Common intervals of trees

Steffen Heber a, Carla D. Savage b,∗,1

a Department of Computer Science, N. C. State University, Box 7566, Raleigh, NC 27695, USA
b Department of Computer Science, N. C. State University, Box 8206, Raleigh, NC 27695, USA

Received 20 May 2004; received in revised form 1 September 2004
Available online 28 October 2004
Communicated by S.E. Hambrusch

Keywords: Combinatorial problems; Algorithms; Labeled trees

1. Introduction

In this paper we consider the problem of finding common intervals of trees, a generalization of the concept of common intervals in permutations. For a permutation $\pi$ of $[n] = \{1, 2, \ldots, n\}$, an interval of $\pi$ is a set of the form $[\pi(i), \pi(i + 1), \ldots, \pi(j)]$ for $1 \leq i < j \leq n$, and any permutation of $[n]$ has $n(n - 1)/2$ intervals. Given a family $\Pi = (\pi_0, \ldots, \pi_{k-1})$ of $k \geq 2$ permutations of $[n]$, a common interval of $\Pi$ is a subset $S$ of $[n]$ such that $S$ is an interval of $\pi_i$ for $0 \leq i \leq k - 1$.

Common intervals have applications in many different fields. Some genetic algorithms using subtour exchange crossover based on common intervals have been proposed for sequencing problems such as the traveling salesman problem or the single machine scheduling problem [4,10,12]. In a bioinformatical context, common intervals are used to detect possible functional associations between genes [8], to compute the reversal distance between genomes [1], and to define a similarity measure for gene order permutations [2].

A related problem is the consecutive arrangement problem, defined as follows [3,5,6]: Given a finite set $X$ and a collection $S$ of subsets of $X$, find all permutations of $X$ where the members of each subset $S \in S$ occur consecutively. Finding all common intervals of a set of permutations reverses this problem.

Uno and Yagiura [14] presented an optimal $O(n + K)$ time and $O(n)$ space algorithm for finding all permutations of $X$ where the members of each subset $S \in S$ occur consecutively. Finding all common intervals of a set of permutations reverses this problem.

Uno and Yagiura [14] presented an optimal $O(n + K)$ time and $O(n)$ space algorithm for finding all permutations of $X$ where the members of each subset $S \in S$ occur consecutively. Finding all common intervals of a set of permutations reverses this problem.
problem of finding all common intervals in multichromosomal permutations, in signed permutations, and in circular permutations were presented in [8].

Here we further generalize the above approaches to find common intervals in a family of labeled trees. The notion of intervals in permutations is replaced by connected components in trees. If we represent permutations as paths, both definitions coincide. Many of the results remain unchanged, but in contrast to the case for permutations, a tree can have exponentially many intervals. Common intervals in trees could be used as a measure of consensus among trees, and to mine tree-structured data like XML documents, or chemical compounds.

A similar problem is the leaf-agree subtree problem [11], also mentioned as the common subtree problem in [15]. Starting from k rooted trees of size n with only leaves carrying labels, the problem is to find all k-tuples of induced subtrees with the same set of leaf labels (leaf-agree subtrees). The major difference between common intervals and leaf-agree subtrees is that the latter do not consider interior nodes. The number of non-trivial leaf-agree subtrees is bounded by the number of interior nodes, and they can be found in time O(kn) [11].

This article is organized as follows. In Section 2, we introduce basic definitions and observations relating to common intervals in trees. In Section 3 we define irreducible common intervals for a family of trees on n vertices and show that they form a generating set for all the common intervals. We prove that although the number of common intervals could be exponential in n, there can be at most n − 1 irreducible common intervals. In Section 4 we show that the irreducible common intervals of a family of k trees on n vertices can be computed in time O(kn²) and space O(kn).

2. Common intervals of trees

A tree is an undirected, connected, acyclic graph. In a tree, any two vertices are joined by a unique path. If \( G = (V, E) \) is an undirected graph and \( U \subseteq V \), \( G[U] \) denotes the subgraph of \( G \) induced by \( U \). That is, the subgraph with vertex set \( U \) and edges of \( U \) joined by an edge in \( G \) if and only if they are joined by an edge in \( G \).

Let \( T \) be a tree with vertex set \([n] = \{1, 2, \ldots, n\} \). A subset \( S \subseteq [n] \) is an interval of \( T \) if \(|S| > 1\) and \( T[S] \) is connected. Let \( T = (T_0, T_1, \ldots, T_{k-1}) \) be a family of trees, each with vertex set \([n]\). A common interval of \( T \) is a set \( S \subseteq [n] \) such that \( S \) is an interval of \( T_i \) for \( i = 0, \ldots, k-1 \). We denote by \( C_T \) the set of all common intervals of \( T \).

What is the maximum number \( m(n) \) of intervals in a tree \( T \) with \( n \) vertices? Let \( u \) be a leaf of \( T \) with neighbor \( v \). If \( I \) is an interval of \( T \), then \( I = [u, v] \) or at least one of \( I, I - \{u\} \) is an interval of \( T - \{u\} \). Thus, \( m(n) \leq 1 + 2m(n - 1) \), with \( m(1) = 0 \). This gives \( m(n) \leq 2^{n-1} - 1 \) and this maximum is achieved by the \( n \)-vertex star consisting of one center vertex adjacent to \( n - 1 \) vertices of degree 1.

**Corollary 1.** For any family of trees \( T \) with vertex set \( V = [n] \) we have

\[ 1 \leq |C_T| < 2^n. \]

**Lemma 1.** Let \( T = (T_0, T_1, \ldots, T_{k-1}) \) and let \( c, d \in C_T \).

(i) If \( c \cap d \neq \emptyset \), \( c \cup d \in C_T \).

(ii) If \( |c \cap d| \geq 2 \), \( c \cap d \in C_T \).

**Proof.** For \( 0 \leq i \leq k - 1 \), both \( T_i[c] \) and \( T_i[d] \) are connected subgraphs of \( T_i[c \cup d] \). For (i), since \( |c \cup d| > 1 \), it suffices to show that \( T_i[c \cup d] \) is connected. This is clear, since if \( x, y \in c \cup d \) and \( z \in c \cap d \), \( x \) and \( y \) are each joined to \( z \) by paths in \( c \cup d \).

For (ii), let \( x, y \in c \cap d \), with \( x \neq y \). There is a unique path \( p \) from \( x \) to \( y \) in \( T_i \). Since \( T_i[c] \) and \( T_i[d] \) are connected, \( p \) is contained in both intervals, and their intersection. Thus, \( T_i[c \cap d] \) is connected and \( c \cap d \) is a common interval of \( T \). \( \Box \)

Given \( T = (T_0, T_1, \ldots, T_{k-1}) \), define the common interval graph of \( T \), to be the graph \( G_T = (C_T, E_T) \), whose vertex set is the set of common intervals of \( T \) and where \( E_T \) is the set of edges

\[ E_T = \{(c,d) \mid c,d \in C_T, c \neq d, c \cap d \neq \emptyset \}. \]

**Example 1.** Let \( V = \{n\} \) and \( T = (T_0, T_1) \) with \( T_0 = P_n \) a path where the vertices are labeled in their natural order, and \( T_1 = S_{n-1} \) a star with center vertex labeled 1. We have \( C_T = \{[2], [3], \ldots, [n]\} \), and
defined, suppose that distinct elements c, d ∈ C, and a subset V ⊆ C, say that V generates c if

(i) for each d ∈ V, d is a proper subset of c,
(ii) c = \bigcup_{d \in V} d, and
(iii) G[V] is connected.

If there is no such V ∈ C then c is irreducible. Let I_T be the set of irreducible intervals of T. Given T = (T_1, T_2, ..., T_k), let T = T_1 and let E_T be the set of edges of T. Define

'\Theta : E_T \rightarrow I_T'

as follows. For e = (x, y) ∈ E_T, let \Theta(e) be the minimal (with respect to \subseteq) element of C_T containing {x, y}.

**Lemma 2.** \(\Theta\) is a well-defined, onto function.

**Proof.** Let e = (x, y) ∈ E_T. Since \([n] \in C_T\), some element of C_T contains \{x, y\}. To show \(\Theta\) is well-defined, suppose that distinct elements c, d ∈ C_T both contain \{x, y\}. Then c ∩ d ∈ C_T and c ∩ d contains \{x, y\}. Since c ∩ d is a proper subset of at least one of c and d, the intervals c and d cannot both be minimal elements of C_T containing \{x, y\}.

To show \(\Theta(E_T) \subseteq I_T\), suppose for some e = (x, y) ∈ E_T that c = \(\Theta(e)\) is reducible. Then there is a subset V ⊆ C_T which generates c. By minimality of c, no element of V contains both x and y, although some element of V contains x and some element of V contains y. Since G_T[V] is connected, let d_1, d_2, ..., d_l be a path in G_T[V] with x ∈ d_1 and y ∈ d_l. Then \(T[d_i]\) is a connected subgraph of \(T - (x, y)\) for i = 1, ..., l and \(d_i ∩ d_{i+1} ≠ \emptyset\) for i = 1, ..., l − 1. Thus, \(T[d_1] ∪ T[d_2] ∪ \cdots ∪ T[d_l]\) is a connected subgraph of \(T - (x, y)\) containing x and y. This is a contradiction since x and y are disconnected in \(T - (x, y)\).

To show \(\Theta(E_T) = I_T\), assume c is an interval of T which is not in \(\Theta(E_T)\). Let E_T[c] be the set of edges of T[c]. Note that since T[c] is connected \(G_T(\Theta(E_T[c]))\) is also connected. Since c is an interval containing x and y for each e = (x, y) ∈ E_T[c], \(\Theta(e)\) is a proper subset of c, and \(\bigcup \Theta(E_T[c]) \subseteq c\). Since \([x, y] \subseteq \bigcup \Theta(E_T[c])\) for each (x, y) ∈ E_T[c], we also have c ⊆ \(\bigcup \Theta(E_T[c])\), and thus \(\Theta(E_T[c])\) generates c. Thus c ∉ I_T. □

**Corollary 2.** For any family of trees T with vertex set \(V = [n]\) we have

\(1 ≤ |I_T| < n\).

Now we show that for any family of trees T the set I_T of irreducible common intervals, generates the set C_T of all common intervals.

**Lemma 3.** For each c ∈ C_T, there is a subset V ⊆ I_T such that V generates c.

**Proof.** If c ∈ I_T, let V = \{c\}. Otherwise, \(\Theta(E_T[c])\) generates c because \(G_T(\Theta(E_T[c]))\) is connected, and c = \(\bigcup \Theta(E_T[c])\), as shown in the proof of Lemma 2. □

**4. The algorithm**

The algorithm is based on the following lemma, which follows from the definitions.

**Lemma 4.** Let I be an interval in tree T, and r ∈ I. If u is a vertex of T, then the minimal interval of T containing I ∪ \{u\} is I ∪ P(T, u, r), where P(T, u, r) is the set of vertices on the unique path from u to r in T.

We compute irreducible common intervals as follows. For convenience, label the trees \(T_0, \ldots, T_{k-1}\). Each tree has vertex set \([n] = \{1, \ldots, n\}\).

**Preprocessing.**

(A) For each \(T_i = (V_i, E_i)\), choose a root vertex, converting \(T_i\) into a rooted tree, and for each \(v \in V_i\), compute \(p_i(v)\), the parent of \(v\) in the rooted \(T_i\).

(B) Preprocess each (rooted) \(T_i\) in order to compute the least common ancestor \(lca(i, u, v)\) for arbitrary vertices \(u, v \in V_i\) in constant time.
Each tree, step A is clearly linear in $n T$ values of $72$, and introduces new vertices, $\Theta(e)$ after executing minimal interval of fixed $e r_j$ and a vertex $T_j$ preprocessing time is $O(n)$ using a result of Harel and Tarjan [7,13]. Thus the total preprocessing time is $O(k n)$.

The algorithm will proceed by computing $\Theta(e)$ for each edge $e$ in $T_0$. Throughout the computation for fixed $e = (x, y)$, we will keep two sets $B_j, L_j \subseteq V_j$, and a vertex $r_j \in B_j$ for $0 \leq j \leq k - 1$. We also store $L = \bigcup_j L_j$.

Initially we set $B_0 = \{x, y\}, L_0 = \{y\}, r_0 = lca(0, x, y)$, and $B_j = \{x\}, r_j = x, L_j = \emptyset$, for $j = 1, \ldots, k - 1$. We iterate circularly through the $T_j$ and, while processing $T_j$, update $B_j$, the current "best guess" of $\Theta(e)$. Given sets $B_j$ and $L$, we update $B_j$ to be the minimal interval of $T_j$ containing $B_j \cup L$. This introduces new vertices, $L_j$, which must be added to $\Theta(e)$. Set $L$ is updated to reflect this. The algorithm terminates when $L = \emptyset$. We outline the computation in Fig. 1, and procedure UPDATE in Fig. 2.

**Fig. 1.** Overview of the computation the irreducible common intervals of $T = (T_0, T_2, \ldots, T_{k-1})$, after preprocessing.

For each tree, step A is clearly linear in $n$, e.g., via depth-first search, and step B can be done in time $O(n)$ using a result of Harel and Tarjan [7,13]. Thus the total preprocessing time is $O(k n)$.

**Lemma 5.** Assume that $B_j$ is an interval of the rooted tree $T_j$, and that subtree $T_j[B_j]$ has root $r_j$. Let $L^*, B^*_j, L^*_j$, and $r^*_j$ denote the values of the variables after executing \textit{UPDATE}(j, L, B_j, L_j, r_j). We have

(i) $B^*_j$ is the minimal interval of $T_j$ which contains $B_j \cup L$, and $r^*_j$ is the root of $T_j[B^*_j]$.

(ii) $B^*_j$ is the disjoint union of $B_j \cup L$ and $L^*_j$.

**Proof.** (i) For $u \in L$, let $z = lca(j, u, r_j)$. By Lemma 4, the minimal interval containing $B_j \cup \{u\}$ is $S = B_j \cup P(T_j, u, z) \cup P(T_j, r_j, z)$. Then $T_j[S]$ is a subtree of $T_j$ rooted at $z$. Starting at $u$, \textit{UPDATE}(j, \ldots)$ first adds the vertices of $P(T_j, u, z)$ to $B_j$, then starting at $r_j$, it adds the vertices of $P(T_j, r_j, z)$; in both cases, it stops when reaching a vertex already in $B_j$. Repeating for each $u \in L$ with the updated $B_j$ gives the result. (ii) Note that no iteration of \textit{UPDATE} ever puts an element into $L$, or a $B_i$ without marking it. Thus, just before \textit{UPDATE}(j, L, B_j, L_j, r_j), all elements of $B_j \cup L$ are marked. At the end of \textit{UPDATE}(j, L, B_j, L_j, r_j), $L^*_j$ consists of the elements of $B^*_j$ which were unmarked at the beginning of the iteration.
To prove correctness, we show that the following invariants hold after the initialization, and are preserved by successive iterations of UPDATE. Based on this result we show furthermore that $L = \emptyset$ and $B_0 = \emptyset = \cdots = B_{k-1} = \Theta((x, y))$ after a finite number of iterations. Let $INV_j$ be the following collection of properties.

**INV**.$j$.

0. $L$ is the pairwise disjoint union of $L_0, \ldots, L_{k-1}$.
1. For $0 \leq i \leq k-1$, $B_i$ is an interval of $T_i$ and $L_i \subseteq B_i \subseteq \Theta((x, y))$.
2. $B_j \supseteq B_{j-1} \pmod{k} \supseteq \cdots \supseteq B_{j-k+1} \pmod{k}$.
3. For $0 \leq i \leq k-1$, and $i \neq j + 1 \pmod{k}$, $L_i = B_i - B_{i-1} \pmod{k}$.
4. $L = (B_j - B_{j-1} \pmod{k}) \cup L_{j+1} \pmod{k}$.

**Lemma 6.** For $0 \leq j \leq k-1$, if the conditions $INV_{j-1} \pmod{k}$ are true before executing UPDATE$(j, L, B_j, L_j, r_j)$, then the conditions $INV_j$ are true after execution.

**Proof.** Assume that the conditions $INV_{j-1}$ are true before executing UPDATE$(j, L, B_j, L_j, r_j)$, and that $L^*$, $B^*$, $L_j^*$ represent the values of these variables at the end of the call. We show that $INV_j(0 - 4)$ hold.

(0) By $INV_{j-1}(0)$, $L = \bigcup L_i$ is a disjoint union. UPDATE$(j, L, B_j, L_j, r_j)$ replaces $L_j$ in this union by $L_j^*$. By Lemma 5(ii), $L_j^*$ is disjoint from $L$.

(1) Using $INV_{j-1}(1)$, it suffices to look at $i = j$. By Lemma 5 we get that $B^*_j$ is the minimal interval of $T_j$ containing $B_j \cup L$, and $L_j^* \subseteq B^*_j$. Using $INV_{j-1}(0)$ and (1), we get $L_i \subseteq B_i$ for all $i$, and $L = \bigcup L_i \subseteq \bigcup B_i \subseteq \Theta((x, y))$. Thus $\Theta((x, y))$ is an interval of $T_j$ which contains $B^*_j \cup L$. Since $B^*_j$ is minimal with this property we have $B^*_j \subseteq \Theta((x, y))$.

(2) Since UPDATE$(j, L, B_j, L_j, r_j)$ does not alter $B_i$ or $L_i$ for $i \neq j$, by $INV_{j-1}(2)$ it suffices to show that $B^*_j \supseteq B_{j-1}$. By Lemma 5(ii), $B^*_j \supseteq B_j \cup L$. By $INV_{j-1}(4)$, $L = (B_{j-1} - B_j) \cup L_j$. By $INV_{j-1}(2)$, $B_{j-1} \supseteq B_j$. Finally, by $INV_{j-1}(0)$ and (4), $L_j \subseteq L \subseteq B_{j-1}$. Putting these together gives the result, since $B_j \cup L = B_j \cup (B_{j-1} - B_j) \cup L_j = B_j \cup B_{j-1} \cup L_j = B_{j-1}$.

(3) By $INV_{j-1}(3)$ it suffices to show that $L_j^* = B^*_j - B_{j-1}$. By Lemma 5(ii), $B^*_j$ is the disjoint union of $B_j \cup L$ and $L_j^*$, so $L_j^* = B^*_j - (B_j \cup L)$. But as in (2) above, $B_j \cup L = B_{j-1}$.

(4) Taking all subscripts modulo $k$ and using $INV_j(2)$ and (3) proved above,

$$B_j^* - B_{j-1} = (B_j^* - B_{j-1}) \cup (B_{j-1} - B_{j-2}) \cup \cdots \cup (B_{j+k-2} - B_{j+k-1}) = L_j^* \cup L_{j+1} \cup \cdots \cup L_{j+k-2}.$$ 

By $INV_j(0)$ we have $L^* = L_j^* \cup L_{j+1} \cup \cdots \cup L_{j+k-2} \cup L_{j+k-1}, so L^* = (B_j^* - B_{j+k-1}) \cup L_{j+k-1}$.

**Theorem 1.** The procedure FIND-IRC$(T)$ computes the irreducible common intervals of $T$.

**Proof.** Let $(x, y)$ be an edge of $T_0$. Initially, $B_0 = \{x, y\}$, $L_0 = \{y\}$, $L = \{y\}$, and for $1 \leq i \leq k-1$, $B_i = \{x\}$, $L_i = \emptyset$, so the conditions $INV_0$ are satisfied. Thus, by Lemma 6, the conditions $INV_j$ hold at the end of any call to UPDATE$(j, \ldots)$.

To show $L = \emptyset$ after a finite number of iterations, we note that by Lemma 6, $INV_{j-1}(4)$, $L = (B_{j-1} - B_j) \cup L_j$, so if $L \neq \emptyset$ either $L_j \neq \emptyset$ and some element leaves $L$ (see procedure UPDATE, line 1) or $B_{j-1} - B_j \neq \emptyset$ and some element enters $B_j$ (see Lemma 5(ii)). No element enters $L$ more than once, and no element is added to any $B_j$ more than once. Thus, after $n + (k-1)n = kn$ iterations, $L$ must be empty.

From $L = \emptyset$ we conclude by Lemma 6, $INV_j(4)$, that $B_j = B_{j+1}$. Thus by $INV_j(2), B_0 = \cdots = B_k - 1$. Since $x, y \in B_0$ by initialization, $INV_j(1)$ yields that $B_0 \subseteq \Theta((x, y))$ is a common interval containing $\{x, y\}$. By minimality of $\Theta((x, y))$ we get $B_0 = \Theta((x, y))$.

To prove time $O(kn)$ for each $(x, y) \in T_0$, we implement the $B_i$ and the mark function as bit vectors, the $L_i$ as lists, and $L$ as a queue, so that all operations needed on these objects can be performed in $O(k)$ time.
constant time. Initializing all $B_j$, $L_j$, $r_j$ and the mark function takes time $O(kn)$. The time for a call to UPDATE($j$, $L$, $B_j$, $L_j$, $r_j$) is linear in the number of elements added to $B_j$, the number of elements $|L_j|$ deleted from the front of $L$, and the number of elements $|L_j^*|$ added to $L$. Furthermore, since $\text{INV}_{j-1}(4)$ holds, at the beginning of update $L = (B_{j-1} - B_j) \cup L_j$, so if $L \neq \emptyset$, either $L_j \neq \emptyset$ and some elements are deleted from $L$, or $L_j = \emptyset$ and some element is added to $B_j$. Since no $u \in [n]$ is added to $B_j$ or $L$ more than once, summing over all calls to UPDATE for fixed $j$ gives total time $O(n)$ and summing over $j$ from 0 to $k − 1$ gives $O(kn^2)$.

Altogether, since preprocessing takes total time $O(kn)$ (and is done only once), finding all irreducible common interval by computing $\Theta(e)$ for each $e \in E_0$ takes time $O(kn^2)$.

5. Conclusion

We generalized the concept of common intervals in multiple permutations of $n$ elements to common subtrees of multiple labeled trees. While there are at most $n(n − 1)/2$ common intervals, now there might be up to $2^{n−1} − 1$ common subtrees. Despite this difference, there is still a generating subset of irreducible elements of maximal cardinality $n − 1$. Our algorithm to compute this set of irreducible elements uses $O(kn^2)$ time and $O(kn)$ space. In contrast to the case for permutations, it can be shown that a common interval could be generated by different sets of irreducible common intervals. This makes the design of an efficient output-sensitive algorithm an interesting open problem.

Acknowledgement

We are grateful to the referees for their suggestions, which improved the presentation of this material.

References


