Constrained Tree Inclusion

14th Annual Symposium on Combinatorial Pattern Matching

Gabriel Valiente
valiente@lsi.upc.es

Technical University of Catalonia
Department of Software
E-08034 Barcelona
Contents

- The tree inclusion problem
Contents

- The tree inclusion problem
- Motivation

Remark
The slides got shuffled as the speaker stumbled over a slope when approaching the podium.
Contents

- The tree inclusion problem
- Motivation
- Constrained tree inclusion

Remark
The slides got shuffled as the speaker stumbled over a slope when approaching the podium.
Contents

- The tree inclusion problem
- Motivation
- Constrained tree inclusion
- Extension for ordered trees

Remark
The slides got shuffled as the speaker stumbled over a slope when approaching the podium.
Contents

- The tree inclusion problem
- Motivation
- Constrained tree inclusion
- Extension for ordered trees
- Solution for unordered trees
Contents

• The tree inclusion problem
• Motivation
• Constrained tree inclusion
• Extension for ordered trees
• Solution for unordered trees
• Related work
Contents

- The tree inclusion problem
- Motivation
- Constrained tree inclusion
- Extension for ordered trees
- Solution for unordered trees
- Related work

**Remark**  The slides got shuffled as the speaker stumbled over a slope when approaching the podium
The Tree Inclusion Problem

- Given a pattern tree $P$ and a text tree $T$, both labeled on the nodes, find the smallest subtrees of $T$ that include $P$
The Tree Inclusion Problem

- Given a pattern tree $P$ and a text tree $T$, both labeled on the nodes, find the smallest subtrees of $T$ that include $P$
  - Minor containment (deleting nodes and permuting siblings)
The Tree Inclusion Problem

- Given a pattern tree $P$ and a text tree $T$, both labeled on the nodes, find the smallest subtrees of $T$ that include $P$
  - Minor containment (deleting nodes and permuting siblings)
The Tree Inclusion Problem

- Given a pattern tree $P$ and a text tree $T$, both labeled on the nodes, find the smallest subtrees of $T$ that include $P$
The Tree Inclusion Problem

- Given a pattern tree $P$ and a text tree $T$, both labeled on the nodes, find the smallest subtrees of $T$ that include $P$
  - Subtree homeomorphism (deleting degree-two nodes and permuting siblings)
The Tree Inclusion Problem

- Given a pattern tree $P$ and a text tree $T$, both labeled on the nodes, find the smallest subtrees of $T$ that include $P$
  - Subtree homeomorphism (deleting degree-two nodes and permuting siblings)
The Tree Inclusion Problem

- Given a pattern tree $P$ and a text tree $T$, both labeled on the nodes, find the smallest subtrees of $T$ that include $P$
The Tree Inclusion Problem

- Given a pattern tree $P$ and a text tree $T$, both labeled on the nodes, find the smallest subtrees of $T$ that include $P$
  - Subtree isomorphism (deleting degree-one nodes and permuting siblings)
The Tree Inclusion Problem

- Given a pattern tree $P$ and a text tree $T$, both labeled on the nodes, find the smallest subtrees of $T$ that include $P$
  - Subtree isomorphism (deleting degree-one nodes and permuting siblings)
The Tree Inclusion Problem

- Given a pattern tree $P$ and a text tree $T$, both labeled on the nodes, find the smallest subtrees of $T$ that include $P$
The Tree Inclusion Problem

- Given a pattern tree $P$ and a text tree $T$, both labeled on the nodes, find the smallest subtrees of $T$ that include $P$
- Constrained tree inclusion (deleting degree-one and degree-two nodes and permuting siblings)
**The Tree Inclusion Problem**

- Given a pattern tree $P$ and a text tree $T$, both labeled on the nodes, find the smallest subtrees of $T$ that include $P$
  - Constrained tree inclusion (deleting degree-one and degree-two nodes and permuting siblings)
Motivation

- Constrained tree inclusion is motivated by the study of query languages for structured text databases
Motivation

- Constrained tree inclusion is motivated by the study of query languages for structured text databases
- Tree inclusion has two main drawbacks
Motivation

- Constrained tree inclusion is motivated by the study of query languages for structured text databases.
- Tree inclusion has two main drawbacks:
  - The solution to a tree inclusion query of a pattern tree in a text tree is not much sensitive to the structure of the query: Many structural forms of the same pattern may be included in the same text tree.
Motivation

- Constrained tree inclusion is motivated by the study of query languages for structured text databases
- Tree inclusion has two main drawbacks
  - The solution to a tree inclusion query of a pattern tree in a text tree is not much sensitive to the structure of the query: Many structural forms of the same pattern may be included in the same text tree
  - Complexity of tree inclusion
Motivation

- Constrained tree inclusion is motivated by the study of query languages for structured text databases
- Tree inclusion has two main drawbacks
  - The solution to a tree inclusion query of a pattern tree in a text tree is not much sensitive to the structure of the query: Many structural forms of the same pattern may be included in the same text tree
  - Complexity of tree inclusion
    - NP-hard for unordered trees
Motivation

- Constrained tree inclusion is motivated by the study of query languages for structured text databases.

- Tree inclusion has two main drawbacks:
  - The solution to a tree inclusion query of a pattern tree in a text tree is not much sensitive to the structure of the query: Many structural forms of the same pattern may be included in the same text tree.
  - Complexity of tree inclusion:
    - NP-hard for unordered trees.
    - Solvable for ordered trees by dynamic programming in $O(mn)$ time and space, in the worst case and also on the average.
Motivation

- Constrained tree inclusion is motivated by the study of query languages for structured text databases
- Tree inclusion has two main drawbacks
  - The solution to a tree inclusion query of a pattern tree in a text tree is not much sensitive to the structure of the query: Many structural forms of the same pattern may be included in the same text tree
  - Complexity of tree inclusion
    - NP-hard for unordered trees
    - Solvable for ordered trees by dynamic programming in $O(mn)$ time and space, in the worst case and also on the average
- These drawbacks stem from the generality of tree inclusion
Motivation

- Three forms of the same query are all included at the node labeled A in the text tree, shown to the right of the picture.
Constrained Tree Inclusion

- A tree $P$ is included in a tree $T$, denoted by $P \sqsubseteq T$, if there is a sequence of nodes $v_1, v_2, \ldots, v_k$ in $V(T)$ such that
Constrained Tree Inclusion

- A tree $P$ is included in a tree $T$, denoted by $P \subseteq T$, if there is a sequence of nodes $v_1, v_2, \ldots, v_k$ in $V(T)$ such that
  - $T_{i+1} \cong \text{delete}(T_i, v_{i+1})$
Constrained Tree Inclusion

- A tree $P$ is included in a tree $T$, denoted by $P \subseteq T$, if there is a sequence of nodes $v_1, v_2, \ldots, v_k$ in $V(T)$ such that
  - $T_{i+1} \cong delete(T_i, v_{i+1})$
  - $outdeg(T_i, v_{i+1}) \leq 1$
Constrained Tree Inclusion

- A tree $P$ is included in a tree $T$, denoted by $P \subseteq T$, if there is a sequence of nodes $v_1, v_2, \ldots, v_k$ in $V(T)$ such that
  - $T_{i+1} \cong \text{delete}(T_i, v_{i+1})$
  - $\text{outdeg}(T_i, v_{i+1}) \leq 1$

for $1 \leq i \leq k - 1$, with $T_0 = T$ and $T_k = P$
Constrained Tree Inclusion

- A tree $P$ is included in a tree $T$, denoted by $P \sqsubseteq T$, if there is a sequence of nodes $v_1, v_2, \ldots, v_k$ in $V(T)$ such that
  - $T_{i+1} \equiv \text{delete}(T_i, v_{i+1})$
  - $\text{outdeg}(T_i, v_{i+1}) \leq 1$
  for $1 \leq i \leq k - 1$, with $T_0 = T$ and $T_k = P$
- $P \sqsubseteq T$, because $P$ can be obtained from $T$ by deleting degree-one and degree-two nodes, as shown from right to left
Constrained Tree Inclusion

- A tree $P$ is included in a tree $T$, denoted by $P \sqsubset T$, if there is a sequence of nodes $v_1, v_2, \ldots, v_k$ in $V(T)$ such that
  - $T_{i+1} \cong \text{delete}(T_i, v_{i+1})$
  - $\text{outdeg}(T_i, v_{i+1}) \leq 1$
  for $1 \leq i \leq k - 1$, with $T_0 = T$ and $T_k = P$

- $P \sqsubset T$, because $P$ can be obtained from $T$ by deleting degree-one and degree-two nodes, as shown from right to left

\[
P = T_2 \cong \text{delete}(T_1, y) \quad T_1 \cong \text{delete}(T_0, w) \quad T_0 = T
\]
Constrained Tree Inclusion

- The number of pattern trees that are included in a text tree is exponential in the size of the text tree
Constrained Tree Inclusion

- The number of pattern trees that are included in a text tree is exponential in the size of the text tree.

includes

and does not include
Constrained Tree Inclusion

- 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.
Constrained Tree Inclusion

- 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.
- In order to determine whether or not $P[v] \subseteq T[w]$ it suffices to know if $P[v] \subseteq T[y]$ and if $P[x] \subseteq T[y]$ for all children $x$ of $v$ and all children $y$ of $w$. 
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.

In order to determine whether or not $P[v] \subseteq T[w]$ it suffices to know if $P[v] \subseteq T[y]$ and if $P[x] \subseteq T[y]$ for all children $x$ of $v$ and all children $y$ of $w$. 

![Diagram showing tree inclusion](image-url)
Constrained Tree Inclusion

- 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.

- In order to determine whether or not \( P[v] \subseteq T[w] \) it suffices to know if \( P[v] \subseteq T[y] \) and if \( P[x] \subseteq T[y] \) for all children \( x \) of \( v \) and all children \( y \) of \( w \).

- That is, it suffices to know if \( P[x] \subseteq T[y] \) for all \( x \in \{v, v_1, v_2, \ldots, v_p\} \) and \( y \in \{w_1, w_2, \ldots, w_t\} \).
Extension for Ordered Trees

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

- 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 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 $(v_j, w_\ell)$ in $M$, $i < j$ if and only if $k < \ell$
Extension for Ordered Trees

• 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 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 $(v_j, w_\ell)$ in $M$, $i < j$ if and only if $k < \ell$

• 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
Extension for Ordered Trees

- 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 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 $(v_j, w_\ell)$ in $M$, $i < j$ if and only if $k < \ell$

- 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

- 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 $w_j \in W$, for $1 \leq j \leq |W|$, gives a noncrossing matching with $|W|$ edges, as long as such a matching does exist
Extension for Ordered Trees

- 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 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 $(v_j, w_\ell)$ in $M$, $i < j$ if and only if $k < \ell$

- 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

- 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 $w_j \in W$, for $1 \leq j \leq |W|$, gives a noncrossing matching with $|W|$ edges, as long as such a matching does exist

- Noncrossing bipartite matching is equivalent to sequence inclusion
Extension for Ordered Trees

- Time complexity is dominated by the solution of a series of small noncrossing bipartite matching problems
Extension for Ordered Trees

- Time complexity is dominated by the solution of a series of small noncrossing bipartite matching problems

\[ \sum_{i=1}^{n} \sum_{j=1}^{m} O(\text{outdeg}(w_i)\text{outdeg}(v_j)) = \sum_{i=1}^{n} O(m \cdot \text{outdeg}(w_i)) = O(mn) \]
Solution for Unordered Trees

• For all $w \in V(T)$, let $S(w) = \{v \in V(P) \mid P[v] \sqsubseteq T[w]\}$
Solution for Unordered Trees

- For all \( w \in V(T) \), let \( S(w) = \{ v \in V(P) \mid P[v] \subseteq T[w] \} \)
- For all nodes \( v \in V(P) \) and \( w \in V(T) \), \( P[v] \subseteq T[w] \) if and only if \( v \in S(w) \)
Solution for Unordered Trees

- For all \( w \in V(T) \), let \( S(w) = \{ v \in V(P) \mid P[v] \sqsubseteq T[w] \} \)
- For all nodes \( v \in V(P) \) and \( w \in V(T) \), \( P[v] \sqsubseteq T[w] \) if and only if \( v \in S(w) \)
- \( P \sqsubseteq T \) if and only if \( \{ w \in V(T) \mid \text{root}(P) \in S(w) \} \neq \emptyset \)
Solution for Unordered Trees

- For all \( w \in V(T) \), let \( S(w) = \{ v \in V(P) \mid P[v] \subseteq T[w] \} \)
- For all nodes \( v \in V(P) \) and \( w \in V(T) \), \( P[v] \subseteq T[w] \) if and only if \( v \in S(w) \)
- \( P \subseteq T \) if and only if \( \{ w \in V(T) \mid \text{root}(P) \in S(w) \} \neq \emptyset \)
- There is a sequence of node deletion operations that transform any given tree \( T \) into the tree \( T' \) with \( V(T') = \{ \text{root}(T) \} \) and \( E(T') = \emptyset \)
Solution for Unordered Trees

- For all \( w \in V(T) \), let \( S(w) = \{ v \in V(P) \mid P[v] \sqsubseteq T[w] \} \)
- For all nodes \( v \in V(P) \) and \( w \in V(T) \), \( P[v] \sqsubseteq T[w] \) if and only if \( v \in S(w) \)
- \( P \sqsubseteq T \) if and only if \( \{ w \in V(T) \mid \text{root}(P) \in S(w) \} \neq \emptyset \)
- There is a sequence of node deletion operations that transform any given tree \( T \) into the tree \( T' \) with \( V(T') = \{ \text{root}(T) \} \) and \( E(T') = \emptyset \)

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
Solution for Unordered Trees

- \(P[v] \subseteq T[parent(w)]\) if \(P[v] \subseteq T[w]\), for all nodes \(v \in V(P)\) and all nonroot nodes \(w \in V(T)\)
Solution for Unordered Trees

- $P[v] \subseteq T[parent(w)]$ if $P[v] \subseteq T[w]$, for all nodes $v \in V(P)$ and all nonroot nodes $w \in V(T)$

![Diagram showing tree structures]

- $P[v] \subseteq T[parent(w)]$ if $P[v] \subseteq T[w]$, for all nodes $v \in V(P)$ and all nonroot nodes $w \in V(T)$
Solution for Unordered Trees

- $P[v] \subseteq T[parent(w)]$ if $P[v] \subseteq T[w]$, for all nodes $v \in V(P)$ and all nonroot nodes $w \in V(T)$

$T[parent(w)]$

$T[x_i] \cdots T[x_j] T[w] T[x_k] \cdots T[x_\ell]$

$P[v] \subseteq T[w]$, and $T[w]$ can be obtained from $T[parent(w)]$ by deleting $T[x]$ for all siblings $x$ of node $w$ and, then, deleting node $parent(w)$, which has become either a degree-one or a degree-two node
Solution for Unordered Trees

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 \( \text{label}(v) = \text{label}(w) \) 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 \( 1 \leq i \leq p \).
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 \( \text{label}(v) = \text{label}(w) \) 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 \( 1 \leq i \leq p \).
Solution for Unordered Trees

- 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 \( \text{label}(v) = \text{label}(w) \) 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 \( 1 \leq i \leq p \).

In the first case, \( P[v] \) can be obtained from \( T[w] \) by deleting \( T[w_1], T[w_2], \ldots, T[w_{j-1}], T[w_j], \ldots, T[w_t] \) and, then, deleting node \( w \), which has become either a degree-one or a degree-two node.
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 \( \text{label}(v) = \text{label}(w) \) 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 \( 1 \leq i \leq p \).

In the first case, \( P[v] \) can be obtained from \( T[w] \) by deleting \( T[w_1], T[w_2], \ldots, T[w_{j-1}], T[w_j], \ldots, T[w_t] \) 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 \( w_i \in \{w_1, w_2, \ldots, w_t\} \setminus \{u_1, u_2, \ldots, u_p\} \).
Solution for Unordered Trees

- A similar result was enunciated without proof in [Chung, 1987] for the 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.
Solution for Unordered Trees

- A similar result was enunciated without proof in [Chung, 1987] for the 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.

- The set of included subtrees $S(w)$ can be computed for each node $w \in V(T)$ in a bottom-up way.
Solution for Unordered Trees

- A similar result was enunciated without proof in [Chung, 1987] for the 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.
- The set of included subtrees $S(w)$ can be computed for each node $w \in V(T)$ in a bottom-up way.
- Time complexity is dominated by the solution of a series of small maximum bipartite matching problems.
Solution for Unordered Trees

- A similar result was enunciated without proof in [Chung, 1987] for the 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.

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

- Time complexity is dominated by the solution of a series of small maximum bipartite matching problems.

\[
\sum_{i=1}^{n} \sum_{j=1}^{m} O(\text{outdeg}(w_i)\text{outdeg}(v_j)^{1.5})
\]

\[
= \sum_{i=1}^{n} O(m^{1.5} \text{outdeg}(w_i)) \quad \text{(because} \sum_{j=1}^{m} \text{outdeg}(v_j) = m - 1)\]

\[
= O(m^{1.5} n) \quad \text{(because} \sum_{i=1}^{n} \text{outdeg}(w_i) = n - 1)\]
Related Work

- Constrained tree inclusion is related to the tree edit problem
Related Work

- Constrained tree inclusion is related to the tree edit problem
  - Insertions are forbidden in tree inclusion

- Constrained tree inclusion is equivalent to degree-two tree edit

- Solvable for unordered trees in $O(mn \min(\deg(P), \deg(T)))$ time
- Solvable for ordered trees in $O(mn)$ time using $O(mn)$ space

- Constrained tree inclusion is easier than degree-two tree edit

- For unordered trees
  - Solvable in $O(mn^{1/2})$ time using $O(mn)$ space
  - Simple algorithm that involves the solution of a series of small maximum bipartite matching problems

- For ordered trees
  - Solvable in $O(mn)$ time using $O(mn)$ space
  - Simple algorithm to find a noncrossing matching covering one of the bipartite sets in an ordered bipartite graph
Related Work

- Constrained tree inclusion is related to the tree edit problem
  - Insertions are forbidden in tree inclusion
  - Deletions of degree-one and degree-two nodes only are allowed in constrained tree inclusion
Related Work

- Constrained tree inclusion is related to the tree edit problem
  - Insertions are forbidden in tree inclusion
  - Deletions of degree-one and degree-two nodes only are allowed in constrained tree inclusion
  - Constrained tree inclusion is equivalent to degree-two tree edit

- Solvable for unordered trees in $O(mn \min(\deg(P), \deg(T)))$ time
- Solvable for ordered trees in $O(mn)$ time using $O(mn)$ space

- Constrained tree inclusion is easier than degree-two tree edit

- For unordered trees
  - Solvable in $O(mn^{1/2})$ time using $O(mn)$ space
  - Simple algorithm that involves the solution of a series of small maximum bipartite matching problems

- For ordered trees
  - Solvable in $O(mn)$ time using $O(mn)$ space
  - Simple algorithm to find a noncrossing matching covering one of the bipartite sets in an ordered bipartite graph
Related Work

- Constrained tree inclusion is related to the tree edit problem
  - Insertions are forbidden in tree inclusion
  - Deletions of degree-one and degree-two nodes only are allowed in constrained tree inclusion
  - Constrained tree inclusion is equivalent to degree-two tree edit
    - Solvable for unordered trees in $O(mn \min(\deg(P), \deg(T)))$ time
- Constrained tree inclusion is easier than degree-two tree edit
  - For unordered trees
    - Solvable in $O(m\sqrt{mn})$ time using $O(mn)$ space
    - Simple algorithm that involves the solution of a series of small maximum bipartite matching problems
  - For ordered trees
    - Solvable in $O(mn)$ time using $O(mn)$ space
    - Simple algorithm to find a noncrossing matching covering one of the bipartite sets in an ordered bipartite graph
Related Work

- Constrained tree inclusion is related to the tree edit problem
  - Insertions are forbidden in tree inclusion
  - Deletions of degree-one and degree-two nodes only are allowed in constrained tree inclusion
  - Constrained tree inclusion is equivalent to degree-two tree edit
    - Solvable for unordered trees in $O(mn \min(\deg(P), \deg(T)))$ time
    - Solvable for ordered trees in $O(mn)$ time using $O(mn)$ space
      [Zhang, 1996; Zhang, Wang, Shasha, 1996]
Related Work

- Constrained tree inclusion is related to the tree edit problem
  - Insertions are forbidden in tree inclusion
  - Deletions of degree-one and degree-two nodes only are allowed in constrained tree inclusion
  - Constrained tree inclusion is equivalent to degree-two tree edit
    - Solvable for unordered trees in $O(mn \min(\text{deg}(P), \text{deg}(T)))$ time
    - Solvable for ordered trees in $O(mn)$ time using $O(mn)$ space
      [Zhang, 1996; Zhang, Wang, Shasha, 1996]
  - Constrained tree inclusion is easier than degree-two tree edit
Related Work

- Constrained tree inclusion is related to the tree edit problem
  - Insertions are forbidden in tree inclusion
  - Deletions of degree-one and degree-two nodes only are allowed in constrained tree inclusion
  - Constrained tree inclusion is equivalent to degree-two tree edit
    - Solvable for unordered trees in $O(mn \min(\text{deg}(P), \text{deg}(T)))$ time
    - Solvable for ordered trees in $O(mn)$ time using $O(mn)$ space
      [Zhang, 1996; Zhang, Wang, Shasha, 1996]
  
- Constrained tree inclusion is easier than degree-two tree edit
  - For unordered trees
Related Work

- Constrained tree inclusion is related to the tree edit problem
  - Insertions are forbidden in tree inclusion
  - Deletions of degree-one and degree-two nodes only are allowed in constrained tree inclusion
  - Constrained tree inclusion is equivalent to degree-two tree edit
    - Solvable for unordered trees in $O(mn \min(\deg(P), \deg(T)))$ time
    - Solvable for ordered trees in $O(mn)$ time using $O(mn)$ space
      [Zhang, 1996; Zhang, Wang, Shasha, 1996]

- Constrained tree inclusion is easier than degree-two tree edit
  - For unordered trees
    - Solvable in $O(m\sqrt{mn})$ time using $O(mn)$ space
Related Work

- Constrained tree inclusion is related to the tree edit problem
  - Insertions are forbidden in tree inclusion
  - Deletions of degree-one and degree-two nodes only are allowed in constrained tree inclusion
  - Constrained tree inclusion is equivalent to degree-two tree edit
    - Solvable for unordered trees in $O(mn \min(\deg(P), \deg(T)))$ time
    - Solvable for ordered trees in $O(mn)$ time using $O(mn)$ space
      [Zhang, 1996; Zhang, Wang, Shasha, 1996]
  - Constrained tree inclusion is easier than degree-two tree edit
    - For unordered trees
      - Solvable in $O(m\sqrt{mn})$ time using $O(mn)$ space
      - Simple algorithm that involves the solution of a series of small maximum bipartite matching problems
Related Work

• Constrained tree inclusion is related to the tree edit problem
  • Insertions are forbidden in tree inclusion
  • Deletions of degree-one and degree-two nodes only are allowed in constrained tree inclusion
  • Constrained tree inclusion is equivalent to degree-two tree edit
    • Solvable for unordered trees in $O(mn \min(\text{deg}(P), \text{deg}(T)))$ time
    • Solvable for ordered trees in $O(mn)$ time using $O(mn)$ space
      [Zhang, 1996; Zhang, Wang, Shasha, 1996]
  • Constrained tree inclusion is easier than degree-two tree edit
    • For unordered trees
      • Solvable in $O(m \sqrt{mn})$ time using $O(mn)$ space
      • Simple algorithm that involves the solution of a series of small maximum bipartite matching problems
    • For ordered trees
Related Work

- Constrained tree inclusion is related to the tree edit problem
  - Insertions are forbidden in tree inclusion
  - Deletions of degree-one and degree-two nodes only are allowed in constrained tree inclusion
  - Constrained tree inclusion is equivalent to degree-two tree edit
    - Solvable for unordered trees in $O(mn \min(\deg(P), \deg(T)))$ time
    - Solvable for ordered trees in $O(mn)$ time using $O(mn)$ space
    [Zhang, 1996; Zhang, Wang, Shasha, 1996]

- Constrained tree inclusion is easier than degree-two tree edit
  - For unordered trees
    - Solvable in $O(m\sqrt{mn})$ time using $O(mn)$ space
    - Simple algorithm that involves the solution of a series of small maximum bipartite matching problems
  - For ordered trees
    - Solvable in $O(mn)$ time using $O(mn)$ space
Related Work

- Constrained tree inclusion is related to the tree edit problem
  - Insertions are forbidden in tree inclusion
  - Deletions of degree-one and degree-two nodes only are allowed in constrained tree inclusion
  - Constrained tree inclusion is equivalent to degree-two tree edit
    - Solvable for unordered trees in $O(mn \min(\deg(P), \deg(T)))$ time
    - Solvable for ordered trees in $O(mn)$ time using $O(mn)$ space
      [Zhang, 1996; Zhang, Wang, Shasha, 1996]

- Constrained tree inclusion is easier than degree-two tree edit
  - For unordered trees
    - Solvable in $O(m \sqrt{mn})$ time using $O(mn)$ space
    - Simple algorithm that involves the solution of a series of small maximum bipartite matching problems
  - For ordered trees
    - Solvable in $O(mn)$ time using $O(mn)$ space
    - Simple algorithm to find a noncrossing matching covering one of the bipartite sets in an ordered bipartite graph