On the Maximum Common Embedded Subtree Problem for Ordered Trees

Antoni Lozano* and Gabriel Valiente**

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

Abstract. The maximum common embedded subtree problem, which generalizes the minor containment problem on trees, is reduced for ordered trees to a variant of the longest common subsequence problem. While the maximum common embedded subtree problem is known to be APX-hard for unordered trees, an exact solution for ordered trees can be found in polynomial time. In this paper, the longest common balanced sequence problem, and thus the maximum common embedded subtree problem, are solved in $O(n_1 n_2 \min(d_1, \ell_1) \min(d_2, \ell_2))$ time, on ordered trees with n_1 and n_2 nodes, of depth d_1 and d_2, and with ℓ_1 and ℓ_2 leaves.

1 Introduction

An important generalization of tree and subtree isomorphism, known as *minor containment*, is the problem of determining whether a tree is isomorphic to an embedded subtree of another tree, where an embedded subtree of a tree is obtained by contracting some of the edges in the tree. A further generalization of minor containment on trees, known as *maximum common embedded subtree*, is the problem of finding or determining the size of a largest common embedded subtree of two trees. The latter also generalizes the maximum common subtree isomorphism problem [1, Sect. 4.3], in which a common subtree of largest size is contained as a subtree, not only embedded, in the two trees.

Minor containment, maximum common subtree isomorphism, and maximum common embedded subtree are fundamental problems on trees, which find application in various areas of computer science. Minor containment is also known as the *tree inclusion* problem [2, 3], which is often formulated as a node deletion rather than an edge contraction problem. One of the main application areas of the tree inclusion problem is information retrieval, where queries are formulated as trees and answers are embedded subtrees in a semi-structured database represented as a collection of trees. The tree inclusion problem can be solved on unordered trees with n_1 and n_2 nodes in $O(n_1^{1.5} n_2)$ time [4], and the subtree isomorphism algorithm of [5] can be changed to solve the subtree homeomorphism problem in $O((n_1^{1.5}/\log n_1) n_2)$ time. However, none of these algorithms can be extended to solve the maximum common embedded subtree problem, because knowing the sizes of maximum common embedded subtrees between all children

* Partially supported by Spanish CICYT project LOGFAC (TIC2001-1577-C03-02)

** Partially supported by Spanish CICYT project MAVERISH (TIC2001-2476-C03-01)

nodes of one tree and all children nodes of the other tree, is not sufficient to know the size of a maximum common embedded subtree between the trees rooted at their parent nodes. The maximum common embedded subtree problem is known to be NP-hard [3] and APX-hard, but approximable within $\log^2 n$, for unordered trees [6–8], where n is the number of nodes in the trees.

Tree inclusion is a particular case of the tree edit problem [9]. In tree edit, the edit distance between two trees is given by the shortest or the least-cost sequence of tree edit operations (insertion, substitution, and deletion of labeled nodes) that allow one to transform one tree to the other. The tree edit distance problem can be solved on ordered trees with n_1 and n_2 nodes, of depth d_1 and d_2, and with ℓ_1 and ℓ_2 leaves, in $O(n_1 n_2 \min(d_1, \ell_1) \min(d_2, \ell_2))$ time [10, 11]. Further, it can be shown that a least-cost transformation between two trees corresponds to a maximum common embedded subtree of the trees, as long as the tree edit operations fulfill the constraint that the cost of inserting a node plus the cost of deleting a node be less than the cost of substituting a node, as shown for general graphs in [12, 13].

A particular case of the maximum common embedded subtree problem, which is used in computational biology for finding the consensus between evolutionary trees (or phylogenies) for the same set of species, is the maximum homeomorphic agreement subtree problem. Evolutionary trees are unordered trees with each internal node having at least two children and with leaves labeled with distinct symbols (species), and an agreement subtree of two evolutionary trees is an evolutionary tree which is included as a minor in the two given trees. The maximum homeomorphic agreement subtree problem can be solved in $O(n^{1.5} \log n)$ time [14] and in $O(n \log n)$ time for binary trees [15], but is NP-complete for more than two trees [16].

In this paper, the (general) maximum common embedded subtree problem on ordered trees is reduced to a variant of the longest common subsequence problem, for which a dynamic programming algorithm is presented which solves the longest common balanced sequence problem, and thus the maximum common embedded subtree problem, in $O(n_1 n_2 \min(d_1, \ell_1) \min(d_2, \ell_2))$ time, on ordered trees with n_1 and n_2 nodes, of depth d_1 and d_2, and with ℓ_1 and ℓ_2 leaves. The maximum common embedded subtree problem is thus solved within the asymptotic time bound of the fastest tree edit algorithm [10, 11], but with a much simpler algorithm.

Similar reductions of pattern matching problems on ordered trees to string matching problems are known. The restricted subtree isomorphism problem (where a subtree is a whole tree rooted at a node) was reduced to a string edit problem in [17–20], string matching with a don't care symbol was reduced to ordered subtree isomorphism in [21], and the tree edit distance algorithm of [11] was reformulated as a string edit problem in [22]. The maximum common subtree problem can also be reduced to a tree edit distance problem, as done for general graphs in [12], and the tree edit problem on ordered trees with n_1 and n_2 nodes can be solved in $O(n_1^{2.5} n_2)$ time [23]. These are just a few examples of the algorithm design technique of reducing problems on trees to problems on strings and sequences.

All trees considered in this paper will be (rooted) ordered trees, unless explicitly mentioned. The rest of the paper is organized as follows. A particular form of well-formed parenthesis strings, which will be used to describe trees, is introduced and their properties are studied in Section 2. The relationship between embedded subtrees and balanced sequences is further established in Section 3, where the longest common balanced sequence problem is introduced. A dynamic programming algorithm for solving the longest common balanced sequence problem, and thus the maximum common embedded subtree problem, is presented in Section 4 and extended to deal with labeled trees. Finally, some conclusions are outlined in Section 5.

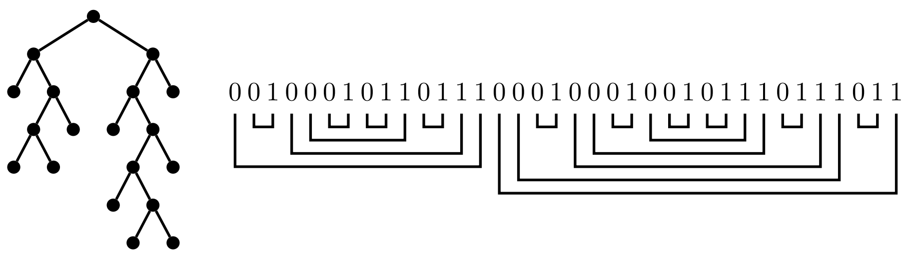
2 Balanced Sequences

Trees will be described by means of a particular form of well-formed parenthesis strings, called *balanced sequences*.

Definition 1. *Let T be a tree with m edges. The balanced sequence of T, denoted by t, is a sequence over $\{0, 1\}$ of $2m$ symbols defined as follows. The balanced sequence of a leaf node is the empty sequence, and the balanced sequence associated to a nonleaf node is obtained by concatenating the balanced sequences of the children of the node, each of them preceded by an additional 0 and followed by an additional 1. The balanced sequence of T is the balanced sequence of the root of T. A string t over $\{0, 1\}$ is a balanced sequence if there is a tree T such that t is the balanced sequence of T.*

The length of a balanced sequence x, denoted by $|x|$, is the number of edges in the tree described by x, that is, the number of 0 characters or the number of 1 characters in x. The empty balanced sequence, which describes the tree with no nodes and no edges, is denoted by λ. Concatenation is indicated by juxtaposition.

Example 1. The sequence shown to the right is the balanced sequence of the tree shown to the left of the following picture. The balanced sequence is annotated with a nested structure, corresponding to the edges of the tree.



Remark 1. Notice that there is a one-to-one correspondence between edges in a tree and edge annotations in the balanced sequence of the tree.

The previous definition yields a recursive algorithm for obtaining the balanced sequence of a tree.

Fact 1 *The balanced sequence of a tree can be obtained in time linear in the size of the tree.*

Remark 2. The balanced sequence of a tree can also be obtained by performing a preorder traversal of the tree, adding a 0 each time an edge is first traversed and adding a 1 each time an edge is traversed in the opposite direction. That is, by performing a leftmost depth-first traversal of the bidirected graph underlying the tree, starting at the root, which is equivalent to finding an Euler trail through the tree [1, Sect. 5.1.2].

A balanced sequence is contained in another balanced sequence if it can be obtained from the latter by deleting edge annotations (character pairs), that is, if the tree represented by the former sequence can be embedded in the tree represented by the latter sequence.

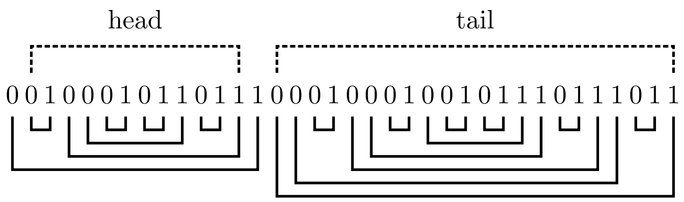
Definition 2. *A balanced sequence s is said to be contained in a balanced sequence t, denoted by $s \subseteq t$, if either $s = t$ or there exist balanced sequences s_1, s_2, s_3 and t_1, t_2, t_3 with $s_i \subseteq t_i$, $1 \leqslant i \leqslant 3$, such that $s = s_1\, s_2\, s_3$ and $t = t_1\, 0\, t_2\, 1\, t_3$. A longest common balanced sequence of s and t is a balanced sequence of largest length among all balanced sequences that are contained in both s and t.*

Example 2. The balanced sequence 00100100101111 contains the following balanced sequences:

00100100101111	000100101111	0000101111	00001111	000111	0011	01	λ
	001000101111	0001001111	00010111	001011	0101		
	001001001111	0001010111	00100111	010011			
	001001010111	0010001111	00101011	010101			
	001010010111	0010010111	01000111				
	010010010111	0010100111	01001011				
		0010101011	01010011				
		0100010111	01010101				
		0100100111					
		0100101011					
		0101001011					

Definition 3. *Let s be a nonempty balanced sequence. The head and the tail of s, denoted respectively by $head(s)$ and $tail(s)$, are the unique balanced sequences such that $s = 0\ head(s)\, 1\ tail(s)$.*

Example 3. The balanced sequence 001000101101110001000100101110111011 of the tree from Example 4 can be partitioned into the head 010001011011 and the tail 0001000100101110111011, both of which are balanced sequences.



Remark 3. The partition of the balanced sequence of a tree into a head and a tail is isomorphic to the partition of a tree into the tree rooted at the first child and the forest whose first tree is rooted at the next sibling of the first child.

Definition 4. *Let s be a balanced sequence. The decomposition of s, denoted by $decomp(s)$, is the set of balanced sequences defined as follows:*

- $s \in decomp(s)$,
- *for all nonempty balanced sequences* $t \in decomp(s)$,
  - $head(t) \in decomp(s)$,
  - $tail(t) \in decomp(s)$,
  - $head(t)\, tail(t) \in decomp(s)$,
- *no other balanced sequence belongs to* $decomp(s)$.

The decomposition of a balanced sequence is thus described by the following recurrence:

$$\begin{aligned} D[\lambda] &= \{\lambda\} \\ D[0x1y] &= \{0x1y\} \cup D[x] \cup D[y] \cup D[xy] \end{aligned}$$

Now, every sequence in the decomposition of a balanced sequence which is a suffix of another balanced sequence, belongs to the decomposition of the latter sequence.

Lemma 1. $D[y] \subseteq D[xy]$ *for all balanced sequences x and y.*

Proof. By induction on $|x|$. Let x and y be balanced sequences. If $|x| = 0$, $D[y] = D[xy]$. Otherwise, let $x = 0x'1y'$. Then,

$$\begin{aligned} D[y] &\subseteq D[y'y] && \text{(by induction hypothesis)} \\ &\subseteq D[xy] && \text{(by definition of } D\text{).} \end{aligned}$$

□

Now, the recurrences

$$\begin{aligned} R[\lambda] &= \{\lambda\} & S[\lambda] &= \{\lambda\} \\ R[0x1y] &= \{0x1y\} \cup R[xy] & S[0x1y] &= R[x] \cup S[xy] \end{aligned}$$

will also be used in the proof of the next lemmata below.

Lemma 2. $D[x] \subseteq D[xy] \cup R[x]$ *for all balanced sequences x and y.*

Proof. By induction on $|x|$. Let x and y be balanced sequences. If $|x| = 0$, $D[x] = \{\lambda\} \subseteq D[xy] \cup R[x]$. Otherwise, let $x = 0x'1y'$. Note first that

$$\begin{aligned} D[x] &= \{x\} \cup D[x'] \cup D[y'] \cup D[x'y'] && \text{(by definition of } D\text{)} \\ &= \{x\} \cup D[x'] \cup D[x'y'] && \text{(by Lemma 1)} \end{aligned}$$

Moreover, $\{x\} \in R[x]$ by definition of R, $D[x'] \subseteq D[xy]$ by definition of D and, further,

$$\begin{aligned} D[x'y'] &\subseteq D[x'y'y] \cup R[x'y'] && \text{(by induction hypothesis)} \\ &\subseteq D[x'y'y] \cup R[x] && \text{(by definition of } R\text{)} \\ &\subseteq D[xy] \cup R[x] && \text{(by definition of } D\text{)} \end{aligned}$$

Therefore, it holds that $D[x] \subseteq D[xy] \cup R[x]$. □

The previous lemmata combine into the following result.

Lemma 3. $D[0x1y] \subseteq \{0x1y\} \cup D[xy] \cup R[x]$ *for all balanced sequences* x *and* y.

Proof. Let x and y be balanced sequences. Then,

$$\begin{aligned} D[0x1y] &= \{0x1y\} \cup D[x] \cup D[y] \cup D[xy] && \text{(by definition of } D\text{)} \\ &= \{0x1y\} \cup D[x] \cup D[xy] && \text{(by Lemma 1)} \\ &\subseteq \{0x1y\} \cup D[xy] \cup R[x] && \text{(by Lemma 2).} \end{aligned}$$ □

The main result in this section is an upper bound on the number of sequences in the decomposition of a balanced sequence.

Lemma 4. $D[zy] \subseteq D[y] \cup R[x]\{y\} \cup S[x]$ *for all* $z \in R[x]$.

Proof. By induction on $|z|$. Let $z \in R[x]$. If $|z| = 0$, $D[zy] = D[y] \subseteq D[y] \cup R[x]\{y\} \cup S[x]$. Otherwise, let $z = 0x'1y'$. Note first that, by Lemma 3, $D[zy] \subseteq \{zy\} \cup D[x'y'y] \cup R[x']$. Now,

- $\{zy\} \subseteq R[x]\{y\}$, by the assumption that $z \in R[x]$.
- Since $|x'y'| < |z|$ and $x'y' \in R[x]$, it follows by induction hypothesis that $D[x'y'y] \subseteq D[y] \cup R[x]\{y\} \cup S[x]$.
- It can be shown by structural induction in $R[x]$ that $R[x'] \subseteq S[x]$. As a matter of fact, if $z = 0x'1y' = x$, then $R[x'] \subseteq S[x]$, by definition of S. Otherwise, $z = 0x'1y' \neq x$ and, by definition of R, it follows that $z = x''y''$ with $0x''1y'' \in R[x]$ and thus, $R[x'] \subseteq S[z] \subseteq S[0x''1y''] \subseteq S[x]$, again by definition of S.

Therefore, $D[zy] \subseteq D[y] \cup R[x]\{y\} \cup S[x]$. □

The following results will also be used in the proof of the upper bound on the cardinal of the decomposition of a balanced sequence.

Corollary 1. $D[x] \subseteq R[x] \cup S[x]$ *for any balanced sequence* x.

Proof. For all balanced sequences x, y, since $x \in R[x]$, it holds by Lemma 4 that $D[xy] \subseteq D[y] \cup R[x]\{y\} \cup S[x]$. Then, taking $y = \lambda$, it follows that $D[x] \subseteq \{\lambda\} \cup R[x] \cup S[x] = R[x] \cup S[x]$. $\square$

Lemma 5. *$S[xy] \subseteq S[x] \cup S[y]$ for all balanced sequences x and y.*

Proof. By induction on $|x|$. If $|x| = 0$, $S[xy] = S[y] \subseteq S[y]$. Otherwise, let $x = 0x'1y'$. Then,

$$\begin{aligned} S[0x'1y'y] &= R[x'] \cup S[x'y'y] \\ &\subseteq R[x'] \cup S[x'y'] \cup S[y] && \text{(by induction hypothesis)} \\ &\subseteq S[x] \cup S[y] && \text{(by definition of } S\text{)}. \end{aligned}$$ $\square$

Now, a bound on the cardinal of R and S will allow one to also bound the cardinal of D. The following fact is easy to prove by induction.

Fact 2 *The cardinal of $R[x]$ is equal to $|x| + 1$, for any balanced sequence x.*

Now, the depth and the number of leaves of a balanced sequence t, denoted respectively by $d(t)$ and $\ell(t)$, are just the depth and the number of leaves of the tree represented by t. They are described by the following recurrences:

$$\begin{aligned} d(\lambda) &= 1 & \ell(\lambda) &= 1 \\ d(0x1y) &= \max(d(x) + 1, d(y)) & \ell(0x1y) &= \ell(x) + \ell(y) \end{aligned}$$

Lemma 6. *$|S[x]| \leqslant |x|d(x) + 1$ for any balanced sequence x.*

Proof. By induction on $|x|$. If $|x| = 0$, $|S[x]| \leqslant |\lambda|d(\lambda) + 1 = 1$. Otherwise, let $x = 0x'1y'$. By Lemma 5, $S[x] \subseteq R[x'] \cup S[x'] \cup S[y']$ and then,

$$\begin{aligned} |S[x]| &\leqslant |R[x']| + |S[x']| + |S[y']| \\ &\leqslant |x'| + 1 + |x'|d(x') + 1 + |y'|d(y') + 1 && \text{(by induction hypothesis)} \\ &\leqslant |x'| + |x'|(d(x) - 1) + |y'|d(x) + 3 && (d(y'), d(x') + 1 \leqslant d(x)) \\ &= (|x'| + |y'|)d(x) + 3 \\ &= (|x| - 1)d(x) + 3 && (|x| = |x'| + |y'| + 1) \\ &= |x|d(x) - d(x) + 3 \\ &\leqslant |x|d(x) + 1 && (x \neq \lambda \text{ and } d(x) \geqslant 2). \end{aligned}$$ $\square$

Lemma 7. *$|S[x]| \leqslant |x|\ell(x) + 1$ for any balanced sequence x.*

Proof. By induction on $|x|$. If $|x| = 0$, $|S[x]| \leqslant |\lambda|\ell(\lambda) + 1 = 1$. Otherwise, let $x = 0x'1y'$. Then, $S[x] = R[x'] \cup S[x'y'] \subseteq R[x'] \cup S[x'] \cup S[y']$, by Lemma 5, and

$$\begin{aligned} |S[x]| &\leqslant |R[x']| + |S[x']| + |S[y']| \\ &\leqslant |x'| + 1 + |x'|\ell(x') + 1 + |y'|\ell(y') + 1 && \text{(by induction hypothesis)} \\ &= |x'| + |x|\ell(x') + |x|\ell(y') + 3 \\ &\quad -(|x| - |x'|)\ell(x') - (|x| - |y'|)\ell(y') \\ &\leqslant |x|\ell(x) + |x'| + 3 - 2|x| + |x'| + |y'| && (\ell(x'), \ell(y') \geqslant 1) \\ &= |x|\ell(x) + |x'| + 3 - 2|x| + |x| - 1 && (|x| = |x'| + |y'| + 1) \\ &= |x|\ell(x) + 2 + |x'| - |x| \\ &\leqslant |x|\ell(x) + 1 && (|x'| + 1 \leqslant |x|). \end{aligned}$$ $\square$

Corollary 2. $|S[x]| \leqslant |x| \min(d(x), \ell(x)) + 1$ *for any balanced sequence* x.

Now, the previous lemmata combine into the following main result.

Theorem 1. $|D[x]| \leqslant |x|(\min(d(x), \ell(x)) + 1) + 1$ *for any balanced sequence* x.

Proof. By Corollary 1, $D[x] \subseteq R[x] \cup S[x]$ and then,

$$\begin{aligned} |D[x]| &\leqslant |R[x]| + |S[x]| - 1 && (\lambda \in R[x] \text{ and } \lambda \in S[x]) \\ &\leqslant |x| + 1 + |x| \min(d(x), \ell(x)) && (\text{by Corollary 2}) \\ &= |x|(\min(d(x), \ell(x)) + 1) + 1. \end{aligned}$$

□

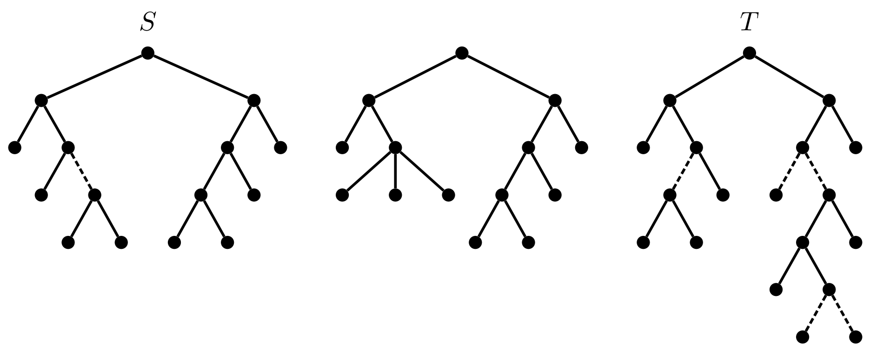
Remark 4. Note that the previous upper bound on the cardinal of the decomposition of a balanced sequence is asymptotically tight, because it is achieved by an infinite number of sequences. As a matter of fact, the decomposition of a balanced sequence that describes the leftist full binary tree with m edges, for all even values of m, which has depth $m/2$ and $m/2 + 1$ leaves, contains $m^2/8 + m/4 + 1 \leqslant m(m/2 + 1) + 1$ sequences.

3 Embedded Subtrees and Balanced Sequences

Maximum common embedded subtree [6, problem GT48] is the problem of finding a tree of largest size that can be embedded into two given trees.

Definition 5. *Let S and T be trees. S is an embedded subtree of T if it can be obtained from T by a series of edge contractions. A common embedded subtree of S and T is a tree which is embedded in both S and T. A maximum common embedded subtree of S and T is a tree of largest size among all common embedded subtrees of S and T.*

Example 4. A maximum common embedded subtree of trees S and T can be obtained by contracting the edges highlighted with dashed lines in the following picture of the trees. The resulting tree, shown in the middle of the picture, has $14 - 1 = 18 - 5 = 13$ edges.

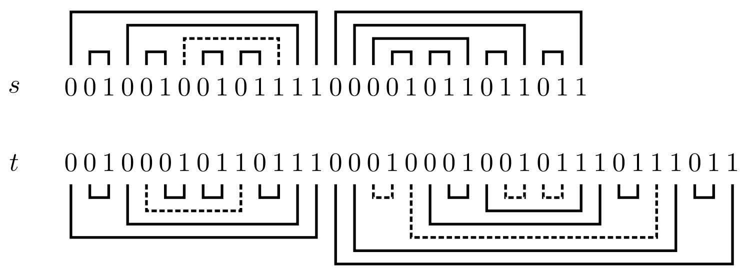


Now, the relationship between embedded subtrees and balanced sequences can be established as follows.

Theorem 2. *A longest common balanced sequence of the balanced sequences of two trees is the balanced sequence of a maximum common embedded subtree of the trees.*

Proof. Contraction of an edge in a tree corresponds to deletion of an edge annotation (one character pair) in the balanced sequence of the tree. A common embedded subtree with the largest number of edges corresponds to a longest common balanced sequence of the balanced sequences of the trees. □

Example 5. A longest common balanced sequence of the balanced sequences for the trees S and T in Exmp. 4 can be obtained by deleting the edge annotations highlighted with dashed lines, together with their adjacent characters. The resulting balanced sequence has $2(14-1) = 2(18-5) = 2 \cdot 13 = 26$ characters.



Remark 5. The longest common balanced sequence problem is related to the longest common subsequence problem for sequences with nested edge annotations, which is useful in the comparison of RNA secondary structures [24]. Although the latter problem is NP-hard [25], it consists in finding a longest common subsequence that preserves all induced edges, and the former problem is the particular case in which all characters are paired by an edge with some other character.

4 Computing Maximum Common Embedded Subtrees

The recursive decomposition of balanced sequences, studied in Section 2, into head, tail, and concatenation of head and tail, is motivated by a natural way in which the maximum common embedded subtree problem can be divided in smaller subproblems.

Lemma 8. *The size of a longest common balanced sequence $lcs(s,t)$ of two balanced sequences s and t is described by the following recurrence:*

$$
\begin{aligned}
&lcs(s,\lambda) = 0\\
&lcs(\lambda,t) = 0\\
&lcs(s,t) = \max \begin{cases} lcs(head(s), head(t)) + lcs(tail(s), tail(t)) + 1,\\ lcs(head(s)\, tail(s), t),\\ lcs(s, head(t)\, tail(t)) \end{cases}
\end{aligned}
$$

Proof. The only common sequence of a balanced sequence and the empty sequence is the empty sequence, which has zero length. Given two nonempty balanced sequences, if the first edge annotation of both sequences belongs to their longest common balanced sequence, then its size is one plus the sum of the sizes of the longest common balanced sequence of the heads and of the longest common balanced sequence of the tails of the two sequences. Otherwise, their longest common balanced sequence is the longest balanced sequence of one of the sequences and the result of deleting the first edge annotation in the other sequence (that is, contracting the first edge in the other tree). □

Now, in order to turn the previous recurrence into an efficient dynamic programming algorithm, a method is needed to map balanced sequences to array positions. Actually, only those sequences resulting from the decomposition of a given sequence need to be taken care of.

Recall from Theorem 1 that the cardinal of the decomposition of a balanced sequence is linear in the size times the minimum of the depth and the number of leaves of the tree represented by the sequence. Definition 4 yields a recursive algorithm for enumerating all sequences in the decomposition of the balanced sequence of a tree, which can be assigned unique numbers by exploiting the idea of [26] that a procedure for dynamically maintaining a global dictionary of unique identifiers, allows partitioning a tree into equivalence classes of restricted subtree isomorphism in expected time linear in the size of the tree. The equivalent problem of assigning unique identifiers to balanced sequences can thus be solved in expected time linear in the cardinal of the decomposition, meaning expected time linear in the size times the minimum of the depth and the number of leaves.

Theorem 3. *The maximum common embedded subtree problem can be solved in in $O(n_1 n_2 \min(d_1, \ell_1) \min(d_2, \ell_2))$ time, on ordered trees with n_1 and n_2 nodes, of depth d_1 and d_2 and with ℓ_1 and ℓ_2 leaves, respectively.*

Proof. Given a balanced sequence s, the unique number $code(t)$ for each sequence $t \in decomp(s)$ is set to either the number already assigned to that sequence (looking it up in a dictionary), or to the next non-assigned number for the decomposition (updating, in this case, the dictionary). Now, for each sequence $t \in decomp(s)$, one unsuccessful dictionary lookup of t and one insertion of $\langle t, code(t) \rangle$ in the dictionary are made. With standard hashing techniques, each such operation takes expected $O(1)$ time and, by Theorem 1, the number of sequences in the decomposition of the balanced sequence of a tree with m edges, depth d and ℓ leaves is $O(m \min(d, \ell))$.

Further, the size of a longest common balanced sequence $lcs(s,t)$ of two sequences s and t is found with standard dynamic programming techniques, where memoization is realized by storing the solution to a subproblem on sequences x and y at entry $a[code(x), code(y)]$ in an integer array a. The size of a longest common balanced sequence $lcs(s,t)$ of two sequences s and t is either looked up at array entry $a[code(s), code(t)]$, or computed according to Lemma 8 and stored in that array entry. There are, by Theorem 1, $O(m_1 \min(d_1, \ell_1))$ and $O(m_2 \min(d_2, \ell_2))$ sequences in the decomposition of the balanced sequences of two trees with respectively m_1 and m_2 edges, depth d_1 and d_2, and ℓ_1 and ℓ_2 leaves and, for each pair of sequences, four array accesses and one array update are made, each taking $O(1)$ time.

The encoding of two trees with m_1 and m_2 edges takes thus expected $O(n_1 \min(d_1, \ell_1) + n_2 \min(d_2, \ell_2))$ time upon which, solving the longest common balanced sequence problem takes $O(n_1 n_2 \min(d_1, \ell_1) \min(d_2, \ell_2))$ time. □

Maximum common labeled subtrees of labeled trees can be found by a simple and straightforward extension of the dynamic programming algorithm. Assume, without loss of generality, that trees have labels on edges. Trees with labels on nodes can be dealt with by shifting node labels to the edge joining the parent with the node, for all nonroot nodes.

Now, for each edge annotation in the balanced sequence of a tree, the edge label can be associated with the corresponding 0 character in the balanced sequence. Then, when computing the size $lcs(s,t)$ of a largest common balanced sequence of sequences s and t, or when mapping the first edge annotation in s to the first edge annotation in t, the case $lcs(head(s), head(t)) + lcs(tail(s), tail(t)) + 1$ of the recurrence in Lemma 8 applies only if the edge labels associated to these first edge annotations are identical, or if they satisfy some predefined criteria, depending on the intended application.

5 Conclusions

The maximum common embedded subtree problem, a generalization of the minor containment problem on trees, is solved in this paper for ordered trees based on a reduction to a variant of the longest common subsequence problem, called the longest common balanced sequence problem. A dynamic programming algorithm is given that solves the longest common balanced sequence problem, and thus the maximum common embedded subtree problem, in $O(n_1 n_2 \min(d_1, \ell_1) \min(d_2, \ell_2))$ time, on ordered trees with n_1 and n_2 nodes, of depth d_1 and d_2 and with ℓ_1 and ℓ_2 leaves, respectively. The maximum common embedded subtree problem is thus solved within the asymptotic time bound of the fastest tree edit algorithm [10, 11], but with a much simpler algorithm.

References

1. Valiente, G.: Algorithms on Trees and Graphs. Springer-Verlag, Berlin (2002)

2. Chen, W.: More efficient algorithm for ordered tree inclusion. Journal of Algorithms **26** (1998) 370–385
3. Kilpeläinen, P., Mannila, H.: Ordered and unordered tree inclusion. SIAM Journal on Computing **24** (1995) 340–356
4. Chung, M.J.: $O(n^{2.5})$ time algorithms for the subgraph homeomorphism problem on trees. Journal of Algorithms **8** (1987) 106–112
5. Shamir, R., Tsur, D.: Faster subtree isomorphism. Journal of Algorithms **33** (1999) 267–280
6. Ausiello, G., Crescenzi, P., Gambosi, G., Kahn, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and Approximation: Combinatorial Optimization Problems and their Approximability Properties. Springer-Verlag (1999)
7. Halldórsson, M.M., Tanaka, K.: Approximation and special cases of common subtrees and editing distance. In: Proc. 7th Ann. Int. Symp. on Algorithms and Computation. Volume 1178 of Lecture Notes in Computer Science., Springer-Verlag (1996) 75–84
8. Zhang, K., Jiang, T.: Some MAX SNP-hard results concerning unordered labeled trees. Information Processing Letters **49** (1994) 249–254
9. Valiente, G.: Constrained tree inclusion. In: Proc. 14th Annual Symp. on Combinatorial Pattern Matching. Volume 2676 of Lecture Notes in Computer Science., Springer-Verlag (2003) 361–371
10. Shasha, D., Zhang, K.: Fast algorithms for the unit cost editing distance between trees. Journal of Algorithms **11** (1990) 581–621
11. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM Journal on Computing **18** (1989) 1245–1262
12. Bunke, H.: On a relation between graph edit distance and maximum common subgraph. Pattern Recognition Letters **18** (1997) 689–694
13. Bunke, H.: Error-correcting graph matching: On the influence of the underlying cost function. IEEE Transactions on Pattern Analysis and Machine Intelligence **21** (1999) 917–922
14. Farach, M., Thorup, M.: Sparse dynamic programming for evolutionary-tree comparison. SIAM Journal on Computing **26** (1997) 210–230
15. Cole, R., Farach, M., Hariharan, R., Przytycka, T.M., Thorup, M.: An $O(n \log n)$ algorithm for the maximum agreement subtree problem for binary trees. SIAM Journal on Computing **30** (2000) 1385–1404
16. Amir, A., Keselman, D.: Maximum agreement subtree in a set of evolutionary trees: Metrics and efficient algorithms. SIAM Journal on Computing **26** (1997) 1656–1669
17. Dublish, P.: Some comments on the subtree isomorphism problem for ordered trees. Information Processing Letters **36** (1990) 273–275
18. Grossi, R.: Further comments on the subtree isomorphism for ordered trees. Information Processing Letters **40** (1991) 255–256
19. Grossi, R.: A note on the subtree isomorphism for ordered trees and related problems. Information Processing Letters **39** (1991) 81–84
20. Mäkinen, E.: On the subtree isomorphism problem for ordered trees. Information Processing Letters **32** (1989) 271–273
21. Verma, R.M.: Strings, trees, and patterns. Information Processing Letters **41** (1992) 157–161
22. Klein, P.N.: Computing the edit-distance between unrooted ordered trees. In: Proc. 6th Annual European Symp. on Algorithms. Volume 1461 of Lecture Notes in Computer Science., Berlin Heidelberg, Springer-Verlag (1998) 91–102

23. Chen, W.: New algorithm for ordered tree-to-tree correction problem. Journal of Algorithms **40** (2001) 135–158
24. Jiang, T., Lin, G.H., Ma, B., Zhang, K.: The longest common subsequence problem for arc-annotated sequences. In: Proc. 11th Annual Symp. on Combinatorial Pattern Matching. Volume 1848 of Lecture Notes in Computer Science., Springer-Verlag (2000) 154–165
25. Lin, G.H., Chen, Z.Z., Jiang, T., Wen, J.: The longest common subsequence problem for sequences with nested arc annotations. In: Proc. 28th Int. Colloquium on Automata, Languages and Programming. Volume 2076 of Lecture Notes in Computer Science., Berlin Heidelberg, Springer-Verlag (2001) 444–455
26. Flajolet, P., Sipala, P., Steyaert, J.M.: Analytic variations on the common subexpression problem. In: Proc. 17th Int. Colloquium on Automata, Languages, and Programming. Volume 443 of Lecture Notes in Computer Science., Springer-Verlag (1990) 220–234