

Reconciling Gene Trees with Apparent Polytomies^{*}

Wen-Chieh Chang and Oliver Eulenstein

Department of Computer Science, Iowa State University, Ames, IA 50011, USA
{wcchang, oeulenstein}@cs.iastate.edu

Abstract. We consider the problem of reconciling gene trees with a species tree based on the widely accepted Gene Duplication model from Goodman *et al.* Current algorithms that solve this problem handle only binary gene trees or interpret polytomies in the gene tree as true. While in practice polytomies occur frequently, they are typically not true. Most polytomies represent unresolved evolutionary relationships. In this case a polytomy is called *apparent*. In this work we modify the problem of reconciling gene and species trees by interpreting polytomies to be apparent, based on a natural extension of the Gene Duplication model. We further provide polynomial time algorithms to solve this modified problem.

1 Introduction

In order to predict the function of genes it is critical to distinguish between speciation and duplication events in the genes' common evolutionary history [1,2]. Duplication events, which are pervasive in many gene families, typically result in incongruence between evolutionary histories of genes and the histories of the species from which the genes were sampled from. Evolutionary histories of either genes or species are represented through rooted phylogenetic trees (where every internal node has at least two children) and we refer to them as either *gene* or *species trees* respectively. An example for incongruence between a gene and its species tree that is caused by gene duplication is depicted in Fig. 1.

The gene duplication (GD) model from Goodman *et al.* [3] infers gene duplication events and losses from the incongruence of a given gene and species tree. While the basic GD model has been widely accepted and utilized through efficient algorithms [4,5,6,7,8], the model is constrained by its interpretation of internal nodes that have more than two children called polytomies. Polytomies can be either 'true' or 'apparent' [9,10]. A polytomy is true if all of its children diverged from it at the same time. A polytomy is apparent when it replaces some phylogenetic subtree that could not be fully resolved in the evolutionary history. In practice, most gene trees contain numerous weakly supported or completely unresolved evolutionary relationships that may be represented most accurately by apparent polytomies. The original GD model is confined to true polytomies.

^{*} This research is supported by NSF Grant No. 0334832.

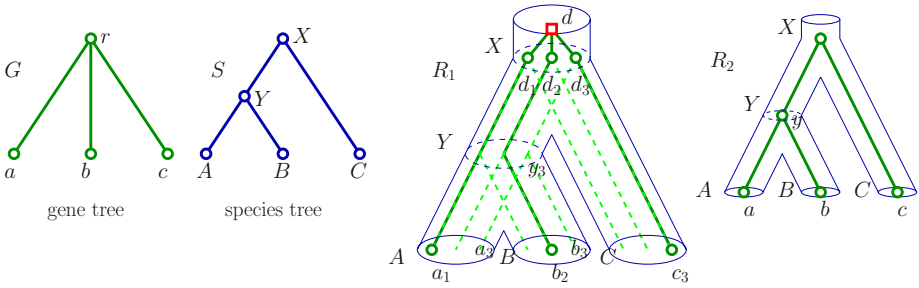


Fig. 1. The reconciled trees R_1 and R_2 explain the inconsistencies between the gene tree G and the species tree S assuming that r is a true and an apparent multifurcation respectively. The reconciled tree R_1 "truly" duplicates node d into the copies d_1 , d_2 , and d_3 in species X . Each of the copies evolves along the species tree and follows speciation events. The embedding of G into R_1 is highlighted by the solid edges. The reconciled tree R_2 explains the inconsistency assuming that r is an apparent multifurcation which is replaced by the "unknown" topology of the species tree S without any duplication event.

Since true polytomies are rare evolutionary events, available algorithms for the GD model were mostly designed for only fully binary input trees.

In this work we introduce the first algorithm that infers gene duplications and losses from gene trees by interpreting polytomies as apparent. We (i) show a natural modification of the basic GD model for apparent polytomies, (ii) formulate the Reconciliation problem that infers duplications and losses from the extended GD model, (iii) present an overview of structural properties of the extended GD model, and (vi) derive from these properties a polynomial time algorithm that solves the reconciliation problem.

1.1 Previous Work and Interpreting the GD Model

Goodman *et al.* [3] introduced the GD model to infer gene duplication and losses for a given rooted gene and species tree. This work was formalized and later refined by Page [4] and others [11,12,6,13,14,15,16].

Given a gene tree G and a species tree S , the GD model assumes that a surjective (onto) mapping from the leaves of the gene tree to the leaves of the species tree is provided. This *leaf association function* maps each leaf gene node to the species which it was sampled from. An LCA mapping function $LCA : V(G) \rightarrow V(S)$ can be computed in linear time on a PRAM [13] (see also [17]) from the leaf association function where $LCA(g)$ is the most recent species in S that theoretically can contain gene g . The node $LCA(g)$ is the *species of g* .

Theoretically, the idea of the basic GD model is to infer all possible gene trees from the species tree S by allowing genes to either duplicate in two or more copies or to speciate. Duplication can only take place in one species at a time, thus the duplicated gene and its copies have the same species. For example in Fig. 1 gene d is duplicated into three copies d_1 , d_2 and d_3 and all genes belong

to the same species. Speciation occurs when a gene in a species evolves into one gene for each child of the species. For example in Fig. 1 gene y_3 in species Y speciates into gene a_3 and b_3 in the species A and B respectively. A gene tree that is derived from the species tree by either duplicating or speciating genes in nodes of a species tree is called a duplication tree. For example the tree R_1 in Fig. 1 is a duplication tree.

Formally D is a *duplication tree* of S , if there exists a function $\text{Dup} : V(D) \rightarrow V(S)$ such that: (i) a leaf in D maps to a leaf in S , and (ii) every internal node u in D is either a duplication or a speciation node. Let $\text{Ch}_T(x)$ denote the children of a node x in a rooted tree T . The node u is a *duplication node* (or d-node) if $\text{Dup}(\text{Ch}_D(u)) = \{\text{Dup}(u)\}$, and it is a *speciation node* (or s-node) if $\text{Dup}(\text{Ch}_D(u)) = \text{Ch}_S(\text{Dup}(u))$. We also call $\text{Dup}(u)$ the *species of u* .

Some of the duplication trees are evolutionary compatible with the gene tree. In this case, the gene tree G can be embedded into the duplication tree D through an *embedding function* $\text{Emb} : V(G) \rightarrow V(D)$ that preserves the pairwise least common ancestor relations in G . Fig. 1 depicts an example where gene tree G can be embedded (solid lines) into the duplication tree R_1 . Duplication trees that allow such an embedding of the gene tree are *explanation trees*. Explanation trees are evolutionary compatible with the gene tree and thus explain the incompatibility between the gene tree and the species tree through gene duplication and losses. A *loss* is a maximum subtree in E that has no embedding from the gene tree G . E.g. in Fig. 1 the subtree rooted at node z_3 of the reconciled tree R_1 is a loss.

Of particular biological interest are two special types of explanation trees for a given gene and a species tree.

Node reconciled trees are explanation trees with the minimum number of nodes [4]. It was shown by [6] and later by [16] that a reconciled tree is uniquely determined by the given gene and species trees.

Dup-loss reconciled trees are explanation trees with the minimum reconciliation cost. The reconciliation cost of an explanation tree E is the duplication cost of E plus the number of losses in E . The *duplication cost* is the overall number of copies minus one for each d-node in E .

Node and dup-loss reconciled trees and their reconciliation cost can be computed in polynomial time for complete binary gene and species trees [4,7,5]. Node reconciled trees and their reconciliation cost can be computed for general gene and species trees under true polytomies [6].

In the case of complete binary gene and species trees the definitions of node and dup-loss reconciled trees are equivalent [16,18].

This is not true for gene and species trees with apparent polytomies as it is shown in Fig. 1. Thus we consider node and dup-loss reconciled trees for our extended GD model.

1.2 Presented Work

In this paper, we modify the basic GD model to interpret polytomies in gene trees as apparent. Apparent polytomies represent unknown phylogenetic subtrees.

Thus, the idea of the modified GD model is to replace the polytomies by complete binary subtrees. Therefore we consider the set \mathcal{G} of all gene trees that we construct from the given gene tree G by replacing star trees, which are the polytomies and their children, with more refined trees. Let $\text{Exp}_{G,S}$ be the set that contains the explanation trees for each combination of a gene tree in \mathcal{G} and the species tree S in the basic GD model. Equivalently $\text{Exp}_{G,S}$ can be described as the duplication trees into which the gene tree G can be embedded using a relaxed embedding function. The *relaxed embedding* is defined similar to the embedding for the basic GD model, but preserves only the tree order, rather than the pairwise least common ancestor relations. Fig. 1 depicts an example. The solid lines in the reconciled tree R_1 represent an embedding under preserving the pairwise least common ancestor relations. In contrast the solid lines in the reconciled tree R_2 represent an embedding that preserves the tree order, but not the pairwise least common ancestor relations.

As we show, node and dup-loss reconciled trees are not necessarily unique thus their definitions are not equivalent. Given a gene and its species tree, the *node reconciliation problem* is to find a node reconciled tree and the *dup-loss reconciliation problem* is to find a dup-loss reconciled tree. In this paper we show that both problems are solvable in polynomial time (an asymptotic upper bound is provided in Section 3.4).

1.3 Outline

We solve the node reconciled tree problem through a divide-and-conquer approach that divides the node reconciled tree problem into independent subproblems that we solve directly through dynamic programming. Section 2 presents an overview of the divide-and-conquer approach and Sections 3 introduces a dynamic programming solution. We briefly describe a similar solution for the dup-loss problem in Section 4. Due to space requirements we refer the reader for most of our proofs to the technical report of Chang and Eulenstein [18].

Let G be a gene tree, S its species tree, and R a node reconciled tree from G to S . Further let (i) LCA be the LCA mapping from G to S , (ii) Dup specify R , and (iii) Emb be the relaxed embedding from G into R . The subtree of a tree T rooted at node $v \in V(T)$ is denoted as T_v .

2 Divide-and-Conquer

The gene duplication problem (GDP) is formally defined as:

Instance: A gene tree G , a species tree S and a leaf association A from G to S .

Find: A reconciled tree R from G to S w.r.t. A .

GDP can be divided into independent subproblems based on the following theorem.

Theorem 1 (LCA-theorem). *For every gene g in the gene tree, it holds that $\text{LCA}(g) = \text{Dup}(\text{Emb}(g))$ if R is a reconciled tree.*

The theorem shows that the species of a gene in G and the species of its embedding in the reconciled tree R are identical. This allows to partition the edges of the gene and species tree into independent subproblems of the node reconciled tree problem, and the edges of the reconciled tree into solutions to these subproblems.

Consider a partition of the edges in the gene tree G into star trees (parent-child edges). Each star tree rooted at node g defines the edges of a subtree in the species tree, called the *environment of g in the species tree S* . This subtree is defined to be rooted at the node $\text{LCA}(g)$ where the subtrees rooted at $\text{LCA}(c)$ for every child c of g are removed. Similarly, we define the *environment of g in the reconciled tree R* .

As a result we partition the original problem instance, the gene and species tree, into subproblems consisting of a star tree in G and its environment in S . The subproblem, called *the core problem*, is defined as follows.

Instance: A star-tree G where $C = \text{Le}(G)$, a tree S and a mapping function $M : V(G) \rightarrow V(S)$ where $M(\text{Root}(G)) = \text{LCA}_S(M(C))$.

Find: A reconciled tree from G to S w.r.t. M .

Our claim is that the solution to the core problem is the environment of g in R . A cut-and-paste argument verifies this claim. Suppose the environment of g in R is not a solution, then there exists an explanation tree that solves the given subproblem with fewer nodes. Replacing the environment of g in R with this explanation tree results in an explanation tree for the original problem that has fewer nodes than R .

Following the same argument, the solutions to the core problems can be combined to obtain a solution to the original GDP.

3 Solving the Core Problem

Here we outline a dynamic programming approach for solving the core problem. To show that solutions to the core problem exhibit optimal substructure we prove that every solution contains node reconciled trees of a particular form, called *normal form*. The normal form allows us to describe the size of a node reconciled tree recursively that can be then computed by dynamic programming.

3.1 A Reconciled Tree in Normal Form

A node reconciled tree R is in *normal form* if for any d-node d in $V(R)$ with copies d_1, \dots, d_k the following two properties are satisfied:

1. **The property of normalized duplication:** There exists no d-node in $V(R_{d_i})$ for $2 \leq i \leq k$.
2. **The property of normalized embedding:** For any node u in $V(R_{d_i}) \cap \text{Emb}(V(G))$ ($1 \leq i < k$), there exists a distinct embedded node v in $V(R_{d_j}) \cap \text{Emb}(V(G))$ ($i < j \leq k$) where $\text{Dup}(v)$ is an ancestor of $\text{Dup}(u)$ in S .

We describe the effect on R if both properties are satisfied. Consider a duplication node d in R and its copies d_1, \dots, d_k ordered from left to right, and let C be the set of all children in the star-tree G that are embedded into the subtree rooted at d . Each subtree rooted at d_1, \dots, d_k exists because it contains an embedding from nodes in C . Thus the subtrees partition the set C into k non-empty sets, C_1, \dots, C_k , based on the nodes that are embedded into each subtree. Recall that R satisfies the first property. Thus the subtrees rooted at d_2, \dots, d_k , called *non-duplication subtrees*, do not contain any duplication. It follows that the species in a non-duplication subtree form an *anti-chain* in S (no two elements are on a same path). Now the second property requires that the elements in the anti-chains in S for each non-duplication subtree are ordered by \leq . We define $S_i \leq S_j$ if for any $i \in S_i$ and $j \in S_j$ that are on a same path, and i has smaller or equal depth then j in S . We refer to the gene nodes in C that form the ordered anti-chains in each subtree rooted at d_2, \dots, d_k as *layers*. An example of layers is depicted in Fig. 2.

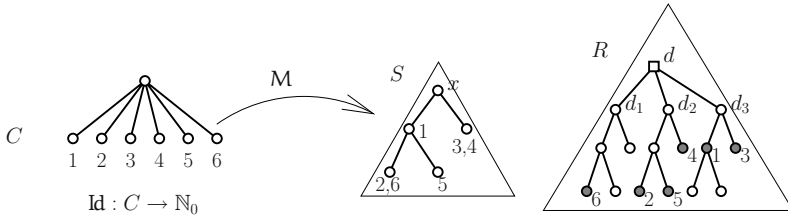


Fig. 2. The problem instance is simplified to a star gene tree where C is the leaf set. Without loss of generality, a total ordering (Id) in C is assumed. The mapping function M is essentially the LCA mapping function. Layers in C are $\{1, 3\}$, $\{2, 5, 4\}$ and $\{6\}$. Please note that in this example, R is not a reconciled tree since it is not optimal.

Theorem 2 (reconciled tree in normal form). *There exists a reconciled tree in normal form.*

The above theorem warrants the existence of a node reconciled tree in normal form in all problem instances. It is shown by an algorithm that transforms any arbitrary reconciled tree into a reconciled tree that satisfies the two properties. A similar approach shows that for the core problem, a reconciled tree in normal form has at most one d-node mapped to a single species node through Dup . This implication helps us to reconstruct an optimal solution in a dynamic programming approach. Furthermore, Theorem 2 identifies an optimal substructure in a node reconciled tree to form a recursive formula which will be introduced later. Because of Theorem 2, we will simply refer to a reconciled tree in normal form as a reconciled tree.

3.2 Optimal Substructure

In order to describe the substructure recursively, we introduce a notation to represent a partition of (a subset of) C . If v is a node in $V(R)$, then C_v is

the subset of C mapped to R_v through Emb , and the definition can be extended recursively to a subset of C . The same notation applies to a node in S . As shown in Fig. 2, we can use C_{d_1}, C_{d_2} and C_{d_3} to denote the layers $\{6\}, \{2, 5, 4\}$ and $\{1, 3\}$ respectively. And the set $\{2, 5, 4\}$ can be further partitioned into layers $\{4\}$ and $\{2, 5\}$ if necessary. Note that we can generalize the definition of layers with the imposed total ordering in C and the ordered anti-chains to describe them uniquely. That is, we call C_{d_3} the first layer of C_x , denoted by $\text{Layer}(C_x, 1)$, where $x = \text{Dup}(d)$, C_{d_2} the second layer of C_x , denoted by $\text{Layer}(C_x, 2)$, and C_{d_1} the third layer of C_x , denoted by $\text{Layer}(C_x, 3)$. We also denote $C_{d_1} \cup C_{d_2}$ as $\text{Remain}(C_x, 1)$, which represents $C_x - \text{Layer}(C_x, 1)$ and are called *remains*, to emphasize that each non-duplication subtree is embedded by a layer. Note that the numbers of non-empty layers and remains are always finite and no greater than $|C|$.

The linear ordering introduced in C simply gives a unique structure among layers and remains, and allows us to find an equivalent reconciled tree. With the generalization in mind, the following theorem concludes the optimal substructure to describe a reconciled tree recursively.

Theorem 3 (layer, remain embedding). *Let v be a vertex in R . It holds that C_v is either a layer or a remain of $C_{\text{Dup}(v)}$, and R_v is a reconciled tree from $G|_{C_v}$ to $S_{\text{Dup}(v)}$.*

The theorem can be shown by using the recursive structure introduced in the normal form, combining the definitions of layers and remains properly. In the next step, we show how to apply the theorem to formulate the substructure recursively.

3.3 Recursive Solution

In the case of node reconciled trees, the objective is to minimize the size of a reconciled tree, which means that we need to count the nodes first. In a top-down fashion, assuming we know whether a node is an s-node or a d-node (and the number of its copies), the formula can be illustrated by Fig. 3 as there are exactly two cases.

For the first case, suppose we know that the genes in C are embedded into the subtree R_d of a node reconciled tree in normal form rooted at a duplication node d with copies d_1, \dots, d_k . The first $(k - 1)$ layers of C are embedded into the subtrees R_{d_2}, \dots, R_{d_k} . Then $\text{Size}(R_d, C)$ is defined as the number of nodes in R_d if we embed C into it. We can then write $\text{Size}(R_d, C)$ as follows

$$\text{Size}(R_d, C) = \text{Size}(R_{d_1}, C_1) + (k - 1) \cdot \text{Size}(S_{d'}) + 1 \tag{1}$$

where C_1 is essentially $\text{Remain}(C, k - 1)$ and d' is the species $\text{Dup}(d)$ in S . $\text{Size}(S_{d'})$ is the size of each non-duplication subtree, since they do not contain any duplication, and it is given by S . We have $(k - 1)$ such subtrees. An additional node is counted for the root of R_d , and $\text{Size}(R_{d_1}, C_1)$ accounts for the size of the remaining subtree rooted at R_{d_1} . Furthermore, as implied by the normal

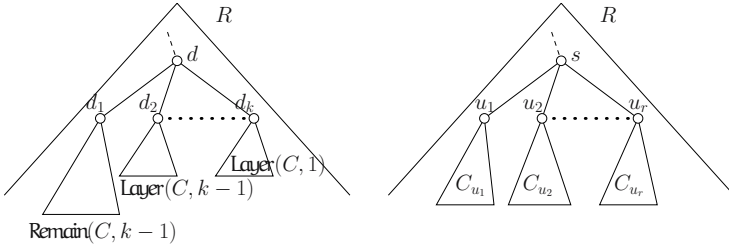


Fig. 3. Each complete subtree of a reconciled tree R has a layer or remain embedded. In the case that s is an s-node in R ; C_{u_1}, \dots, C_{u_r} form a partition of C_s . In the case that d is a d-node in R ; $R_{d_2} \dots R_{d_k}$ are embedded with the first $k - 1$ layers of C_d (denoted as C above).

form properties, d_1 in R has to be an s-node, which automatically advances the recursion into the second scenario.

In the second scenario, also illustrated in Fig. 3, if a node s in R is an s-node, we know exactly its children, as in the species tree. Therefore the objective function $\text{Size}(R_s, C)$ can be expressed similarly to Equation (1)

$$\text{Size}(R_s, C) = \sum_{u \in \text{Ch}(s)} \text{Size}(R_u, C_u) \tag{2}$$

The above two equations describe the size of a reconciled tree precisely in a top-down recursive fashion. By reversing the direction, we can find an optimal solution (as they are not necessarily unique) from bottom up. In the next step, we present the general idea behind the optimization.

3.4 Optimal Node Reconciled Tree

The goal of optimization is essentially finding the optimal number of copies of each node in a reconciled tree. Since a d-node has at least two copies, we can generalize the notion by saying an s-node has exactly one copy. As demonstrated in the previous discussion, the number of allowed copies for each node is bounded by the number of non-empty layers in the remain, which may be reduced by a duplication event of some ancestor node. Hence, the solution can be found by trying all possible subproblems in a given subtree then finding the optimal number of copies.

The process can be memoized using an $O(|S| \cdot |C|)$ table for all feasible combinations of vertices and their non-empty remains. One also needs to know $|S_v|$ for each $v \in V(S)$, in which a DFS traversal is sufficient. All leaf nodes can be initialized immediately since they are all base cases. Each cell stores two values: $\text{Size}(v, C'_v)$ and $k(v, C'_v)$, where C'_v is a non-empty remain of v , as the solution to the subproblem under the given condition. Once the table is filled up in a bottom-up fashion, $\text{Size}(\text{Root}(S), C)$ represents $|R|$. The whole optimization process takes $O(|S| \cdot |C|^2)$ time as each cell has at most $|C|$ possible values for k (and those values are not all necessarily feasible).

3.5 Constructing a Node Reconciled Tree

It suffices to build a node reconciled tree R , if the optimal k is found for each $v \in V(S)$. Starting at $r = \text{Root}(S)$, $k(r, C)$ determines whether the root of the resulting node reconciled tree is a d-node or an s-node. For $v \in \text{In}(S)$, if $k(v, C'_v) = k'$ is optimal, we know there are k' copies of the node x in R such that $\text{Dup}(x) = v$ and the embedding nodes in children of x are also determined accordingly for the next step. Essentially by backtracking $k(v, C'_v)$, there is a duplication function, and we also know the unique layer structures embedded into each non-duplication subtree, which gives an embedding function.

4 Dup-Loss Reconciled Tree

In order to minimize the duplication and loss cost, it is necessary to know how to calculate losses, since the duplication cost in a reconciled tree R is clear. As losses are closely related to the layers, a lookup table associates the number of losses in each subtree of S and a given layer can be calculated recursively as a part of pre-processing. The loss table is of size $O(|S| \cdot |C|)$, and we claim it can be filled up properly in $O(|S| \cdot |C|)$ time. The recursion is similar to Equation (1) and (2), but it computes the duplication and loss cost instead of the size of a subtree. Hence, to determine a dup-loss reconciled tree takes $O(|S| \cdot |C|^2)$ time as well.

5 Outlook

Of biological interest is the extension of the GD model that considers in addition to apparent polytomies in the gene tree also apparent topologies in the species tree. This extension requires a biologically meaningful definition of a reconcile tree. Using the relaxed embedding of our extended GD model for the reconciliation, a solution similar to the presented one might be sufficient.

Further *dup reconciled trees* that are explanation trees with the minimum duplication cost under the extended GD model might be valuable to biologists. A modification of Theorem 1 should allow to compute dup reconciled trees using our presented approach.

References

1. Page, R.D.M., Holmes, E.C.: Molecular Evolution: A Phylogenetic Approach. Blackwell Science Ltd, Osney Mead, Oxford (1998)
2. Felsenstein, J.: Inferring phylogenies. Sinauer Associates, Inc., Sunderland, Massachusetts (2004)
3. Goodman, M., Czelusniak, J., Moore, G.W., Romero-Herrera, A.E., Matsuda, G.: Fitting the gene lineage into its species lineage, a parsimony strategy illustrated by cladograms constructed from globin sequences. *Systematic Zoology* **28** (1979) 132–163
4. Page, R.D.M.: Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas. *Systematic Biology* **43**(1) (1994) 58–77

5. Eulenstein, O.: A linear time algorithm for tree mapping (1997) <http://taxonomy.zoology.gla.ac.uk/rod/genetree/maths/maths.html>, (access date: February 19, 2006), Arbeitspapiere der GMD, 1046.
6. Eulenstein, O.: Vorhersage von Genduplikationen und deren Entwicklung in der Evolution. Ph.d. dissertation, Bonn University (1998) <http://www.bi.fraunhofer.de/publications/research/1998/020/Text.pdf>, GMD Research Series, 20.
7. Ma, B., Li, M., Zhang, L.: On reconstructing species trees from gene trees in term of duplications and losses. In: RECOMB. (1998) 182–191
8. Zmasek, C.M., Eddy, S.R.: A simple algorithm to infer gene duplication and speciation events on a gene tree. *Bioinformatics* **17**(9) (2001) 821–828
9. Maddison, W.P.: Reconstructing character evolution on polytomous cladgrams. *Cladistics* **5** (1989) 365–377
10. Slowinski, J.B.: Molecular polytomies. *Molecular Phylogenetics and Evolution* **19**(1) (2001) 114–120
11. Guigó, R., Muchnik, I., Smith, T.F.: Reconstruction of ancient molecular phylogeny. *Molecular Phylogenetics and Evolution* **6**(2) (1996) 198–213
12. Mirkin, B., Muchnik, I., Smith, T.F.: A biologically meaningful model for comparing molecular phylogenies. *Journal of Computational Biology* **2** (1995) 493–507
13. Zhang, L.: On a mirkin-muchnik-smith conjecture for comparing molecular phylogenies. *Journal of Computational Biology* **4**(2) (1997) 177–187
14. Chen, K., Durand, D., Farach-Colton, M.: Notung: Dating gene duplications using gene family trees. In: RECOMB. (2000) 96–106
15. Bonizzoni, P., Vedova, G.D., Dondi, R.: Reconciling gene trees to a species tree. In: CIAC2003 - Italian Conference on Algorithms and Complexity, Rome, Italy (2003)
16. Górecki, P., Tiuryn, J.: On the structure of reconciliations. In: *Recomb Comparative Genomics Workshop 2004*. Volume 3388. (2004)
17. Bender, M.A., Farach-Colton, M.: The lca problem revisited. *Latin American Theoretical Informatics* (2000) 88–94 Apr.
18. Chang, W.C., Eulenstein, O.: Reconciling gene trees with apparent polytomies. Technical report, Department of Computer Science, Iowa State University (2006)