# Efficient Merging, Construction, and Maintenance of Evolutionary Trees *

Andrzej Lingas Hans Olsson

Anna Östlin

Department of Computer Science, Lund University
Box 118, S-221 00 Lund, Sweden
e-mail: {Andrzej.Lingas, Hans.Olsson, Anna.Ostlin}@cs.lth.se.

**Abstract**

We present new techniques of *efficiently merging and updating partial evolutionary trees* in the so called experiment model. We show that two partial evolutionary trees for disjoint sets of species can be merged using experiments in time $O(dn)$, where $n$ is the size of the resulting evolutionary tree and $d$ is its maximum degree. We prove our upper time bound for merging evolutionary trees to be asymptotically optimal. We show also that after $O(n \log n)$-time preprocessing, a partial evolutionary tree can be maintained under a sequence of $m$ species insertions or deletions in time $O(dm \log(n+m))$. By applying our algorithm for merging evolutionary trees, or alternatively, our algorithm for updating evolutionary trees, we obtain an $O(dn \log n)$-time bound on the problem of constructing an evolutionary tree of size $n$ and maximum degree $d$ from experiments. The classic $O(n \log n)$-time bound on sorting in the comparison-based model can be seen as a very special case of this upper bound.

*An extended abstract of a preliminary version of the paper appeared in the proceedings of ICALP'99

# 1 Introduction

Consider the problem of constructing an unknown tree using some partial information on the tree topology available at some cost. More precisely, the partial information is in the form of the topological subtree induced by a subset of the leaves and the cost corresponds to the time taken by the construction of the subtree in a given model. We introduce techniques of efficiently merging and updating such partial subtrees and apply them to the efficient construction and maintenance of evolutionary trees in the experiment model.

The problem of constructing evolutionary trees is basic in computational biology. It has been studied extensively in several papers [1, 2, 3, 4, 5, 6, 7]. An *evolutionary tree* is a tree where the leaves represent species and internal nodes represent their common ancestors, see Fig. 1. There are many different approaches to the problem of constructing an evolutionary tree reflecting, among other things, different kinds of available data.

A well known approach to the problem of constructing an evolutionary tree is based on experiments. An *experiment* determines how three species are related in the evolutionary tree, i.e., returns the topological subtree (without weights on the edges) for the three species. For example, the experimental technique of Ahlquist and Sibley [8] may be used to perform such an experiment.

The problem of constructing the unknown evolutionary tree $T$ for the set of $n$ species in the experiment model consists in constructing $T$ within a standard computational model additionally equipped with an oracle which for each query composed of three of the species returns the outcome of the experiment on the three species within one step.

Fig. 1 shows an example of a tree and the outcome of experiments. Note that there are two different types of trees one can get from an experiment.

Interestingly, the problem of sorting distinct numbers $a_1, a_2, \ldots, a_n$ in the comparison-based model can be modeled as a special case of the problem of constructing an evolutionary tree from experiments as follows. The binary tree to construct is like a caterpillar. It is composed of a single rooted path on $n$ nodes and a set of $n$ leaves, each pending from a unique node of the path and labeled uniquely by one of the input numbers, so the labels of the leaves along the path form, say, an decreasing sequence. The outcome of the experiment for a triple $a_{i_1}, a_{i_2}, a_{i_3}$ gives the maximum of the numbers in the triple. Thus, it can be obtained by two comparisons. In case the maximum in the triple is known, such an experiment is equivalent to a single comparison.

The problem of constructing evolutionary trees using experiments have been studied by Kannan, Lawler, and Warnow [6]. In [6], they present several algorithms for this problem. The fastest of them applies to binary evolutionary

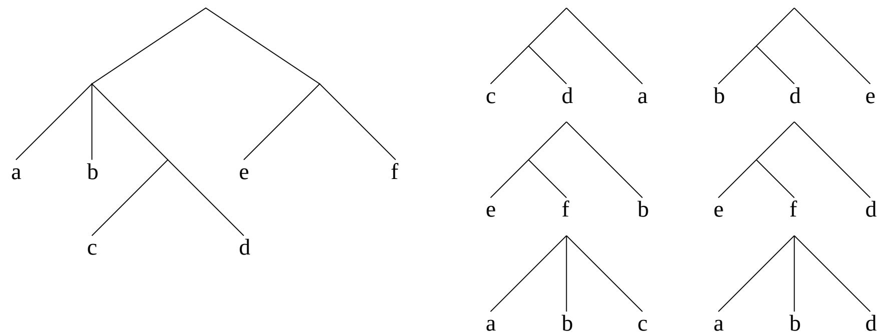


Figure 1: An example an evolutionary tree and the experiment results.

trees, it runs in time $O(n \log n)$ and performs at most $4n\log_2 n$ experiments. Experiments are expensive and hence it is important to minimize their number. The algorithm for binary trees that performs the smallest number of experiments presented in [6] runs in time $O(n^2)$ and requires at most $n \log_2 n$ experiments. For trees not restricted to be binary Kannan *et al.* present an $O(n^2)$-time algorithm which requires $O(dn \log n)$ experiments, where $d$ is the maximum degree in the tree [6]. For the unrestricted degree case Kannan *et al.* present an $O(n^2)$-time algorithm using $O(n^2)$ experiments [6]. Recently, Kao *et al.*, by using the so called randomized tree splitting [7], have shown that an evolutionary tree for $n$ species can be determined from experiments in expected time $O(dn \log n \log\log n)$. In [7], there is also an $\Omega(n(\log n + d))$ bound on the number of experiments required to construct an evolutionary tree of maximum degree $d$ for $n$ species.

In this paper, we introduce new techniques of merging evolutionary trees and updating them for the purpose of their efficient construction and dynamic maintenance in the experiment model. We are not aware of any prior methods for the important issues of efficiently merging and updating evolutionary trees.

Our first main result is an algorithm for merging two evolutionary trees for disjoint sets of species running in time $O(dn)$, where $n$ is the size of the resulting evolutionary tree and $d$ is its maximum degree. Our merging algorithm yields a deterministic algorithm for constructing an evolutionary tree for $n$ species from experiments running in time $O(dn \log n)$. This is a substantial improvement over the corresponding deterministic $O(n^2)$-time bound from [6], and the corresponding randomized $O(dn \log n \log\log n)$-time bound from [7]. Our second main result is a lower bound on the number of experiments necessary to perform in order to merge two evolutionary trees. It asymptotically matches the upper time provided by our merging algorithm proving its asymptotic optimality. Our third main result is an efficient algorithm for updating evolutionary trees in the experiment model. After $O(n \log n)$-time preprocessing, it can maintain a partial evolutionary tree under a sequence of $m$ species insertions or deletions in time $O(dm \log(n+m))$. Observe that this dynamic algorithm yields an alternative proof of our $O(dn \log n)$-time bound on evolutionary tree construction.

In the literature, another variant of the experiment model based on quartets of species is often considered, see e.g. [2]. In this variant, the input to the experiment consists of four different species, say $a,\ b,\ c$ and $d$, and the output is an undirected tree on four leaves labeled with $a,\ b,\ c$ and $d$ respectively, revealing the topology of the subtree of the evolutionary tree induced by the quartet. Note that if we fix some species as the root of the evolutionary tree then all the experiments on quartets including this particular species are equivalent to triple experiments for the other three species. By this simple observation, all upper bounds established in the triple variant of the experiment model immediately carry over to the quartet variant. As for lower bounds, note that the output of an experiment on $a,\ b,\ c$ and $d$ returning a directed tree on four leaves labeled with $a,\ b,\ c,\ d$ can easily be deduced from the output of $O(1)$ experiments on triples in $\{a, b, c, d\}$. Since such a directed tree on $a,\ b,\ c,\ d$ immediately yields the topology of an undirected one on $a,\ b,\ c,\ d$, also the lower bounds valid in the triple variant of the experiment model carry over up to constants to the (undirected) quartet variant.

Section 2 presents our efficient method of merging partial evolutionary trees with disjoint sets of leaves. Section 3 presents our lower bound on the number of experiments necessary to merge two evolutionary trees. Our method of efficiently updating evolutionary trees in the experiment model is given in Section 4.

## 2 Efficiently merging partial evolutionary trees

Throughout the paper, we shall denote by $T$ an arbitrary fixed rooted evolutionary tree for a set of $n$ species. We shall identify the leaves of $T$ with the unique species labeling them. By the *degree* of a node $v$ in a rooted tree, $deg(v)$ for short, we shall mean the number of children of $v$ in the tree.

Given a subset $S$ of leaves of $T$, the *partial evolutionary tree* induced by $S$, denoted by $T \parallel S$, is the minimum size tree that has $S$ as the set of its leaves and is homeomorphic to the subtree of $T$ composed of paths in $T$ joining each pair of leaves in $S$. The node of $T \parallel S$ corresponding to the root of the subtree of $T$ is the root of $T \parallel S$. For a partial evolutionary tree $U$, the set of its leaves will be denoted by $S_U$.

The problem of *merging two partial evolutionary trees* $L$ and $R$ consists in constructing the partial evolutionary tree $T \parallel (S_L \cup S_R)$, denoted by $L \cup R$, on the basis of $L$, $R$, and experiments on triples in $(S_L \cup S_R)^3$.

**Theorem 1** *Let $L$ and $R$ be two partial evolutionary trees with disjoint sets of leaves. The merging of $L$ and $R$ into $L \cup R$ can be done in time $O(\sum_{v \in L \cup R} deg^2(v))$ using at most $\sum_{v \in L \cup R} deg^2(v)$ experiments.*

**Proof.** Let $L_1, ..., L_k$ denote the complete subtrees of $L$ respectively rooted at the children of the root of $L$, and let the subtrees $R_1, ..., R_m$ of $R$ be defined symmetrically.

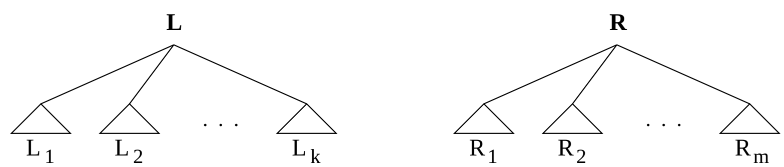


Figure 2: The trees $L$ and $R$.

Six different types of the tree $L \cup R$ can be distinguished, see Fig. 3.

1. The roots of $L$ and $R$ are exactly the children of the root of $L \cup R$.

2. The sequence of the complete subtrees rooted at the children of the root of $L \cup R$ has the form $L_1, ..., L_{i-1}, L_i \cup R, L_{i+1}, ..., L_k$, where $1 \leq i \leq k$.

3. The sequence of the complete subtrees rooted at the children of the root of $L \cup R$ has the form $R_1, ..., R_{i-1}, R_i \cup L, R_{i+1}, ..., R_m$, where $1 \leq i \leq m$.

4. The sequence of the complete subtrees rooted at the children of the root of $L \cup R$ has the form $L_{i_1} \cup R_{j_1}, L_{i_2} \cup R_{j_2}, ..., L_{i_p} \cup R_{j_p}, L_{i_{p+1}}, ..., L_{i_k}, R_{j_{p+1}}, ..., R_{j_m}$.

5. The roots of $L_1, ..., L_k$ and $R$ are exactly the children of the root of $L \cup R$.

6. The roots of $L$ and $R_1, ..., R_m$ are exactly the children of the root of $L \cup R$.

To see that $L \cup R$ is always of one of the six types note that otherwise one of the complete subtrees rooted at the children of the root of $L \cup R$ would exactly

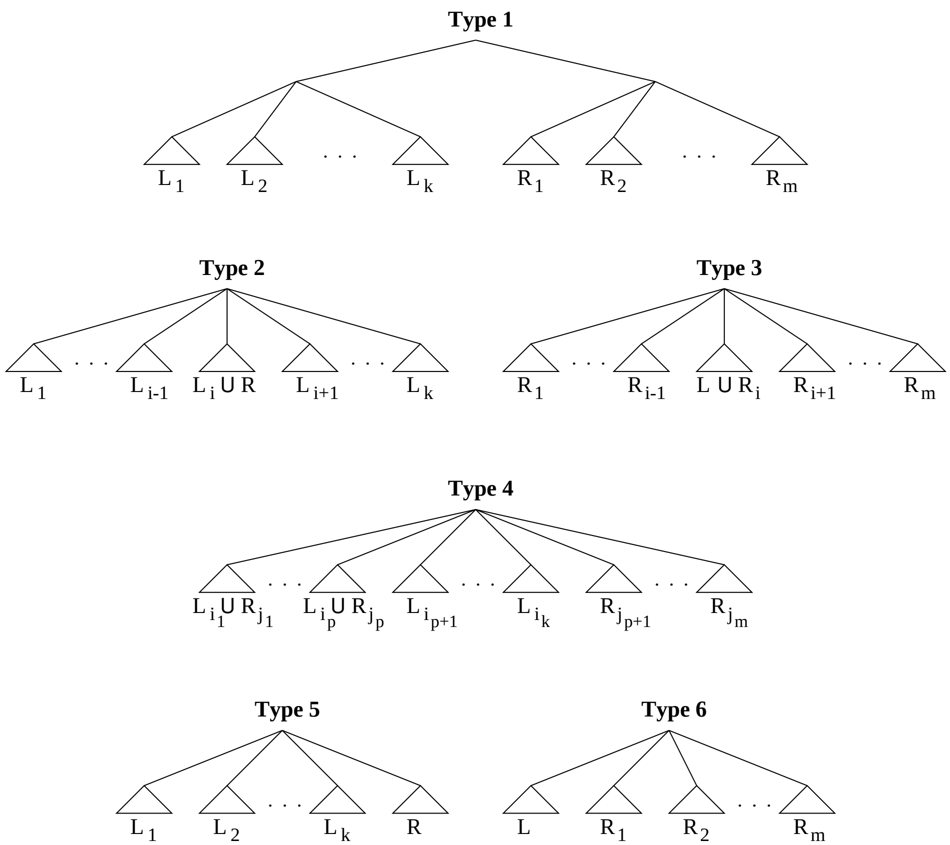


Figure 3: The six different types of the tree $L \cup R$.

include some proper and non-empty subset of $S_{L_i}$ or $S_{R_j}$ for some $1 \leq i \leq k$ or some $1 \leq j \leq m$. Then, the root of $L_i$ or $R_j$ would overlap with that of $L \cup R$ which would contradict the topology of $L$ or $R$.

For $i = 1, ..., k$, let $l_i$ be a fixed leaf in $S_{L_i}$. Similarly, for $j = 1, ..., m$, let $r_j$ be a fixed leaf in $S_{R_j}$.

The time complexity of the following multi-case merging method will depend only on the squares of the degrees in $L \cup R$ and not on the product of the degrees in $L$ and $R$. The method starts by performing the experiment on the triple $l_1,\ l_2,\ r_1$. Then, it branches depending on the outcome of the experiment. To describe the possible outcomes we shall denote the fact that the lowest common ancestor of leaves $a,\ b$ is below that of leaves $a$ and $c$ in $T$ by $((a, b), c)$. If the lowest common ancestors for all pairs from $\{a,\ b,\ c\}$ overlap, we just write $(a, b, c)$.

**Case** $((\mathbf{l_1}, \mathbf{r_1}), \mathbf{l_2})$ $L \cup R$ cannot be of type 1, 5 or 6 here. If it is of type 2 then one of the complete subtrees rooted at the children of the root of $L \cup R$ is $L_1 \cup R$. If it is of type 3 then one of the complete subtrees rooted at the children of the root of $L \cup R$ is $L \cup R_1$. Finally, if it is of type 4 then one of the complete subtrees rooted at the children of the root of $L \cup R$ is $L_1 \cup R_1$.

To determine $L \cup R$, perform the experiment on the triple $l_1,\ l_2,\ r_2$. There are four possible outcomes:

$((\mathbf{l_1}, \mathbf{r_2}), \mathbf{l_2})$, then $L \cup R$ is of type 2 and it remains to construct $L_1 \cup R$ in order to determine $L \cup R$.

$((\mathbf{l_1}, \mathbf{l_2}), \mathbf{r_2})$, then $L \cup R$ is of type 3 and symmetrically it remains to construct $L \cup R_1$ in order to determine $L \cup R$.

$((\mathbf{l_2}, \mathbf{r_2}), \mathbf{l_1})$, then $L \cup R$ is of type 4 and two of the complete subtrees rooted at the children of the root of $L \cup R$ are $L_1 \cup R_1$ and $L_2 \cup R_2$, respectively. To determine the remaining subtrees $L_{i_q} \cup R_{j_q}$ perform the experiments on the triples $l_1,\ l_i,\ r_j$ for all $i,\ j$ satisfying $2 < i \leq k$ and $2 < j \leq m$. Whenever the outcome is $((l_i, r_j), l_1)$, $L_i \cup R_j$ must be one of the complete subtrees rooted at the children of the root of $L \cup R$.

$(\mathbf{l_1}, \mathbf{l_2}, \mathbf{r_2})$, then $L \cup R$ is also of type 4 and similarly it is sufficient to perform the experiments on all the triples $l_1,\ l_i,\ r_j$ where $2 \leq i \leq k$ and $2 \leq j \leq m$.

**Case** $((\mathbf{l_2}, \mathbf{r_1}), \mathbf{l_1})$. This case reduces to the previous one by swapping $L_1$ and $L_2$.

**Case** $(\mathbf{l_1}, \mathbf{l_2}, \mathbf{r_1})$. Similarly as in the two previous cases, $L \cup R$ cannot be of type 1 or 6 here and if it is of type type 3 then one of the complete subtrees rooted at the children of the root of $L \cup R$ is $L \cup R_1$.

To determine $L \cup R$, perform the experiment on the triple $l_1,\ r_1,\ r_2$. If its outcome is:

$((\mathbf{r_1}, \mathbf{r_2}), \mathbf{l_1})$ then $L \cup R$ is of type 2 or 5. In this subcase if there is an index $i$ such that $L_i \cup R$ is one of the complete subtrees rooted at a child of the root of $L \cup R$ then $L \cup R$ is of type 2. We can easily verify whether or not such an index exists, and if so, find it, e.g., by performing the experiments on the triples $l_i,\ l_{i+1},\ r_1$ for all $i$ satisfying $1 \leq i < k$.

$((\mathbf{l_1},\mathbf{r_1}),\mathbf{r_2})$ then $L \cup R$ is of type 3 and $L \cup R_1$ is one of the complete subtrees rooted at the children of the root of $L \cup R$.

$(\mathbf{l_1},\mathbf{r_1},\mathbf{r_2})$ **or** $((\mathbf{l_1},\mathbf{r_2}),\mathbf{r_1})$ then $L \cup R$ is of type 4. By performing the experiments on all the triples $l_i,\ r_j,\ r_{j+1}$ where $1 \leq i \leq k$ and $1 \leq j \leq m-1$, we can easily determine all the subtrees $L_{i_q} \cup R_{j_q}$ where $1 \leq q \leq k$.

**Case** $((\mathbf{l_1},\mathbf{l_2}),\mathbf{r_1})$. $L \cup R$ can be only of type 1, 3 or 6 here. To determine $L \cup R$, perform the experiment on the triple $l_1,\ r_1,\ r_2$. If its outcome is:

$((\mathbf{r_1},\mathbf{r_2}),\mathbf{l_1})$, we are done since $L \cup R$ is simply of type 1.

$(\mathbf{l_1},\mathbf{r_1},\mathbf{r_2})$, $L \cup R$ is of type 3 or 6. In this subcase, if there is an index $j$ such that $L \cup R_j$ is one of the complete subtrees rooted at a child of the root of $L \cup R$ then $L \cup R$ is of type 3, otherwise it is of type 6. We can easily verify whether or not such an index exists, and if so, find it, by performing the experiments on all the triples $l_1,\ r_j,\ r_{j+1}$, where $1 \leq j \leq m-1$.

$((\mathbf{l_1},\mathbf{r_1}),\mathbf{r_2})$ ( or $((\mathbf{l_1},\mathbf{r_2}),\mathbf{r_1})$, respectively) then $L \cup R$ is of type 3 and $L \cup R_1$ (or $L \cup R_2$, respectively) is one of the complete subtrees rooted at the children of the root of $L \cup R$.

**Complexity analysis.** Let $w$ be the root of $L \cup R$. Observe that if $L \cup R$ is respectively of type 1, 2, 3, 4, 5, or 6 then $deg(w) = 2$, $deg(w) = k$, $deg(w) = m$, $deg(w) \geq \max\{k, m\}$, $deg(w) = k+1$, or $deg(w) = m+1$ respectively. Also, we have $k,\ m \geq 2$. Hence, by straightforward examination of all the cases, we infer that for any type of the resulting $L \cup R$, we use at most $deg(w)^2$ experiments apart from the experiments used to merge respective subtrees on subsequent recursion levels. The upper bound on the number of experiments easily follows by induction. To complete the proof it is sufficient to observe that our method requires a number of steps proportional to the number of experiments. □

By Theorem 1 and the sparsity of trees, we obtain the following corollary.

**Corollary 2** *Two evolutionary trees with disjoint sets of leaves can be merged in time $O(dn)$ using at most $dn$ experiments, where $n$ is the size of the resulting tree and $d$ is its maximum degree.*

**Theorem 3** *An evolutionary tree for $n$ species, of maximum degree $d$ can be constructed using experiments in time $O(dn \log n)$.*

**Proof.** Split the set of species (i.e., leaves) into two parts, of size $\lceil \frac{n}{2} \rceil$ and $\lfloor \frac{n}{2} \rfloor$, respectively. Next, we recursively construct the partial evolutionary tree for each of the parts, and merge the two resulting trees. Now, Corollary 2 together with the logarithmic recursion depth yield the theorem thesis. $\square$

If the maximum degree $d$ in the tree is known in advance it is possible to improve the upper time bound in Theorem 3. Simply, we can limit the recursion depth to $\lceil \log_2(n/d) \rceil$, and use the quadratic time algorithm by Kannan *et al.* for the $O(n/d)$ subtrees of size at most $d$ to reduce the upper bound to $O(nd + dn \log(n/d))$.

# 3 The lower bound

To prove that $\Omega(dn)$ experiments are needed to merge two trees into one tree with $n$ nodes and maximum degree $d$, we consider first the case of merging two $d$-stars. A $d$-star is a rooted tree with $d$ leaves which are the only children of the root.

**Lemma 4** *Merging two evolutionary $d$-stars into one tree with $2d$ leaves requires, in the worst case, $\Omega(d^2)$ experiments.*

**Proof.** Consider two isomorphic $d$-stars $L$ and $R$ (i.e., $L$ and $R$ has a root with $d$ leaf-children). Next, consider an algorithm, say $A$, for merging evolutionary trees. We shall act as an adversary constructing the tree $L \cup R$ forcing $A$ to perform $\Omega(d^2)$ experiments. The tree $L \cup R$ after removing its leaves will be isomorphic to $L$ and $R$. All its nodes adjacent to leaves will have degree exactly 2, see Fig. 4.

Let $l_1, ..., l_d$ and $r_1, ..., r_d$ denote the leaves of $L$ and $R$ respectively. The subtrees rooted at the children of the root of $L \cup R$ will have two leaves, one from $L$ and one from $R$. We claim that the algorithm $A$ in order to determine the pairs $l_{i_1},\ r_{j_1}, ..., l_{i_d},\ r_{j_d}$ forming the aforementioned subtrees has to perform in the worst case so many experiments on triples $l_i,\ r_j,\ x$, where $l_i$ is any leaf in $L$ and $r_i$ is any leaf in $R$, that the number of different pairs $(l_i, r_j)$ of leaves induced by the triples is at least $\binom{d}{2}$.

To prove the claim we play the following game with $A$ as long as the following condition holds: *There are still at least two perfect pairings (matchings) of the $l_i$'s with $r_j$'s so the respective pairs could be merged without contradicting experiments performed so far and our current intentions.* To model the game we use an originally complete, bipartite graph $G$ on $\{l_1, ..., l_d\} \cup \{r_1, ..., r_d\}$. Whenever $A$ performs an experiment on a triple $l_i,\ r_j,\ x$, where $x$ is any leaf, we set the outcome of the experiment to be $(l_i,\ r_j,\ x)$. Note that such an outcome

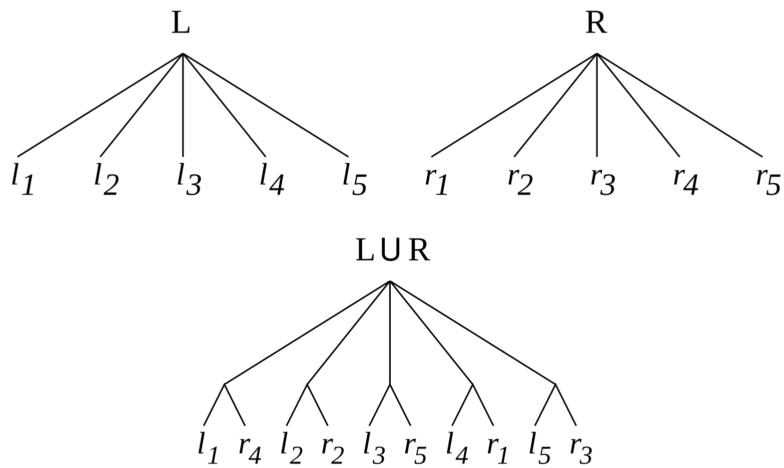


Figure 4: An example of $L \cup R$.

excludes the possibility that the pairs of subtrees corresponding to the pairs of leaves in the experiment performed are the subject of merging. Therefore, we remove the edge (if not already removed) connecting $l_i$ with $r_i$ from the bipartite graph $G$. Also, depending on whether $x$ is a leaf $l_{i'}$ or some $r_{j'}$, we remove the edge connecting $l_{i'}$ with $r_j$ or $l_i$ with $r_{j'}$, respectively. The question is what is the minimum number of aforementioned pairs of leaves induced by the experiments performed by $A$ necessary to finish the game. This minimum number is clearly at least the minimum number of edges necessary to delete from $G$, originally isomorphic to $K_{d,d}$, so there is left a unique perfect matching. The latter number is in turn at least the number of pairs of edges in the perfect matching, i.e., $\binom{d}{2}$, since for each such pair, say $(l_{i_1}, r_{j_1})$, $(l_{i_2}, r_{j_2})$, at least one of the two cross edges $(l_{i_1}, r_{j_2})$, $(l_{i_2}, r_{j_1})$ has to be deleted.

Each experiment triple induces at most two different pairs. We conclude that the algorithm $A$ has to perform at least $\binom{d}{2}/2 = \Omega(d^2)$ experiments to merge the two trees. $\square$

**Theorem 5** *Merging two evolutionary trees with disjoint sets of leaves requires in the worst case $\Omega(dn)$ experiments where $n$ is the number of leaves in the resulting tree and $d$ is its maximum degree.*

**Proof.** Let $l$ be the maximum integer such that $n = 2ld + x$, for some integer $x < 2d$. Consider an arbitrary tree $L$ with maximum degree $d$, $dl$ leaves and exactly $l$ parents of the leaves. Observe that each of the $l$ parents of the leaves

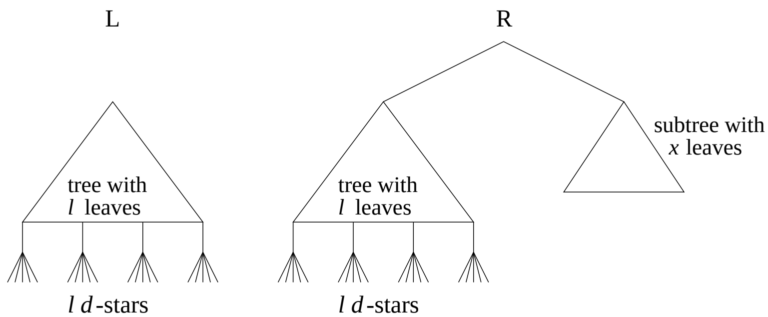


Figure 5: The input trees $L$ and $R$.

is the root of a $d$-star shaped subtree of $L$. Let $R$ be another tree whose root has two children, one is the root of any subtree with $x$ leaves and maximum degree $d$ and the other is the root of a subtree that is isomorphic to $L$ after neglecting leaf labels. See figure 5 for an example.

As an adversary we can force an algorithm for merging $L$ with $R$ to pairwise merge the $l$ $d$-star shaped subtrees of $L$ with the subtrees of $R$ of the same form. The algorithm then has to solve the problem of merging $l$ times two $d$-stars. Note that experiments performed when merging one pair of $d$-stars cannot give any information useful when merging another pair of $d$-stars, since an experiment only includes three leaves, i.e., at most one pair of $d$-stars has at least two leaves in the experiment. From Lemma 4 it follows that any algorithm merging two $d$-stars into one tree with $2d$ leaves and maximum degree $d$, has to perform in the worst case $\Omega(d^2)$ experiments. Since this has to be done $l$ times and $l \cdot \Omega(d^2) = \Omega(dn)$, the theorem holds. $\square$

# 4 Dynamic Construction of Evolutionary Trees

In this section, we present an efficient dynamic algorithm for maintaining an evolutionary tree under a sequence of leaf (species) insertions and deletions in the experiment model. Our algorithm relies on the following fact.

**Fact 6 (see [6])** *Let $v$ be an internal node of the evolutionary tree for a set $U$ of species. Let $W \subseteq U$, and let $W_1, ..., W_q$ be the splitting of $W$ into non-empty components induced by $v$. For any $u \in U \setminus W$, we can determine whether $u$ belongs to any of the components of $U$ induced by $v$ that is a superset of one of the components $W_1, ..., W_q$, and if so, also the index of the component, in time*

*$O(q)$ by performing $\lceil \frac{q}{2} \rceil$ experiments.*

The basic data structure used by our dynamic algorithm is a *tree of leaf separators with threshold $b$* for the current evolutionary tree $T$ of maximum degree $d$ and $0 < b < 1$. The root of the tree of leaf separators is a vertex of $T$ whose removal disconnects $T$ into subtrees of leaf size non-exceeding $b$ times that for $T$. Importantly, only the original leaves of $T$ are accounted for in the leaf size of its subtrees. The children of the root are in turn the roots of the trees of leaf separators for the aforementioned subtrees. Each edge of the tree of leaf separators is labeled by the number of leaves in the subtree of $T$ for which its lower endpoint is the root of the (sub)tree of leaf separators.

The following lemma can be easily proved using the standard idea for finding vertex separators in rooted trees (e.g., see Lemma 3 in [6]).

**Lemma 7** *The tree of leaf separators with threshold $\frac{1}{2}$ for $T$ can be constructed in time $O(l \log n)$, where $l$ is the number of vertices in $T$, and $n$ is the number of leaves in $T$.*

**Proof.** If the root of $T$ satisfies the requirements for the root for the tree of leaf separators with threshold $\frac{1}{2}$, we recursively construct the (sub)trees of leaf separators for the subtrees of $T$ respectively rooted at its children. Then, it remains to append the roots of the (sub)trees of leaf separators to the root. Otherwise, we iterate the aforementioned test for the child of the root for which the subtree rooted at it has more than half of the leaves of $T$, and continue until a vertex passing the test is found. Then we proceed analogously with the subtrees of $T$ resulting from removing the vertex. Importantly, the father of the vertex in $T$ is treated as the root of the subtree of $T$ resulting from removing the vertex and all the subtrees of $T$ rooted at its children. Since each vertex of $T$ is involved in $O(\log n)$ tests, the total time complexity is $O(l \log n)$. □

Given a tree $S$ of leaf separators, with the initial threshold $\frac{1}{2}$ and the maintained threshold $\frac{2}{3}$ for $T$, our dynamic algorithm is simple. In case of leaf insertion, we start from the root of $S$. By Fact 6, we can determine whether the leaf to insert should extend one of the leaf subsets induced by the root or form a new separate one by performing $O(d)$ experiments. In the former case, we go to the child of the root in $S$ corresponding to the leaf subset and proceed recursively. Otherwise, it is sufficient to hang the new leaf at the root. Note that the depth of $S$ is $O(\log n)$, where $n$ is the number of leaves in $T$. Hence, the total time, including experiments, taken by the insertion is $O(d \log n)$. After an insertion of a new single leaf $w$ in $T$, $S$ is not necessarily a tree of leaf separators for the updated $T$. For this reason, we increase by one each number labeling the edges on the path in $S$ tracing the insertion of $w$, provided that the augmented number does not exceed the threshold $\frac{2}{3}$ on the leaf size of the corresponding subtree of $T$. Otherwise, we simply compute from scratch a (sub)tree of leaf separators with threshold $\frac{1}{2}$ for the whole subtree $T'$ of $T$ (extended by $w$) for which the upper endpoint of the edge is the root of the (sub)tree of leaf separators. Consequently, we replace the complete subtree of $S$ rooted at the upper endpoint with the (sub)tree computed from scratch by substituting the root of the latter for the upper endpoint. Since initially the replaced (sub)tree of leaf separators had threshold $\frac{1}{2}$ we can charge at least $\frac{1}{6}$ times the number of leaves in $T'$ leaf insertions with this costly operation. It follows from Lemma 7 that this additional cost of maintaining the tree of leaf separators for $T$ under leaf insertions is at most proportional to the number of leaf insertions times the logarithm of the number of species.

Leaf deletions can be handled similarly. First, we trace the leaf to delete using the tree of leaf separators, then we delete the leaf and update the tree of leaf separators similarly.

**Theorem 8** *After $O(n \log n)$-time preprocessing, the evolutionary tree for $n$ species can be maintained under a sequence of $m$ species insertions and deletions in time $O(dm \log(n+m))$, where $d$ is the maximum degree in the tree.*

Note that as an immediate corollary of Theorem 8, we obtain the $O(dn \log n)$-time bound on the construction of an evolutionary tree of maximum degree $d$ for $n$ species derived in Section 2. This gives an alternative proof of Theorem 3.

# 5 Final Remarks

The merging and updating techniques for evolutionary trees presented in this paper should be useful in finding efficient algorithms for several other problems involving construction and maintenance of unknown or partially unknown trees (both within computational biology as well as outside it, e.g., in distributed computing).

In Introduction we have observed that the problem of sorting distinct numbers in the comparison-based model can be expressed as a special case of the problem of constructing an evolutionary tree from experiments. Our observation immediately yields an $\Omega(n \log n)$ bound on the number of experiments that have to be performed in the worst case in order to construct an evolutionary tree for $n$ species, confirming the information-theoretic $\Omega(n \log n)$ bound from [6]. Our $\Omega(dn)$ bound on the number of experiments that have to be performed in the worst case in order to merge two evolutionary trees into one of size $n$

and maximum degree $d$ clearly yields the corresponding $\Omega(dn)$ bound for the problem of constructing an evolutionary tree. This confirms the $\Omega(dn)$ bound for the evolutionary tree construction problem from [7]. It is an intriguing open problem whether one could establish a lower bound on the number of experiments for the evolutionary tree construction problem that would be superlinear in $nd$ for $d = O(n/\log n)$.

# References


[1] A. Amir and D. Keselman. Maximum agreement subtree in a set of evolutionary trees: Metrics and efficient algorithms. *SIAM Journal on Computing*, 26:1656–1669, 1997.

[2] D. Bryant and M. Steel. Fast algorithms for constructing optimal trees from quartets. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 147–155, 1999.

[3] J. C. Culbertson and P. Rudnicki. A fast algorithm for constructing trees from distance matrices. *Information Processing Letters*, 30(4):215–220, 1989.

[4] M. Farach and M. Thorup. Fast comparison of evolutionary trees (extended abstract). In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 481–488, 1994.

[5] M. R. Henzinger, V. King, and T. J. Warnow. Constructing a tree from homeomorphic subtrees, with applications to computational biology. In *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 333–340, 1996.

[6] S. K. Kannan, E. L. Lawler, and T. J. Warnow. Determining the evolutionary tree using experiments. *Journal of Algorithms*, 21:26–50, 1996.

[7] M. Y. Kao, A. Lingas, and A. Östlin. Balanced randomized tree splitting with applications to evolutionary tree constructions. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, pages 184–196, 1999.

[8] C.G. Sibley and J.E. Ahlquist. Phylogeny and classification of birds based on the data of dna-dna-hybridization. *Current Ornithology*, 1:245–292, 1983.