ELSEVIER

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Discrete Algorithms 3 (2005) 431–447

JOURNAL OF DISCRETE ALGORITHMS

www.elsevier.com/locate/jda

Constrained tree inclusion

Gabriel Valiente

Department of Software, Technical University of Catalonia, E-08034 Barcelona, Spain

Available online 11 September 2004

Abstract

The tree matching problem is considered of given labeled trees P and T, determining if the pattern tree P can be obtained from the text tree T by deleting degree-one and degree-two nodes and, in the case of unordered trees, by also permuting siblings. The constrained tree inclusion problem is more sensitive to the structure of the pattern tree than the general tree inclusion problem. Further, it can be solved in polynomial time for both unordered and ordered trees. Algorithms based on the restricted subtree homeomorphism algorithm of M.-J. Chung [J. Algorithms 8 (1) (1987) 106–112] are presented that solve the constrained tree inclusion problem in $O(m^{1.5}n)$ time on unordered trees with m and n nodes, and in $O(mn)$ time on ordered trees, using $O(mn)$ additional space.

Keywords: Combinatorial algorithms; Tree inclusion; Tree pattern matching; Subtree homeomorphism; Subtree isomorphism; Noncrossing bipartite matching

1. Introduction

The tree inclusion problem was introduced in [1, Ex. 2.3.2-22] and further studied in [2], motivated by the study of query languages for structured text databases, and has since received much attention, from the theoretical side and also as a primitive for querying collections of XML documents [3–5]. Given a pattern tree P and a text tree T, both with labels on the nodes, the tree inclusion problem consists in locating the smallest subtrees of T that include P, where a tree is included in another tree if can be obtained from the latter by deleting nodes and, in the case of unordered trees, by also permuting siblings. Deleting

E-mail address: valiente@lsi.upc.es (G. Valiente).
URL: http://www.lsi.upc.es/~valiente (G. Valiente).

1570-8667/$ – see front matter
doi:10.1016/j.jda.2004.08.017

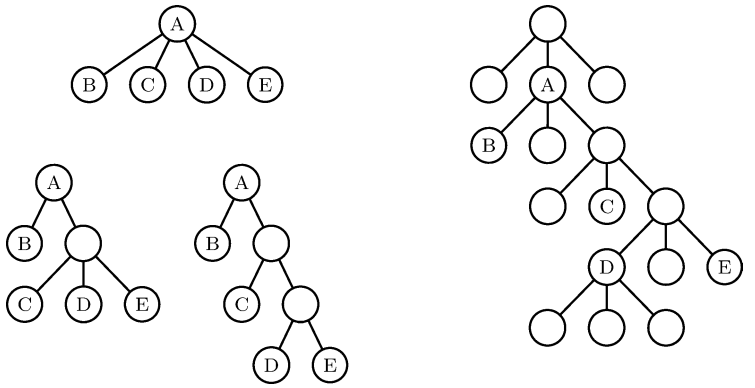


Fig. 1. The three forms of the same query (to the left) are all included at the node labeled A in the text tree (to the right).

a node v from a tree entails deleting all edges incident to v and inserting edges connecting the parent of v (for nonroot nodes) with the children (if any) of v.

Tree inclusion has two main drawbacks. First, the solution to a tree inclusion query of a pattern tree in a text tree is not sensitive to the structure of the query, in the sense that even for ordered trees, many structural forms of the same pattern (that is, many different pattern trees with the same labeling) may be included in the same text tree. As a matter of fact, any smallest supertree under minor containment [6] of a set of pattern trees includes all of the pattern trees (although not necessarily in a minimal way). For instance, three different structural forms of the same query or pattern tree are shown in Fig. 1 which are all included in the same text tree.

The second drawback is the complexity of tree inclusion. The tree inclusion problem on unordered trees is NP-hard [2,7], although it can be solved in polynomial time on ordered trees. A dynamic programming algorithm was given in [2] that solves the ordered tree inclusion problem in $O(mn)$ time and space in the worst case and also on the average, for a pattern tree with m nodes and a text tree with n nodes. Several improvements were since proposed [8–10].

These drawbacks stem from the generality of tree inclusion. In this paper, a constrained form of the tree inclusion problem is presented which is more sensitive to the structure of the pattern tree and can be solved in polynomial time, on both unordered and ordered trees. In the constrained formulation, a tree is included in another tree if can be obtained from the latter by deleting degree-one and degree-two nodes and, in the case of unordered trees, by also permuting siblings. Therefore, it is related to the problem of finding homeomorphic subtrees in a tree. The solution method used is based on the restricted subtree homeomorphism algorithm of [11], which involves solving a series of maximum bipartite matching problems. The constrained tree inclusion algorithm takes $O(m^{1.5}n)$ time using $O(mn)$ additional space on unordered trees with m and n nodes, and can be improved using a result of [12] to run in $O((m^{1.5}/\log m)n)$ time, on trees labeled from an alphabet of size bounded by a constant [13, Lemma 6].

Remark 1. Note that the subtree homeomorphism algorithm of Chung [11] solves a restricted form of the problem only, in which a pattern tree is homeomorphic to the whole subtree rooted at some node of the text tree. Further, both the subtree homeomorphism algorithm of [11] and the subtree isomorphism algorithm of [12] deal with unlabeled trees only. The constrained tree inclusion algorithm presented in this paper solves the general subtree homeomorphism problem on labeled trees, without the aforementioned restrictions.

For ordered trees, a simple algorithm for the noncrossing bipartite matching problem is also presented that takes time linear in the number of vertices of the bipartite graph. The constrained tree inclusion algorithm takes thus $O(mn)$ time using $O(mn)$ additional space on ordered trees with m and n nodes.

Constrained tree inclusion is related to the tree edit problem. As a matter of fact, in tree inclusion the elementary edit operation of insertion is forbidden and thus, constrained tree inclusion is polynomially equivalent to degree-two tree edit distance, in which a node can be deleted only when it is a leaf or has one child. The degree-two tree edit distance problem can be solved in $O(mn \min(\deg(P), \deg(T)))$ time on unordered trees, and in $O(mn)$ time on ordered trees P and T with m and n nodes, respectively [14,15], using $O(mn)$ additional space.

The rest of the paper is organized as follows. The constrained tree inclusion problem is defined in Section 2, where the necessary notation is also introduced. The solution of constrained tree inclusion problems on unordered trees in addressed in Section 3, where an algorithm based on the restricted subtree homeomorphism algorithm of [11] is described that solves the constrained tree inclusion problem in $O(m^{1.5}n)$ time using $O(mn)$ additional space, on unordered trees with m and n nodes. The algorithm, which involves the solution of a series of small maximum bipartite matching problems, is extended in Section 4 to ordered trees by means of a simple algorithm to find a noncrossing matching covering one of the bipartite sets in an ordered bipartite graph. The constrained tree inclusion problem is thus solved in $O(mn)$ time and space, on ordered trees with m and n nodes. Finally, some conclusions are outlined in Section 5.

2. Constrained tree inclusion

In the tree inclusion problem, a pattern tree P is included in a text tree T if P can be obtained by deleting some nodes from T and, in the case of unordered trees, by also permuting sibling nodes [2]. In constrained tree inclusion, these deletion operations are allowed on degree-one nodes (leaves) and degree-two nodes (with one child) only.

All trees considered in this paper will be rooted trees, and the following notation will be used. The set of nodes and the set of edges of a tree T are denoted by $V(T)$ and $E(T)$, respectively. The number of children of node v in tree T is denoted by $\mathit{outdeg}(T, v)$ or just $\mathit{outdeg}(v)$, if T is clear from the context. The subtree of T rooted at node $v \in V(T)$ is denoted by $T[v]$. The label of node $v \in V(T)$ is denoted by $\mathit{label}(v)$. Further, $\mathit{root}(T)$ denotes the root of tree T, and $P \cong T$ denotes that trees P and T are isomorphic.

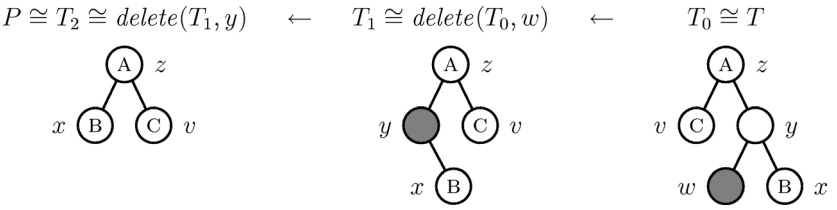


Fig. 2. A pattern tree P (to the left) included in a text tree T (to the right). P can be obtained from T by deleting degree-one and degree-two nodes, as shown from right to left.

Definition 2. A tree P is included in a tree T, denoted by $P \sqsubseteq T$, if either $P \cong T$ or there is a sequence of nodes $v_1, v_2, \ldots, v_k$ in $V(T)$ such that

- $T_{i+1} \cong delete(T_i, v_{i+1})$,
- $outdeg(T_i, v_{i+1}) \leqslant 1$

for $1 \leqslant i \leqslant k-1$, with $T_0 \cong T$ and $T_k \cong P$.

Example 3. The pattern tree P is included in the text tree T shown in Fig. 2. $P \cong T_2$ can be obtained from $T \cong T_0$ by deleting degree-one and degree-two nodes: $T_1 \cong delete(T_0, w)$ and $T_2 \cong delete(T_1, y)$.

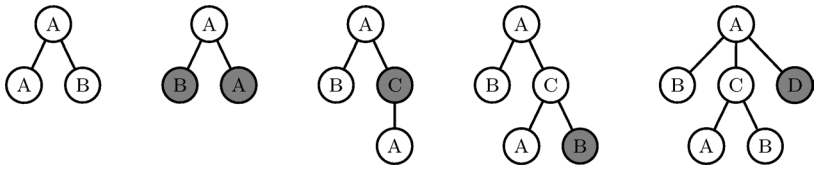
It follows from Definition 2 that the constrained tree inclusion relation is both reflexive and transitive.

3. Solving constrained tree inclusion problems on unordered trees

Constrained tree inclusion is not a trivial problem. The number of pattern trees that are included in a text tree is exponential in the number of nodes of the text tree.

Example 4. The ordered tree represented by $a(b, c(a, b), d)$ has 32 nonempty included subtrees, represented by a, $a(a)$, $a(a, d)$, $a(b)$, $a(b, a)$, $a(b, a, d)$, $a(b, b)$, $a(b, b, d)$, $a(b, c(a))$, $a(b, c(a), d)$, $a(b, c(a, b))$, $a(b, c(a, b), d)$, $a(b, c(b))$, $a(b, c(b), d)$, $a(b, c)$, $a(b, c, d)$, $a(b, d)$, $a(c(a))$, $a(c(a), d)$, $a(c(a, b))$, $a(c(a, b), d)$, $a(c(b))$, $a(c(b), d)$, $a(c)$, $a(c, d)$, $a(d)$, b, c, $c(a)$, $c(a, b)$, $c(b)$, d, but does not include the four subtrees represented by $a(a, b)$, $a(a, b, d)$, $a(b, a, b)$, $a(b, a, b, d)$, which would require the deletion of a degree-three node (with two children). Further, there are $32 + 50 = 82$ nonempty subtrees included in the unordered tree represented by the same term: the previous 32 subtrees, and $a(a(a, b), b)$, $a(a, b)$, $a(a, b, d)$, $a(a, d, b)$, $a(b, c(b, a))$, $a(b, c(b, a), d)$, $a(b, d, a)$, $a(b, d, b)$, $a(b, d, c(a))$, $a(b, d, c(a, b))$, $a(b, d, c(b))$, $a(b, d, c(b, a))$, $a(b, d, c)$, $a(c(a), b)$, $a(c(a), b, d)$, $a(c(a), d, b)$, $a(c(a, b), b, d)$, $a(c(a, b), d, b)$, $a(c(b), b)$, $a(c(b), b, d)$, $a(c(b), d, b)$, $a(c(b, a))$, $a(c(b, a), b)$, $a(c(b, a), b, d)$, $a(c(b, a), d)$, $a(c(b, a), d, b)$, $a(c, b)$, $a(c, b, d)$, $a(c, d, b)$, $a(d, a)$, $a(d, a, b)$, $a(d, b)$, $a(d, b, a)$, $a(d, b, b)$, $a(d, b, c(a))$, $a(d, b, c(a, b))$, $a(d, b, c(b))$, $a(d, b, c(b, a))$, $a(d, b, c)$, $a(d, c(a))$, $a(d, c(a), b)$,

$a(d, c(a, b))$, $a(d, c(a, b), b)$, $a(d, c(b))$, $a(d, c(b), b)$, $a(d, c(b, a))$, $a(d, c(b, a), b)$, $a(d, c)$, $a(d, c, b)$, $c(b, a)$. Again, the 14 subtrees represented by $a(a, b, b)$, $a(a, b, b, d)$, $a(a, b, d, b)$, $a(a, d, b, b)$, $a(b, a, b)$, $a(b, a, d, b)$, $a(b, b, a)$, $a(b, b, a, d)$, $a(b, b, d, a)$, $a(b, d, a, b)$, $a(b, d, b, a)$, $a(d, a, b, b)$, $a(d, b, a, b)$, $a(d, b, b, a)$ are not included in the previous unordered tree, because that would require the deletion of a degree-three node. Notice that $a(a, b)$ is not included as an ordered tree, because it would require deletion of the node labeled c, which has degree three in $a(b, c(a, b), d)$, although it is indeed included as an unordered tree.



As a matter of fact, $a(a, b)$ can be obtained from $a(b, c(a, b), d)$ by deleting two degree-one nodes and one degree-two node, and permuting sibling nodes.

The key to an efficient solution lies in the fact that a constrained tree inclusion problem instance can be decomposed into a series of smaller, independent problem instances. As a matter of fact, it will be shown that in order to determine whether or not $P[v] \sqsubseteq T[w]$, where node $v \in V(P)$ has children $v_1, v_2, \ldots, v_p$ and node $w \in V(T)$ has children $w_1, w_2, \ldots, w_t$, it suffices to know if $P[x] \sqsubseteq T[y]$ for all $x \in \{v, v_1, v_2, \ldots, v_p\}$ and $y \in \{w_1, w_2, \ldots, w_t\}$.

Definition 5. Let P and T be unordered trees, and let $w \in V(T)$. The included subtrees at node w, denoted by $S(w)$, are the set

$$S(w) = \big\{v \in V(P) \mid P[v] \sqsubseteq T[w]\big\}.$$

That is, $S(w)$ is the set of roots of those subtrees of P that are included in $T[w]$. Two direct consequences of the previous definition are given next.

Fact 6. *For all nodes $v \in V(P)$ and $w \in V(T)$, $P[v] \sqsubseteq T[w]$ if and only if $v \in S(w)$.*

Fact 7. *$P \sqsubseteq T$ if and only if $\{w \in V(T) \mid root(P) \in S(w)\} \neq \emptyset$.*

The next result ensures correctness of decomposing a constrained tree inclusion problem in independent subproblems.

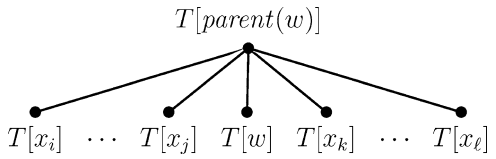
Lemma 8 (Chung). *Let $v \in V(P)$ have children $v_1, v_2, \ldots, v_p$, and let $w \in V(T)$ have children $w_1, w_2, \ldots, w_t$. Then, $P[v] \sqsubseteq T[w]$ if and only if either there is a child w_j of w such that $P[v] \sqsubseteq T[w_j]$, or $label(v) = label(w)$ and there is a subset of p different nodes $\{u_1, u_2, \ldots, u_p\} \subseteq \{w_1, w_2, \ldots, w_t\}$ such that $P[v_i] \sqsubseteq T[u_i]$ for $1 \leqslant i \leqslant p$.*

Lemma 9. *Let T be a tree. There is a sequence of node deletion operations that transform T into the tree T' with $V(T') = \{root(T)\}$ and $E(T') = \emptyset$.*

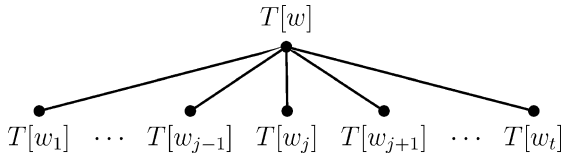
Proof. By deleting all nonroot nodes of T in postorder, the children (if any) of a node will have already been deleted when the node is considered for deletion, meaning the node has become a degree-one node (a leaf), which can thus be deleted. □

Corollary 10. *$P[v] \sqsubseteq T[\mathit{parent}(w)]$ if $P[v] \sqsubseteq T[w]$, for all nodes $v \in V(P)$ and all nonroot nodes $w \in V(T)$.*

Proof. $P[v] \sqsubseteq T[w]$, and $T[w]$ can be obtained from $T[\mathit{parent}(w)]$ by deleting $T[x]$ for all siblings x of node w and, then, deleting node $\mathit{parent}(w)$, which has become either a degree-one or a degree-two node. □



Proof of Lemma 8. (If.) Immediate. (Only if.) In the first case, $P[v]$ can be obtained from $T[w]$ by deleting $T[w_1], T[w_2], \ldots, T[w_{j-1}], T[w_{j+1}], \ldots, T[w_t]$ and, then, deleting node w, which has become either a degree-one or a degree-two node. In the second case, $P[v]$ can be obtained from $T[w]$ by deleting $T[w_i]$ for all $w_i \in \{w_1, w_2, \ldots, w_t\} \setminus \{u_1, u_2, \ldots, u_p\}$. □



Remark 11. A result similar to Lemma 8 was enunciated without proof in [11, Lemma 1] for the restricted subtree homeomorphism problem but does not carry over to constrained tree inclusion (it does not even hold for subtree homeomorphism) because deletion of degree-one nodes, not only of degree-two nodes, is required for Lemma 9 to hold.

Now, the set of included subtrees $S(w)$ can be computed for each node $w \in V(T)$ in a bottom-up way, as follows.

Algorithm 1. A procedure call of the form *included*(P, T, S) to the algorithm shown in Fig. 3, where P and T are unordered trees, computes $S = \{w \in V(T) \mid \mathit{root}(P) \in S(w)\}$. Thus, $P \sqsubseteq T$ if and only if $S \neq \emptyset$.

Lemma 12. *The constrained tree inclusion algorithm is correct.*

```
procedure included P, T, S
    S := ∅
    unmark all nodes w ∈ V(T)
    for all nodes w ∈ V(T) in postorder do
      S(w) := {v ∈ V(P) | outdeg(v) = 0 ∧ label(v) = label(w)}
      if outdeg(w) ≠ 0 then
        for all children y of w do
          S(w) := S(w) ∪ S(y)
        end for
        for all nonleaf nodes v ∈ V(P) with outdeg(v) ⩽ outdeg(w) and
        label(v) = label(w) in postorder do
          if v ∉ S(w) then
            construct a bipartite graph G
              with vertices all children x of v and all children y of w
              and x adjacent with y if and only if x ∈ S(y)
            if there is a matching in G with outdeg(v) edges then
              S(w) := S(w) ∪ {v}
            end if
          end if
    end for
    end if
    if node w is unmarked and root(P) ∈ S(w) then
      S := S ∪ {v}
    mark w and all ancestors of w
    end if
    end for
end procedure
```

Fig. 3. Constrained tree inclusion algorithm for unordered trees. Given unordered trees P and T, $\mathit{included}(P, T, S)$ computes $S = \{w \in V(T) \mid \mathit{root}(P) \in S(w)\}$.

Proof. Follows from Lemma 8. Notice that, all leaves $v \in V(P)$ with $\mathit{label}(v) = \mathit{label}(w)$ are added to $S(w)$ for all nodes $w \in V(T)$, both leaves and nonleaf nodes, in postorder. A leaf $v \in V(P)$ with $v \in S(w)$ for a nonleaf node $w \in V(T)$ corresponds to a constrained tree inclusion $P[v] \sqsubseteq T[w]$ in which $T[w_j]$ is deleted for all children w_j of node w. Note also that by marking all ancestors of a node $w \in V(T)$ with $\mathit{root}(P) \in S(w)$, only the (roots of) smallest subtrees of T that include P are collected in S. □

Lemma 13. *The constrained tree inclusion algorithm takes* $O(m^{1.5}n)$ *time using* $O(mn)$ *additional space, on unordered trees with* m *and* n *nodes.*

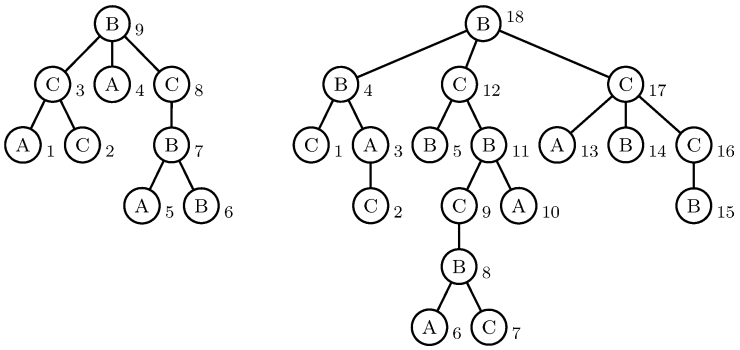
Proof. Let P and T be unordered trees with nodes $\{v_1, v_2, \ldots, v_m\}$ and $\{w_1, w_2, \ldots, w_n\}$, respectively. The time complexity of the algorithm is dominated by the solution of a series of small maximum bipartite matching problems. Since a maximum matching problem on a bipartite graph with r and s vertices can be solved in $O(r^{1.5}s)$ time [16], the time

complexity of the algorithm is

$$\begin{aligned}
&\sum_{i=1}^{n}\sum_{j=1}^{m}\mathrm{O}\big(\mathit{outdeg}(w_i)\mathit{outdeg}(v_j)^{1.5}\big)\\
&\quad=\sum_{i=1}^{n}\mathrm{O}\big(m^{1.5}\mathit{outdeg}(w_i)\big)\quad\left(\text{because }\sum_{j=1}^{m}\mathit{outdeg}(v_j)=m-1\right)\\
&\quad=\mathrm{O}(m^{1.5}n)\quad\left(\text{because }\sum_{i=1}^{n}\mathit{outdeg}(w_i)=n-1\right).
\end{aligned}$$

Regarding space complexity, the collection of sets $S(w)$ for $w \in V(T)$ is stored in a two-dimensional array S of m by n integers, indexed by the postorder number of the nodes in the trees and with the representation invariant that $S(\mathit{order}(v), \mathit{order}(w))$ if and only if $v \in S(w)$, for all nodes $v \in V(P)$ and $w \in V(T)$. The algorithm thus uses $\mathrm{O}(mn)$ additional space. □

Example 14. Consider the pattern tree P and text tree T shown in Fig. 4. The constrained tree inclusion algorithm computes the set $S(w)$ of pattern nodes v such that $P[v] \sqsubseteq T[w]$, for each text node w. Then, $P \sqsubseteq T$, because $\mathit{root}(P) = 9 \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\} = S(18) = S(\mathit{root}(T))$ and, furthermore, $\mathit{root}(P) \notin S(w)$ for all nonroot nodes w of T. That is, the pattern is included in the text but not in any proper subtree of the text.



w	$S(w)$
1	{2}
2	{2}
3	{1, 2, 4, 5}
4	{1, 2, 4, 5, 6}
5	{6}
6	{1, 4, 5}

w	$S(w)$
7	{2}
8	{1, 2, 4, 5, 6}
9	{1, 2, 4, 5, 6}
10	{1, 4, 5}
11	{1, 2, 4, 5, 6, 7}
12	{1, 2, 4, 5, 6, 7, 8}

w	$S(w)$
13	{1, 4, 5}
14	{6}
15	{6}
16	{2, 6}
17	{1, 2, 3, 4, 5, 6}
18	{1, 2, 3, 4, 5, 6, 7, 8, 9}

Fig. 4. Unordered pattern tree P with $m = 9$ nodes (to the left) and text tree T with $n = 18$ nodes (to the right). The postorder number is show next to each node.

Now, the output of the constrained tree inclusion algorithm, that is, the set $S(w)$ of pattern nodes v such that $P[v] \sqsubseteq T[w]$, for each text node w, is sufficient for obtaining the actual pattern as a constrained subtree of the text, for instance in the form of a mapping of pattern nodes to text nodes. Then, those text nodes which no pattern node is mapped to can be deleted (in an appropriate order, to ensure that only degree-one and degree-two nodes are deleted) from the text to obtain the pattern.

Remark 15. Notice that the construction problem is not dealt with in related algorithms for restricted subtree homeomorphism [11] and subtree isomorphism [12], in which only the decision problem is solved. Furthermore, the approach used in [17, Section 4.2] of constructing a subtree isomorphism based on the solutions to all the maximum bipartite matching problems solved by the algorithm does not extend to subtree homeomorphism or constrained tree inclusion, because it is based on the existence of a unique subtree isomorphism contained in these solutions.

Given a pattern tree P, a text tree T, and the solution S computed by the constrained tree inclusion algorithm, together with the solutions B to all the bipartite matching problems, an actual mapping M of pattern nodes to text nodes can be obtained by performing a simultaneous traversal of P and T, mapping each node v of P to the first node w of T found such that

- $v \in S(w)$,
- $\mathit{label}(v) = \mathit{label}(w)$,
- $\{v, w\} \in B$.

However, in the constrained tree inclusion algorithm one bipartite matching problem is solved involving the children of each nonleaf node v of P and the children of each nonleaf node w of T with $\mathit{outdeg}(v) \leqslant \mathit{outdeg}(w)$ and $\mathit{label}(v) = \mathit{label}(w)$, while only one bipartite matching problem needs to be solved for each nonleaf node v of P in order to obtain the node mapping underlying the solution S computed by the constrained tree inclusion algorithm. Those solutions to bipartite matching problems needed in order to construct the mapping of pattern nodes to text nodes are computed again by the following algorithm, in order to keep the presentation clear.

Algorithm 2. A procedure call of the form $\mathit{subtree}(P, T, S, M)$ to the algorithm shown in Fig. 5, where P and T are unordered trees and $S(v) = \{w \in V(T) \mid v \in S(w)\}$ for all $v \in V(P)$, computes a node mapping $M : V(P) \to V(T)$ such that a subtree of T isomorphic to P can be obtained by deleting those nodes $w \in V(T)$ for which there is no node $v \in V(P)$ with $M(v) = w$.

Lemma 16. *The constrained tree inclusion construction algorithm is correct.*

Proof. Let B be the solutions to all maximum bipartite matching problems, and let $B(v)$ be the unique text node associated with v in B, for all nonroot pattern nodes v. Let also $B(\mathit{root}(P))$ be the first node w of T in preorder such that $\mathit{root}(P) \in S(w)$. It follows from

```
procedure subtree P, T, S, M
    if root(P) ∈ S(root(T)) then
        let w be the first node of T in preorder
          such that root(P) ∈ S(w) and label(root(P)) = label(w)
        set M(root(P)) to w
        for all nonleaf nodes v of P in preorder do
          construct a bipartite graph G
              with vertices all children x of v and all children y of M(v)
              and x adjacent with y if and only if x ∈ S(y)
          find a maximum matching in G
          for all children x of v do
              let z be the node adjacent with x in the maximum matching
              let y be the first node of T[z] in preorder
                such that x ∈ S(y) and label(x) = label(y)
              set M(x) to y
          end for
        end for
    end if
end procedure
```

Fig. 5. Constrained tree inclusion construction algorithm for unordered trees. Given unordered trees P and T, and given $S(v) = \{w \in V(T) \mid v \in S(w)\}$ for all nodes v of P, *subtree*(P, T, S, M) computes a mapping M of P into T as a constrained-included subtree.

Lemma 12 that there is a constrained tree inclusion mapping M associating a text node $M(v)$ of $T[B(v)]$ with each pattern node v.

These subtrees $T[B(v)]$ for all pattern nodes v are not pairwise disjoint, but the simultaneous preorder traversal of P and T ensures that if v comes before v' in preorder, then v is mapped to a node $M(v)$ of $T[B(v)] - T[B(v')]$. □

Corollary 17. *The constrained tree inclusion construction algorithm also takes* $O(m^{1.5}n)$ *time using* $O(mn)$ *additional space, on unordered trees with* m *and* n *nodes.*

Example 18. Consider again the pattern tree P with $m = 9$ nodes and text tree T with $n = 18$ nodes of Example 14, shown in Fig. 6. The mapping of pattern nodes to text nodes produced by the previous algorithm, illustrated in the drawing by highlighting mapped nodes in the text tree and showing to the left of each mapped node the postorder number of the pattern node which the text node is mapped to, is the following:

v	$B(v)$	$M(v)$
9	–	18
3	17	17
1	13	13

$0v$	$B(v)$	$M(v)$
2	16	16
4	4	3
8	12	12

v	$B(v)$	$M(v)$
7	11	11
5	10	10
6	9	8

$B(v)$ is the text node associated with nonroot pattern node v in the maximum bipartite patching. The pattern tree can be obtained from the text tree by deleting all unmapped nodes, that is, the nodes with postorder number 1, 2, 4, 5, 6, 7, 9, 14, 15, for instance during a postorder traversal of the text tree.

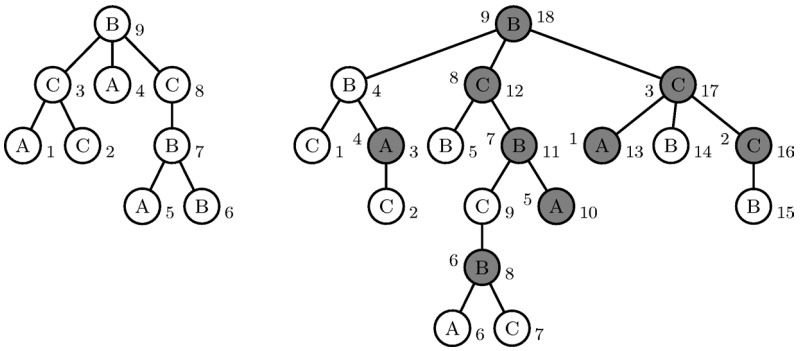


Fig. 6. Constrained inclusion of an unordered pattern tree P with $m = 9$ nodes (to the left) in a text tree T with $n = 18$ nodes (to the right). The postorder number is shown to the right of each node. Mapped text nodes are highlighted, with the postorder number of the corresponding pattern node shown to the left.

4. Solving constrained tree inclusion problems on ordered trees

When it comes to ordered trees, a necessary (but not sufficient) condition for a pattern tree to be included in a text tree is that the postorder traversal of the pattern be a subsequence of the postorder traversal of the text. Such a test can be used as a fast filter in pattern matching applications, because the problem of searching a string of length m as a subsequence of a string of length n can be solved in $O(m + n)$ time [18–20].

Now, it turns out that the previous algorithm for constrained tree inclusion on unordered trees can be used to solve the constrained tree inclusion problem on ordered trees as well. The series of maximum bipartite matching problems become now matching problems on *ordered bipartite graphs*.

Definition 19. An ordered bipartite graph is a bipartite graph $G = (V \cup W, E)$ with orderings $V = (v_1, v_2, \ldots, v_p)$ and $W = (w_1, w_2, \ldots, w_q)$.

A matching in an ordered bipartite graph is a noncrossing matching in the bipartite graph, with respect to the given orderings on the bipartite sets of vertices.

Definition 20. A noncrossing matching M in an ordered bipartite graph $G = (V \cup W, E)$ is a subset of edges $M \subseteq E$ such that no two edges are incident to the same vertex and no two edges are crossing, that is, for all edges (v_i, w_k) and (v_j, w_ℓ) in M, $i < j$ if and only if $k < \ell$.

The problem of finding a noncrossing matching of a bipartite graph with n vertices can be solved in $O(n \log \log n)$ time [21,22]. However, the decision problem of whether an ordered bipartite graph $(V \cup W, E)$ has a noncrossing matching of size $|W|$ can be solved in $O(n)$ time, where $n = |V| + |W|$.

As a matter of fact, given an ordered bipartite graph $(V \cup W, E)$, the greedy strategy of always choosing the first noncrossing edge joining some vertex $v_i \in V$ with vertex $w_j \in W$,

```
function matching V, W, E
    i := 1
    for j := 1 to |W| do
      while i ⩽ |V| − |W| + j and (v_i, w_j) ∉ E do
        i := i + 1
      end while
      if i ⩽ |V| − |W| + j then
        do nothing: edge (v_i, w_j) belongs to the matching
        i := i + 1
      else
        return false
      end if
    end for
    return true
end function
```

Fig. 7. Noncrossing bipartite matching algorithm. Given a bipartite graph $(V \cup W, E) = G$, *matching*(V, W, E) returns true if and only if there is a noncrossing matching of G covering all vertices of W.

for $1 \leqslant j \leqslant |W|$, gives a noncrossing matching with $|W|$ edges, as long as such a matching does exist. The following algorithm is an efficient implementation.

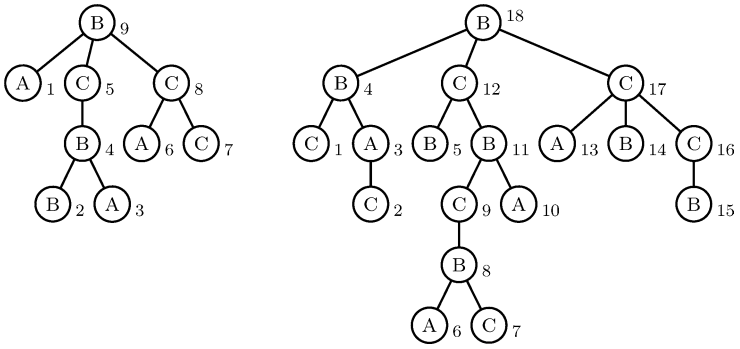
Algorithm 3. A function call of the form *matching*(V, W, E) to the algorithm shown in Fig. 7, returns true if and only if there is a noncrossing matching of the ordered bipartite graph $(V \cup W, E)$ covering all vertices of W.

Remark 21. Note that the noncrossing bipartite matching problem is polynomially equivalent to a variant of the sequence inclusion problem. As a matter of fact, let $G = (V \cup W, E)$ be an ordered bipartite graph with $V = (v_1, v_2, \ldots, v_p)$ and $W = (w_1, w_2, \ldots, w_q)$, where $p \geqslant q$. Then, there is a noncrossing matching in G covering all vertices of W if and only if there is a subsequence $v_{i_1} v_{i_2} \ldots v_{i_q}$ of $v_1 v_2 \ldots v_p$ such that $(v_{i_j}, w_j) \in E$ for $1 \leqslant j \leqslant q$, if and only if $w_1 \in v_1 v_2 \ldots v_p$ at position, say, k_1 and $w_i \in v_{k_{i-1}+1} \ldots v_p$ at position k_i, for $2 \leqslant i \leqslant q$.

Lemma 22. *The noncrossing bipartite matching algorithm is correct.*

Proof. Let $V = (v_1, v_2, \ldots, v_p)$ and $W = (w_1, w_2, \ldots, w_q)$. For $1 \leqslant j \leqslant q$, the edge (v_i, w_j) with the smallest i for $i \leqslant p - (q - j)$ is added to the matching. Choosing the smallest such i guarantees that the matching is noncrossing, and the upper bound $p - (q - j)$ guarantees that the noncrossing matching can be completed to size q with the remaining $q - j$ edges (if they exist). Now, as soon as no such noncrossing edge (v_i, w_j) exists for a vertex w_j, no noncrossing matching of size q exists and the matching procedure fails. Otherwise, one noncrossing edge belongs to the matching for each j with $1 \leqslant j \leqslant q$, and the matching procedure is thus successful. □

Lemma 23. *The noncrossing bipartite matching algorithm takes time linear in the number of vertices of the bipartite graph.*



w	$S(w)$
1	{7}
2	{7}
3	{1, 3, 6, 7}
4	{1, 2, 3, 6, 7}
5	{2}
6	{1, 3, 6}

w	$S(w)$
7	{7}
8	{1, 2, 3, 6, 7}
9	{1, 2, 3, 6, 7}
10	{1, 3, 6}
11	{1, 2, 3, 4, 6, 7}
12	{1, 2, 3, 4, 5, 6, 7}

w	$S(w)$
13	{1, 3, 6}
14	{2}
15	{2}
16	{2, 7}
17	{1, 2, 3, 6, 7, 8}
18	{1, 2, 3, 4, 5, 6, 7, 8, 9}

Fig. 8. Ordered pattern tree P with $m = 9$ nodes (to the left) and text tree T with $n = 18$ nodes (to the right). The postorder number is show next to each node.

Proof. On an ordered bipartite graph $G = (V \cup W, E)$, the inner loop is executed at most $|V| + |W|$ times and, thus, the number of edge existence tests made is linear in the number of vertices of G. □

Corollary 24. *The constrained tree inclusion algorithm takes* $O(mn)$ *time using* $O(mn)$ *additional space, on ordered trees with* m *and* n *nodes.*

Proof. Let P and T be ordered trees with nodes $\{v_1, v_2, \ldots, v_m\}$ and $\{w_1, w_2, \ldots, w_n\}$, respectively. The time complexity of the algorithm is, by Lemma 23, dominated by the construction of a series of small maximum bipartite matching problems. Then, as in the proof of Lemma 13, the time complexity of the algorithm is

$$\sum_{i=1}^{n}\sum_{j=1}^{m} O\big(outdeg(w_i)outdeg(v_j)\big) = \sum_{i=1}^{n} O\big(m \cdot outdeg(w_i)\big) = O(mn). \qquad \square$$

Example 25. Consider the pattern tree P and text tree T shown in Fig. 8. The constrained tree inclusion algorithm computes the set $S(w)$ of pattern nodes v such that $P[v] \sqsubseteq T[w]$, for each text node w. Then, $P \sqsubseteq T$, because $root(P) = 9 \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\} = S(18) = S(root(T))$ and, furthermore, $root(P) \notin S(w)$ for all nonroot nodes w of T. That is, the pattern is included in the text but not in any proper subtree of the text.

Again, as in the case of unordered trees, the output of the constrained ordered tree inclusion algorithm, that is, the set $S(w)$ of pattern nodes v such that $P[v] \sqsubseteq T[w]$, for each text node w, is sufficient for obtaining the actual pattern as a constrained subtree of

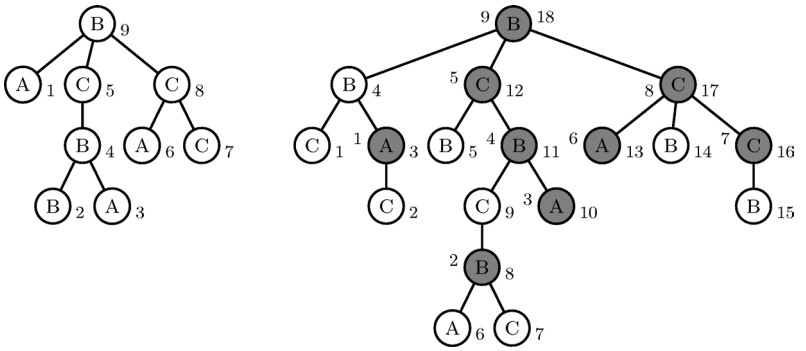


Fig. 9. Constrained inclusion of an ordered pattern tree P with $m = 9$ nodes (to the left) in a text tree T with $n = 18$ nodes (to the right). The postorder number is shown to the right of each node. Mapped text nodes are highlighted, with the postorder number of the corresponding pattern node shown to the left.

the text, in the form of a mapping of pattern nodes to text nodes. Those text nodes which no pattern node is mapped to can be deleted (in an appropriate order, to ensure that only degree-one and degree-two nodes are deleted) from the text to obtain the pattern.

The algorithm for constructing the node mapping underlying the solution to a constrained tree inclusion problem, carries over to ordered trees in a straightforward way. The bipartite matching problems become noncrossing matching problems.

Corollary 26. *The constrained tree inclusion construction algorithm takes* $O(mn)$ *time using* $O(mn)$ *additional space, on ordered trees with* m *and* n *nodes.*

Proof. Follows from Corollary 17 and the proof of Corollary 24. □

Example 27. Consider again the pattern tree P with $m = 9$ nodes and text tree T with $n = 18$ nodes of Example 25, shown in Fig. 9. The mapping of pattern nodes to text nodes produced by the constrained ordered tree inclusion construction algorithm, illustrated in the drawing by highlighting mapped nodes in the text tree and showing to the left of each mapped node the postorder number of the pattern node which the text node is mapped to, is the following:

v	$B(v)$	$M(v)$
9	–	18
1	4	3
5	12	12

v	$B(v)$	$M(v)$
4	11	11
2	9	8
3	10	10

v	$B(v)$	$M(v)$
8	17	17
6	13	13
7	16	16

$B(v)$ is the text node associated with nonroot pattern node v in the noncrossing bipartite patching. The pattern tree can be obtained from the text tree by deleting all unmapped nodes, that is, the nodes with postorder number 1, 2, 4, 5, 6, 7, 9, 14, 15, for instance during a postorder traversal of the text tree.

5. Conclusions

A constrained form of the tree inclusion problem is addressed in this paper in which a pattern tree can be obtained from a text tree by deleting degree-one and degree-two nodes and, in the case of unordered trees, by also permuting siblings. The constrained tree inclusion problem is more sensitive to the structure of the pattern tree than the general tree inclusion problem, in which there is no restriction of node degree for deletion and, unlike the latter, can be solved in polynomial time for both unordered and ordered trees.

Based on the restricted subtree homeomorphism algorithm of [11], an algorithm is given that solves the constrained tree inclusion problem in $O(m^{1.5}n)$ time using $O(mn)$ additional space, on unordered trees with m and n nodes. The algorithm, which involves the solution of a series of small maximum bipartite matching problems, is extended to ordered trees by means of a simple algorithm to find a noncrossing matching covering one of the bipartite sets in an ordered bipartite graph, a problem that is polynomially equivalent to a variant of the sequence inclusion problem, solving thus the constrained tree inclusion problem in $O(mn)$ time and space, on ordered trees with m and n nodes. The algorithm can be improved using a result of [12] to run in $O((m^{1.5}/\log m)n)$ time, on trees labeled from an alphabet of size bounded by a constant [13, Lemma 6].

The constrained tree inclusion algorithm also solves, with a minor modification, the subtree isomorphism problem. While there are efficient algorithms for unordered subtree isomorphism [12,23–25], known ordered subtree isomorphism algorithms solve the problem in a restricted form, in which a subtree is either a prefix of the tree [26,27] or the whole tree rooted at a node [28–33]. Further details about unordered and ordered subtree isomorphism can be found in [17, Section 4.2].

Acknowledgements

The author would like to acknowledge with thanks the anonymous referees, whose comments and criticism have led to a substantial improvement of this paper, and would also like to acknowledge detailed comments by Ron Y. Pinter and Michal Ziv-Ukelson on a preliminary version of this paper.

The research described in this paper has been partially supported by Spanish CICYT project MAVERISH (TIC2001-2476-C03-01) and by the Ministry of Education, Science, Sports and Culture of Japan through Grant-in-Aid for Scientific Research B-15300003 for visiting JAIST (Japan Advanced Institute of Science and Technology). A preliminary version of this paper has appeared in [34].

References

[1] D.E. Knuth, Fundamental Algorithms, vol. 1: The Art of Computer Programming, third ed., Addison-Wesley, Reading MA, 1997.

[2] P. Kilpeläinen, H. Mannila, Ordered and unordered tree inclusion, SIAM J. Comput. 24 (2) (1995) 340–356.

[3] S. Abiteboul, P. Buneman, D. Suciu, Data on the Web: From Relations to Semistructured Data and XML, Morgan Kaufmann, 2000.
[4] P. Kilpeläinen, H. Mannila, Retrieval from hierarchical texts by partial patterns, in: Proc. 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM Press, 1993, pp. 214–222.
[5] T. Schlieder, H. Meuss, Querying and ranking XML documents, J. Amer. Soc. Inform. Sci. Technol. 53 (6) (2002) 489–503.
[6] N. Nishimura, P. Ragde, D.M. Thilikos, Finding smallest supertrees under minor containment, Internat. J. Found. Comput. Sci. 11 (3) (2000) 445–465.
[7] J. Matoušek, R. Thomas, On the complexity of finding isomorphisms and other morphisms for partial k-trees, Discrete Math. 108 (1–3) (1992) 343–364.
[8] L. Alonso, R. Schott, On the tree inclusion problem, Acta Informatica 37 (9) (2001) 653–670.
[9] W. Chen, More efficient algorithm for ordered tree inclusion, J. Algorithms 26 (2) (1998) 370–385.
[10] T. Richter, A new algorithm for the ordered tree inclusion problem, in: Proc. 8th Annual Symp. Combinatorial Pattern Matching, in: Lecture Notes in Computer Science, vol. 1264, Springer-Verlag, Berlin, 1997, pp. 150–166.
[11] M.-J. Chung, $O(n^{2.5})$ time algorithms for the subgraph homeomorphism problem on trees, J. Algorithms 8 (1) (1987) 106–112.
[12] R. Shamir, D. Tsur, Faster subtree isomorphism, J. Algorithms 33 (2) (1999) 267–280.
[13] R.Y. Pinter, O. Rokhlenko, D. Tsur, M. Ziv-Ukelson, Approximate labelled subtree homeomorphism, in: Proc. 15th Annual Symp. Combinatorial Pattern Matching, in: Lecture Notes in Computer Science, vol. 3109, Springer-Verlag, Berlin, 2004, pp. 55–69.
[14] K. Zhang, Efficient parallel algorithms for tree editing problems, in: Proc. 7th Annual Symp. Combinatorial Pattern Matching, in: Lecture Notes in Computer Science, vol. 1075, Springer-Verlag, Berlin, 1996, pp. 361–372.
[15] K. Zhang, J.T.-L. Wang, D. Shasha, On the editing distance between undirected acyclic graphs, Internat. J. Found. Comput. Sci. 7 (1) (1996) 43–57.
[16] J.E. Hopcroft, R.M. Karp, An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs, SIAM J. Comput. 2 (4) (1973) 225–231.
[17] G. Valiente, Algorithms on Trees and Graphs, Springer-Verlag, Berlin, 2002.
[18] A. Apostolico, M. Crochemore, Optimal canonization of all substrings of a string, Inform. and Comput. 95 (1) (1991) 76–95.
[19] R.M. Karp, M.O. Rabin, Efficient randomized pattern-matching algorithms, IBM J. Res. Develop. 31 (2) (1987) 249–260.
[20] D.E. Knuth, J.H.M. Jr., V.R. Pratt, Fast pattern matching in strings, SIAM J. Comput. 6 (2) (1977) 323–350.
[21] F. Malucelli, T. Ottmann, D. Pretolani, Efficient labelling algorithms for the maximum noncrossing matching problem, Discrete Appl. Math. 47 (2) (1993) 175–179.
[22] M.-S. Yu, L.Y. Tseng, S.-J. Chang, Sequential and parallel algorithms for the maximum-weight independent set problem on permutation graphs, Inform. Process. Lett. 46 (1) (1993) 7–11.
[23] D.W. Matula, Subtree isomorphism in $O(n^{5/2})$, Ann. Discrete Math. 2 (1) (1978) 91–106.
[24] S.W. Reyner, An analysis of a good algorithm for the subtree problem, SIAM J. Comput. 6 (4) (1977) 730–732.
[25] R.M. Verma, S.W. Reyner, An analysis of a good algorithm for the subtree problem, corrected, SIAM J. Comput. 18 (5) (1989) 906–908.
[26] A.T. Berztiss, Data Structures: Theory and Practice, second ed., Academic Press, New York, 1975.
[27] G. Valiente, On the algorithm of Berztiss for tree pattern matching, in: Proc. 5th Mexican Internat. Conf. Computer Science, IEEE Computer Science Press, Piscataway, NJ, 2004.
[28] P. Dublish, Some comments on the subtree isomorphism problem for ordered trees, Inform. Process. Lett. 36 (5) (1990) 273–275.
[29] R. Grossi, A note on the subtree isomorphism for ordered trees and related problems, Inform. Process. Lett. 39 (2) (1991) 81–84.
[30] R. Grossi, Further comments on the subtree isomorphism for ordered trees, Inform. Process. Lett. 40 (5) (1991) 255–256.

[31] E. Mäkinen, On the subtree isomorphism problem for ordered trees, Inform. Process. Lett. 32 (5) (1989) 271–273.
[32] G. Valiente, An efficient bottom-up distance between trees, in: Proc. 8th Internat. Symp. String Processing and Information Retrieval, IEEE Computer Science Press, Piscataway, NJ, 2001, pp. 212–219.
[33] R.M. Verma, Strings, trees, and patterns, Inform. Process. Lett. 41 (3) (1992) 157–161.
[34] G. Valiente, Constrained tree inclusion, in: Proc. 14th Annual Symp. Combinatorial Pattern Matching, in: Lecture Notes in Computer Science, vol. 2676, Springer-Verlag, Berlin, 2003, pp. 361–371.