New algorithms for the duplication-loss model

New Algorithms for the Duplication-Loss Model

M. T. Hallett
Computational Biochemistry Research Group
Dept. of Computer Science, ETH Zürich, Zürich, Switzerland
hallett@inf.ethz.ch
J. Lagergren
Dept. of Numerical Analysis of Computing Science, KTH, Stockholm, Sweden
jensl@nada.kth.se

Abstract. We consider the problem of constructing a species tree given a number of gene trees. In the frameworks introduced by Goodman et al. [GCMRM79], Page [P94], and Guigó, Muchnik, and Smith [GMS96] this is formulated as an optimization problem; namely, that of finding the species tree requiring the minimum number of *duplications* and/or *losses* in order to explain the gene trees.
In this paper, we introduce the WIDTH k DUPLICATION-LOSS and WIDTH k DUPLICATION problems. A gene tree has *width* k w.r.t. a species tree, if the species tree can be reconciled with the gene tree using at most k *simultaneously* active copies of the gene along its branches. We explain w.r.t. to the underlying biological model, why this width typically is very small in comparison to the total number of duplications and losses needed. We show polynomial time algorithms for finding optimal species trees having *bounded width* w.r.t. at least one of the input gene trees. Furthermore, we present the first algorithm for input gene trees that are unrooted. Lastly, we apply our algorithms to a dataset from [GMS96] and show species trees requiring significantly fewer duplications and fewer duplications/losses than those trees given in the original paper.

1 Introduction

The following strategy has become the *de facto* norm for reconstructing the evolutionary relationships of a set of species (i.e. their *species tree*): One begins by constructing phylogenetic trees for a set of distinct gene families (i.e. *gene trees*). Typically, these gene trees are built using one of the standard techniques for constructing phylogenetic trees from molecular sequence data. However, for many gene families, the gene tree differs from the species tree (using another terminology, their topologies disagree). Hence, a single gene tree is not considered sufficient for inferring a species tree. For this reason, a set of gene trees is often used in order to increase the reliability of the resulting species tree. This approach does of course demand a procedure to reconcile the contradictory information contained in the differing gene trees in order to obtain a species tree. Before trying to find such a procedure, one must ask *why* some gene trees differ from the species tree. There are a number of answers to this question e.g. *partial domain agreement*, *horizontal transfer*, *long distance homology* and *parology via gene duplication and loss events*. See [BDDEHY98,GCMRM79,KTG98,TKL97] for general discussions on these subjects.

With respect to paralogy, Goodman et al. [GCMRM79] introduced the concept of *reconciling gene trees with a species tree*; this was later formalized by Page [P94] and Guigó, Muchnik, and Smith [GMS96]. Basically, the authors show how gene tree disagreement can stem from a series of *speciation*, *duplication* and *loss* events. Duplications are genome level evolutionary events that, in essence, copy a contiguous strand of DNA in the genome of a taxa; any genes located along this strand are copied and proceed through evolution independently of each other. Two genes are *paralogous* if their lowest common ancestor can be traced back to a such a duplication event. Two homologous genes are said to be *orthologous* if they evolved from a single gene existing in the genome of their lowest common ancestor taxa. A *loss* is an event where a copy of a gene is lost in the genome of a species. The investigations of [GCMRM79,GMS96,P94] led to a set of optimization problems that ask to find the species tree that requires the minimum number of postulated duplications (the DUPLICATION problem) or the minimum number of postulated duplications and losses (the DUPLICATION-LOSS problem) needed in order to reconcile the set of input gene trees.

A heuristic based on neighbor swapping was given for the DUPLICATION-LOSS problem in [GMS96]; and then applied to a set of 53 gene trees over 16 taxa. Ma et al. [MLZ98] and later [FHKS98] show that the DUPLICATION and DUPLICATION-LOSS problems remain NP-hard for various parameterizations and restrictions. The DUPLICATION-LOSS problem is known to be approximable to within a factor of 2 in polynomial time [MLZ98]; here the authors also give a heuristic derived from their approximation algorithm. It is also known that the DUPLICATION problem is *fixed parameter tractable* when parameterized by the number of duplications [S99].

Here we study, among other things, the WIDTH k DUPLICATION-LOSS problem. A gene tree has *width* k with respect to a species tree, if the species tree can explain the gene tree using at most k *simultaneously* active copies of the gene along any of its branches. The WIDTH k DUPLICATION-LOSS problem asks for the species tree S requiring the minimum number of duplications and losses with the proviso that at least one of the input gene trees has width $\leq k$ w.r.t. to S. The reason for introducing such a parameter is simply that, in many cases, the optimum species tree has small width w.r.t. the input gene trees and, when the width is bounded, the optimum species tree can be found in polynomial time (i.e. fast)[1].

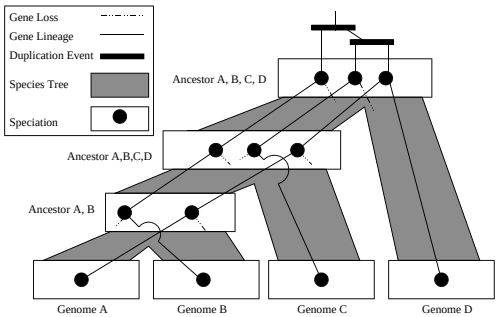

Fig. 1. Gene tree reconciled with species tree.

Figure 1 contains a small example of a species tree $(((A, B), C), D)$ that has large width w.r.t. to the gene tree $((A, D), C), B)$. The width here is 3, since edge $(ABC, ABCD)$ of S has 3 simultaneously active copies of the gene. Note that many of the *NP-hardness gadgets* used in [MLZ98] employ a generalization of this construction using n taxa with conflicting gene trees of the form $(A1, (A2, \ldots, (A_{n-1}, A_n)\ldots)$ and $(A_n, (A_{n-1}, \ldots, (A_2, A_1)\ldots)$. In such a case, the width is $n-1$ and, when interpreted in terms of the underlying biological model, it requires that $n-1$ copies of the same gene co-existed in the genome of the ancestral taxa, only to have $n-2$ of these lost in each of the extant taxa. Although there is evidence that a species benefits from having more copies of some genes in its genome (a possible example is the Hox gene family), such a behavior where many copies co-exist and then are lost in such a pattern seems very degenerate. The *width* k model disallows this.

The optimization problem introduced here asks for the species tree that requires the fewest number of postulated duplications (or, duplications and losses) in order to reconcile the set of input gene trees with this species tree with the proviso that no more than a bounded number k of simultaneously active genes are located in any species at any time. Our four optimization functions are the WIDTH k DUPLICATION problem (find the species tree minimizing duplications with $\leq k$ simultaneously active genes) for rooted and unrooted gene trees, and the WIDTH k DUPLICATION-LOSS problem for rooted and unrooted gene trees.

[1] The species tree given in [GMS96] for 53 gene trees over 16 taxa has a width of only 3 w.r.t. to its gene trees. In contrast, this species tree requires the postulation of 46 duplications (172 duplications and losses). An algorithm bounded by a polynomial with a degree bound of k, the width, will out-perform a similar algorithm with the degree bounded by the duplication cost.

The contributions of this paper are as follows. Section 3 introduces our width k algorithm and its proof of correctness. Section 4 provides the first model and algorithm for input gene trees that are unrooted. Working with unrooted gene trees is advantageous as the determination of roots in phylogenetic tree analysis remains a hard problem, and belongs to the domain of biology rather than to that of mathematics or computer science [HMM96]. Section 5 extends the algorithm from Section 3 to handle *losses*. Lastly, our algorithms are practical and have yielded good results. In Section 6 we present our experimentation with a dataset from [GMS96] and show trees that score better in many respects.

2 Definitions

A graph G consist of a set of vertices denoted $V(G)$ and a set of edges denoted $E(G)$. A directed graph D consist of a set of vertices denoted $V(D)$ and a set of arcs denoted $A(D)$. For a directed graph D, $E(D)$ denotes the edges of the underlying undirected graph of D. A *pseudo tree* is a graph G given together with a set $L_G \subseteq V(G)$, called *leaves*, such that each cycle of G contains a leaf. The vertices of $V(G) \setminus L_G$ are called *internal vertices*. A *leaf path* of a pseudo tree G is path that starts in a non-leaf, ends in a leaf l, and contains no other leaf than l. A *rooted pseudo tree* is a directed graph D given together with a vertex $r \in V(D)$, called the root, and a set $L_D \subseteq V(D)$, called *leaves*, such that each maximal directed path of D starts in the root and ends in a leaf and only leaves have indegree ≥ 2. The vertices of $V(D) \setminus (L_D \cup \{r\})$ are called *internal vertices*. A *directed tree* is a rooted pseudo tree D such that the underlying undirected graph of D is a tree. A *binary pseudo tree* is a pseudo tree where all internal vertices have degree 3. A *rooted binary pseudo tree* is a directed tree, where the root has outdegree 2 and indegree 0, and the rest of the internal vertices have indegree 1 and outdegree 2. Let T be a rooted binary pseudo tree. For any vertex $v \in V(T)$, $L_T(v)$ is the set of leaves of T reachable from v by directed paths. If T is a rooted binary pseudo tree and $v \in V(T)$, then any vertex reachable from v by a directed path is a *descendant* of v (this means that v is a descendant of v). If T is a directed tree and $L \subseteq L_T$, then the *least common ancestor (LCA)* of L in T is defined as follows: if $L = \{l\}$, then the LCA is l; otherwise, the LCA is the vertex v such that $L \subseteq L_T(v)$ but $L \not\subseteq L_T(u)$ for each descendant $u \neq v$ of v. Disjoint union is denoted by $\biguplus$. By a *gene tree* we mean a rooted binary pseudo tree. By a *simple gene tree* we mean a rooted binary tree (i.e. a rooted binary pseudo tree which is also directed). By a *unrooted gene tree* we mean a binary pseudo tree. By a *species tree* we mean a rooted binary tree. A gene tree T and a species tree S are *compatible* if $L_T \subseteq L_S$.

Definition 1. *Let T be a gene tree and S a compatible species tree. The mapping $\lambda_{T,S} : V(T) \to V(S)$ is defined as follows: $\lambda_{T,S}(v)$ is the least common ancestor of $L_T(v)$ in S.*

The width parameter is introduced below.

Definition 2. *Let T be a gene tree and S a compatible species tree. The* width *of T with respect to S is defined to be the maximum of*

$$|\{\overline{(x,y)} \in A(T) : \lambda_{T,S}(x) = a', \lambda_{T,S}(y) = b' \text{ where } a\ (b') \text{ is a descendant of } a'\ (b) \text{ in } S\}|.$$

over all $\overline{(a,b)} \in A(S)$.

Our polynomial time algorithms are based on dynamic programming. The most crucial observation necessary to make is that, for a width bounded by a constant, we need only consider a polynomial number of subproblems. These subproblems are constructed from combining the *partitions* of the input gene trees.

Definition 3. *If T is an unrooted gene tree, then for each edge $(x,y) \in E(T)$, $\mathcal{P}_T(\epsilon) = \{L_1, L_2\}$ where L_1 and L_2 are the leaves reachable by leaf paths in $T \setminus (x,y)$ from x and y, respectively . Moreover, $\mathcal{P}_T = \cup_{e \in E(T)} \mathcal{P}_T(e) \cup \{L_T\}$. Similarly, if T is a (simple) rooted gene tree, then for each arc $\overrightarrow{(x,y)} \in A(T)$, $\mathcal{P}_T(\overrightarrow{x,y}) = L_T(y)$ (this is similar to the unrooted case since $L_T(y)$ is the set of leaves reachable from y by leaf paths). Moreover, $\mathcal{P}_T = \{\mathcal{P}_T(\epsilon) : \epsilon \in A(T)\} \cup \{L_T\}$ and for any $F \subseteq A(T)$, $\mathcal{P}_T(F) = \cup_{\epsilon \in F} \mathcal{P}_T(\epsilon)$. In the rooted as well as the unrooted case, we call $\mathcal{P}_T$ the partitions of T.*

Following [GMS96], we define the duplication cost as follows.

Definition 4. *Let T be a gene tree and S a compatible species tree. Let $\Delta_{T,S} = |\{x : \exists \overrightarrow{(x,y)} \in A(T), \lambda_{T,S}(x) = \lambda_{T,S}(y)\}|$, that is, $\Delta_{T,S}$ is the number of duplications needed to explain the gene tree T under the species tree S. Moreover, let $\Delta_{T_1,\ldots,T_r,S}$ denote $\sum_{i=1}^{r} \Delta_{T_i,S}$, and let $\delta(T_1,\ldots,T_r)$ denote the tree S that minimize $\Delta_{T_1,\ldots,T_r,S}$.*

3 The Rooted Gene Tree Duplication Problem

In this section, we consider the WIDTH k DUPLICATION problem for gene trees (i.e. rooted pseudo trees). The most important elements are the (1) basic dynamic programming algorithm and (2) the algorithm for choosing an appropriate set of partitions $\mathcal{P}$.

3.1 The basic dynamic programming algorithm

If S is the optimal tree, then the dynamic programming algorithm will return an optimal solution when $\mathcal{P}_S$ is guaranteed to be a subset of the subproblems considered.

Definition 5. *A species tree S is $\mathcal{P}$-restricted if and only if $\mathcal{P}_S \subseteq \mathcal{P}$.*

The following algorithm accepts as input a set of gene trees and a $\mathcal{P}$ such that $\delta(T_1,\ldots,T_r)$ is $\mathcal{P}$-restricted and computes $\Delta_{T_1,\ldots,T_r,\delta(T_1,\ldots,T_r)}$. The algorithms runs in polynomial time in the size of the input $T_1,\ldots,T_r$ and $\mathcal{P}$. It is straightforward to modified it so that it also computes $\delta(T_1,\ldots,T_r)$. Let $\mathcal{P}^{(i)}$ denote the sets in $\mathcal{P}$ of size i, i.e. $\{A \in \mathcal{P} : |A| = i\}$. Let $\mathcal{P}^{(\leq i)}$ denote the sets in $\mathcal{P}$ of size $\leq i$, i.e. $\{A \in \mathcal{P} : |A| \leq i\}$.

ALGORITHM MINIMUM NUMBER OF DUPLICATIONS (ROOTED CASE)
input: gene trees $T_1,\ldots,T_r$ **(s.t.** $L(T_i) = L$**) and** $\mathcal{P}$ **(s.t.** $\cup_{A \in \mathcal{P}} A = L$ **and** $L \in \mathcal{P}$**)**
  For $i = 1$ **to** $|L|$
    For each $A \in \mathcal{P}^{(i)}$
      Let $M(A) := \min M(A_1) + M(A_2) + d$ **where** $A_1 \uplus A_2 = A$**,** $A_1, A_2 \in \mathcal{P}^{(\leq i)}$ **and** $d = |\{x : \exists \overrightarrow{(x,y)} \in A(T_i); 1 \leq i \leq r; L(x), L(y) \subseteq A; L(x), L(y) \not\subseteq A_1; L(x), L(y) \not\subseteq A_2\}|$
output: $M(L)$**.**

Here d should be interpreted as the number of duplications needed at vertex a with leaf set $L_S(a) = A$ and with children a_1 and a_2 with $L_S(a_1) = A_1$ and $L_S(a_2) = A_2$ resp.

We now give a definition for an optimal subsolution.

Definition 6. *Let $A \in \mathcal{P}$. We define $\mathcal{D}_{T,S}(A)$ to be the number*

$$|\{x : \exists \overrightarrow{(x,y)} \in A(T) : \lambda_{T,S}(x) = \lambda_{T,S}(y), \text{ and } \lambda_{T,S}(x) \text{ is a descendant of the LCA of } A \text{ in } S\}|$$

(Intuitively, $\mathcal{D}_{T,S}(A)$ is the number of duplications required below the edge e, where $A = \mathcal{P}_T(e) \in A(S)$ is the set that minimizes this number). Moreover, $\mathcal{D}_{T_1,\ldots,T_r}(A,\mathcal{P})$ is defined to be the minimum, taken over all $\mathcal{P}$-restricted species trees S such that $A \in \mathcal{P}_S$, of $\sum_{i=1}^{r} \mathcal{D}_{T_i,S}(A)$.

By the following observation, an optimal solution to a subproblem is equivalent to an optimal solution to the original problem.

Observation 1 *If $\delta(T_1,\ldots,T_r)$ is $\mathcal{P}$-restricted, then $\Delta_{T_1,\ldots,T_r,\delta(T_1,\ldots,T_r)} = \mathcal{D}_{T_1,\ldots,T_r}(L_S,\mathcal{P})$.*

It follows from Lemma 1 that our algorithm correctly computes optimal subsolutions.

Lemma 1. *If $A \in \mathcal{P}^{(i)}$, then $\mathcal{D}_{T_1,\ldots,T_r}(A,\mathcal{P}) = \min \mathcal{D}_{T_1,\ldots,T_r}(A_1,\mathcal{P}) + \mathcal{D}_{T_1,\ldots,T_r}(A_2,\mathcal{P}) + d(A,A_1,A_2)$ where the minimum is taken over all $A_1, A_2 \in \mathcal{P}^{(\leq i-1)}$ such that: $A_1 \cup A_2 = A$ and $A_1 \cap A_2 = \emptyset$, and where $d(A,A_1,A_2)$ equals*

$$|\{x : \exists \overline{(x,y)} \in A(T_i); 1 \leq i \leq r; L(x), L(y) \subseteq A; L(x), L(y) \not\subseteq A_1; L(x), L(y) \not\subseteq A_2\}|$$

Proof. We first prove the inequality

$$\mathcal{D}_{T_1,\ldots,T_r}(A,\mathcal{P}) \leq \min \mathcal{D}_{T_1,\ldots,T_r}(A_1,\mathcal{P}) + \mathcal{D}_{T_1,\ldots,T_r}(A_2,\mathcal{P}) + d(A,A_1,A_2). \tag{1}$$

Assume that S_1 and S_2 are $\mathcal{P}$-restricted gene trees such that: $\mathcal{D}_{T_1,\ldots,T_r}(A_1,\mathcal{P}) = \sum_{i=1}^{r} \mathcal{D}_{T_i,S_1}(A_1)$ and $A_1 \in \mathcal{P}_{S_1}$, and $\mathcal{D}_{T_1,\ldots,T_r}(A_2,\mathcal{P}) = \sum_{i=1}^{r} \mathcal{D}_{T_i,S_2}(A_2)$ and $A_2 \in \mathcal{P}_{S_2}$, respectively. Let $\overline{(a_1',a_1)} \in A(S_1)$ and $\overline{(a_2',a_2)} \in A(S_2)$ be edges such that $A_1 = \mathcal{P}_{S_1}\overline{(a_1',a_1)}$ and $A_2 = \mathcal{P}_{S_2}\overline{(a_2',a_2)}$, respectively. Let S_1' be the subtree of S_1 rooted at a_1. Let S_2' be the subtree of S_2 rooted at a_2. Let S' be the rooted binary tree obtained by attaching a S_1' as left subtree and S_2' as right subtree to a root a. Since S_1 and S_2 are $\mathcal{P}$-restricted, and moreover $A \in \mathcal{P}$, S' is $\mathcal{P}$-restricted. Let S be any $\mathcal{P}$-restricted gene tree with S' as a rooted subtree. It is straightforward to verify that $\sum_{i=1}^{r} \mathcal{D}_{T_i,S}(A) = \mathcal{D}_{T_1,\ldots,T_r}(A_1,\mathcal{P}) + \mathcal{D}_{T_1,\ldots,T_r}(A_2,\mathcal{P}) + d(A,A_1,A_2)$. This equality yields (1).

We now prove the inequality

$$\mathcal{D}_{T_1,\ldots,T_r}(A,\mathcal{P}) \geq \min \mathcal{D}_{T_1,\ldots,T_r}(A_1,\mathcal{P}) + \mathcal{D}_{T_1,\ldots,T_r}(A_2,\mathcal{P}) + d(A,A_1,A_2). \tag{2}$$

Assume that S is a $\mathcal{P}$-restricted gene tree such that: $\mathcal{D}_{T_1,\ldots,T_r}(A,\mathcal{P}) = \sum_{i=1}^{r} \mathcal{D}_{T_i,S}(A)$ and $A \in \mathcal{P}_S$. Let $\overline{(a',a)} \in A(S)$ be an edge such that $A = \mathcal{P}_S\overline{(a',a)}$. Let S' be the subtree of S rooted at a. Let S_1 and S_2 be the left and the right subtree of S at a; moreover, let $A_1 = L_{S_1}$ and $A_2 = L_{S_2}$. It is straightforward to verify that $\mathcal{D}_{T_1,\ldots,T_r}(A,\mathcal{P}) = \sum_{i=1}^{r} \mathcal{D}_{T_i,S}(A) = \sum_{i=1}^{r} \mathcal{D}_{T_i,S_1}(A_1) + \sum_{i=1}^{r} \mathcal{D}_{T_i,S_2}(A_2) + d(A,A_1,A_2)$ This equality yields (2).

We are now ready to state the main result of this section.

Lemma 2. *Let $T_1,\ldots,T_r$ be gene trees. Assume that $\delta(T_1,\ldots,T_r)$ is $\mathcal{P}$-restricted. Given $\mathcal{P}$ and $T_1,\ldots,T_r$ the value $\Delta_{T_1,\ldots,T_r,\delta(T_1,\ldots,T_r)}$ (and the tree $\delta(T_1,\ldots,T_r)$) can be computed in polynomial time (the exact running time is stated below).*

Proof. The lemma follows easily from Observation 1 and Lemma 1 above.

The running time of the algorithm is $O(p^2 \cdot a \cdot r \cdot l^2)$ where $p = |P|$, $l = |\cup L_{T_i}|$, and a is the time needed to access a set $A \in \mathcal{P}$. Notice that $a \leq \log p \leq l$ and that $a = O(1)$ expected time. Each element $A \in P$ has at most p candidate sets A_1 and A_2, since A and A_1 uniquely determines A_2. For each triplet A, A_1, A_2, the number of duplications at this vertex in the species trees must be calculated for each of the $O(l)$ edges in each of the r gene trees. Each such edge requires $O(l)$ time giving a bound of $O(r \cdot l^2)$ in total.

3.2 How to find the right $\mathcal{P}$

In this subsection, we begin by showing that, if all but one copy of each species is removed from a gene tree, then the width will not increase. We will use this result later when we show how an adequate set $\mathcal{P}$ can be generated from such simple gene trees (i.e. a set $\mathcal{P}$ adequate for the original - not necessarily simple - gene trees). First, we formally define the simplification procedure.

Definition 7. *Let T be a gene tree (i.e. a rooted binary pseudo tree). Let T' be obtained by taking a rooted binary subtree T'' of T with the same leaf set as T and removing each vertex with indegree 1 and outdegree 1 (recursively) by connecting its two neighbors. The simple gene tree T' is said to be obtained by* simplifying T using T''. *Notice that $V(T'') = V(T)$.*

The following observation is a straightforward consequence of the definition of LCA.

Observation 2 *Let T be a gene tree and S a compatible species tree. If y is a descendant of x in T, the vertex $\lambda_{T,S}(y)$ is a descendant of $\lambda_{T,S}(x)$ in S. Let T' be a simple gene tree obtained by simplifying T. For each t of $V(T')$, the vertex $\lambda_{T',S}(t)$ is a descendant of $\lambda_{T,S}(t)$.*

By the lemma below, simplification does not increase the width. Since simplification clearly does not change the leaf set, it follows that it is sufficient to consider subproblems in our dynamic programming algorithm that are created from simple gene trees.

Lemma 3. *Let T be a gene tree and S a compatible species tree. If T' is a simple gene tree obtained by simplifying T (using T''), then the width of T' w.r.t. S is at most the width of T w.r.t. S.*

Proof. Let $\overline{(a,b)}$ be an edge of S. Let

$$A = \{\overline{(x,y)} \in A(T) : \lambda_{T,S}(x) = a', \lambda_{T,S}(y) = b' \text{ where } a \ (b') \text{ is a descendant of } a' \ (b) \text{ in } S\}$$

and define A' similarily but over all $\overline{(x,y)} \in A(T')$. For any vertex y of T'', let $f_{T''}(y)$ be the parent of y in T''. For any vertex y of T', let $f_{T'}(y)$ be the parent of y in T'. We now prove the theorem by showing the existence of an injective function $g : A' \to A$.

Let $(f_{T'}(y), y) \in A'$. Notice that, by Observation 2, $\lambda_{T',S}(f_{T'}(y))$ is a descendant of $\lambda_{T,S}(f_{T'}(y))$. Hence, there is a unique descendant z of $f_{T'}(y)$ in T'' which also is an ancestor of y in T'' such that $(f_{T''}(z), z) \in A$. Let $g(f_{T'}(y), y) = (f_{T''}(z), z)$.

Assume that $(f_{T'}(y_1), y_1), (f_{T'}(y_2), y_2) \in A'$ and that $g(f_{T'}(y_1), y_1) = g(f_{T'}(y_2), y_2)$. It follows that there is vertex z of T'' such that (1) z is an ancestor of y_1 as well as of y_2 in T'', and (2) z has a parent $f_{T''}(z)$ in T'' such that $(f_{T''}(z), z) \in A$. Notice that (1) implies that z is an ancestor of y_1 as well as y_2 in T'. This together with Observation 2 shows that $(f_{T''}(z), z) \in A$ contradicts $(f_{T'}(y_1), y_1) \in A'$.

We now show how to construct an adequate polynomial sized set $\mathcal{P}$ from the gene trees. Recall that we are given gene trees $T_1, T_2, \ldots, T_r$ such that $T_i \subseteq L_S$ and T_i has width $\leq k$ w.r.t. S. By the above lemma, we may w.l.o.g. assume that they all are simple gene trees. The simplest strategy would be use all of the r gene trees, thus generating a set $\mathcal{P}$ of size $\prod_{i=1}^{r} |A(T_i)|^k$. For large r and k, the size of this set will approach $2^{|L_S|}$. However, we may be able to do better than this via Lemma 4. The intuition behind this technique is to choose a subset of the gene trees (call them $T_1, \ldots, T_\rho$) from which to construct our $\mathcal{P}$ set.

Lemma 4. *Let $T_1, \ldots, T_\rho$ be simple gene trees and S a compatible species tree. If $T_1, \ldots, T_\rho$ all have width $\leq k$ with respect to S and $L_S \setminus (\cup_{i=1}^{\rho} L_{T_i}) = R$, then $\mathcal{P}_S \subseteq \mathcal{P}$ where*

$$\mathcal{P} = \{B \cup \bigcup_{1 \leq i \leq \rho 1 \leq j \leq k} A_{i,j} : A_{i,j} \in \mathcal{P}_{T_i}, B \subseteq R.\}$$

Moreover, $|\mathcal{P}| \leq 2^{|R|} \prod_{i=1}^{\rho} |A(T_i)|^k$.

Proof. Let $\mathcal{P}'_S = \{A \setminus R : A \in \mathcal{P}_S\}$, $\mathcal{P}' = \{A \setminus R : A \in \mathcal{P}\}$, $\mathcal{P}(F) = \cup_{i=1}^{\rho} \mathcal{P}_{T_i}(F)$, $\mathcal{P}'(F) = \{A \setminus R : A \in \mathcal{P}(F)\}$, and $\mathcal{P}'_{T_i}(F) = \{A \setminus R : A \in \mathcal{P}_{T_i}(F)\}$ For each $\overline{(a,b)} \in A(S)$, let $F_i\overline{(a,b)}$ be the set

$$\{\overline{(x,y)} \in A(T_i) : \lambda_{T,S}(x) = a', \lambda_{T,S}(y) = b' \text{ where } a \ (b') \text{ is a descendant of } a' \ (b) \text{ in } S\}$$

and let $F\overline{(a,b)} = \cup_{i=1}^{\rho} F_i\overline{(a,b)}$. Observe that, since T_i has width $\leq k$ with respect to S, $|F_i(\overline{e})| \leq k$ for each $e \in A(S)$.

Define the height of an arc $\overline{(a,b)} \in A(S)$ to be the length of the longest directed path from b to a leaf of S. We prove, by induction on the height of $\overline{e}$, that

$$\mathcal{P}'_S(\overline{e}) \subseteq \mathcal{P}'(F(\overline{e})), \tag{3}$$

for each $\overline{e} \in A(S)$ which yields

$$\mathcal{P}'_S \subseteq \mathcal{P}' \tag{4}$$

After (3) is proven, we derive the lemma from (4).

Since $\lambda_{T,S}|_{L_T}$ is the identity mapping, (3) holds for each arc of height 0 of S. Assume that $\overline{(a,b)} \in A(S)$ has height $i > 0$. Let $\overline{(b,c_1)}$ and $\overline{(b,c_2)}$ be two distinct arcs of S. Since $\overline{(a,b)} \in A(S)$ has height $i > 0$, $\overline{(b,c_1)}$ as well as $\overline{(b,c_2)}$ has height at most $i-1$, and hence satisfy (3). This means that we will be able to conclude that $\overline{(a,b)}$ satisfies (3), if the following two statements hold:

$$\mathcal{P}'_S(\overline{(a,b)}) = \mathcal{P}'_S(\overline{(b,c_1)}) \cup \mathcal{P}'_S(\overline{(b,c_2)}) \tag{5}$$

$$\mathcal{P}'(F\overline{(b,c_1)}) \cup \mathcal{P}'(F\overline{(b,c_2)}) \subseteq \mathcal{P}'(F\overline{(a,b)}) \tag{6}$$

The equality (5) is straightforward to verify. We will now end the proof of (3) by showing

$$\mathcal{P}'_{T_i}(F_i\overline{(b,c_1)}) \cup \mathcal{P}'_{T_i}(F_i\overline{(b,c_2)}) \subseteq \mathcal{P}'_{T_i}(F_i\overline{(a,b)}) \tag{7}$$

which clearly implies (6).

Assume that $\overline{(x,y)} \in F_i\overline{(b,c_i)}$ where $i \in \{1,2\}$. If $\overline{(x,y)} \notin F_i\overline{(a,b)}$ then $\lambda_{T_i,S}(x) = b$. Let r be the root of T. Clearly $\lambda_{T_i,S}(r)$ is the root of S and a is a descendant of $\lambda_{T_i,S}(r)$. It follows that there are vertices x' and x'' of T such that x' is the parent of x'', x is a descendant of x'', $\lambda_{T_i,S}(x'') = b$, and a is a descendant of $\lambda_{T_i,S}(x')$. From this, it follows that $\overline{(x',x'')} \in F_i\overline{(a,b)}$ which gives $\mathcal{P}_{T_i}(F_i\overline{(b,c_i)}) \subseteq \mathcal{P}_{T_i}(F_i\overline{(a,b)})$. Thus (7) holds and hence also (6) which yields (3).

We now show that (4) implies the lemma. Assume that $A \in \mathcal{P}_S$. Let $A' = A \setminus R$. Clearly $A' \in \mathcal{P}'_S$ and hence $A' \in \mathcal{P}'$ which yields that $A = A' \cup (A \cap R) \in \mathcal{P}$ and the lemma follows. QED

Thus, by Lemma 4, we may choose a subset of the gene trees to cover the total leaf set L_S as completely as possibly but simultaneously manage to keep ρ reasonably small. In particular, we want to choose gene trees in a way that minimizes $2^{|R|} \prod_{i=1}^{\rho} |A(T_i)|^k$, where R is the set of leaves not appearing in any of $T_1, \ldots, T_\rho$.

Corollary 1 *Let T be a simple gene tree and S a species tree such that $L_T = L_S$. If T has width $\leq k$ with respect to S, then $\mathcal{P}_S \subseteq \{\cup_{i=1}^{k} A_i : A_i \in \mathcal{P}_T\}$.*

4 The Unrooted Gene Tree Duplication Problem

Previous work on the DUPLICATION/DUPLICATION-LOSS problems focused on input gene trees that are rooted. For various reasons, the determination of the root of a phylogenetic tree is a difficult problem that, in many cases, requires the expertise of biologists. For this reason, we introduce the UNROOTED DUPLICATION problem. Within this paper, we do not consider the DUPLICATION-LOSS problem but note that it is straightforward to modify the following algorithm for this case (Section 5).

As input, we are given unrooted gene trees $T_1, \ldots, T_r$ and asked to compute $\Delta(T'_1, \ldots, T'_r)$ where (1) T'_i is T_i rooted at one of its vertices and (2) $T'_1, \ldots, T'_r$ minimize $\Delta(T'_1, \ldots, T'_r)$ with respect to (1). Since $\mathcal{P}_{T'_1} \subseteq \mathcal{P}_{T_1}$, the set $\mathcal{P}_S$ can be constructed as before from $T_1, \ldots, T_r$. Given the subproblems defined by $\mathcal{P}_S$, the algorithm proceeds in the same manner as in Section 3 with one exception. Presently, we must store subsolutions for each of the possible ways of rooting each unrooted gene tree. Fortunately, each tree can be considered separately.

ALGORITHM MINIMUM NUMBER OF DUPLICATIONS (UNROOTED CASE)
input: unrooted gene trees $T_1, \ldots, T_r$ **(s.t.** $L(T_i) = L$**) and** $\mathcal{P}$ **(s.t.** $\cup_{A \in \mathcal{P}} A = L$ **and** $L \in \mathcal{P}$**)**
  For $i = 1$ **to** $|L|$
    For each $A \in \mathcal{P}^{(i)}$
      Let $T_{i,j}$ **be** T_i **rooted at its** j^{th} **vertex.**
      (These trees do not have to be constructed).
      Let $M(A)_{i,j} := \min M(A_1)_{i,j} + M(A_2)_{i,j} + d_{i,j}$**, where** $A_1 \biguplus A_2 = A$**,** $A_1, A_2 \in \mathcal{P}^{(\leq i)}$
        and
  $d_{i,j} = |\{x : \exists \overline{(x,y)} \in A(T_{i,j}); 1 \leq i \leq r; L(x), L(y) \subseteq A; L(x), L(y) \not\subseteq A_1; L(x), L(y) \not\subseteq A_2\}|$
output: $\sum_{1 \leq i \leq r} \min_j \{M(L)_{i,j}\}$.

In the above algorithm, $d_{i,j}$ is the number of duplications needed with respect to $T_{i,j}$ at a vertex a with leaf set A and children a_1 and a_2 s.t. $L_S(a_1) = A_1$ and $L_S(a_2) = A_2$.

Theorem 1. *The algorithm correctly computes* $\Delta(T'_1, \ldots, T'_r)$ *and* $T'_1, \ldots, T'_r$ *minimize* $\Delta(T'_1, \ldots, T'_r)$. *The running time of the algorithm is* $O(p^2 \cdot r \cdot a \cdot l^3)$, *where* $p = |P|$, $l = |\cup L_{T_i}|$, *and* a *is the time needed to access a set* $A \in \mathcal{P}$.

Proof. The proof is similar to that of Section 3.2.

5 The Rooted Gene Tree Duplication-Loss Problem

Following [GMS96], we define the *losses* at a vertex a with children b and c of a gene tree T with respect to a species tree S as:

$$l_S(a) = \begin{cases} 0 & \text{if } \lambda_{T,S}(a) = \lambda_{T,S}(b) = \lambda_{T,S}(c) \\ |d_S(\lambda_{T,S}(a), \lambda_{T,S}(b)) - 1| + |d_S(\lambda_{T,S}(a), \lambda_{T,S}(c)) - 1| & \text{otherwise} \end{cases}$$

where $d_S(x, y)$ is the distance from x to y in S (i.e. the number of edges on the unique path between them).

Definition 8. *Let* $\Lambda_{T,S}$ *be the number of losses needed to explain the gene tree* T *under the species tree* S, *i.e.* $\Lambda_{T,S} = \sum_{x \in V(T_i)} l_S(x)$. *Moreover, let* $\Sigma_{T_1,\ldots,T_r,S}$ *denote* $\sum_{i=1}^{r} \Delta_{T_i,S} + \Lambda_{T_i,S}$, *and let* $\sigma(T_1, \ldots, T_r)$ *denote the tree* S *that minimize* $\Sigma_{T_1,\ldots,T_r,S}$.

To facilitate the formulation of a dynamic programming algorithm that also handles losses, we give a procedure to count losses by only inspecting a parent and its two children. Let

$\chi(L, A, A_1, A_2)$ be true if and only if $L \subseteq A$ and $L \not\subseteq A_i$ for $i = 1, 2$. Let x be a vertex of a tree T with a parent f and children s_1, s_2, and let A, A_1, A_2 be three sets such that A is the disjoint union of A_1 and A_2. We say that a loss happens at A, A_1, A_2 w.r.t. x and T if and only if either (1) or (2) holds:

(1) $L_T(x) \subseteq A$, $\chi(L_T(x), A, A_1, A_2)$, $\chi(L_T(s_1), A, A_1, A_2)$, and not $\chi(L_T(s_2), A, A_1, A_2)$
(2) $L_T(x) \subseteq A$, $L_T(f) \not\subseteq A$, and not $\chi(L_T(x), A, A_1, A_2)$

Let l be the function defined as follows

$$l(T, x, A, A_1, A_2) = \begin{cases} 1 \text{ if a loss happens at } A, A_1, A_2 \text{ w.r.t. } x \text{ and } T \\ 0 \text{ otherwise} \end{cases}$$

ALGORITHM MINIMUM NUMBER OF DUPLICATIONS AND LOSSES (ROOTED CASE)
input: gene trees $T_1, \ldots, T_r$ (such that $L(T_i) \subseteq L$) and $\mathcal{P}$ (such that $\cup_{A \in \mathcal{P}} A = L$ and $L \in \mathcal{P}$)
For $i = 1$ to $|L|$
For each $A \in \mathcal{P}^{(i)}$
Let $M(A) := \min M(A_1) + M(A_2) + d + l$ where $A_1 \uplus A_2 = A$, $A_1, A_2 \in \mathcal{P}^{(\leq i)}$,
$d = |\{x : \exists (x, y) \in A(T_i); 1 \leq i \leq r; L(x), L(y) \subseteq A; L(x), L(y) \not\subseteq A_1; L(x), L(y) \not\subseteq A_2\}|$,
and
$l = \sum_{x \in V(T_i)} l(T_i, x, A, A_1, A_2)$
output: $M(L)$.

Let l_x be defined as above. The following holds: $\sum_{v \in V(S)} l(T_i, x, L_S(v), L_S(l(v)), L_S(r(v))) = l_S(x)$. From the above equality, it follows that $\sum_{v \in V(S) x \in V(T_i)} l(T_i, x, L_S(v), L_S(l(v)), L_S(r(v))) = \sum_{x \in V(T_i)} l_S(x)$. That is, the above algorithm counts the number of losses correctly.

Theorem 2. *The running time for this algorithm is $O(p^2 \cdot r \cdot a \cdot l^2)$, where $p = |P|$, $l = |\cup L_{T_i}|$, and a is the time needed to access a set $A \in \mathcal{P}$.*

6 Experimental Results

We focused our attention to the dataset provided in [GMS96]; this consists of 53 gene trees formed over 16 taxa. The gene trees given in the paper have width 3 w.r.t. the species tree. In total, the best tree they found has 17 of the 53 gene trees agree with this species tree (in other words, they have width 1) and the total score for this species tree is 190 (47 duplications, 143 losses).

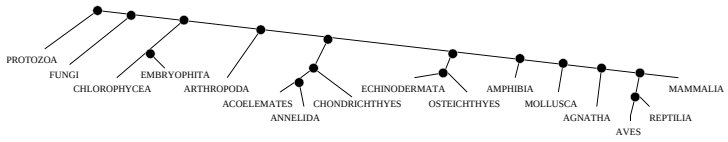

Fig. 2. The best tree for the dataset from [GMS96] found w.r.t. the DUPLICATION-LOSS problem.

Figure 2 gives a tree we found using our algorithm from Section 5 that scores better under the DUPLICATION-LOSS model. In total, the tree scores 159 (36 duplications and 123 losses). The overall width is 4. (This tree was later verified to be optimal via an exhaustive dynamic programming algorithm using all possible partitions.)

We coded our algorithms in the bioinformatics programming language DARWIN [GH99] and, although much slower than native C code, the implementations were fast enough to run our

algorithms for $\mathcal{P}$ sets created with respect to width 4 (at this width, the P set has size 58757 out of a possible 65536). For larger datasets, it may be infeasible to compute such large sets exactly. We offer the following heuristic: *Choose an initial species tree S. Repeat the following until there is no improvement: Generate $\mathcal{P} = \{\cup_{i=1}^{2} A_i : A_i \in \mathcal{P}_T\}$. Let S be equal to the results of applying the* DUPLICATION-LOSS *algorithm of Section 5 using all of the gene trees $T_1, \ldots, T_r$.*

This heuristic, using the [GMS96] tree as the initial species tree, managed to find the optimal species tree for the DUPLICATION problem after only 2 iterations. It was also extremely fast, since the size of the $\mathcal{P}$ set was very small (below $\binom{|A(S)|}{2} = 435$ out of a possible size of $2^{L_S} = 65536$). In general, a good candidate for choosing the initial species tree might be a tree which agrees with many of the gene trees. Or, one might use the algorithm in Section 5 with a $\mathcal{P}$ set created from all the gene trees and $k = 1$ to form the initial topology. Initial experiments on random data have given good results with the number of iterations, before finding the optimal species tree, always below 4.

7 Open Problems

One current avenue for future research would be to determine if these problems are hard for any level of the W-hierarchy - the appropriate framework for these questions - when parameterized by the number of gene trees and/or the width. If this is the case, an approximation algorithm or solid heuristic for choosing good $\mathcal{P}$ sets will be needed for larger widths.

Also, our algorithms should be compared against the [P94] and [MLZ98] implementations on gene trees having large width w.r.t. their species tree. The heuristic presented in Section 6 might complement the work done in these other systems.

References

[BDDEHY98] Bork, P. et. al. (1998) Predicting function: from genes to genomes and back. *J. Mol. Biol.* 283, 707–725.

[FHKS98] Fellows, M. R. et. al. (1998) Analogs & duals of the MAST problem for sequences & trees. *European Symposium on Algorithms (ESA).*

[GCMRM79] Goodman, M. et. al. (1979) Fitting the Gene Lineage into its Species Lineage: A parsimony strategy illustrated by cladograms constructed from globin sequences, *Syst.Zool.*, 28.

[GH99] Gonnet, G. and Hallett, M. T. (1999). Darwin: A User Manual. *To be published.* book, Available at http://cbrg.inf.ethz.ch.

[GMS96] Guigó, R. et al. (1996) Reconstruction of Ancient Molecular Phylogeny. *Molec. Phylogenet. and Evol.*, 6(2), pp. 189–213, 1996.

[HMM96] *Molecular Systematics.* (1996) Sinauer Assoc. Inc, USA. ed. Hillis, D. M., Moritz, C. and Mable, B.

[KTG98] Koonin, E. V. et. al. (1998) Beyond complete genomes: from sequence to structure and function. *Curr Opin Struct Biol.* 8(3), 355-63.

[MLZ98] Ma, B. et. al. (1998) On Reconstructing Species Trees from Gene Trees in Term of Duplications and Losses. *Recomb 98.*

[MMS95] Mirkin, B. et. al. (1995) A biologically consistent model for comparing molecular phylogenies. *Journal of computational biology*, 2(4), 493–507.

[P94] Page, R. (1994) Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas. *Syst. Biol.*, 43, p. 58–77.

[P98] Page, R. (1998) GeneTree: comparing gene and species phylogenies using reconciled trees. *Bioinformatics*, 14(9), 819–820.

[PC97] Page, R. and M. Charleston, M. (1997) ¿From Gene to organismal phylogeny: reconciled trees and the gene tree/species tree problem. *Molec. Phyl. and Evol.* 7, 231–240.

[S99] Stege, U. (1999) Gene trees and species trees: The gene-duplication problem is fixed-parameter tractable. *Proceedings of the 6th International Workshop on Algorithms and Data Structures (WADS'99)*, Vancouver, Canada, LNCS 1663.

[TKL97] Tatusov, R. L. et. al. (1997) A genomic perspective on protein families. Science, 278(5338), 631-7.

[YEVB98] Yuan, Y. P. et. al. (1998) Towards detection of orthologues in sequence databases. *Bioinformatics*, 14(3), 285–289.