus U, E)$, where possibly $|V| \neq |U|$. In the following, we exploit the fact that during the run of our algorithm, we get sets of "similar" bipartite graph instances. After computing an MWM on one graph in one of the sets, we can derive MWMs for all the other bipartite graphs in that set very efficiently. Finally, we provide an upper bound to compute an MWM in all the occurring bipartite graphs.



(a) Input graph B. **(b)** Reduced graph B'. **(c)** MWMs M', M. **(d)** B'_4 with M'_4.

Figure 2 Weighted bipartite graph B (a); reduced graph B' with initial duals in green (vertices without label have dual value 0) and initial matching M'' in blue (b); MWM M' of B' in blue, M of B in thick blue c; B'_4 with matching M'_4 in blue (d). - cf. proofs of Lemma 5 and Lemma 6.

Computing an MWM is closely related to finding an MWPM and there is extensive literature on both problems [8]. Gabow and Tarjan [10] describe a reduction to solve the MWM problem using any algorithm for MWPM, without altering the algorithm's asymptotic time bound, which we will make use of. For computing an MWPM, we use the well known Hungarian method, which has at most n iterations in its outer loop and a total running time of $\mathcal{O}(n^3)$ or $\mathcal{O}(n(m + n \log n))$ using Fibonacci heaps, where n and m denote the number of vertices and edges of the bipartite graph. We denote this algorithm by $\mathcal{A}_{\text{PM}}$.

The Hungarian method is a primal-dual algorithm. It starts with an empty matching and computes a new matching with one more edge in each iteration, maintaining a feasible dual solution of a primal linear program. The complementary slackness theorem ensures, that the obtained perfect matching after n iterations is a MWPM. Note, by using the reduction in [10], we always have at least one perfect matching.

▶ **Lemma 5.** *Let $B = (V \uplus U, E)$ be a bipartite graph with edge weights $w : E \to \mathbb{R}$. Let $k := |V|$, $l := |U|$, and $k \leq l$. An MWM M on B can be computed in time $\mathcal{O}(kl(k + \log l))$.*

Proof. Let $\{v_1, \ldots, v_k\} = V, \{u_1, \ldots, u_l\} = U$ be the two vertex sets of B. First, we remove all edges from B with negative edge weight, because they never contribute to an MWM. Then, we add a copy B^{C} of B to the graph. For each vertex $v \in V \uplus U$ we denote its copy v^{C} and for each edge $e \in E$ we denote its copy e^{C}. We then copy the edge weights, i.e., $w(e^{\text{C}}) := w(e)$ for each edge $e \in E$. Next we insert a new edge of weight 0 between each vertex $v \in V \uplus U$ and its copy v^{C}. This graph is called *reduced graph* B'. Figures 2a and b show an example of B and B'. An MWPM M' of B' yields an MWM M of B: $vu \in M \Leftrightarrow vu \in M'$ and $v \in V, u \in U$. This follows from the construction of B'.

In the following, we prove an upper time bound to compute M'. An initial feasible dual solution $d : V_{B'} \to \mathbb{R}$, i.e., $d(v) + d(u) \geq w(vu)$ for all edges $vu \in E_{B'}$, including l matching edges vu with $d(v) + d(u) = w(vu)$, is computed as follows: We set $d(u) = 0$ for all $u \in U$ and $d(v) := \max\{w(vu) \mid u \in U\}$ for all $v \in V$. Next, for each $v \in V \uplus U$ the vertex v^{C} obtains the dual value $d(v^{\text{C}}) := d(v)$. We define an initial matching $M'' := \{uu^{\text{C}} \mid u \in U\}$. Note, $d(u) + d(u^{\text{C}}) = 0 = w(uu^{\text{C}})$.

The dual solution d is feasible and can be computed in time $\mathcal{O}(kl)$. Let $n := |V_{B'}| = 2(k+l) \in \Theta(l)$ and $m := |E_{B'}| \leq 2kl+k+l \in \mathcal{O}(kl)$. Increasing the number of matching edges by one using a single iteration of $\mathcal{A}_{\text{PM}}$ is possible in time $\mathcal{O}(m + n \log n) = \mathcal{O}(l(k + \log l))$. To obtain an MWPM M' form M'' in B' we need to increase the number of matching edges by k, therefore the time to compute M' and thus M is $\mathcal{O}(kl(k + \log l))$. ◀

Algorithm 2: Computing MWMs on B and B_j, cf. Lemmas 5, 6

Input : Bipartite graph $B = (V \uplus U, E), |U| \geq 2, U = \{u_1, u_2, \ldots\}$ with edge weights $w : E \to \mathbb{R}$

Output : MWMs M, M_j on $B, B_j := G[V \uplus U \setminus \{u_j\}]$ for each $j \in [1..|U|]$.

1 **if** $|V| \leq |U|$ **then** ▷ *Compute MWM M of B*

2 Let $B' := (V', E')$, where $V' := V \cup U \cup \{v^{\mathrm{C}} \mid v \in V \cup U\}$ and $E' := E \cup \{e^{\mathrm{C}} \mid e \in E\} \cup \{vv^{\mathrm{C}} \mid v \in V \cup U\}$.

3 $w(e^{\mathrm{C}}) \leftarrow w(e)$ for all $e \in E$ ▷ *Weights of additional edges*

4 $w(vv^{\mathrm{C}}) \leftarrow 0$ for all $v \in V \cup U$

5 $d(u^{\mathrm{C}}) \leftarrow d(u) \leftarrow 0$ for all $u \in U$ ▷ *Dual values*

6 $d(v^{\mathrm{C}}) \leftarrow d(v) \leftarrow \max\{w(vu) \mid u \in U\}$ for all $v \in V$

7 $M'' \leftarrow \{uu^{\mathrm{C}} \mid u \in U\}$ ▷ *Initial matching edges*

8 Starting with M'' and d, compute an MWPM M' on B' using $|V|$ iterations of $\mathcal{A}_{\mathrm{PM}}$

9 $M \leftarrow \{vu \mid vu \in M', v \in V, u \in U\}$

10 **else**

11 Exchange the vertices of V and U.

12 Compute M as in lines 2 to 9 and exchange V and U back.

13 $d \leftarrow$ The dual values obtained while computing M'.

14 **foreach** $j \in [1..|U|]$ **do** ▷ *MWMs M_j on B_j*

15 **if** u_j *is not matched by* M **then**

16 $M_j \leftarrow M$

17 **else**

18 $B'_j \leftarrow B' \setminus \{u_j, u_j^{\mathrm{C}}\}$

19 $M'_j \leftarrow M'$ without the matching edges incident to u_j, u_j^{C} ▷ *Initial matching*

20 Compute an MWPM M'_j on B'_j using d and a single iteration of $\mathcal{A}_{\mathrm{PM}}$.

21 $M_j \leftarrow \{vu \mid vu \in M'_j, v \in V, u \in U\}$

▶ **Lemma 6.** *Let $B = (V \uplus U, E)$ be a weighted bipartite graph with $k := |V|, U = \{u_1, \ldots, u_l\}, l \geq 2$. Let $B_j := G[V \uplus U \setminus \{u_j\}]$ for each $j \in [1..l]$. Computing MWMs for all graphs $B, B_1, \ldots, B_l$ is possible in total time $\mathcal{O}(kl(\min\{k,l\} + \log\max\{k,l\}))$.*

Proof. According to Lemma 5 we obtain an MWM M of B in time $\mathcal{O}(kl(\min\{k,l\} + \log\max\{k,l\}))$. We compute an MWM on each B_j as follows: Let d be an optimal dual solution obtained while computing M' (on B', see proof of Lemma 5). If u_j is not matched by M, i.e., $u_j \notin e$ for all $e \in M$, then $M_j := M$ is an MWM of B_j. Otherwise let B'_j be the reduced graph as explained in the proof of Lemma 5. We obtain a feasible dual solution d_j on the bipartite graph B'_j by taking the dual values from d, i.e., $d_j(v) := d(v)$ for all $v \in V(B'_j)$. Note, we have $2(k+l)$ vertices in B', and exactly two less in B'_j, i.e., a perfect matching in B'_j consists of $k + l - 1$ matching edges.

We can derive an initial matching M'_j on B'_j with $k+l-2$ edges from M'; M'_j contains the matching edges that are not incident to the two removed vertices from B' to B'_j. Therefore only one more iteration of $\mathcal{A}_{\mathrm{PM}}$ is needed, which is possible in time $\mathcal{O}(\max\{k,l\}(\min\{k,l\} + \log\max\{k,l\}))$, cf. proof of Lemma 5. We then obtain M_j from M'_j as previously described. The complementary slackness conditions ensure M'_j and therefore M_j is of maximum weight. An example of M' and B'_j ($j = 4$) is shown in Figures 2c and d.

We need to compute an MWM different from M for at most $\min\{k,l\}$ of the l graphs $B_1, \ldots, B_l$, because at most k vertices of U are matched by M, cf. Figure 2c: only u_3

and u_4 of U are matched by M. Therefore the time bound to compute MWMs for all the graphs $B_1, \ldots, B_l$ is $\mathcal{O}(\min\{k,l\} \max\{k,l\}(\min\{k,l\} + \log \max\{k,l\})) = \mathcal{O}(kl(\min\{k,l\} + \log \max\{k,l\}))$. ◀

We call B and the graphs $B_j, j \in [1..l]$, a set of "similar" bipartite graph instances. Algorithm 2 shows how we compute an MWM for each graph in this set. Next, we apply Lemma 6 to Algorithm 1. For each pair $u \in V_G, v \in V_H$ of vertices, selected in line 2 and 4, respectively, the algorithm computes up to $|N(v)| + 1$ MWMs, cf. lines 5, 10. A close look at Algorithm 1 reveals this as a set of "similar" bipartite graph instances. The first graph B is obtained by selecting $s = v$ in line 5. The other graphs B_j are obtained by selecting all the vertices $s \in N(v)$. This observation allows to prove the following theorem.

▶ **Theorem 7.** *All the MWMs in Algorithm 1 can be computed in total time* $\mathcal{O}(kl(\min\{\Delta(G), \Delta(H)\} + \log \max\{\Delta(G), \Delta(H)\}))$*, where* $k = |G|$ *and* $l = |H|$*.*

Proof. For each pair $(v,u) \in V_G \times V_H$ we compute an MWM on each of the "similar" graphs, where $B = (C(v) \uplus N(u), E)$ and edge weights as determined by Eq. (1). Let $d_{\min} := \min\{\Delta(G), \Delta(H)\}$ and $d_{\max} := \max\{\Delta(G), \Delta(H)\}$. For all the pairs (v,u) we obtain a time complexity of

$$
\begin{aligned}
&\mathcal{O}\left(\sum_v \sum_u \delta(v)\delta(u)(\min\{\delta(v), \delta(u)\} + \log \max\{\delta(v), \delta(u)\})\right) \\
\subseteq\ &\mathcal{O}\left(\sum_v \delta(v) \sum_u \delta(u)(d_{\min} + \log d_{\max})\right) \\
=\ &\mathcal{O}\left((d_{\min} + \log d_{\max}) \sum_v \delta(v) l\right) \\
=\ &\mathcal{O}((d_{\min} + \log d_{\max})kl).
\end{aligned}
$$

◀

5 Lower Bounds on the Time Complexity and Optimality

Providing a tight lower bound on the time complexity of a problem is generally a non-trivial task. We obtain this for trees of bounded degree and reason why the existence of an algorithm with subcubic running time for unrestricted trees is unlikely. In order to solve MCST with an arbitrary weight function ω for two trees G and H, all values $\omega(u,v)$ for $u \in V(G)$ and $v \in V(H)$ must be considered. This directly leads the lower bound of $\Omega(|G||H|)$ for the time complexity of MCST. For trees of bounded degree our approach achieves running time $\mathcal{O}(|G||H|)$ according to Theorem 4 and, thus, has an optimal worst-case running time in the considered setting.

For unrestricted trees of order n our approach has a running time of $\mathcal{O}(n^3)$ according to Theorem 4. In the next paragraph we present a linear time reduction from the assignment problem to MCST, which preserves the time complexity. Therefore solving MCST in time $o(n^3)$ yields an algorithm to solve the assignment problem in time $o(n^3)$. The Hungarian method solves the assignment problem in $\mathcal{O}(n^3)$, which is the best known time bound for bipartite graphs with $\Theta(n^2)$ edges of unrestricted weight for more than 30 years.

Let $B = (U \uplus V, E, w)$ be a weighted bipartite graph on which we want to solve the assignment problem, i.e., to find an MWPM. We assume weights to be non-negative, which can be achieved by adding a sufficiently large constant to every edge weight to obtain an

assignment problem that is equivalent w.r.t. the MWPMs. We construct a star graph G with center c and leaves U and another star graph H with center c' and leaves V. Let $n = |U| = |V|$ and $N = \max_{e \in E} w(e)$. We define ω such that $\omega(u, v) = w(uv) + nN$ for all $uv \in E$, $\omega(c, c') = nN$ and $\omega(u, v) = -\infty$ for all other pairs of vertices. For all pairs of edges we define $\omega(e, e') = 0$. Let ϕ be an MCSI between G and H w.r.t. w and $p := |\operatorname{dom}(\phi)|$. It directly follows from the construction that $M := \{uv \in E \mid \phi(u) = v\}$ is an MWM in B with $W(M) = \mathcal{W}(\phi) - pnN$. Furthermore, the incremented weights ensure that M is perfect, i.e., $p = n + 1$, whenever B admits a perfect matching. Therefore we obtain:

▶ **Proposition 8.** *Only if we can solve the assignment problem on a graph with n vertices and $\Theta(n^2)$ edges of unrestricted weight in time $o(n^3)$, we can solve MCST on two unrooted trees of order $\Theta(n)$ in time $o(n^3)$.*

6 Output-Sensitive Algorithms for Listing All Solutions

Algorithm 1 can easily be modified to not only output the weight of an MCSI, but also an associated isomorphism. Let $D(u, s, v)$ be a maximum entry in D. Then $\phi(u) = v$. Further mappings are defined by the matching edges occurring in Eq. (1). In the example of Figure 1d we obtain $\phi(c_1) = d_1$ and $\phi(c_2) = d_3$. Since in general there is no single unique MCSI, it is of interest to find and list all of them. In this section we show how our techniques can be combined with the enumeration algorithm from [7], which lists all the different MCSIs of two trees exactly once. We obtain the best known time bound for listing all solutions by an improved analysis.

Since the number of MCSIs is not polynomially bounded in the size of the input trees, we cannot expect polynomial running time. An algorithm is said to be *output-sensitive* if its running time depends on the size of the output in addition to the size of the input.

The basic idea to enumerate all MCSIs is to first compute the weight of an MCSI. Then for each maximum table entry $D(u, v, v), u \in V_G, v \in V_H$, all the different rooted MCSIs on the rooted subtrees G_u^r, H_v^v are listed. Note, we omit maximum table entries $D(u, s, v)$, where $s \neq v$. We do this, because every MCSI of G_u^r, H_v^s is also an MCSI of G_u^r, H_v^v. As an example let u be the root of G in Figure 1. Then $D(u, v, v) = D(u, s, v) = 7$. For both table entries we obtain the same MCSI ϕ with $\phi(u) = v, \phi(r) = d_1, \phi(c_1) = d_2, \phi(c_2) = d_3, \ldots$.

We enumerate the MCSIs on a pair of rooted subtrees by enumerating all MWMs of the associated bipartite graphs of Eq. (1) and then expanding ϕ recursively along all the different MWMs of the mapped children. For the problem depicted in Figure 1c there are two different MWMs: $M_1 = \{c_1d_1, c_2d_3\}$ and $M_2 = \{c_1d_2, c_2d_3\}$. Therefore we first expand along M_1 as explained in the first paragraph of this section and then along M_2. We do this recursively for each occurring matching instance. The enumerated isomorphisms of each maximum entry are pairwise different, based on the different MWMs. They are also pairwise different between two different maximum entries. The proof of the latter claim is similar to the proof of Lemma 2. Thus we do not enumerate an MCSI twice. Further we do not omit an MCSI, because we consider all necessary maximum table entries and their rooted subtrees, as well as all possible expansions along the MWMs.

Note, the enumeration algorithm of [7] uses a somewhat different table to store maximum solutions. The basic idea to list all solutions is the same. For trees of sizes $k := |G|$ and $l := |H|$, $k \leq l$, their enumeration algorithm requires total time $\mathcal{O}(k^2l^4 + Tl^2)$, where T is the number of different MCSIs. The $\mathcal{O}(k^2l^4)$ term of the running time is caused by computing the weight of an MCSI in time $\mathcal{O}(kl^4)$ and repeated deletions of single edges in one tree and recalculations of the weight of an MCSI to avoid outputting an MCSI twice. We have

improved the time bound to compute the weight of an MCSI, cf. Theorem 4. Therefore we can improve the $\mathcal{O}(k^2l^4)$ term to $\mathcal{O}(kl(\min\{\Delta(G),\Delta(H)\}+\log\max\{\Delta(G),\Delta(H)\}))$.

The $\mathcal{O}(Tl^2)$ term in the original running time is caused by the enumeration of MWMs. For each MCSI ϕ several MWMs have to be enumerated, let this number be m_ϕ. The time to do this can be bounded by $\mathcal{O}(l^2)$, when using a variant of the enumeration algorithm for perfect matchings presented in [21]. The running time follows from the fact, that for each MWM two depth first searches (DFS) in a directed subgraph of B', cf. Figure 2, are computed. The running time of DFS is linear in the number of edges and vertices. Let k_i, l_i be the sizes of the disjoint vertex sets of the i-th bipartite graph, on which we enumerate the MWMs, $i \in [1..m_\phi]$. Then $\sum_i k_i \leq k$ and $\sum_i l_i \leq l$, because all the vertices in all the m_ϕ bipartite graphs are pairwise disjoint. The running time of DFS in the directed subgraphs of the i-th bipartite graph is $\mathcal{O}(k_il_i)$, cf. Figure 2b or d. For all m_ϕ DFS runs we have $\sum_i k_il_i \leq \sum_i k_i\Delta(H) \leq k\Delta(H)$ as well as $\sum_i k_il_i \leq \sum_i \Delta(G)l_i \leq \Delta(G)l$. Hence, the time to enumerate ϕ is bounded by $\mathcal{O}(\min\{k\Delta(H),\Delta(G)l\})$.

Both improvements combined together, the initial computation of the weight of an MCSI and the MWM enumeration, improve the enumeration time from $\mathcal{O}(n^6+Tn^2)$ to $\mathcal{O}(n^3+Tn\min\{\Delta(G),\Delta(H)\})$. More precisely we obtain the following theorem.

▶ **Theorem 9.** *Enumerating all MCSIs of two unrooted trees G and H is possible in time $\mathcal{O}(\,|G|\,|H|\,(\,\min\{\Delta(G),\Delta(H)\}\,+\,\log\max\{\Delta(G),\Delta(H)\}\,)+T(\,\min\{|G|\Delta(H),\Delta(G)|H|\}\,)\,)$, where T is the number of different MCSIs.*

7 Experimental Comparison

In this section we experimentally evaluate the running time of our approach (DKM) on synthetic and real-world instances. We compare our algorithm to the approach of [18] which also solves MCST when the input graphs are trees. The corrected analysis of the approach yields a running time of $\mathcal{O}(n^4)$, which aligns better with our experimental findings of $\Omega(n^5)$. The implementation was provided by the authors as part of the FOG package.[2] Both algorithms were implemented in C++ and compiled with GCC v.4.8.4. Running times were measured on an Intel Core i7-3770 CPU with 16 GB of RAM using a single core only. We generated random trees by iteratively adding edges to a randomly chosen vertex and averaged over 40 to 100 pairs of instances depending on their size. The weight function ω was set to 1 for each pair of vertices and edges, i.e., we compute isomorphisms of maximum size. This matches the setting in FOG.

Table 1 summarizes our results and we observe that the running time of our approach aligns with our theoretical analysis. In comparison, FOG's running time is much higher and increases to a larger extent with the input size. The running times of both algorithms show a low standard deviation for random trees, cf. Tables 1a, b. Table 1c shows the running time in star graphs, which are worst-case examples for some approaches, see Sec. 3. Our theoretical proven cubic running time matches the experimental results, while FOG's running time increases drastically. Table 1d summarizes the computation time under different weight functions. We defined ω such that different labels are simulated, i.e., vertices and edges with different labels have weight $-\infty$, which again matches FOG's setting. Both algorithms clearly benefit from the fact that less MWMs have to be computed. The results on random trees are also shown in Figure 3.

[2] https://dtai.cs.kuleuven.be/software/PMCSFG

Table 1 Average running time in ms ± RSD in % and speedup factor $q := \text{FOG/DKM}$.

(a) Random trees of the same order.

Order	DKM	FOG	q
20	$0.9 \pm 8\%$	$40 \pm 7\%$	44.1
40	$3.5 \pm 6\%$	$221 \pm 5\%$	62.7
80	$15.2 \pm 4\%$	$1\,286 \pm 5\%$	84.8
160	$58.9 \pm 3\%$	$8\,342 \pm 5\%$	141.7
320	$237.4 \pm 2\%$	$63\,327 \pm 8\%$	266.9

(b) Random trees with $|G| = 80$ fixed.

$\lvert H\rvert$	DKM	FOG	q
20	$3.6 \pm 8\%$	$192 \pm 4\%$	53.6
40	$7.3 \pm 7\%$	$504 \pm 4\%$	68.7
80	$15.2 \pm 4\%$	$1\,286 \pm 5\%$	84.8
160	$30 \pm 9\%$	$3080 \pm 4\%$	103.3
320	$59.5 \pm 3\%$	$6842 \pm 4\%$	114.9

(c) Star graphs.

Order	DKM	FOG	q
10	0.1	18	117.6
20	1	489	458.5
40	8.9	18 722	2109.9
80	77.5	929 784	11 992.1

(d) Different ω-functions, order 80

#labels	DKM	FOG	q
1	$15.2 \pm 4\%$	$1\,286 \pm 5\%$	84.8
2	$5.4 \pm 8\%$	$217 \pm 8\%$	40
3	$3.3 \pm 7\%$	$118 \pm 12\%$	36.1
4	$2.6 \pm 8\%$	$83 \pm 9\%$	31.9

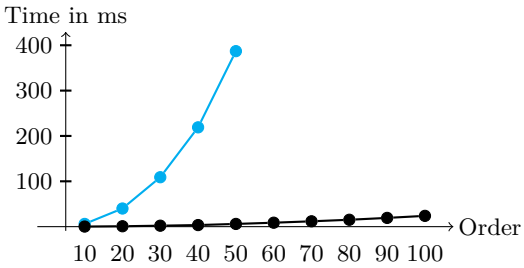


Figure 3 Average running time in ms (y-axis) for MCSI computation on random trees of order n (x-axis). Black = Our implementation (DKM). Blue = FOG implementation.

From a chemical database of thousands of molecules[3] we extracted 100 pairs of graphs with block-cut trees (BC-trees) consisting of more than 40 vertices. BC-trees are a representation of graphs, where each maximal biconnected component is represented by a B-vertex. If two such components share a vertex, the corresponding B-vertices are connected through a C-vertex representing this shared vertex. The running time for MCST on BC-trees is an important factor for the total running time of MCS algorithms for outerplanar and series-parallel graphs like [2, 13, 18]. The average running time of our algorithm was 11.2 ms, compared to FOG's 481.3 ms. The speedup factor ranges from 24 to 59, with an average of 43. This indicates that the above mentioned approaches could greatly benefit from the techniques presented in this paper.

8 Conclusions

We have presented a novel algorithm for MCST which (i) considers only the subproblems required to guarantee that an optimal solution is found and (ii) solves groups of related matching instances efficiently in one pass. Rigorous analysis shows that the approach achieves cubic time in general trees and quadratic time in trees of bounded degree. Our analysis of the

[3] NCI Open Database, GI50, `http://cactus.nci.nih.gov`

problem complexity reveals that there is only little room for possible further improvements. The practical efficiency is documented by an experimental comparison.

If the weight function is restricted to integers of a bounded value, scaling approaches [8] to the corresponding matching problems become applicable. It remains future work to improve the running time for this case.

References

1 Amir Abboud, Arturs Backurs, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Or Zamir. Subtree isomorphism revisited. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 1256–1271. SIAM, 2016.

2 Tatsuya Akutsu and Takeyuki Tamura. A polynomial-time algorithm for computing the maximum common connected edge subgraph of outerplanar graphs of bounded degree. *Algorithms*, 6(1):119–135, 2013. doi:10.3390/a6010119.

3 Tatsuya Akutsu, Takeyuki Tamura, Avraham A. Melkman, and Atsuhiro Takasu. On the complexity of finding a largest common subtree of bounded degree. In Leszek Gasieniec and Frank Wolter, editors, *Fundamentals of Computation Theory*, volume 8070 of *LNCS*, pages 4–15. Springer, 2013. doi:10.1007/978-3-642-40164-0_4.

4 Moon Jung Chung. $O(n^{2.5})$ time algorithms for the subgraph homeomorphism problem on trees. *Journal of Algorithms*, 8(1):106–112, 1987. doi:10.1016/0196-6774(87)90030-7.

5 Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 2004. doi:10.1142/S0218001404003228.

6 Erik D. Demaine, Shay Mozes, Benjamin Rossman, and Oren Weimann. An optimal decomposition algorithm for tree edit distance. *ACM Trans. Algorithms*, 6(1):2:1–2:19, December 2009. doi:10.1145/1644015.1644017.

7 Andre Droschinsky, Bernhard Heinemann, Nils Kriege, and Petra Mutzel. Enumeration of maximum common subtree isomorphisms with polynomial-delay. In Hee-Kap Ahn and Chan-Su Shin, editors, *Algorithms and Computation (ISAAC)*, LNCS, pages 81–93. Springer, 2014. doi:10.1007/978-3-319-13075-0_7.

8 Ran Duan and Hsin-Hao Su. A scaling algorithm for maximum weight matching in bipartite graphs. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 1413–1424. SIAM, 2012.

9 Hans-Christian Ehrlich and Matthias Rarey. Maximum common subgraph isomorphism algorithms and their applications in molecular science: a review. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 1(1):68–79, 2011. doi:10.1002/wcms.5.

10 Harold N. Gabow and Robert Endre Tarjan. Faster scaling algorithms for network problems. *SIAM J. Comput.*, 18(5):1013–1036, 1987.

11 Arvind Gupta and Naomi Nishimura. Finding largest subtrees and smallest supertrees. *Algorithmica*, 21:183–210, 1998. doi:10.1007/PL00009212.

12 Ming-Yang Kao, Tak-Wah Lam, Wing-Kin Sung, and Hing-Fung Ting. An even faster and more unifying algorithm for comparing trees via unbalanced bipartite matchings. *Journal of Algorithms*, 40(2):212–233, 2001. doi:10.1006/jagm.2001.1163.

13 Nils Kriege, Florian Kurpicz, and Petra Mutzel. On maximum common subgraph problems in series-parallel graphs. In Kratochvíl Jan, Mirka Miller, and Dalibor Froncek, editors, *IWOCA 2014*, volume 8986 of *LNCS*, pages 200–212. Springer, 2014. doi:10.1007/978-3-319-19315-1_18.

14 Nils Kriege and Petra Mutzel. Finding maximum common biconnected subgraphs in series-parallel graphs. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, edit-

ors, *MFCS 2014*, volume 8635 of *LNCS*, pages 505–516. Springer, 2014. doi:10.1007/978-3-662-44465-8_43.

15 David W. Matula. Subtree isomorphism in $O(n^{5/2})$. In P. Hell B. Alspach and D.J. Miller, editors, *Algorithmic Aspects of Combinatorics*, volume 2 of *Annals of Discrete Mathematics*, pages 91–106. Elsevier, 1978. doi:10.1016/S0167-5060(08)70324-8.

16 Matthias Rarey and J.Scott Dixon. Feature trees: A new molecular similarity measure based on tree matching. *Journal of Computer-Aided Molecular Design*, 12(5):471–490, 1998. doi:10.1023/A:1008068904628.

17 Steven W. Reyner. An analysis of a good algorithm for the subtree problem. *SIAM J. Comput.*, 6(4):730–732, 1977.

18 Leander Schietgat, Jan Ramon, and Maurice Bruynooghe. A polynomial-time maximum common subgraph algorithm for outerplanar graphs and its application to chemoinformatics. *Annals of Mathematics and Artificial Intelligence*, 69(4):343–376, 2013. doi:10.1007/s10472-013-9335-0.

19 Ron Shamir and Dekel Tsur. Faster subtree isomorphism. *Journal of Algorithms*, 33(2):267–280, 1999. doi:10.1006/jagm.1999.1044.

20 A. Torsello, D. Hidovic-Rowe, and M. Pelillo. Polynomial-time metrics for attributed trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1087–1099, July 2005. doi:10.1109/TPAMI.2005.146.

21 Takeaki Uno. Algorithms for enumerating all perfect, maximum and maximal matchings in bipartite graphs. In *Algorithms and Computation (ISAAC)*, volume 1350 of *LNCS*, pages 92–101. Springer, 1997.

22 Gabriel Valiente. *Algorithms on Trees and Graphs*. Springer-Verlag, Berlin, 2002.

23 Rakesh M. Verma and Steven W. Reyner. An analysis of a good algorithm for the subtree problem, corrected. *SIAM J. Comput.*, 18(5):906–908, 1989.