An optimal reconciliation algorithm for gene trees with polytomies

Manuel Lafond[1], Krister M. Swenson[2], and Nadia El-Mabrouk[3]

[1] DIRO, Université de Montréal, H3C 3J7, Canada, lafonman@iro.umontreal.ca
[2] DIRO, swensonk@iro.umontreal.ca
[3] DIRO, mabrouk@iro.umontreal.ca

Abstract. Reconciliation is a method widely used to infer the evolutionary relationship between the members of a gene family. It consists of comparing a gene tree with a species tree, and interpreting the incongruence between the two trees as evidence of duplication and loss. In the case of binary rooted trees, linear-time algorithms have been developed for the duplication, loss, and mutation (duplication + loss) costs. However, a strict prerequisite to reconciliation is to have a gene tree free from error, as few misplaced edges may lead to a completely different result in terms of the number and position of inferred duplications and losses. How should the weak edges be handled? One reasonable answer is to transform the binary gene tree into a non-binary tree by removing each weak edge and collapsing its two incident vertices into one. The created polytomies are "apparent" as they do not reflect a true simultaneous divergence of many copies from a common ancestor, but rather a lack of resolution. In this paper, we consider the problem of reconciling a non-binary rooted gene tree G with a binary rooted species tree S, were polytomies of G are assumed to be apparent. We give a linear-time algorithm that infers a reconciliation of minimum mutation cost between a binary refinement of a polytomy and S, improving on the best known result, which is cubic. This implies a straightforward generalization to a gene tree G with nodes of arbitrary degree, that runs in time $O(|S||G|)$, which is shown to be an optimal algorithm.

1 Introduction

The evolutionary history of a gene family is determined by a combination of microevolutionary events at the sequence level, and macroevolutionary events (duplications, losses, horizontal gene transfer) affecting the number and distribution of genes among genomes [11]. While sequence similarity can be considered as a footprint of microevolution and used to construct a gene tree G for the gene family, macroevolution is harder to predict as it is not explicitely reflected by the gene tree. Having a clear picture of the speciation, duplication and loss mechanisms that have shaped a gene family is however crucial to the study of gene function. Indeed, following a duplication, the most common occurrence is for only one of the two gene copies to maintain the parental function, while the other becomes non-functional (pseudogenization) or acquires a new function (neofunctionalization) [19].

Reconciliation, first introduced by Goodman in 1979 [13] — and since widely studied and implemented in comparative genomics software [12] — is a method that compares the gene tree G with a phylogeny S of the considered species (species tree), and interprets the incongruence between the two trees as evidence describing evolution of the gene family through duplication and loss. A reconciliation $R(G, S)$ is a tree obtained from G by inserting "lost" branches so that the obtained tree is in agreement with the phylogeny S. As there can be several reconciliations for a given tree pair, a natural approach is then to select one, or a subset, that optimize some probabilistic [1, 2] or combinatorial [16] criterion such as the number of duplications (duplication cost), losses (loss cost) or both combined (mutation cost). Reconciliation of binary rooted trees is a well-studied problem, and linear-time algorithms based on the so called lowest common ancestor (LCA) mapping have been developed for the duplication, loss and mutation costs [6, 20, 22]. Generalizations of reconciliation accounting for horizontal gene tranfers have also been considered [9]. In particular, minimizing the number of duplications, losses and transfers has been shown to be computationally hard [18], but feasible in polynomial time if the input species tree is dated [10].

The fundamental hypothesis behind reconciliation is that the gene tree reflects the true phylogeny of the gene family. Therefore, a strict prerequisite is to have both gene tree and species tree free from error [8, 15]. Unfortunately gene trees are not always well-supported, and frequently many equally-supported trees are obtained as the output of a phylogenetic method. Typically bootstrap values are used as a measure of confidence in each edge of a phylogeny. How should the weak edges of a gene tree be handled? One strategy adopted in [6] is to explore the space of gene trees obtained from the original tree G by performing Nearest Neighbor Interchanges around weakly-supported edges. Another reasonable answer is to transform the binary gene tree into a non-binary tree by removing each weak edge and collapsing its two incident vertices into one. A polytomy (node with more than two children) in a gene tree is called *true* (or *hard*) if it reflects a true simultaneous divergence of its children from a common ancestor, and it is called *apparent* (or *soft*) otherwise [17]. Implicitly, polytomies of a gene tree obtained by the method of collapsing short or poorly supported internal branches are apparent polytomies, reflecting a lack of resolution.

In this paper, we consider the problem of reconciling a non-binary rooted gene tree G with a binary rooted species tree S, where polytomies of G are assumed to be apparent. More precisely, we seek out a reconciliation of minimum mutation cost between a binary refinement of G and S. Chang and Eulenstein were the first to consider this problem [3]. They showed that each polytomy P can be treated independently in $O(|S| \times |P|^2)$ time, implying an $O(|S| \times |G|^2)$ algorithm for the entire tree. In a recent paper [21], a linear-time algorithm is developed for reconciling a non-binary gene tree G with a binary species tree S, but for the duplication cost. The output is a reconciliation with optimal loss cost over all the reconciliations with the optimal duplication cost, which does not necessarily minimize the mutation cost. Here, we describe an algorithm that infers the minimum mutation cost reconciliation between P and S in $O(max(|P|, |S|))$ time, implying an $O(|G| \times |S|)$ algorithm over the entire gene tree. This algorithm is optimal, since there exists a family of instances leading to a most parsimonious reconciliation of size $\Omega(|G| \times |S|)$.

2 Preliminary notation

In this paper, all the trees are considered rooted (we ommit to mention it each time). Given a tree T, we denote by T_x the subtree of T rooted at x, and by $L(T_x)$ (or simply $L(x)$ if unambiguous) the set of leaves of T_x. We also denote by $root(T)$ the root of T, by $V(T)$ the set of nodes of T and by $|T|$ the number of nodes $|V(T)|$ of T. The *degree* of an internal node x in a tree T is the number of children of x. If T is binary, we denote by x_l and x_r the two children of x in T.

A *phylogeny* over a set L is a tree with internal nodes of degree 2 or more, uniquely leaf-labeled by L. A *polytomy* (or star tree) over a set of L is a phylogeny with a single internal node, which is of degree $|L|$, adjacent to each leaf of L. For example, the tree G in Figure 2 is a polytomy.

A *species tree* S is a phylogeny over a set of species Σ, which represents the evolutionary relationship between these species. Similarly, we can consider the evolutionary relationship between a family of genes Γ, that appear in the genomes of Σ: a *gene tree* G for Γ is a phylogeny accompanied by a function $g : \Gamma \rightarrow \Sigma$ indicating the species where each gene is found. See Figure 1 for an example. Given a gene tree G, we denote by $\mathcal{S}(G_x)$ the subset of Σ corresponding to $L(G_x)$ (i.e. $\mathcal{S}(G_x) = \{g(l) \mid l \in L(G_x)\}$).

In this paper, we assume a binary species tree S and a non-binary gene tree G. As stated in the introduction, the polytomies of G are considered apparent (i.e. reflecting non resolved parts of the tree). The goal is then to find a "binary refinement" of G. For any internal node x of G with children $\{x_1, x_2, \ldots, x_n\}$, any rooted binary tree on the set of leaves $\{G_{x_1}, G_{x_2}, \ldots, G_{x_n}\}$ is a *refinement* of the polytomy G_x. The following definition generalizes this fact.

Definition 1 (binary refinement). *A* binary refinement *$B(G)$ of a gene tree G is defined as follows.*

- *If r is a leaf then $B(G) = G$;*
- *Otherwise, $B(G)$ is a rooted binary tree on the set $\{B(G_1), B(G_2) \ldots, B(G_n)\}$, where G_i is the tree rooted at the ith child of root(G) (for some ordering of the children), and $B(G_i)$ is a binary refinement of G_i.*

2.1 Histories and reconciliation

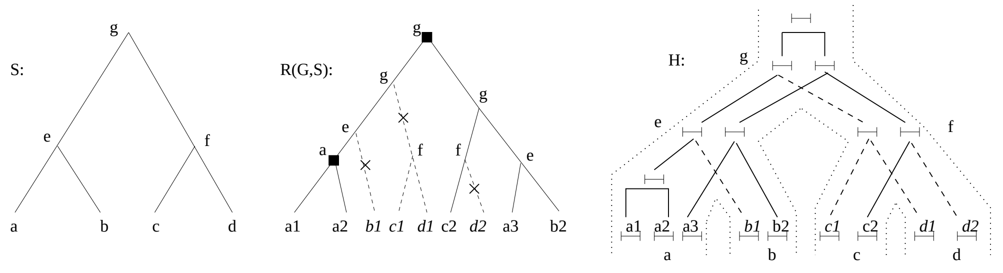

Fig. 1: S is a species tree over $\Sigma = \{a, b, c, d\}$; $R(G, S)$ is a reconciliation between S and the gene tree G represented by plain lines. Here $\Gamma = \{a_1, a_2, a_3, b_2, c_2\}$, and for each $x_i \in \Gamma$, $g(x_i) = x$. Internal nodes of $R(G, S)$ are labeled according to the LCA mapping. Artificial genes $\{b_1, c_1, d_1, d_2\}$ are added to illustrate lost branches. Duplication nodes are indicated by bold squares, and loss leaves are represented by crosses. This reconciliation has cost 5: 2 duplications and 3 losses; H illustrates the history that has led to the gene family Γ. H is the same tree as $R(G, H)$, but represented differently (embedded in the species tree).

We study the evolution of a family of genes Γ taken from genomes Σ through duplication and loss. Conceptually, a *duplication/loss/speciation history* (or simply *history*) is a tree H reflecting the evolution from a single ancestral gene to a set of genes through duplication, loss, and speciation events. Given a binary gene tree G for the gene family and a species tree S for Σ, a reconciliation is a history obtained from G, in "agreement" with the phylogeny S. In this section we formally define history and reconciliation, as well as presenting tools for working with them. All these concepts are illustrated in Figure 1.

The most popular method for finding a parsimonious reconciliation is based on the "LCA mapping". The *LCA mapping* between G and S, denoted by $\mu()$, maps every node x of G to the *lowest common ancestor* of $\mathcal{S}(G_x)$ in S, which is the common ancestor of $\mathcal{S}(G_x)$ in S that is farthest from the root. We call $\mu(x)$ the *label* of x. A node x of G is considered a *duplication* with respect to S if and only if $\mu(x_\ell) = \mu(x)$ and/or $\mu(x_r) = \mu(x)$. Any node of G that is not a duplication node, is a *speciation* node.

Take a binary tree T, labeled by the LCA mapping, where there exists exactly one leaf labeled by each gene in Γ, and a function $g : \Gamma \mapsto \Sigma$ indicating the species where each gene is found. A *duplication-free restriction* $D(T)$ of a tree T is obtained by removing either T_{x_ℓ} or T_{x_r} for each duplication node x, along with x, and if x is not the root, joining the parent of x and the remaining child by a new edge. Each duplication-free restriction $D(T)$ can be considered to be a copy of a species tree S, in which case each loss leaf u corresponds subtree S_u of the species tree that is missing in $D(T)$.

A duplication-free restriction $D(T)$ *agrees* with a species tree S iff relabeling each leaf l of $D(T)$ by $g(l)$, and replacing each loss leaf u in $D(T)$ with the subtree S_u, results in a tree isomorphic to S.

Definition 2 (consistent). *Take a species tree S and a rooted binary tree T where there exists exactly one leaf labeled by each gene in Γ, and all other leaves are labeled as losses. T is said to be* consistent *with S iff every duplication-free restriction of T agrees with S.*

Definition 3 (history). *A* history *H is a rooted binary tree uniquely leaf-labeled by a gene set Γ, and function $g : \Gamma \mapsto \Sigma$ (indicating the species where each gene is found) with the following properties:*

1. *Any leaf not labeled by a member of Γ is a loss.*
2. *Each internal node is a duplication or speciation node.*
3. *There exists a species tree S consistent with H.*

Definition 4 (reconciliation). *A* reconciliation *$R(G, S)$ between a binary gene tree G and a species tree S is a history that can be obtained from G by inserting loss leaves and labeling internal nodes as speciations or duplications so that it is consistent with S.*

The parsimony criteria used to choose among the large set of possible reconciliations are the number of duplications (*duplication cost*), the number of losses (*loss cost*) or both combined (*mutation cost*). The LCA mapping induces a reconciliation $R(G, S)$ between G and S, where an internal node x of G leads to a duplication node in R if and only if x is a duplication node of G with respect to S. Moreover, $R(G, S)$ is a reconciliation that minimizes the duplication, loss, and mutation costs [5, 14].

In the rest of this paper, the *cost* of a reconciliation refers to its mutation cost.

2.2 Problem statement

Given a binary species tree S and a non-binary gene tree G, we seek out a full resolution of G leading to a reconciliation of minimum mutation cost. We formally define the notion of a resolution of G as being a reconciled refinement of G.

Definition 5 (Resolution). *A tree $R(G, S)$ is a* resolution *of G with respect to S if and only if $R(G, S)$ is a reconciliation between a binary refinement $B(G)$ of G, and S.*

We are now ready to state our optimization problem.

MINIMUM RESOLUTION:
Input: A binary species tree S and a non-binary gene tree G.
Output: A *Minimum Resolution* of G with respect to S (or simply a *Minimum Resolution of* G if there is no ambiguity on S), e.g. a resolution of G with respect to S of minimum mutation cost.

We first show that each polytomy of G can be resolved independently.

Theorem 1. *Let $\{G_{x_i},$ for $1 \leq i \leq p\}$ be the set of subtrees of G rooted at the p children $\{x_i,$ for $1 \leq i \leq p\}$ of the root of G. Let $R_{min}(G_{x_i}, S)$ be a minimum resolution of G_{x_i} w.r.t. S. Let G' be the tree obtained from G by replacing each G_{x_i} by $R_{min}(G_{x_i}, S)$. Then a minimum resolution of G' is a minimum resolution of G.*

Proof. This statement was proved by Chang and Eulenstein in [4], which led them to a dynamic programming algorithm with running-time complexity $O(|S| \times |G|^2)$, for the MINIMUM RESOLUTION problem. The reader interested in this proof might refer to Chang's MSc. Thesis written in 2006.

It follows from Theorem 1 that a minimum resolution of G can be obtained by a depth-first procedure that solves each polytomy G_x iteratively, for each internal node x of G. At each step, whether the children of the polytomy G_x are internal nodes or leaves of G, they are treated as leaves of the polytomy and we refer to each leaf l by its label $\mu(l)$.

In the next section, we consider G as a polytomy whose leaves are labeled (not uniquely) by nodes of S. Furthermore, as the subtrees S_x of S such that $V(S_x) \setminus \{x\}$ has an empty intersection with $\mathcal{S}(G)$, will never be considered in the resolution of G, we can ignore them. We say that S is a *species tree linked to the polytomy* G if and only if any internal node of S has a descendant included in $\mathcal{S}(G)$ and the root of S is the lowest common ancestor of $\mathcal{S}(G)$. For example, in Figure 2, S is a species tree linked to the polytomy G.

3 Method

In this section, we consider G to be a polytomy whose leaves are labeled (not uniquely) by nodes of a species tree S. Notice that a leaf labeled x actually represents a whole subtree of the considered gene tree, which has already been resolved, and thus is consistent with $S_{\mu(x)}$. We assume that S is a species tree linked to G. We describe an approach for computing a minimum resolution $R(G, S)$ of G based on the observation that for any node x in $R(G, S)$, all nodes on a path from x to a leaf in $R(G, S)$ will map to a node that is on a path from $\mu(x)$ to a leaf of S. Thus, we decompose the computation of a minimum resolution of G according to a depth-first traversal of the nodes of S; for each node s of S we consider the cost of having k maximal subtrees of $R(G, S)$ whose roots map to s. For example, Figures 2c and 2d represent two such partial resolutions where there are three maximal subtrees whose roots map to e. Given, for all k, the minimum cost of a so-called "k-partial resolution" corresponding to a node s, we show how to compute the cost of a partial resolution corresponding to the parent of s. Clearly a solution of the Minimum Resolution problem is a minimum 1-partial resolution of G at the root of S.

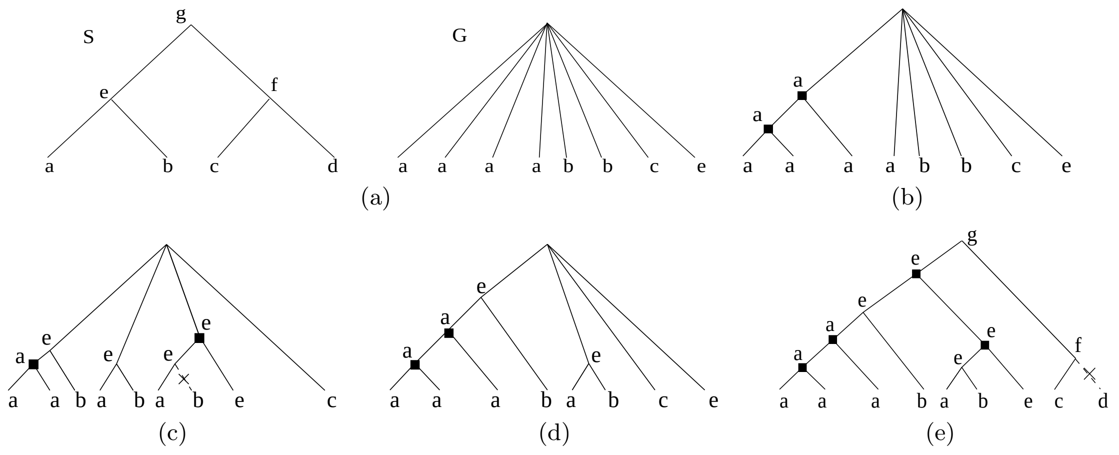

Fig. 2: (a) A species tree S and a polytomy G; (b) A 2-partial resolution of G at a of cost 2 (2 duplications); (c) A 3-partial resolution of G at e of cost 3 (2 duplications and one loss); (d) A 3-partial resolution of G at e of cost 2 (2 duplications); (e) A full resolution of G with minimum cost (4 duplications, 1 loss).

3.1 Partial resolutions

Let s be a node of S. The restriction of G by node s, denoted $G_{/s}$, is the tree obtained from G by removing the set of leaves $\mathcal{L}_s$ whose labels are not in S_s.

Definition 6 (partial resolution). *Let s be a node of S. A partial resolution $P(G, S, s)$ of G at s is a polytomy on a set $\mathcal{F}_s \cup \mathcal{L}_s$, where $\mathcal{F}_s$ is obtained from a resolution $R(G_{/s}, S)$ as follows: $\mathcal{F}_s$ is* *a forest of subtrees of $G_{/s}$, rooted at nodes labeled s, partitioning the set of leaves of $G_{/s}$ (i.e. each leaf of $G_{/s}$ is in a unique tree of the forest).*

The cost of a partial resolution $P(G, S, s)$ on a set $\mathcal{F}_s \cup \mathcal{L}_s$ is the sum of the cost of all the trees (reconciliations) of $\mathcal{F}_s$. See Figure 2 for an example.

Definition 7 (k-partial resolution). *Let s be a node of S. A k-*partial resolution *$P^k(G, S, s)$ of G at s is a partial resolution of G at s on a set $\mathcal{F}_s \cup \mathcal{L}_s$ with exactly k trees in $\mathcal{F}_s$.*

For example, the tree G in Figure 2 is itself a 4-partial resolution of G at a, whereas the tree (b) is a 2-partial resolution of G at a, and (c) and (d) are two different 3-partial resolutions of G at e.

Notation 1 *For any integer $k \geq 1$, we denote by $M_{s,k}$ the minimum cost of a k-partial resolution of G at s. We also denote by M_s the vector $(M_{s,k})_{k\geq 1}$.*

A solution for the Minimum Resolution problem is a resolution of G with cost $M_{root(S),1}$. In this section, we describe an algorithm that computes $M_{root(S),1}$ based on the costs $M_{s,k}$ of all partial resolutions of G over all k and s. Before giving a recursive formulation of $M_{s,k}$, we need to introduce a subset of k-partial resolutions, leading to an intermediate cost $C_{s,k}$ for internal nodes s, which can be computed directly from k-partial resolutions corresponding to the children of s.

Definition 8 (k-speciation resolution). *Let s be an internal node of S. A k-partial resolution $P^k(G, S, s)$ of G at node s is a k-*speciation resolution *of G at s if and only if each node of $P^k(G, S, s)$ labeled s is a speciation or a leaf.*

A k-speciation resolution at s contains no duplication node nor loss leaf labeled s. In Figure 2, the tree G is a 4-speciation resolution of G at node a, while the tree (d) is a 3-speciation resolution at e. Neither the 2-partial resolution of G at a (b) nor the 3-partial resolutions of G at e (c) is a speciation resolution, as in the first case the left-most child of the root labeled a is a duplication node, while in the second case the right-most child of the root labeled e is a duplication node.

Notation 2 *For any node s of S, we denote by $nb(s)$ the number of leaves of G labeled s.*

Note that there is no k-speciation resolution for $k \leq nb(s)$. Indeed, as G has $nb(s)$ leaves labeled s, any speciation resolution of G has at least $nb(s)$ speciation nodes labeles s, and thus $k \geq nb(s)$. Moreover, as S is a species tree linked to G, at least one descendant of the internal node s of S should be a leaf of G, and thus any partial resolution at s needs to have at least one additional speciation node labeled s.

Notation 3 *For any internal node s of S and any integer $k > nb(s)$, we denote by $C_{s,k}$ the cost of a minimum k-speciation resolution of G at s. For technical reasons, we set $C_{s,k} = \infty$ for $1 \leq k \leq nb(s)$. We denote by C_s the vector $(C_{s,k})_{k\geq 1}$.*

3.2 A recursive formulation

There is an infinite range of values of k for which $M_{s,k}$ and $C_{s,k}$ correspond to valid resolutions. However, the following remark is easy to validate and implies that, for some input, we need only consider a fixed-size table of values.

Remark 1. There exists a value $n \in \mathbb{N}$ such that $M_{s,k} < M_{s,n}$, for any node s of S and any integer $0 < k < n$.

The intuition behind this remark is that when k is large enough a k-partial resolution would contain too many losses, so could never be part of an optimal solution.

The following lemma exhibits a relationship between two entries of vector M_s.

Lemma 1. *For any node s of S and any integers $k, i \geq 1$, we have $M_{s,k} \leq M_{s,i} + |k - i|$.*

Proof. Let $P^i(G, S, s)$ be an i-partial resolution at s of cost $M_{s,i}$. If $i < k$, inserting $k - i$ losses of s at the root of $P^i(G, S, s)$ gives us a k-partial resolution of cost $M_{s,i} + k - i$. If $i > k$, joining $i - k + 1$ subtrees rooted at s in $P^i(G, S, s)$ (by duplication nodes) gives us a k-partial resolution of cost $M_{s,i} + i - k$. Both cases imply that $M_{s,k} \leq M_{s,i} + |k - i|$. $\square$

We use Lemma 1 to prove the main recurrence defining $C_{s,k}$ and $M_{s,k}$.

Theorem 2. *Let s be a node of S and $1 \leq k \leq n$.*

1. *If s is a leaf of S, then $M_{s,k} = |k - nb(s)|$;*
2. *Otherwise, let s_l and s_r be the two children of s in S. Then,*
   (a) $C_{s,k} = M_{s_\ell,k-nb(s)} + M_{s_r,k-nb(s)}$ if $k > nb(s)$ (∞ otherwise);
   (b) $M_{s,k} = \min\left(C_{s,k}, \min_{1 \leq i \leq n} (M_{s,i} + |k - i|)\right)$

Proof. Let $P^k(G, S, s)$ be a minimum k-partial resolution of G at node s of cost $M_{s,k}$. Suppose $P^k(G, S, s)$ is defined on the set of nodes $\mathcal{F}_s \cup \mathcal{L}_s$.

1. Suppose s is a leaf of S. If $k = nb(s)$, then each tree in $\mathcal{F}_s$ is a single node labeled s, with reconciliation cost 0, and thus $M_{s,k} = 0 = |k - nb(s)|$. If $k < nb(s)$ (respectively $k > nb(s)$), then at least $nb(s) - k$ duplication nodes (respec. $k - nb(s)$ losses) should be present in the trees of $\mathcal{F}_s$. As the trees of $\mathcal{F}_s$ are part of an optimal k-partial resolution, the number of duplications (losses) should be exactly $nb(s) - k$ ($k - nb(s)$), and thus $M_{s,k} = |k - nb(s)|$.

2. Otherwise, s is an internal node of S.
2(a) Let $P^k(G, S, s)$ be a k-speciation resolution of G at node s of minimum cost $C_{s,k}$. Since none of the trees in $\mathcal{F}_g$ are rooted at a duplication node nor labeled as a loss, there must be exactly $k - nb(s)$ trees in $\mathcal{F}_s$ that are rooted at speciation nodes labeled by s. Any such node must have one child labeled s_ℓ and one child labeled s_r. Since we are going through each node of S in a depth-first manner, we assume that the values of M_{s_ℓ} and M_{s_r} have been computed. The result follows from the fact that $M_{s_\ell,k-nb(s)}$ (resp. $M_{s_r,k-nb(s)}$) gives the optimal configuration yielding $k - nb(s)$ trees rooted at nodes labeled s_ℓ (resp. s_r).
2(b) If $P^k(G, S, s)$ is a k-speciation resolution of G at s, then clearly $M_{s,k} = C_{s,k}$. Otherwise, let k' be the number of trees rooted at duplication nodes of $\mathcal{F}_s$. For positive k', if each of the two children of those duplication nodes were taken as the roots of two new trees, then we would have a forest of $i' = k + k'$ trees. This gives us $M_{s,k} \geq M_{s,i'} + k' = M_{s,i'} + |k - i'|$. If $k' = 0$ (and $P^k(G, S, s)$ is not a k-speciation resolution), then $\mathcal{F}_s$ must have, say k'', trees corresponding to losses. Consider the $i'' = k - k''$ trees of $\mathcal{F}_s$ that are not losses. This gives us $M_{s,k} \geq M_{s,i''} + k'' = M_{s,i''} + |k - i''|$ as well. Therefore $M_{s,k} \geq \min_i(M_{s,i} + |k - i|)$. On the other hand, Lemma 1 gives us $M_{s,k} \leq \min_i(M_{s,i} + |k - i|)$. $\square$

3.3 A dynamic programming approach

The recurrence 2.b in Theorem 2 induces a circular argument for computing the entries of M_s, as for two different constants k and i, $M_{s,k}$ may be computed from $M_{s,i}$, which in turn may be computed from $M_{s,k}$. In other words, the recurrences of Theorem 2 cannot be used directly in a dynamic programming algorithm for the computation of $C_{s,k}$ and $M_{s,k}$. The rest of this section focuses on reformulating recurrence 2.b. We start by giving two important properties relating M_s to C_s.

Lemma 2. *For an internal node s of S, there exists at least one k such that $M_{s,k} = C_{s,k}$.*

Proof. Let $P^k(G, S, s)$ be a k-partial resolution of G at node s of cost $M_{s,k}$, defined on the set $\mathcal{F}_s \cup \mathcal{L}_g$. Assume that $M_{s,k} \neq C_{s,k}$ for all k. This implies $M_{s,k} < C_{s,k}$ for all k. Consider the subforest $\mathcal{F}'_s$ consisting of the j maximal subtrees of $\mathcal{F}_s$ that are rooted at speciation nodes; $\mathcal{F}'_s$ defines a j-speciation resolution with cost $C' < M_{s,k}$. Since $C_{s,j} \leq C'$, we have $C_{s,j} < M_{s,k}$. But $M_{s,j} < C_{s,j}$, so in general, for any k there exists a j such that $M_{s,j} < M_{s,k}$, a contradiction since the minimum value in M_s must occur in within a finite range (by Remark 1). □

Theorem 2 shows that for an internal node s of S, $M_{s,k}$ can be computed from $C_{s,k}$, or from some other value in M_s. However, we have not characterized how to easily discern which case will be used, and we have no information about which i gives $M_{s,i} = C_{s,i}$. The following lemma addresses this matter by narrowing the possibilities.

Lemma 3. *For some internal node s of S and integer $k \geq 1$, if $M_{s,k} \neq C_{s,k}$ then there exists an i such that $M_{s,i} = C_{s,i}$ and $M_{s,k} = M_{s,i} + |k - i|$.*

Proof. By the recurrence 2(b) of Theorem 2, if $M_{s,k} \neq C_{s,k}$, then we should have $M_{s,k} = M_{s,i}+|k-i|$ for some i. If $M_{s,i} = C_{s,i}$, then the lemma is verified. Otherwise, $M_{s,i} = M_{s,h} + |i - h|$ for some h. This gives $M_{s,k} = M_{s,h} + |k - i| + |i - h| \geq M_{s,h} + |k - h|$. By lemma 2 we know that there must be some value α for which $M_{s,k_\alpha} = C_{s,k_\alpha}$, so in general, for some integers k_0 and α we have

$$\begin{aligned} M_{s,k_0} &= M_{s,k_\alpha} + |k_\alpha - k_{\alpha-1}| + |k_{\alpha-1} - k_{\alpha-2}| + \cdots + |k_2 - k_1| + |k_1 - k_0| \\ &= M_{s,k_\alpha} + \sum_{i=1}^{\alpha} |k_i - k_{i-1}| \\ &\geq M_{s,k_\alpha} + |k_\alpha - k_0| = C_{g,k_\alpha} + |k_\alpha - k_0|. \end{aligned}$$

Lemma 1 gives the complementary bound $M_{s,k_0} \leq C_{s,k_\alpha} + |k_\alpha - k_0|$, so equality holds. □

Lemma 3 allows us to rewrite the recurrence 2(b) of Theorem 2 as follows:

$$M_{s,k} = \min_{nb(s)<i\leq n} C_{s,i} + |k - i| \qquad (Eq.1)$$

With this new formulation of recurrence 2(b), Theorem 2 leads to a cubic-time dynamic programming algorithm for the computation of the cost of a solution of the Minimum Resolution problem. Indeed, let the height of a node s of S be the maximum number of nodes in a path from s to a leaf of S. Consider an ordering $s_1, s_2 \cdots s_p$ of the nodes of S by increasing height, where $p = |S|$. In other words, leaves are listed before nodes of height 1, etc. In particular $s_p = root(S)$. Consider the tables M and C of $|S|$ lines, where each line i of M and C corresponds respectively to the vectors M_{s_i} and C_{s_i}. The table C is defined only for lines $i > L(S)$. We first compute the $L(S)$ first lines of M in $O(n)$ steps using recurrence (1) of Theorem 2. Then, for each line i, we successively compute C_{s_i} and M_{s_i} for increasing values of i, by using the recurrences (2) a. and b. of Theorem 2. Each line representing C_{s_i} is computed in time $O(n)$, while each line representing M_{s_i} is computed in $O(n^2)$ steps, leading to an $O(n^2|S|)$ algorithm for filling the two tables. The final result (cost of a solution of the Minimum Resolution problem) is just $M_{s_p,1}$. An example is given in Figure 3.

3.4 A linear-time approach

We show in this section that the recurrence (2)b of Theorem 2 can further be simplified in a way leading to a constant time update for each M_s vector according to the M_{s_l} and M_{s_r} vectors. This implies a linear time algorithm for the computation of $M_{root(S),k}$.

We first show that $M_{s,k} = C_{s,k}$ when $C_{s,k}$ is the minimum value among the entries of C_s.

Lemma 4. *For k such that $C_{s,k} = \min_{nb(s)<i\leq n} C_{s,i}$, we have $M_{s,k} = C_{s,k}$.*

Proof. If $M_{s,k} \neq C_{s,k}$, then by Lemma 3, there must exist an i such that $M_{s,k} = M_{s,i} + |k - i|$ and $M_{s,i} = C_{s,i}$. This implies that $C_{s,i} < C_{s,k}$, contradicting the minimality of $C_{s,k}$. □

The key observation allowing a constant-time computation of any entry in C_s and M_s is that these vectors can be seen as two functions with a cup shape. The following definition formally introduces the notion of a "cup function".

Definition 9 (cup function). *A* cup function *is a convex piecewise linear function $m()$ which, for a minimum value $\gamma_m \in \mathbb{Z}$ and two* breakpoints *$m_1, m_2 \in \mathbb{N}$, is strictly decreasing linearly for $x < m_1$, equal to γ_m when $m_1 \leq x \leq m_2$, and strictly increasing linearly when $x > m_2$. It can be written as*

$$m(x) = \begin{cases} \gamma_m + m_1 - x + \mathcal{P}(x) & \text{if } x < m_1 \\ \gamma_m & \text{if } m_1 \leq x \leq m_2 \\ \gamma_m + x - m_2 + \mathcal{Q}(x) & \text{if } x > m_2 \end{cases}$$

where $\mathcal{P} : \mathbb{N} \to \mathbb{Z}$ is non-increasing and $\mathcal{Q} : \mathbb{N} \to \mathbb{Z}$ is non-decreasing.

We say the function $m(x)$ is a simple cup function *iff $\mathcal{P}(x) = \mathcal{Q}(x) = 0$ for all x. Roughly speaking, a simple cup function viewed from left to right has a slope of -1, a plateau of minimum values, and a slope of* 1.

Assume for now that M_s can be associated with a simple cup function $m()$ such that $M_{s,k} = m(k)$ for $1 \leq k \leq n$. Recall that the values of C_s are obtained by adding the values of M_{s_ℓ} and M_{s_r} (recurrence 2(a) of Theorem 2). We show that C_s can be associated with a cup function by proving that the addition of two simple cup functions yields a cup function.

Lemma 5. *If $\ell()$ and $r()$ are two simple cup functions, then the function $m()$ defined by $m(k) = \ell(k) + r(k)$, is a cup function.*

Furthermore, if ℓ_1, ℓ_2 and r_1, r_2 respectively denote the breakpoints of $\ell()$ and $r()$, and γ_ℓ, γ_r respectively denote the minimum values of $\ell()$ and $r()$, then the breakpoints m_1, m_2 and minimum value γ_m of $m()$ are computed according to the following table:

Condition	γ_m	m_1	m_2
If $\ell_1 < r_1, \ell_2 < r_1$	$\gamma_\ell + \gamma_r + r_1 - \ell_2$	ℓ_2	r_1
If $\ell_1 < r_1, r_1 \leq \ell_2 \leq r_2$	$\gamma_\ell + \gamma_r$	r_1	ℓ_2
If $\ell_1 < r_1, \ell_2 > r_2$	$\gamma_\ell + \gamma_r$	r_1	r_2
If $r_1 \leq \ell_1 \leq r_2, r_1 \leq \ell_1 \leq r_2$	$\gamma_\ell + \gamma_r$	ℓ_1	ℓ_2
If $r_1 \leq \ell_1 \leq r_2, \ell_2 > r_2$	$\gamma_\ell + \gamma_r$	ℓ_2	r_2
If $\ell_1 > r_2, \ell_2 > r_2$	$\gamma_\ell + \gamma_r + \ell_1 - r_2$	r_2	ℓ_1

Proof. The complete proof of this lemma is given in Appendix A. Moreover, a more general version of this lemma has already been proven by Csűrös in [7], where it is shown that the sum of an arbitrary number of cup functions yields a cup function □

The following theorem states that M_s and C_s can be associated with two cup functions with the same breakpoints m_1, m_2 and minimum value γ_m. Moreover, M_s is associated with a simple cup function. For example, each line of the table of Figure 3 can be rewritten as a cup function, with the breakpoint and minimum values indicated in vectors m_1, m_2 and γ_m.

Theorem 3. *For any node s of S, there exists a simple cup function $m()$ with breakpoints $m_1, m_2 \geq 1$ and minimum value γ_m, such that $M_{s,k} = m(k)$ for $1 \leq k \leq n$.*

Furthermore, if s is an internal node of S, there exists a cup function $c()$ with the same breakpoints $m_1, m_2 > nb(s)$ and same minimum value γ_m, such that $C_{s,k} = c(k)$ for $nb(s) < k \leq n$.

Proof. We prove the theorem by induction over the nodes visited in a postorder traversal of S.

Base Case: If s is a leaf and $nb(s) > 0$, let $m(k) = |k - nb(s)|$. It is clear that $m(k)$ is a simple cup function with breakpoints $m_1 = m_2 = nb(s)$ and $\gamma_m = 0$. If $nb(s) = 0$, let $m(k)$ be a simple cup function with breakpoints $m_1 = m_2 = 1$ and minimum value $\gamma_m = 1$. We have $m(k) = |k - nb(s)|$ for $k \geq 1$. Then from Theorem 2.1, both cases give $M_{s,k} = m(k)$ for $1 \leq k \leq n$.

Induction Step: If s is an internal node, then from the inductive hypothesis, there exist two simple cup functions $\ell(), r()$ such that $M_{s_\ell,k} = \ell(k)$ and $M_{s_r,k} = r(k)$ for $1 \leq k \leq n$. Let $f(k) = \ell(k - nb(s)) + r(k - nb(s))$. By Lemma 5, $f()$ is a cup function. Let f_1, f_2 be the breakpoints of $f()$ and γ_f its minimum value. From part *(2a)* of Theorem 2 we have $C_{s,k} = M_{s_\ell,k-nb(s)} + M_{s_r,k-nb(s)}$, implying that $C_{s,k} = f(k)$ for $nb(s) < k \leq n$. However, it is possible that $f_1 \leq nb(s)$ or $f_2 \leq nb(s)$, making the theorem statement invalid.

Let $c()$ be another cup function with breakpoints $m_1 = \max(f_1, nb(s)+1), m_2 = \max(f_2, nb(s)+1)$ and minimum value $\gamma_m = f(m_2) = \min\limits_{nb(s)<k\leq n} C_{s,k}$. It can be verified that $c(k) = f(k)$ when $nb(s) < k \leq n$ and thus, $C_{s,k} = c(k)$ for $nb(s) < k \leq n$. From this, we have

$$C_{g,k} = \begin{cases} \infty & \text{if } k \leq nb(s) \\ \gamma_m + m_1 - k + \mathcal{P}(k) & \text{if } nb(s) < k < m_1 \\ \gamma_m & \text{if } m_1 \leq k \leq m_2 \\ \gamma_m + k - m_2 + \mathcal{Q}(k) & \text{if } k > m_2 \end{cases}$$

for $1 \leq k \leq n$.

By Lemma 4, we know that $M_{s,k} = C_{s,k} = \gamma_m$ for $m_1 \leq k \leq m_2$.

If $k < m_1$, we show that $M_{s,k} = \gamma_m + m_1 - k$. From the equation (Eq.1), we have $M_{s,k} = \min_j(C_{s,j} + |k - j|)$. If $j > m_1$, then by the definition of $C_{s,j}$ given above, we have $C_{s,j} \geq \gamma_m$ and thus $C_{s,j} + j - k \geq \gamma_m + m_1 - k$. If $j < m_1$, then $C_{s,j} + |k - j| \geq \gamma_m + m_1 - j + |k - j| \geq \gamma_m + m_1 - k$ since we can reformulate this inequality as $|k - j| \geq j - k$. Therefore, $C_{s,j} + |k - j|$ has the minimum value when $j = m_1$ and it follows that $M_{s,k} = C_{s,m_1} + m_1 - k = \gamma_m + m_1 - k$ when $k < m_1$.

If $k > m_2$, we show that $M_{s,k} = \gamma_m + k - m_2$. From the (Eq.1), we have $M_{s,k} = \min_j(C_{s,j} + |k - j|)$. If $j < m_2$, we have $C_{s,j} \geq \gamma_m$ and thus $C_{s,j} + k - j \geq \gamma_m + k - m_2$. If $j > m_2$, then $C_{s,j} + |k - j| \geq \gamma_m + j - m_2 + |k - j| \geq \gamma_m + k - m_2$ since we can reformulate this inequality as $|k - j| \geq k - j$. Therefore, $C_{s,j} + |k - j|$ is minimum when $j = m_2$ and it follows that $M_{s,k} = C_{s,m_2} + k - m_2 = \gamma_m + k - m_2$ when $k > m_2$.

The three cases verify that M_s can be associated with a cup function. □

Denote by $m_{1,s}, m_{2,s}$ and γ_s the breakpoints and minimum value of the cup function associated with M_s (and thus from Theorem 3 also with C_s). Theorem 2.(1) allows us to compute $m_{1,s}$, $m_{2,s}$ and γ_s for any leaf s of S. Finally, Theorem 2.(2a) and Lemma 5 allow us to compute, in constant time, the breakpoints $m_{1,s}$, $m_{2,s}$ and minimum value γ_s associated with C_s and M_s, given those associated with M_{s_l} and M_{s_r}.

Stated differently, let $s_1, s_2, \cdots s_p$ be the ordering of the nodes of S defined at the end of Section 3.3, and consider the three vectors $m_1 = (m_{1,s_i})_{1\leq i\leq s}$, $m_2 = (m_{2,s_i})_{1\leq i\leq s}$ and $\gamma = (\gamma_{s_i})_{1\leq i\leq s}$. Then each entry of each of these vectors can be computed in constant-time. Theorem 3 ensures that these vectors allow us to completely define the simple cup function M_s. This leads to an $O(max(|G|, |S|))$ algorithm for computing any value $M_{s,k}$, and in particular the cost $M_{s_p,1}$ of a solution of the minimum Resolution problem.

In Algorithm CupValues(s), we detail the steps required to compute $m_{1,s}, m_{2,s}$ and γ_s for a given node s. Lines 1 to 6 follow from the Theorem 3's base case proof. Line 9 follows from Theorem 2.(2a) and Lemma 5 (discussion above). Line 10 is a correction needed because the table in Lemma 5 gives the result for the addition of two simple cup functions for the same value of k, e.g. $c(k) = \ell(k) + r(k)$.

Since $C_{s,k} = M_{s_\ell,k-nb(s)} + M_{s_r,k-nb(s)}$, we need to shift the obtained breakpoints by adding $nb(s)$ to them. Lines 11 and 12 ensure that $m_{1,s}, m_{2,s} > nb(s)$ and that γ_s is the minimum entry in C_s, as stated in Theorem 3.

Algorithm 1 $CupValues(s)$

```
1: if $s$ is a leaf then
2:     if $nb(s) > 0$ then
3:         $m_{1,s} := nb(s); m_{2,s} := nb(s); \gamma_s := 0;$
4:     else
5:         $m_{1,s} := 1; m_{2,s} := 1;\ \gamma_s := 1;$
6:     end if
7: else
8:     Let $s_\ell, s_r$ be the two children of $s$:
9:     Compute $m_{1,s}, m_{2,s}$ and $\gamma_s$ using the children values $m_{1,s_\ell}, m_{2,s_\ell}, m_{1,s_r}, m_{2,s_r}, \gamma_\ell, \gamma_r$ and the table
       given in Lemma 5;
10:    $m_{1,s} := m_{1,s} + nb(s);\ m_{2,s} := m_{2,s} + nb(s);$
11:    If $m_{1,s} \leq nb(s)$ then $m_{1,s} := nb(s) + 1;$
12:    If $m_{2,s} \leq nb(s)$ then $m_{2,s} := nb(s) + 1$ and $\gamma_s := M_{s_\ell,1} + M_{s_r,1};$
13: end if
```

	1	2	3	4	5	6	m_1	m_2	γ_m
M_a	3	**2**	1	0	1	2	4	4	0
M_b	1	**0**	1	2	3	4	2	2	0
M_c	**0**	1	2	3	4	5	1	1	0
M_d	**1**	2	3	4	5	6	1	1	1
C_e	∞	4	2	2	2	4	3	5	2
M_e	**4**	3	2	2	2	3	3	5	2
C_f	1	3	5	7	9	11	1	1	1
M_f	**1**	2	3	4	5	6	1	1	1
C_g	5	5	5	6	7	9	1	3	5
M_g	**5**	5	5	6	7	8	1	3	5

Fig. 3: An illustration of the algorithms for the gene tree G and species tree S of Figure 2(a). The cost of a most parsimonious resolution of G is $M_{g,1} = 5$. The gray cells are those considered by Algorithm DupLoss for computing the *Dup* and *Loss* vectors, the values in bold being the first ones evaluated by the algorithm on a given row. The obtained values are $Dup(e) = 2, Dup(a) = 2, Loss(d) = 1$ and $Dup(s) = Loss(s) = 0$ for any other node s. The corresponding resolution is given in Figure 2(e).

3.5 Constructing an optimal resolution

Starting with $s = root(S)$ and $k = 1$, we recursively compute the number of losses and duplications required for each node s of S in an optimal reconciliation, based on partial resolutions at s_l and s_r, for s_l and s_r being the children of s. The algorithm presented in this section is based on the following result, which is a corollary of Theorem 3.

Corollary 1 *Let s be a node of S with children s_ℓ and s_r, and $P^k(G, S, s)$ be a minimum k-partial resolution at s defined on the set $\mathcal{F}_g \cup \mathcal{L}_g$ for $1 \leq k \leq n$.*

1. *If $M_{s,k} = M_{s_\ell,k-nb(s)} + M_{s_r,k-nb(s)}$, then the k roots of $\mathcal{F}_s$ are all speciation nodes. Otherwise,*
2. *either $M_{s,k} = \gamma_s + k - m_{2,s}$, in which case $P^k(G,S,s)$ has $k - m_{2,s}$ loss leaves labeled s,*
3. *or $M_{s,k} = \gamma_s + m_{1,s} - k$, in which case $P^k(G,S,s)$ has $m_{1,s} - k$ duplications labeled s.*

Proof. In the first case, $M_{s,k} = C_{s,k}$, indicating that the optimal k-partial resolution at s is a k-speciation resolution. case, taking the $m_{2,s}$-speciation resolution at s of cost γ_s and adding $k - m_{2,s}$ loss leaves labeled s yields a k-partial resolution at s with minimum score $\gamma_s + k - m_{2,s}$. In the third case, taking the $m_{1,s}$-speciation resolution at s of cost γ_s and creating $m_{1,s} - k$ duplications labeled s yields a k-partial resolution at s with minimum score $\gamma_s + m_{1,s} - k$. □

Algorithm DupLoss(s,k) computes the values $Dup(s)$ and $Loss(s)$, being respectively the number of duplications labeled s and the number of loss leaves labeled s in a minimum resolution of G. Starting with a call to Algorithm DupLoss$(root(S), 1)$, the output is a pair $(Dup(s), Loss(s))$ for each node s of S. From these values, it is easy to reconstruct the corresponding solution $R(G,S)$ of the Minimum Resolution problem.

Algorithm 2 *DupLoss*(s,k)

```
if $s$ is a leaf and $k \geq nb(s)$ then
    $Dup(s) := 0; Loss(s) := k - nb(s);$
else if $s$ is a leaf and $k < nb(s)$ then
    $Dup(s) := nb(s) - k; Loss(s) := 0;$
else if $k - nb(s) > 0$ and $M_{s,k} = M_{s_\ell,k-nb(s)} + M_{s_r,k-nb(s)}$ then
    $Dup(s) := 0; Loss(s) := 0;$
    $DupLoss(s_\ell, k - nb(s)); DupLoss(s_r, k - nb(s));$
else if $k < m_{1,s}$ then
    $Dup(s) := m_{1,s} - k; Loss(s) := 0;$
    $DupLoss(s_\ell, m_{1,s} - nb(s)); DupLoss(s_r, m_{1,s} - nb(s));$
else if $k > m_{2,s}$ then
    $Dup(s) := 0; Loss(s) := k - m_{2,s};$
    $DupLoss(s_\ell, m_{2,s} - nb(s)); DupLoss(s_r, m_{2,s} - nb(s));$
end if
```

Once the vectors Dup and $Loss$ have been computed, an optimal resolution of G can be constructed easily, knowing $nb(g)$ for each node, and knowing how many of these nodes are joined under duplication or speciation and how many are inserted as losses.

4 Discussion

We have developed an algorithm for constructing the most parsimonious reconciliation, in term of number of duplications + losses, of a polytomy G with a binary species tree S, running in time $O(|S|)$. It naturally leads to an $O(|G| \times |S|)$ algorithm for the reconciliation of a gene tree with an arbitrary number of polytomies. Indeed, it is sufficient to traverse the tree in a depth-first manner, and resolve each polytomy at a time. Interestingly, we can find an example of trees G and S leading to a reconciliation of size $|G| \times |S|$. Indeed, let $\Sigma = \{1, 2, \cdots s\}$, and consider the species tree S over Σ to be a caterpilar tree $(1, 2, \cdots s)$ with leaves ordered from 1 to s, where $s = |S|$. Consider the gene tree G to be the caterpilar tree $((l_1, r_1), \cdots (l_g, r_g))$ composed of g cherries (l_1, r_s), where the l leaves are labeled 1, and the r leaves are labeled s. We have $g = |G|$. Then clearly a most parsimonious reconciliation of G and S is one with $s - 2$ leaves inserted in each cherry of G, which leads to a tree of size $|G| \times |S|$. Therefore, the algorithm is optimal for our considered Minimum Resolution problem. It is likely however that finding the mutation cost of an optimal reconciliation, without displaying the actual reconciliation, can be done in linear-time.

References

1. O. Akerborg, B. Sennblad, L. Arvestad, and J. Lagergren. Simultaneous bayesian gene tree reconstruction and reconciliation analysis. *Proceedings of the National Academy of Sciences USA*, 106(14):5714-5719, 2009.
2. L. Arvestad, A.-C. Berglung, J. Lagergren, and B. Sennblad. Gene tree reconstruction and orthology analysis based on an integrated model for duplications and sequence evolution. In D. Gusfield, editor, *RECOMB '04: Proceedings of the Eighth Annual International Conference on Research in Computational Molecular Biology*, pages 326–335, New York, 2004. ACM.
3. W.C. Chang and O. Eulenstein. Reconciling gene trees with apparent polytomies. In D.Z. Chen and D. T. Lee, editors, *Proceedings of the 12th Conference on Computing and Combinatorics (COCOON)*, volume 4112 of *Lecture Notes in Computer Science*, pages 235–244, 2006.
4. W.C. Chang and O. Eulenstein. Reconciling gene trees with apparent polytomies, technical report. In *Department of Computer Science, Iowa State University*, 2006.
5. C. Chauve and N. El-Mabrouk. New perspectives on gene family evolution: losses in reconciliation and a link with supertrees. In *RECOMB 2009*, volume 5541 of *LNCS*, pages 46-58. Springer, 2009.
6. K. Chen, D. Durand, and M. Farach-Colton. Notung: Dating gene duplications using gene family trees. *Journal of Computational Biology*, 7:429–447, 2000.
7. M. Csűrös. Ancestral reconstruction by asymmetric wagner parsimony over continuous characteand squared parsimony over distributions. *Sixth RECOMB Satellite Workshop on Comparative Genomics*, pages 72–86, 2008.
8. A. Doroftei and N. El-Mabrouk. Removing noise from gene trees. In *WABI*, volume 6833 of *LNBI/LNBI*, pages 76-91, 2011.
9. J-P. Doyon, V. Ranwez, V. Daubin, and V. Berry. Models, algorithms and programs for phylogeny reconciliation. *Brief Bioinform*, 12:392–400, 2011.
10. J.P. Doyon, C. Scornavacca, and G.J. SzÃPllösi et al. An efficient algorithm for gene/species trees parsimonious reconciliation with losses, duplications and transfers. *Proc 14th Int Conf Res Comput Mol Biol (RECOMB-CG) 2011*, pages 93–108, 2011.
11. D. Durand, B.V. Haldórsson, and B. Vernot. A hybrid micro-macroevolutionary approach to gene tree reconstruction. *Journal of Computational Biology*, 13:320–335, 2006.
12. Gang Fang, Nitin Bhardwaj, Rebecca Robilotto, and Mark B. Gerstein. Getting started in gene orthology and functional analysis. *PLoS Comput Biol*, 6(3):e1000703, 03 2010.
13. M. Goodman, J. Czelusniak, G.W. Moore, A.E. Romero-Herrera, and G. Matsuda. Fitting the gene lineage into its species lineage, a parsimony strategy illustrated by cladograms constructed from globin sequences. *Systematic Zoology*, 28:132–163, 1979.
14. P. Gorecki and J. Tiuryn. DLS-trees: a model of evolutionary scenarios. *Theoretical Computer Science*, 359:378–399, 2006.
15. M.W. Hahn. Bias in phylogenetic tree reconciliation methods: implications for vertebrate genome evolution. *Genome Biology*, 8(R141), 2007.
16. B. Ma, M. Li, and L. Zhang. From gene trees to species trees. *SIAM J. on Comput.*, 30:729–752, 2000.
17. J.B. Slowinski. Molecular polytomies. *Molecular Phylogenetics and Evolution*, 19(1):114-120, 2001.
18. A. Tofigh, M. Hallett, and J. Lagergren. Simultaneous identification of duplications and lateral gene transfers. *IEEE/ACM Trans Comput Biol Bioinform 2011*, 8:517–535., 2011.
19. J. Zhang. Evolution by gene duplication: an update. *TRENDS in Ecology and Evolution*, 18(6):292-298, 2003.
20. L.X. Zhang. On Mirkin-Muchnik-Smith conjecture for comparing molecular phylogenies. *Journal of Computational Biology*, 4:177–188., 1997.
21. Y. Zheng, T. Wu, and L. Zhang. Reconciliation of gene and species trees with polytomies. eprint arXiv:1201.3995, 2012.
22. C. M. Zmasek and S. R. Eddy. A simple algorithm to infer gene duplication and speciiation events on a gene tree. *Bioinformatics*, 17:821– 828, 2001.

Appendix

A Proof of lemma 5

Proof. Let

$$\ell(k) = \begin{cases} \gamma_\ell + \ell_1 - k & \text{if } k < \ell_1 \\ \gamma_\ell & \text{if } \ell_1 \leq k \leq \ell_2 \\ \gamma_\ell + k - \ell_2 & \text{if } k > \ell_2 \end{cases}$$

$$r(k) = \begin{cases} \gamma_r + r_1 - k & \text{if } k < r_1 \\ \gamma_r & \text{if } r_1 \leq k \leq r_2 \\ \gamma_r + k - r_2 & \text{if } k > r_2 \end{cases}$$

and $m(k) = \ell(k) + r(k)$. The addition of $\ell(k)$ and $r(k)$ yields a function with nine possible cases.

$$m(k) = \ell(k) + r(k) = \begin{cases} \gamma_\ell + \ell_1 - k + \gamma_r + r_1 - k & \text{if } k < \ell_1, k < r_1 \\ \gamma_\ell + \ell_1 - k + \gamma_r & \text{if } k < \ell_1, r_1 \leq k \leq r_2 \\ \gamma_\ell + \gamma_r + r_1 - k & \text{if } \ell_1 \leq k \leq \ell_2, k < r_1 \\ \gamma_\ell + \gamma_r & \text{if } \ell_1 \leq k \leq \ell_2, r_1 \leq k \leq r_2 \\ \gamma_\ell - \ell_2 + \gamma_r + r_1 & \text{if } k > \ell_2, k < r_1 \\ \gamma_\ell + \ell_1 + \gamma_r - r_2 & \text{if } k < \ell_1, k > r_2 \\ \gamma_\ell + \gamma_r + k - r_2 & \text{if } \ell_1 \leq k \leq \ell_2, k > r_2 \\ \gamma_\ell + k - \ell_2 + \gamma_r & \text{if } k > \ell_2, r_1 \leq k \leq r_2 \\ \gamma_\ell + k - \ell_2 + \gamma_r + k - r_2 & \text{if } k > \ell_2, k > r_2 \end{cases}$$

These nine cases are not all compatible with each other depending on ℓ_1, ℓ_2, r_1, r_2. In fact, there are only six possible cases of how these breakpoints cross eachother. If $\ell_1 < r_1$, either $\ell_2 < r_1, r_1 \leq \ell_2 \leq r_2$ or $\ell_2 > r_2$. If $r_1 \leq \ell_1 \leq r_2$, either $r_1 \leq \ell_2 \leq r_2$. If $\ell_1 > r_2$, then $\ell_2 > r_2$. Let $\alpha = \gamma_\ell + \gamma_r$. Then, $m(k)$ can be rewritten as :

$$\text{If } \ell_1 < r_1, \ell_2 < r_1, \text{then } m(k) = \begin{cases} \alpha - 2k + \ell_1 + r_1 & \text{if } k < \ell_1, k < r_1 \\ \alpha - k + r_1 & \text{if } \ell_1 \leq k \leq \ell_2, k < r_1 \\ \alpha + r_1 - \ell_2 & \text{if } k > \ell_2, k < r_1 \\ \alpha + k - \ell_2 & \text{if } k > \ell_2, r_1 \leq k \leq r_2 \\ \alpha + 2k - \ell_2 - r_2 & \text{if } k > \ell_2, k > r_2 \end{cases}$$

$$\text{If } \ell_1 < r_1, r_1 \leq \ell_2 \leq r_2, \text{then } m(k) = \begin{cases} \alpha - 2k + \ell_1 + r_1 & \text{if } k < \ell_1, k < r_1 \\ \alpha - k + r_1 & \text{if } \ell_1 \leq k \leq \ell_2, k < r_1 \\ \alpha & \text{if } \ell_1 \leq k \leq \ell_2, r_1 \leq k \leq r_2 \\ \alpha + k - \ell_2 & \text{if } k > \ell_2, r_1 \leq k \leq r_2 \\ \alpha + 2k - \ell_2 - r_2 & \text{if } k > \ell_2, k > r_2 \end{cases}$$

$$\text{If } \ell_1 < r_1, \ell_2 > r_2, \text{then } m(k) = \begin{cases} \alpha - 2k + \ell_1 + r_1 & \text{if } k < \ell_1, k < r_1 \\ \alpha - k + r_1 & \text{if } \ell_1 \leq k \leq \ell_2, k < r_1 \\ \alpha & \text{if } \ell_1 \leq k \leq \ell_2, r_1 \leq k \leq r_2 \\ \alpha + k - r_2 & \text{if } \ell_1 \leq k \leq \ell_2, k > r_2 \\ \alpha + 2k - \ell_2 - r_2 & \text{if } k > \ell_2, k > r_2 \end{cases}$$

$$\text{If } r_1 \le \ell_1 \le r_2, r_1 \le \ell_2 \le r_2, \text{then } m(k) = \begin{cases} \alpha - 2k + \ell_1 + r_1 & \text{if } k < \ell_1, k < r_1 \\ \alpha - k + \ell_1 & \text{if } k < \ell_1, r_1 \le k \le r_2 \\ \alpha & \text{if } \ell_1 \le k \le \ell_2, r_1 \le k \le r_2 \\ \alpha + k - \ell_2 & \text{if } k > \ell_2, r_1 \le k \le r_2 \\ \alpha + 2k - \ell_2 - r_2 & \text{if } k > \ell_2, k > r_2 \end{cases}$$

$$\text{If } r_1 \le \ell_1 \le r_2, \ell_2 > r_2, \text{then } m(k) = \begin{cases} \alpha - 2k + \ell_1 + r_1 & \text{if } k < \ell_1, k < r_1 \\ \alpha - k + \ell_1 & \text{if } k < \ell_1, r_1 \le k \le r_2 \\ \alpha & \text{if } \ell_1 \le k \le \ell_2, r_1 \le k \le r_2 \\ \alpha + k - r_2 & \text{if } \ell_1 \le k \le \ell_2, k > r_2 \\ \alpha + 2k - \ell_2 - r_2 & \text{if } k > \ell_2, k > r_2 \end{cases}$$

$$\text{If } \ell_1 > r_2, \ell_2 > r_2, \text{then } m(k) = \begin{cases} \alpha - 2k + \ell_1 + r_1 & \text{if } k < \ell_1, k < r_1 \\ \alpha - k + \ell_1 & \text{if } k < \ell_1, r_1 \le k \le r_2 \\ \alpha - r_2 + \ell_1 & \text{if } k < \ell_1, k > r_2 \\ \alpha + k - r_2 & \text{if } \ell_1 \le k \le \ell_2, k > r_2 \\ \alpha + 2k - \ell_2 - r_2 & \text{if } k > \ell_2, k > r_2 \end{cases}$$

Using the breakpoints and minimum values provided by the table in the theorem statement, it can be verified that each of these cases is a cup function. □