Faster Subtree Isomorphism

Ron Shamir Dekel Tsur
Tel-Aviv University
Dept. of Computer Science
Tel Aviv 69978, Israel
{shamir,dekel}@math.tau.ac.il

Abstract

We study the subtree isomorphism problem: Given trees H and G, find a subtree of G which is isomorphic to H or decide that there is no such subtree. We give an $O(\frac{k^{1.5}}{\log k}n)$-time algorithm for this problem, where k and n are the number of vertices in H and G respectively. This improves over the $O(k^{1.5}n)$ algorithms of Chung and Matula. We also give a randomized (Las Vegas) $O(\min(k^{1.45}n, kn^{1.43}))$-time algorithm for the decision problem.

1. Introduction

A fundamental problem in graph theory is the *subgraph isomorphism problem*: Given two graphs G and H, find a subgraph of G which is isomorphic to H (if there is one). This problem is NP-hard as it includes for example the clique problem.

When restricting the graph G, the subgraph isomorphism problem may become easier. Polynomial-time algorithms for subgraph isomorphism problem were given for trees [16], two-connected outerplanar graphs [12], and two-connected series-parallel graphs [13]. All these graphs have a treewidth at most 2. The subgraph isomorphism problem is NP-hard when the graph G is an arbitrary graph with treewidth at most 2 [15]. However, the problem is polynomial for the family of graphs with treewidth at most p for every $p \geq 2$, if we also add the restriction that G is p-connected, or that H has bounded degree [15]. A faster algorithm for the former case was given in [4].

In this paper we study *subtree isomorphism*, i.e., the subgraph isomorphism problem when G and H are trees. Throughout this work n and k denote the number of vertices in G and H, respectively. When $k = n$ we get the *tree isomorphism problem*, which has a linear time algorithm due to Hopcroft and Tarjan [10]. In contrast, the problem is NP-complete when G is a tree and H is a forest (*subforest isomorphism* [7]). Polynomial algorithms for subtree isomorphism were first given by Matula [16] and by Edmonds (cf. [17]). Faster algorithms, with worst case time complexity of $O(k^{1.5}n)$, were given independently by Matula [17] and Chung [2].

Our main result here is an $O(\frac{k^{1.5}}{\log k}n)$-time algorithm. Our algorithm, like most previous studies of the problem, is based on the close relationship between subtree isomorphism and maximum matching in bipartite graphs. (This relationship was also utilized to obtain fast parallel algorithms for subtree isomorphism [8, 11].) The subtree isomorphism problem is recursively translated into a collection of smaller subtree isomorphism problems, that are solved using maximum matching algorithms. The improved complexity is achieved by a combinatorial lemma that bounds the possible number of distinct subtrees involved, and by using the notion of clique partition and its application by Feder and Motwani to finding maximum matching in bipartite graphs [6]. We show that for the matching problem resulting from the subtree isomorphism problem, we can find a good partition in a simple way.

We also give a randomized (Las Vegas) $O(\min(k^{1.45}n, kn^{1.43}))$-time algorithm for the decision problem. This algorithm follows directly from a randomized $O(n^{2.38})$ algorithm for the cardinality of a maximum matching given by Cheriyan [1].

The rest of the paper is organized as follows: In section 2 we prove some simple lemmas on matching that are needed later. Section 3 describes an $O(k^{1.5}n)$ algorithm for subtree isomorphism, and section 4 proves the combinatorial lemma and uses it together with the clique partition in order to improve the running time of the algorithm. Finally, section 5 describes the randomized algorithm. For lack of space, some of the proofs are omitted in this abstract.

Before we finish this section we give some basic definitions. A *rooted tree* is a triplet $G = (V, E, r)$ where (V, E) is an unrooted tree, and r is some vertex in V which is called the *root*. We will sometime write G^r to denote that r is the root of the rooted tree G. Also, for an unrooted tree G, we denote by G^r the rooted tree formed by choosing the vertex r to be its root. We denote by G_v^r the rooted subtree of G^r whose vertices are all descendants of v, and its root is v.

We say that two rooted trees G^r and $H^{r'}$ are *isomorphic* if there is an isomorphism between G and H which maps r to r'. Likewise, we write $H^{r'} \subseteq_R G^r$ if there is a rooted subtree G_1^r of G^r which is isomorphic to $H^{r'}$ (note that the subtree G_1^r must have the same root as G^r).

We define the *open neighborhood* of a vertex v in graph G by $N(v) = \{u : uv \in E\}$, and the *closed neighborhood* by $N[v] = N(v) \cup \{v\}$.

2. Matching

In this section, we give several lemmas about matchings which we will need later. For the following lemmas, let $B = (X, Y, E)$ be a bipartite graph, where $X = \{x_1, x_2, \ldots, x_s\}, Y = \{y_1, y_2, \ldots, y_t\}$ and $s \leq t$. Denote $X_0 = X$ and $X_i = X - \{x_i\}$ for $1 \leq i \leq s$. Denote by m_i the size of a maximum matching between X_i and Y. Clearly for every i, either $m_i = m_0$ or $m_i = m_0 - 1$.

Lemma 2.1 *The problem of finding a maximum matching in B can be reduced in $O(s^2)$ time to the problem of finding a maximum matching in a subgraph of B with $O(s^2)$ vertices and edges and with maximum degree at most s.*

Proof: Let X' be all the vertices of X with degree less than s. Let B' the subgraph induced from B by the vertices of X' and their neighbors. Building B' takes $O(s^2)$ time. We claim that given a maximum matching M' of B' we can build a maximum matching M of B in $O(s^2)$ time: First, add all the edges of M' to M. For each vertex $x \in X - X'$ find an unmatched neighbor $y \in Y$ (which must exist since each such x has at least $s - |M'|$ unmatched neighbors) and add xy to M. ∎

Combining lemma 2.1 with the algorithm of Hopcroft and Karp [9] we obtain

Corollary 2.2 *A maximum matching in B can be found in $O(s^{2.5})$ time.*

This improves over a direct application of [9] which would give $O(ts^{1.5})$.

An important definition in matching theory is of critical vertices (see e.g. [14]): A vertex x in a graph G is *critical* if the size of a maximum matching in $G - x$ is strictly less than the size of a maximum matching in G. Here, x_i is critical means that $m_i = m_0 - 1$. Let M be a maximum matching of B. We define the directed graph B_M by $B_M = (X \cup Y, E_M)$ where

$$E_M = \{(x,y) : xy \in E - M, x \in X, y \in Y\} \cup \{(y,x) : xy \in M, x \in X, y \in Y\}.$$

We denote by X_M all the vertices from X which are unmatched in M.

Lemma 2.3 *For any maximum matching M of B and any vertex $x_i \in X$, x_i is critical if and only if x_i is matched in M, and there is no directed path in B_M from a vertex in X_M to x_i.*

Proof: omitted. ∎

Corollary 2.4 *Given a maximum matching M of B, we can compute the value of m_i for $0 \leq i \leq s$ in $O(st)$ time.*

3. An $O(k^{1.5}n)$ algorithm

In this section we describe an $O(k^{1.5}n)$ algorithm for the subtree isomorphism problem that will be the basis for the improved algorithms in sections 4,5. It is based on Chung's algorithm [2] and has the same asymptotic time complexity, but is simpler as the tree G is traversed only once, compared to twice in Chung's algorithm. We briefly describe the algorithm for completeness. We note that the improvements of sections 4,5 can also be applied to Chung's original algorithm. For simplicity, we describe an algorithm for the decision problem. It can be extended easily to an algorithm for the search problem.

Let $G = (V, E), H = (V_H, E_H)$ be the input trees. Select a vertex r of G to be the root. We wish to know for each $v \in V, u \in H$ whether $H^u \subseteq_R G_v^r$. In order to compute this efficiently we will also need to know for each neighbor w of u if $H_u^w \subseteq_R G_v^r$ (notice that H_u^w is the graph formed by removing H_w^u from H^u). This information is relevant because $H^w \subseteq_R G_v^r$ iff for every child u_i of w there is a distinct child v_j of v such that $H_{u_i}^w \subseteq_R G_{v_j}^r$. We store this information in sets $S(v,u)$ defined as follows: for every $v \in V$, and for every $u \in V_H$

$$S(v,u) = \{w \in N[u] : H_u^w \subseteq_R G_v^r\}.$$

See figure 1 for an example for this definition. Notice that $u \in S(v,u)$ if and only if $H^u = H_u^u \subseteq_R G_v^r$. If v is a leaf of G^r then $S(v,u)$ can be computed for all u as follows: If u is a leaf of H, then $S(v,u)$ consists of one vertex which is the single neighbor of u in H. Otherwise, if u is an internal vertex, $S(v,u)$ is empty. If v is an internal vertex, we first need to compute $S(v_i, u)$ for all the children v_i of v, for all $u \in V_H$. Then we can use the following simple lemma.

Lemma 3.1 *Let v be a vertex of G^r with children $v_1, v_2, \ldots, v_t$. Let u be a vertex of H. Let $w \in N[u]$ and let $u_1, u_2, \ldots, u_s$ be the children of u in H_u^w. Then $w \in S(v,u)$ iff there are distinct indices $i_1, \ldots, i_s$, such that $u \in S(v_{i_p}, u_p)$ for all p $(1 \leq p \leq s)$.*

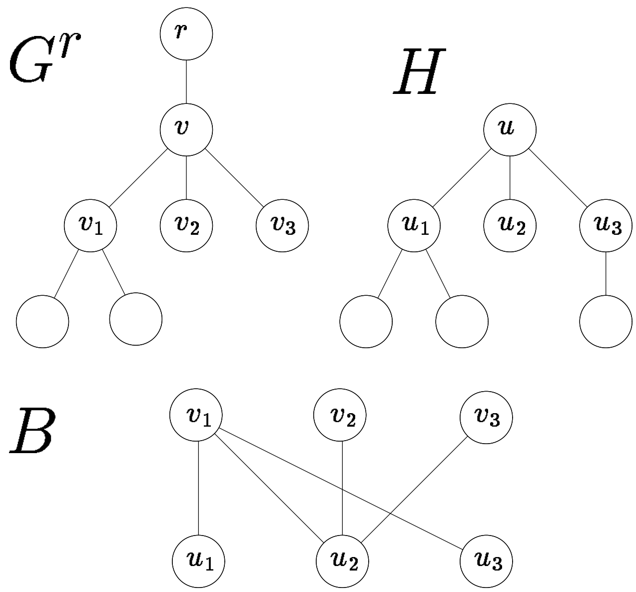

Figure 1. An instance of subtree isomorphism. Here, we have $H^u, H_u^{u_2} \not\subseteq_R G_v^r$ and $H_u^{u_1}, H_u^{u_3} \subseteq_R G_v^r$, so $S(v,u) = \{u_1, u_3\}$. The graph B is the bipartite graph which we construct in order to compute $S(v,u)$. There is an edge u_iv_j in this graph iff $u \in S(v_j, u_i)$. $H^u \not\subseteq_R G_v^r$ as B does not contain a matching of size 3. $H_{u_1}^u \subseteq_R G_v^r$ as $B - u_1$ contains a matching of size 2.

Thus, in order to decide if $w \in S(v,u)$ we construct a bipartite graph B with the two parts X and Y, where X is the set of children of u in H_u^w, Y is the set of children of v, and u_iv_j is an edge of B iff $u \in S(v_j, u_i)$. By lemma 3.1, w is in $S(v,u)$ iff B has a matching of size $|X|$. By lemma 2.3, in order to compute $S(v,u)$, it suffices to construct a single graph MATCH(v,u) and compute the critical vertices and the size of a maximum matching in this graph.

We can now describe the algorithm Subtree-Isomorphism. Its code is given in figure 2.

Theorem 3.2 *Algorithm Subtree-Isomorphism solves the subtree isomorphism problem in $O(k^{1.5}n)$ time.*

The correctness follows from lemma 3.1. The proof of the complexity uses corollary 2.4, Dinic's algorithm [5] and some simple data structures. We omit the proof here.

We note that the above algorithm can be changed to solve the *subtree homeomorphism* problem without changing the asymptotic complexity. The same modification applies to the algorithms in the sections below.

Select a vertex r of G to be the root of G.
for all $u \in H, v \in G$ **do** $S(v,u) \leftarrow \phi$.
for all leaves v of G^r **do**
  for all leaves u of H **do** $S(v,u) \leftarrow N(u)$.
for all internal vertices v of G^r in a postorder **do**
  Let $v_1, v_2, \ldots, v_t$ be the children of v.
  for all vertices $u = u_0$ of H with degree $\leq t+1$ **do**
    Let $u_1, u_2, \ldots, u_s$ be the neighbors of u.
    Construct a bipartite graph
      MATCH$(v,u) = (X, Y, E_{vu})$
      where $X = \{u_1, \ldots, u_s\}$, $Y = \{v_1, \ldots, v_t\}$,
      and $E_{vu} = \{u_iv_j : u \in S(v_j, u_i)\}$.
    Denote $X_0 = X$ and $X_i = X - \{u_i\}$.
    Compute the size m_i of a maximum matching
      between X_i and Y for $0 \leq i \leq s$.
    $S(v,u) \leftarrow \{u_i : m_i = |X_i|, 0 \leq i \leq s\}$.
    if $u \in S(v,u)$ **then** answer YES and **stop**
  end for
end for
Answer NO.

Figure 2. Algorithm Subtree-Isomorphism(G, H)

4. An $O(\frac{k^{1.5}}{\log k}n)$ algorithm

We now improve the algorithm from the last section by a $\log k$ factor. We use the previous algorithm but we solve the maximum matching problems more efficiently using the idea of clique partition of a bipartite graph and its usage in finding maximum matching [6]. The algorithm of Feder and Motwani, originally stated for bipartite graphs with equal size parts, can be extended to general bipartite graphs. This allows to give an algorithm for subtree isomorphism whose time complexity is $O(k\frac{n^{1.5}}{\log n})$. In contrast with [6] where the denseness of the graph is exploited, we achieve the reduction in time complexity by utilizing the special structure of the matching problems that must be solved in the subtree isomorphism algorithm. We find a good partition in a simpler way, and this results in an $O(\frac{k^{1.5}}{\log k}n)$ algorithm for subtree isomorphism.

Before we describe the improved algorithm, we need a simple combinatorial lemma. Let $g(n)$ denote the number of distinct (i.e. non-isomorphic) rooted trees with n vertices. We shall use the following result:

Lemma 4.1 *(see e.g. [18, p. 1197])* $g(n) = 2^{\Theta(n)}$.

Let $f(n)$ denote the maximum number of distinct rooted trees in a forest of n nodes.

Lemma 4.2 $f(n) = \Theta(\frac{n}{\log n})$.

Proof: As it is easy to verify using lemma 4.1 that $f(n) = \Omega(\frac{n}{\log n})$, we will only show the upper bound.

If we have a forest of rooted trees and r_i is the number of trees with i vertices, then the number of distinct trees in this forest is at most $\sum_i \min(r_i, g(i))$. Hence,

$$f(n) \leq \max\{\sum_i \min(r_i, g(i)) : r_1, \ldots, r_n \in N, \sum_{i=1}^{n} ir_i \leq n\}$$

By lemma 4.1,

$$f(n) \leq \max\{\sum_i \min(r_i, c^i) : r_1, \ldots, r_n \in N, \sum_{i=1}^{n} ir_i \leq n\}$$

for some integer constant c. Let x be the minimum integer for which $\sum_{i=1}^{x} ic^i \geq n$. Let $r_1, \ldots, r_n$ be the integers that maximize $\sum_i \min(r_i, c^i)$ under the constraint $\sum_{i=1}^{n} ir_i \leq n$. We can assume that $r_i \leq c^i$ for all i, because if $r_j > c^j$ for some j, we can set $r_j = c^j$ and the value of $\sum_i \min(r_i, c^i)$ does not change. Now, suppose $r_j > 0$ for some $j > x$. This implies that there is a $k \leq x$ for which $r_k < c^k$ (because otherwise $\sum_{i=1}^{n} ir_i \geq \sum_{i=1}^{x} ir_i + jr_j = \sum_{i=1}^{x} ic^i + jr_j > n$, a contradiction). If we decrease r_j by one, and increase r_k by one, then the value of $\sum_i \min(r_i, c^i)$ does not change, and the constraint $\sum_{i=1}^{n} ir_i \leq n$ still holds (as $k < j$). We can repeat this process until $r_j = 0$ for all $j > x$ and therefore

$$f(n) \leq \sum_{i=1}^{n} \min(r_i, c^i) \leq \sum_{i=1}^{x} c^i = O(c^x).$$

The lemma follows from the fact that $x = \log_c n - \log_c \log_c n - O(1)$. ∎

We now describe a modified algorithm, called Improved-Subtree-Isomorphism: The algorithm is the same as the algorithm Subtree-Isomorphism with the exception that we solve the maximum matching problems differently. Let v be some vertex in G^r whose children are $v_1, v_2, \ldots, v_t$, and let u be a vertex in H whose neighbors are $u_1, u_2, \ldots, u_s$. We now consider finding a maximum matching in $B = \mathrm{MATCH}(v, u)$. Recall that $B = (X, Y, E)$ with $X = \{u_1, \ldots, u_s\}$ and $Y = \{v_1, \ldots, v_t\}$. We assume that $s \leq t + 1$ (because otherwise the algorithm doesn't find a maximum matching in B).

We first apply lemma 2.1 and build a subgraph B' of B. Like in [6], we will partition the edges of B' into complete bipartite graphs $C_1, C_2, \ldots, C_p$. We do the partition in the following way: First, we sort the vertices of X' where the key of a vertex u is $N(u)$. Afterwards, we split X' into sets of equal keys $X^1, X^2, \ldots, X^p$ (i.e., all the vertices in a set X^i have the same neighbors in B'). Now, for $1 \leq i \leq p$ we set C_i to be the subgraph induced by the vertices of X_i and all their neighbors. We now follow the method of [6] and build a network B^* whose vertices are $X' \cup Y' \cup \{c_1, \ldots, c_p, a, b\}$. The arcs are $E_1 \cup E_2$ where

$$E_1 = \{au_i : u_i \in X'\} \cup \{v_i b : v_i \in Y'\}$$

$$E_2 = \{u_i c_j : j \leq p, u_i \in C_j\} \cup \{c_j v_i : j \leq p, v_i \in C_j\}$$

All arcs have capacity 1. The source is a and the sink is b. We find a maximum (integral) flow in B^*, and construct from this flow a maximum matching in B' in the obvious way.

We will now analyze the time complexity of the algorithm described above. We denote by $D(u)$ the number of distinct trees in the forest $H^u_{u_1}, \ldots, H^u_{u_s}$.

Lemma 4.3 *Algorithm Improved-Subtree-Isomorphism finds the maximum matching in MATCH(v,u) in $O(st + ts^{0.5}D(u))$ time.*

Proof: Note that if for some i, j the rooted trees $H^u_{u_i}$ and $H^u_{u_j}$ are isomorphic, then in $B = \mathrm{MATCH}(v, u)$ the vertices u_i, u_j have exactly the same neighbors, and this remains true in B' (assuming u_i and u_j were not deleted in B'). Therefore $p \leq D(u)$.

The time for constructing B, B' and B^* is $O(st)$ (we sort the vertices of X' using radix-sort). We now bound the time for finding a maximum flow in B^*: The size of E_2 is at most $2sp$ as the number of arcs in E_2 incident on some vertex c_j is at most $|X'| + \deg_{B'}(u_i) \leq 2s$ where $u_i \in C_j$. Since B^* does not have isolated vertices, the size of E_1 is at most $|X'| + |E_2|$. Hence, the number of arcs in B^* is $O(sp)$. Now, Dinic's algorithm [5] performs $O(\sqrt{|X'| + |Y'|}) = O(\sqrt{t})$ stages (see [6]), and each stage is linear in the number of arcs. Hence, the total time is $O(t^{0.5}sp) = O(ts^{0.5}p) = O(ts^{0.5}D(u))$. ∎

We denote $V = \{x_1, x_2, \ldots, x_n\}$ and $V_H = \{y_1, y_2, \ldots, y_k\}$, where x_i has degree t_i, and y_i has degree s_i. By lemma 4.3, the time complexity of algorithm Improved-Subtree-Isomorphism is $O(\sum_i \sum_j (s_j t_i + t_i s_j^{0.5} D(y_j))) = O(kn + n \sum_j s_j^{0.5} D(y_j))$. We will now bound $\sum_j s_j^{0.5} D(y_j)$.

Let ϵ be some constant $0 < \epsilon < 1/3$. We call a vertex of H *heavy* if its degree is at least $2k^{1-\epsilon}$, and otherwise it is called *light*. If u is a heavy vertex and v is a neighbor of u such that H^u_v does not contain a heavy vertex, then we call every vertex in H^u_v a *private vertex* of v. We denote by l_i the number of the private vertices of a heavy vertex y_i.

Lemma 4.4 *For any heavy vertex y_j, $s_j \leq k^\epsilon + l_j$ and $D(y_j) \leq k^\epsilon + f(l_j)$.*

Proof: Each neighbor of y_j is either a private vertex of y_j or not. The number of private neighbors of y_j is at most the total number of private vertices of y_j which is l_j. For each non-private neighbor v of u, we can choose a heavy vertex

v' in H^v_u. As all vertices we chose are distinct, we have that the number of non-private neighbors of u is less than the number of heavy vertices which is at most k^ϵ. Hence, $s_j \le k^\epsilon + l_j$.

Let $u_1, \ldots, u_s$ be the neighbors of u. Among the trees $H^u_{u_1}, \ldots, H^u_{u_s}$ at most k^ϵ contain a heavy vertex. The rest of the trees constitute all the private vertices of u, and therefore they have a total of l_j vertices and there are at most $f(l_j)$ distinct trees among them. Hence, $D(y_j) \le k^\epsilon + f(l_j)$. ■

Lemma 4.5 $\sum_j s_j^{0.5} D(y_j) = O(\frac{k^{1.5}}{\log k})$.

Proof:

We split $\sum_{j=1}^k s_j^{0.5} D(y_j)$ into two sums. Summing over the light vertices of H, as $D(y_j) \le s_j$ we have

$$\sum_{j: y_j \text{ is light}} s_j^{0.5} D(y_j) \le \sum_{j: s_j \le 2k^{1-\epsilon}} s_j^{1.5}$$

and as $\sum_j s_j = 2k - 2$,

$$\sum_{j: s_j \le 2k^{1-\epsilon}} s_j^{1.5} \le \frac{2k-2}{2k^{1-\epsilon}} (2k^{1-\epsilon})^{1.5} \le 2^{1.5} k^{1.5-\epsilon/2}$$

Now, summing over the heavy vertices and using lemma 4.4

$$\sum_{j: y_j \text{ is heavy}} s_j^{0.5} D(y_j) \le \sum_{j: y_j \text{ is heavy}} (k^{\epsilon/2} + l_j^{0.5})(k^\epsilon + f(l_j))$$

Since $f(l_j) \le l_j \le k$ and since the number of heavy vertices is at most k^ϵ, we have

$$\le \sum_{j: y_j \text{ is heavy}} (k^{3\epsilon/2} + k^{0.5+\epsilon} + k^{1+\epsilon/2} + l_j^{0.5} f(l_j))$$
$$\le 3k^{1+3\epsilon/2} + \sum_{j: y_j \text{ is heavy}} l_j^{0.5} f(l_j)$$

and by lemma 4.2, the fact that $\sum_j l_j \le k$, and the fact that the function $h(x) = x^{1.5}/\log x$ is convex,

$$\le 3k^{1+3\epsilon/2} + \sum_{j: y_j \text{ is heavy}} c \frac{l_j^{1.5}}{\log l_j} \le 3k^{1+3\epsilon/2} + c \frac{k^{1.5}}{\log k}$$

■

We therefore proved the following theorem:

Theorem 4.6 *Algorithm Improved-Subtree-Isomorphism solves the subtree isomorphism problem in* $O(\frac{k^{1.5}}{\log k} n)$ *time.*

5. An $O(\min(k^{1.45} n, kn^{1.43}))$ algorithm

In this section we give a randomized algorithm for the decision problem of subtree isomorphism. The algorithm is more efficient asymptotically than the deterministic algorithm of the previous section. We use the following result:

Theorem 5.1 *(Cheriyan [1]) There is a randomized (Las Vegas) algorithm that finds all critical vertices and the size of a maximum matching of a graph in* $\tilde{O}(n^\omega)$ *expected time where* n *is the number of vertices of the graph (*ω *denotes the exponent of matrix multiplication).*

Again, the algorithm Randomized-Subtree-Isomorphism is based on the algorithm Subtree-Isomorphism but solving the maximum matching problems is done differently. Consider some vertex v in G^r whose children are $v_1, v_2, \ldots, v_t$, and some vertex u in H whose neighbors are $u_1, u_2, \ldots, u_s$. If $s \le t^{0.4\omega}$, then we use corollary 2.2 to find a maximum matching in MATCH(v, u) and then we compute the value of all m_i using corollary 2.4, and the time is $O(s^{2.5} + st)$. If $s > t^{0.4\omega}$, we use theorem 5.1 to compute the values of all m_i in $\tilde{O}(t^\omega)$ time. Hence, the time the algorithm spends on the pair v, u is $\tilde{O}(st + \min(t^\omega, s^{2.5}))$. In order to estimate the overall time complexity, let x, y be two numbers satisfying

$$x \ge 1, y \ge 1, \omega - 0.4\omega x \le y$$

Then $\min(t^\omega, s^{2.5}) = O(s^x t^y)$ and by summing over all vertices of G and H we obtain an $\tilde{O}(k^x n^y)$ bound on the running time of algorithm Randomized-Subtree-Isomorphism. The constraints above on x, y create a polyhedron with two vertices ($x = 2.5 - \frac{2.5}{\omega}, y = 1$ and $x = 1, y = 0.6\omega$) and for any fixed values of k and n, the minimum of $k^x n^y$ under the constraints is achieved in some vertex. Hence,

Theorem 5.2 *Algorithm Randomized-Subtree-Isomorphism solves the decision problem in* $\tilde{O}(\min(k^{2.5-\frac{2.5}{\omega}} n, kn^{0.6\omega}))$ *expected time.*

Since $\omega < 2.376$ [3], we have

Corollary 5.3 *Algorithm Randomized-Subtree-Isomorphism solves the decision problem in* $O(\min(k^{1.45} n, kn^{1.43}))$ *expected time.*

References

[1] J. Cheriyan. A las vegas $O(n^{2.38})$ algorithms for the cardinality of a maximum matching. In *Proc. fifth annual ACM-SIAM Symp. on Discrete Algorithms (SODA 94)*, pages 297–309. ACM Press, 1994.

[2] M. J. Chung. $O(n^{2.5})$ time algorithms for the subgraph homeomorphism problem on trees. *J. Algorithms*, 8:106–112, 1987.

[3] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symbolic Comp.*, 9:23–52, 1990.

[4] A. Dessmark, A. Lingas, and A. Proskurowski. Faster algorithms for subgraph isomorphism of k-connected partial k-trees. In *Proc. of the Fourth Annual European Symp. on Algorithms, (ESA 96)*, pages 501–513. Springer-Verlag Lecture Notes in Computer Science, 1996.

[5] E. A. Dinic. An algorithm for solution of a problem of maximum flow in a network with power estimation. *Soviet Math. Doklady*, 11:1277–1280, 1970.

[6] T. Feder and R. Motwani. Clique partitions, graph compression and speeding-up algorithms. In *Proc. 23rd STOC*, pages 123–133. ACM Press, 1991.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., San Francisco, 1979.

[8] P. B. Gibbons, R. M. Karp, G. L. Miller, and D. Soroker. Subtree isomorphism is in random NC. *Discrete Applied Mathematics*, 29:35–62, 1990.

[9] J. E. Hopcroft and R. M. Karp. A $n^{5/2}$ algorithm for maximum matching in bipartite graphs. *SIAM J. Computing*, 2:225–231, 1973.

[10] J. E. Hopcroft and R. E. Tarjan. Isomorphism of planar graphs. In *Raymond E. Miller and James W. Thatcher, Complexity of Computer Computations, Plenum Press*. 1972.

[11] M. Karpinski and A. Lingas. Subtree isomorphism is NC reducible to bipartite perfect matching. *Information Processing Letters*, 30(1):27–32, 1989.

[12] A. Lingas. Subgraph isomorphism for biconnected outerplanar graphs in cubic time. *Theoretical Computer Science*, 63:295–302, 1989.

[13] A. Lingas and M. M. Sysło. A polynomial-time algorithm for subgraph isomorphism of two-connected series-parallel graphs. In *Proc. 15th Int. Colloq. Automata, Languages and Programming*, volume 317, pages 394–409. Springer-Verlag Lecture Notes in Computer Science, 1988.

[14] L. Lovasz and M. D. Plummer. *Matching Theory*. North-Holland, Amsterdam, 1986.

[15] J. Matoušek and R. Thomas. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Math.*, 108:343–364, 1992.

[16] D. W. Matula. An algorithm for subtree identification. *SIAM Rev.*, 10:273–274, 1968.

[17] D. W. Matula. Subtree isomorphism in $O(n^{5/2})$. *Ann. Discrete Math.*, 2:91–106, 1978.

[18] A. M. Odlyzko. Asymptotic enumeration methods. In *Handbook of Combinatorics, edited by R. L. Graham, M. Grotschel, and L. Lovasz, Elsevier and The MIT Press, 1995*, volume 2. 1995.