An $\Omega(n^2/\log n)$ Speed-Up of TBR Heuristics for the Gene-Duplication Problem [⋆]

Mukul S. Bansal and Oliver Eulenstein

Department of Computer Science, Iowa State University, Ames, IA, USA
{bansal, oeulenst}@cs.iastate.edu

Abstract. The gene-duplication problem is to infer a species supertree from gene trees that are confounded by complex histories of gene duplications. This problem is NP-hard and thus requires efficient and effective heuristics. Existing heuristics perform a stepwise search of the tree space, where each step is guided by an exact solution to an instance of a local search problem. We improve on the time complexity of the local search problem by a factor of $n^2/\log n$, where n is the size of the resulting species supertree. Typically, several thousand instances of the local search problem are solved throughout a stepwise heuristic search. Hence, our improvement makes the gene-duplication problem much more tractable for large-scale phylogenetic analyses.

1 Introduction

An abundance of potential information for phylogenetic analyses is provided by the rapidly increasing amount of available genomic sequence information. Most phylogenetic analyses combine genomic sequences, from presumably orthologous loci, or loci whose homology is the result of speciation, into gene trees. These analyses largely have to neglect the vast amounts of sequence information, in which gene duplication generates gene trees that differ from the actual species tree. Phylogenetic information from such gene trees can be utilized through a species tree obtained by solving the gene-duplication problem [1]. This problem is a type of *supertree problem*, that is, assembling from a set of gene trees a supertree that contains all species found in at least one of the input trees. The decision version of the gene-duplication problem is NP-complete [2]. Existing heuristics aimed at solving the gene-duplication problem search the space of all possible supertrees guided by a series of exact solutions to instances of a local search problem [3]. The gene-duplication problem has shown much potential for building phylogenetic species trees for snakes [4], vertebrates [5, 6], *Drosophia* [7], and plants [8]. Yet, the computation time of local search problems which are solved by existing heuristics has largely limited the size of such studies.

Throughout the current section n denotes the number of leaves in the resulting species tree, and, for brevity in stating time complexities, gene trees and the resulting species tree are assumed to have similar sizes.

[⋆] This research was supported in part by NSF grant no. 0334832

We improve on the best existing solution for a particular local search problem, the TBR local search problem, by a factor of $n^2/\log n$. Heuristics solving the TBR local search problem, TBR heuristics, were rarely applied in practice due to inefficient running times. Our method greatly improves the speed of TBR based heuristics for the gene-duplication problem and makes it possible to infer larger supertrees that were previously difficult, if not impossible, to compute.

For convenience, we use the term "tree" to refer to a rooted and full-binary tree. The terms "leaf-gene" and "leaf-species" refer to a gene or species that is represented by a leaf of a gene or species tree respectively throughout this work unless otherwise stated.

Previous Results: The gene-duplication problem is based on the Gene Duplication model from Goodman et al. [9]. In the following, we (i) describe the Gene Duplication model, (ii) formulate the gene-duplication problem, and (iii) describe a heuristic approach of choice [3] to solve the gene-duplication problem.

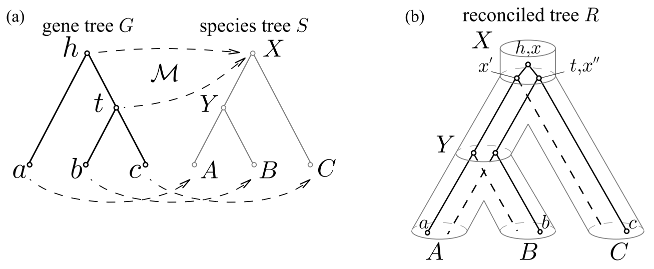

Fig. 1. (a) Gene trees G and species tree S are comparable, as the mapping from the leaf-genes to the leaf-species indicates. $\mathcal{M}$ is the lca-mapping from G to S. (b) R is the reconciled tree for G and S. In species X of R gene x duplicates into the genes x' and x''. The solid lines in R represent the embedding of G into R.

Gene Duplication model: The Gene Duplication (GD) model [1, 10–16] explains incompatibilities between a pair of "comparable" gene and species trees through gene duplications. A gene and a species tree are *comparable*, if a sample mapping, called *leaf-mapping*, exists that maps every leaf-gene to the leaf-species from which it was sampled. Figure 1 depicts an example. Gene tree G is inferred from the leave-genes that were sampled from the leaf-species of the species tree described by the leaf-mapping. However, both trees describe incompatible evolutionary histories. The GD model explains such incompatibilities by reconciling the gene tree with postulated gene duplications. For example, in Figure 1 a reconciled gene tree R can be theoretically inferred from the species tree S by duplicating a gene x in species X into the copies x' and x'' and letting both copies speciate according to the topology of S. In this case, the gene tree can be embedded into the reconciled tree. Thus, the gene tree can be reconciled by using the duplication of gene x to explain the incompatibility. The gene duplications that are necessary under the GD model to reconcile the gene tree can be described by the mapping $\mathcal{M}$, which is an extension of the given leaf-mapping. $\mathcal{M}$ maps every gene in the gene tree to the most recent species in the species tree that could have contained the gene (i.e. their least common ancestor). A gene in the gene tree is a *(gene) duplication* if it has a child with the same mapping under $\mathcal{M}$. In Figure 1 gene h and its child t map under the mapping $\mathcal{M}$ to the same species X. The *reconciliation cost* for a gene tree and a comparable species tree is measured in the number of duplications in the gene tree induced by the species tree. The *reconciliation cost* for a given set of gene trees and a species tree is the sum of the reconciliation costs for every gene tree in the set and the species tree. The reconciliation cost is linear time computable [13, 17, 18].

Gene-duplication problem and heuristic: The *gene-duplication problem* is to find, for a given set of gene trees, a comparable species tree with the minimum reconciliation cost. The decision variant of this problem and some of its characterizations are NP-complete [2, 19] while some parameterizations are fixed parameter tractable [20, 21]. However, GeneTree [3], an implementation of a standard local search heuristic for the gene-duplication problem, was used to show that the gene-duplication problem can be an effective approach. Therefore, in practice, heuristics are commonly applied to solve the gene-duplication problem, even if they are unable to guarantee an optimal solution. While the local search heuristic for the gene-duplication problem performs reasonably well in computing smaller sized instances, it does not allow the computation of larger species supertrees. In this heuristic, a *tree graph* is defined for the given set of gene trees and some, typically symmetric, tree edit operation. The nodes in the tree graph are the species trees which are comparable with every given gene tree. An edge adjoins two nodes exactly if the corresponding trees can be transformed into each other by the tree edit operation. The *reconciliation cost of a node* in the graph is the reconciliation cost of the species tree represented by that node and the given gene trees. Given a starting node in the tree graph, the heuristic's task is to find a maximal-length path of steepest descent in the reconciliation cost of its nodes and to return the last node on such a path. This path is found by solving the local search problem for every node along the path. The *local search* problem is to find a node with the minimum reconciliation cost in the neighborhood (all adjacent nodes) of a given node. The neighborhood searched depends on the edit operation. Edit operations of interest are rooted subtree pruning and regrafting (SPR) [22–24] and rooted tree bisection and reconnection (TBR) [22, 23, 25]. We defer the definition of these operations to Section 2. The best known run times for the SPR and TBR local search problems are $O(kn^2)$ [26] and $O(kn^4)$ (naive solution) respectively, where k is the number of input gene trees.

Our Contribution: The efficient solution for the SPR local search problem makes SPR based heuristics suitable for large-scale phylogenetic analyses. Currently, TBR based heuristics are not applicable for phylogenetic analyses because no efficient solution is known for the TBR local search problem. However, TBR based heuristics are more desirable because they significantly extend the search space explored at each local search step. In particular, TBR heuristics search a neighborhood of $\Theta(n^3)$ nodes, including the $\Theta(n^2)$ nodes of the SPR neighborhood, at each local search step. Our contribution is an $O(kn^2 \log n)$ algorithm for the TBR local search problem. This makes TBR heuristics almost as efficient as SPR heuristics for large-scale phylogenetic analyses.

2 Basic Definitions, Notation, and Preliminaries

In this section we first introduce basic definitions and notation and then define preliminaries required for this work.

2.1 Basic Definitions and Notation

A *tree* T is a connected graph with no cycles, consisting of a node set $V(T)$ and an edge set $E(T)$. The nodes in $V(T)$ of degree at most one are called *leaves* and denoted by $\mathsf{Le}(T)$. A node in $V(T)$ that is not a leaf is called an *internal* node. T is *rooted* if it has exactly one distinguished node called the *root* which we denote by $\mathsf{Ro}(T)$. Let T be a rooted tree. For any pair of nodes $x, y \in V(T)$ where y is on a path from $\mathsf{Ro}(T)$ to x we call (i) y an *ancestor* of x, and (ii) x a *descendant* of y. If $\{y, x\} \in E(T)$ then we call y the *parent* of x denoted by $\mathsf{Pa}(x)$ and we call x a *child* of y. We write (y, x) to denote the edge $\{y, x\}$ where $y = \mathsf{Pa}(x)$. The set of all children of y is denoted by $\mathsf{Ch}(y)$. If two nodes in T have the same parent, they are called *siblings*. T is (fully) *binary* if every internal node has exactly two children. A *subtree* of T rooted at node $x \in V(T)$, denoted by T_x, is the tree induced by x and all its descendants. The *depth* of a node $x \in V(T)$ is the number of edges on the path from $\mathsf{Ro}(T)$ to x. The *least common ancestor* of a non-empty subset $L \subseteq \mathsf{V}(T)$, denoted as $\mathrm{lca}(L)$, is the common ancestor of all nodes in L with maximum depth.

2.2 The Gene Duplication Problem

We now introduce necessary definitions to state the gene duplication problem. A *species tree* is a tree that depicts the evolutionary relationships of a set of species. Given a gene family for a set of species, a *gene tree* is a tree that depicts the evolutionary relationships among the sequences encoding only that gene family in the given species. Thus the nodes in a gene tree represent genes. In order to compare a gene tree G with a species tree S a mapping from each gene $g \in V(G)$ to the most recent species in S that could have contained g is required.

Definition 1 (Mapping). *The* leaf-mapping $\mathcal{L}_{G,S} \colon \mathsf{Le}(G) \to \mathsf{Le}(S)$ *specifies the species $\mathcal{M}_{G,S}(g)$ from which gene g was sampled from. An extension of $\mathcal{L}_{G,S}$ to $\mathcal{M}_{G,S} \colon V(G) \to V(S)$ is the* mapping *where $\mathcal{M}_{G,S}(g) = \mathcal{L}_{G,S}(g)$, if $g \in \mathsf{Le}(G)$, and $\mathcal{M}_{G,S}(g) = \mathrm{lca}(\mathcal{M}_{G,S}(\mathsf{Le}(G_g))$ otherwise.*

Definition 2 (Comparability). *The trees G and S are* comparable *if there exists a leaf-mapping $\mathcal{L}_{G,S}$. A set of gene trees $\mathcal{G}$ and S are* comparable *if each gene tree in $\mathcal{G}$ is comparable with S.*

Let G and S be comparable trees for the remainder of this section.

Definition 3 (Duplication). *A node $v \in V(G)$ is a* (gene) duplication *if $\mathcal{M}_{G,S}(v) = \mathcal{M}_{G,S}(u)$ for some $u \in \mathsf{Ch}(v)$ and we define $\mathsf{Dup}(G, S) = \{g \in V(G)\colon$ g is a duplication $\}$.*

Definition 4 (Reconciliation cost). *We define reconciliation costs for gene and species trees as follows:*

1. *$\Delta(G, S) = |\,\mathsf{Dup}(G, S)|$ is the* reconciliation cost from G to S.
2. *$\Delta(\mathcal{G}, S) = \sum_{G\in\mathcal{G}} \Delta(G, S)$ is the* reconciliation cost *from $\mathcal{G}$ to S.*
3. *Let $\mathcal{T}$ be the set of species trees that is comparable with $\mathcal{G}$. We define $\Delta(\mathcal{G}) = \min_{S\in\mathcal{T}} \Delta(\mathcal{G}, S)$ to be the* reconciliation cost of $\mathcal{G}$.

Problem 1 (Duplication)
Instance: *A set $\mathcal{G}$ of gene trees.*
Find: *A species tree S^* such that $\Delta(\mathcal{G}, S^*) = \Delta(\mathcal{G})$.*

2.3 Local Search Problems

Here we first provide definitions for the TBR [25] and SPR [24] edit operations and then formulate the related local search problems that were motivated in the Introduction.

Definition 5 (RR operation). *Let T be a tree and $x \in V(T)$. $\mathsf{RR}(T, x)$ is defined to be the tree T, if $x = \mathsf{Ro}(T)$. Otherwise, $\mathsf{RR}(T, x)$ is the tree that is obtained from T by (i) suppressing $\mathsf{Ro}(T)$, and (ii) subdividing the edge $\{\mathsf{Pa}(x), x\}$ by a new root node. We define the following extension: $\mathsf{RR}(T) = \bigcup_{x\in V(T)}\{\mathsf{RR}(T, x)\}$.*

Definition 6 (TBR operation). *For technical reasons we first define for a tree T the* planted tree *$P(T)$ that is the tree obtained by adding an additional edge, called* root edge, *$\{u, \mathsf{Ro}(T)\}$ to T.*

Let T be a tree, $e = (u, v) \in E(T)$ and X, Y be the connected components that are obtained by removing edge e from T where $v \in X$ and $u \in Y$. We define $\mathsf{TBR}_T(v, x, y)$ for $x \in X$ and $y \in Y$ to be the tree that is obtained from $P(T)$ by first removing edge e, then replacing the component X by $\mathsf{RR}(X, x)$, and then adjoining a new edge f between $x' = \mathsf{Ro}(\mathsf{RR}(X, x))$ and Y as follows:
1. *Create a new node y' that subdivides the edge $(\mathsf{Pa}(y), y)$.*
2. *Adjoin the edge f between nodes x' and y'.*
3. *Suppress the node u, and rename x' as v and y' as u.*

We say that the tree $\mathsf{TBR}_T(v, x, y)$ is obtained from T by a tree bisection and reconnection (TBR) *operation that* bisects *the tree T into the components X, Y and* reconnects *them above the nodes x, y.*

We define the following extensions for the TBR operation:
1. $\mathsf{TBR}_T(v, x) = \bigcup_{y\in Y}\{\mathsf{TBR}_T(v, x, y)\}$
2. $\mathsf{TBR}_T(v) \quad = \bigcup_{x\in X} \mathsf{TBR}_T(v, x)$
3. $\mathsf{TBR}_T \qquad = \bigcup_{(u,v)\in E(T)} \mathsf{TBR}_T(v)$

An SPR operation for a given tree T can be briefly described through the following three steps: (i) prune some subtree P from T, (ii) add a root edge to the remaining tree S, (iii) regraft P into an edge of the remaining tree S. For our purposes we define the SPR operation as a special case of the TBR operation.

Definition 7 (SPR operation). *Let T be a tree, $e = (u, v) \in E(T)$ and X, Y be the connected components that are obtained by removing edge e from T where $v \in X$ and $u \in Y$. We define $\mathsf{SPR}_T(v, y)$ for $y \in Y$ to the tree $\mathsf{TBR}_T(v, v, y)$. We say that the tree $\mathsf{SPR}_T(v, y)$ is obtained from T by a* subtree prune and regraft (SPR) *operation that* prunes *subtree T_v and* regrafts *it above node y.*

We define the following extensions of the SPR operation:

1. $\mathsf{SPR}_T(v) = \bigcup_{y\in Y}\{\mathsf{SPR}_T(v, y)\}$
2. $\mathsf{SPR}_T \quad = \bigcup_{(u,v)\in E(T)} \mathsf{SPR}_T(v)$

Problem 2 (TBR-Scoring (TBR-S))

Instance: *A gene tree set $\mathcal{G}$, and a comparable species tree S.*

Find: *A tree $T^* \in \mathsf{TBR}_S$ such that $\Delta(\mathcal{G}, T^*) = \min_{T\in\mathsf{TBR}_S} \Delta(\mathcal{G}, T)$.*

Problem 3 (TBR-Restricted Scoring (TBR-RS))

Instance: *A triple $(\mathcal{G}, S, v)$, where $\mathcal{G}$ is a set of gene trees, S is a comparable species tree, and $(u, v) \in E(S)$.*

Find: *A tree $T^* \in \mathsf{TBR}_S(v)$ such that $\Delta(\mathcal{G}, T^*) = \min_{T\in\mathsf{TBR}_S(v)} \Delta(\mathcal{G}, T)$.*

The problems SPR-***Scoring (***SPR-***S)*** and SPR-***Restricted Scoring (***SPR-***RS)*** are defined analogously to the problems TBR-S and TBR-RS respectively.

Throughout this paper we use the following terminology: (i) $\mathcal{G}$ is a set of gene trees, (ii) S denotes a compatible species tree, (ii) $r = \mathsf{Ro}(S)$, (iii) P denotes a proper (pruned) subtree of S, and (iv) $v = \mathsf{Ro}(P)$.

3 Solving the TBR-S problem

In this section we study the TBR-S problem in more detail. First, we show how the algorithm developed by Bansal et al. [26] to solve the SPR-RS problem can be slightly modified to solve the TBR-S problem. This already improves the running time of the existing solution considerably. Second, we show how the inherent structure of the TBR-S problem can be used to further improve the running time. To do this, we define the "BestRooting" problem, and show how an efficient solution for this problem leads to an efficient solution for the TBR-S problem.

3.1 Relating Scores of TBR and SPR Neighborhoods

The following algorithm Alg-SPR-RS is a brief restatement of the algorithm presented in [26] to solve the SPR-RS instance $(\mathcal{G}, S, v)$ efficiently.

Algorithm Alg-SPR-RS

1. Prune P from S, regraft P above node r to obtain the resulting tree denoted by $\Re(P)$. Compute the reconciliation cost of $\Re(P)$.

2. Compute the difference between the reconciliation cost of each tree in $\mathsf{SPR}_S(v)$ and $\Re(P)$. This gives the reconciliation cost of each tree in $\mathsf{SPR}_S(v)$.

Observe that $\mathsf{SPR}_S(v) = \mathsf{TBR}_S(v, v)$. In fact, Alg-$\mathsf{SPR}$-RS can be modified to efficiently compute the reconciliation costs of all trees in $\mathsf{TBR}_S(v, x)$ for any node $x \in V(P)$. To do this, we simply modify Step 1 of Alg-SPR-RS as follows:

1. Prune P from S, re-root P to obtain $P' = \mathsf{RR}(S, x)$, and regraft P' above node r to obtain $\Re(P')$. Compute the reconciliation cost of $\Re(P')$.

Note, this modification does not change the algorithms's complexity.

Observation 1 *The* TBR*-RS problem on* $(\mathcal{G}, S, v)$ *can be solved by computing the reconciliation cost of each tree in* $\mathsf{TBR}_S(v, x)$*, for all* $x \in V(P)$*. The* TBR*-S problem in turn can be solved by solving the* TBR*-RS problem* $|V(S)| - 1$ *times.*

Let us assume, for convenience, similar gene tree and species tree sizes. It is known that the SPR-RS problem is solvable in $O(kn)$ time [26], where $k = |\mathcal{G}|$. Based on Observation 1, and the modification described above, the TBR-S problem can then be solved in $O(kn^3)$ time. This already gives us a speed up of $\Theta(n)$ over known algorithms for this problem. We will show how to solve the TBR-S problem in $O(kn^2 \log n)$ time. This gives a speed-up of $\Theta(n^2/\log n)$ over existing algorithms. Also, it should be noted that the correctness or efficiency of our algorithm does not depend on the simplifying assumption of similar gene and species tree sizes.

It is interesting to note that the size of the set TBR_S is $\Theta(n^3)$. Thus, for one gene tree the time complexity of computing and enumerating the reconciliation costs of all trees in TBR_S is $\Omega(n^3)$.

However, to solve the TBR-S problem one is only interested in finding a tree with the minimum reconciliation cost. This lets us solve the TBR-S problem in time that is sub-linear in the size of TBR_S, and obtain a time complexity of $O(n^2 \log n)$ for the TBR-S problem. In fact, after the initial $O(n^2 \log n)$ preprocessing step, our algorithm can output the reconciliation cost of any tree in TBR_S in $O(1)$ time.

3.2 Relating TBR-RS with SPR-RS

To obtain our speed-up, we concentrate on improving the complexity of solving the TBR-RS problem. To do this, we take a closer look at Step 2 of Alg-SPR-RS. This part of the algorithm computes the difference in reconciliation cost of each tree in $\mathsf{SPR}_S(v)$ and the tree $\Re(P)$. To compute this difference, the algorithm considers only the leaf set of P, and not its topology. This means that the difference values would be the same if P was replaced by any tree $P' \in \mathsf{RR}(P)$. Based on this observation, we have the following theorem. In the interest of brevity, this theorem is stated here without proof.

Theorem 1. *Let* $x', x'' \in V(P)$*, and* $y', y'' \in V(S) \setminus (V(P) \cup \{r\})$*. Let* $T_1 = \mathsf{TBR}_S(v, x', y')$*,* $T_2 = \mathsf{TBR}_S(v, x', y'')$*, and,* $T_3 = \mathsf{TBR}_S(v, x'', y')$*,* $T_4 = \mathsf{TBR}_S(v, x'', y'')$*. Then,* $\Delta(\mathcal{G}, T_1) - \Delta(\mathcal{G}, T_2) = \Delta(\mathcal{G}, T_3) - \Delta(\mathcal{G}, T_4)$*.*

Corollary 1. *To obtain the reconciliation cost of each tree in* $\mathsf{TBR}_S(v)$*, it is sufficient to compute the reconciliation cost of* $\Re(P')$ *for each* $P' \in \mathsf{RR}(P)$*, and then perform Step 2 of Alg-*SPR*-RS starting with any* $\Re(P')$*,* $P' \in \mathsf{RR}(P)$*.*

This is because the output of Step 2 of Alg-SPR-RS will be the same for all $\Re(P')$ where $P' \in \mathsf{RR}(P)$.

To solve the TBR-RS problem it is sufficient to find one tree in $\mathsf{TBR}_S(v)$ with minimum reconciliation cost. Based on Alg-SPR-RS and Corollary 1 we have the following theorem.

Theorem 2. *Let* T_1 *be a tree with minimum reconciliation cost in* $\mathsf{TBR}_S(v)$*. Consider tree* $P' \in \mathsf{RR}(P)$ *where* $\Re(P')$ *has minimum reconciliation cost and let* $P' = \mathsf{RR}(P, x)$*. Then, there exists a tree* $T_2 \in \mathsf{TBR}_S(v, x)$ *such that* $\Delta(\mathcal{G}, T_1) = \Delta(\mathcal{G}, T_2)$*.*

In other words, to obtain a solution for the TBR-RS problem for instance $(\mathcal{G}, S, v)$, it is sufficient to obtain the reconciliation costs of only the trees in $\mathsf{TBR}_S(v, x)$, where $P' = \mathsf{RR}(P, x)$ such that $\Re(P')$ has the minimum reconciliation cost. Based on Corollary 1 and Theorem 2 we have the following corollary.

Corollary 2. *The minimum reconciliation cost of a tree in* $\mathsf{TBR}_S(v)$ *can be obtained by performing Step 2 of Alg-*SPR*-RS starting with* $\Re(P')$*, where* $P' \in \mathsf{RR}(P)$ *such that* $\Re(P')$ *has minimum reconciliation cost.*

Problem 4 (BestRooting (BR))

Instance:	*A set of gene trees* $\mathcal{G}$*, a compatible species tree* S*, and a proper subtree* P *of* S*.*
Find:	*A tree* $P' \in \mathsf{RR}(P)$ *for which* $\Delta(\mathcal{G}, \Re(P'))$ *is minimum.*

Thus, based on Observation 1, Theorems 1 and 2, and, Corollaries 1 and 2, an efficient solution to the BR problem leads naturally to an efficient solution for the TBR-S problem. The remainder of this paper deals mostly with our solution to solve the BR problem efficiently. In the next section we take a closer look at the BR problem and study some of its structural properties.

4 Structural Properties of the BR Problem

Our solution to solve the BR problem for a set of input gene trees involves computing the reconciliation cost of $\Re(P')$, where $P' \in \mathsf{RR}(P)$, for each gene tree separately, and then combining the results to obtain the final solution. The solution for the BR problem is easily obtained by picking that $P' \in \mathsf{RR}(P)$ for which the sum of the reconciliation costs from each gene tree is minimum. Therefore, in the remainder of this section we assume that there is only one input gene tree G for the BR problem. Thus, the problem to be solved is the following:

Problem 5 (Rooting)

Instance:	*A triple* (G, S, P)*, where* G *is a gene tree,* S *a compatible species tree, and* P *a proper subtree of* S*.*
Find:	*The reconciliation cost* $\Delta(G, \Re(P'))$ *for each* $P' \in \mathsf{RR}(P)$*.*

To solve the ROOTING problem we first calculate the reconciliation cost of $\Re(P)$. As P is re-rooted to form P', the duplication status of some of the nodes from G may change, which changes the reconciliation cost. We show how to efficiently compute this difference between the reconciliation cost of $\Re(P)$ and the reconciliation cost of $\Re(P')$ for each $P' \in \mathsf{RR}(P)$.

To realize this strategy it is imperative to study the change in the duplication status of nodes in the gene tree as P is re-rooted step-by-step.

Lemma 1. *The duplication status of any node $g \in G$ for which $\mathcal{M}_{G,S}(g) \notin V(P)$ remains the same for each $\Re(P')$, $P' \in \mathsf{RR}(P)$.*

Thus, under our strategy, we only need to consider those nodes in G that map to a node in $V(P)$ under $\mathcal{M}_{G,S}$. These are the nodes that are responsible for any difference in the reconciliation costs of $\Re(P)$ and $\Re(P')$, where $P' \in \mathsf{RR}(P)$.

Definition 8. *An internal node $g \in V(G)$ is* relevant *if $\mathcal{M}_{G,S}(g) \in V(P)$.*

For the remainder of this section let $g \in V(G)$ be relevant, and $\mathsf{Ch}(g) = \{g', g''\}$.

Lemma 2. *If $\mathcal{M}_{G,\Re(P')}(g) = \mathcal{M}_{G,\Re(P')}(g') = \mathcal{M}_{G,\Re(P')}(g'')$ for some $P' \in \mathsf{RR}(P)$, then g remains a duplication under $\mathcal{M}_{G,\Re(P'')}$ for every $P'' \in \mathsf{RR}(P)$.*

Lemma 3. *Let $a = \mathcal{M}_{G,\Re(P)}(g)$. The duplication status of g under $\mathcal{M}_{G,\Re(P)}$ is preserved under $\mathcal{M}_{G,\Re(P')}$ where $P' = \mathsf{RR}(P, x)$ for $x \in V(P) \setminus (V(P_a) \setminus \{a\})$.*

Lemma 4. *Suppose g is not a duplication under $\mathcal{M}_{G,\Re(P)}$. Let $b = \mathcal{M}_{G,\Re(P)}(g')$, $c = \mathcal{M}_{G,\Re(P)}(g'')$. Then g is a duplication under $\mathcal{M}_{G,\Re(P')}$ where $P' = \mathsf{RR}(P, x)$ for $x \in (V(P_b) \setminus \{b\}) \cup (V(P_c) \setminus \{c\})$. And, g is not a duplication under $\mathcal{M}_{G,\Re(P')}$ for any other P'.*

Lemma 5. *Let $a = \mathcal{M}_{G,\Re(P)}(g) = \mathcal{M}_{G,\Re(P)}(g')$, and $b = \mathcal{M}_{G,\Re(P)}(g'')$. Let α denote the node closest to b along the path from a to b in $\Re(P)$, such that there exists a node $v \in V(G_{g'})$ with $\mathcal{M}_{G,\Re(P)}(v) \in P_\alpha$. If $\alpha \neq b$, then let β be the child of α that lies along the path from α to b. Then,*

1. *If $\alpha = b$ then g is a duplication under mapping $\mathcal{M}_{G,\Re(P')}$ for each $P' \in \mathsf{RR}(P)$.*
2. *Otherwise,*
   (a) *g is not a duplication under $\mathcal{M}_{G,\Re(P')}$ where $P' = \mathsf{RR}(P, x)$ and $x \in V(P_\beta) \setminus \{V(P_b) \setminus b\}$, and,*
   (b) *g is a duplication under $\mathcal{M}_{G,\Re(P')}$ for every other P'.*

5 Description of the Algorithm

We first design an efficient algorithm, called ROOTINGCOSTTREE (Alg-RCT), which solves the ROOTING problem. Based on the lemmas seen in Section 4, we then show how this algorithm fits into our algorithm for solving the TBR-S problem. Finally we analyze the complexity of our algorithm for solving the TBR-S problem.

5.1 Algorithm Alg-RCT(G, S, P)

The input for Alg-RCT is the instance (G, S, P) of the ROOTING problem. The first step in the algorithm is to obtain the tree $\Re(P)$.

The output $\widetilde{P}$ is a $W\colon V(\widetilde{P}) \to \mathbb{N}_0$ node weighted version of tree P, where $W(s) = \Delta(G, \Re(P'))$ for $P' = \mathsf{RR}(P, s)$.

Initialization: Construct $\Re(P)$ and initialize two counters $\mathsf{g}(s)$ and $l(s)$ with 0, for each node $s \in V(P)$. Then, compute $\mathcal{M}_{G,\Re(P)}$. Create two empty sets "start" and "end" at each node in P.

Partially updating the values for g and l: For each relevant node g do the following: If g is not a duplication under $\mathcal{M}_{G,\Re(P)}$, then $\mathsf{g}(\mathcal{M}_{G,\Re(P)}(c)) \leftarrow \mathsf{g}(\mathcal{M}_{G,\Re(P)}(c)) + 1$ for each $c \in \mathsf{Ch}(g)$. If g is a duplication where $a = \mathcal{M}_{G,\Re(P)}(g) = \mathcal{M}_{G,\Re(P)}(u)$, and $b = \mathcal{M}_{G,\Re(P)}(v)$, for $\mathsf{Ch}(g) = \{u, v\}$ and $b \neq a$. Add u to the "start" set of node a and the "end" set of node b.

Fully updating the values for g and l: We now update the l and g values for those nodes that satisfy the condition of Lemma 5. Lets call these nodes "special". Following the notation from Lemma 5, the goal is to find node $\alpha \in P$ for each special node from G. In the interest of brevity we only give a high level idea of the algorithm to be followed for this step. An in-order labeling of G lets us store the subtree G_g for any special node $g \in V(G)$ as an interval. These intervals can be stored in an interval tree, so that stabbing queries can be performed efficiently. We traverse P in post-order, and for each node, say x, we keep track of those nodes from the gene tree that might have a descendant mapping to x and for which α can be deduced from x. This is done by making use of the "start" and "end" sets established in the previous step. This 'currently active' set of nodes (intervals) is maintained dynamically in the interval tree. Suitably querying the interval tree allows us to obtain those special nodes for which the α nodes can be deduced easily from x. This step can be shown to run in time $O(|V(P) + V(G)| \log(|V(P) + V(G)|))$.

Computing $\widetilde{P}$: The tree $\widetilde{P}$ is initialized to be P and its node weights are set to 0. Set $d \leftarrow \Delta(G, \Re(P))$. For each node s in a preorder traversal on the tree $\widetilde{P}$, we calculate the weight of that node as follows: If $s \in \mathsf{Ch}(\mathsf{Ro}(P))$ then $W(s) \leftarrow d$. Otherwise, set $W(s) \leftarrow W(\mathsf{Pa}(s)) + \mathsf{g}(\mathsf{Pa}(s)) - l(s)$.

Note: The value $\mathsf{g}(s)$, represents the number of additional nodes from G that will become duplications when $P' = \mathsf{RR}(P, s)$ is re-rooted to form $P'' = \mathsf{RR}(P, t)$, $t \in \mathsf{Ch}(s)$. The value $l(s)$ represents the number of nodes from G that will lose their duplication status when $P' = \mathsf{RR}(S, \mathsf{Pa}(s))$ is re-rooted to form $P'' = \mathsf{RR}(S, s)$.

5.2 Algorithm Alg-TBR$(\mathcal{G}, S, P)$

This algorithm solves the TBR-S problem. The algorithm is as follows: We first use Algorithm Alg-RCT to solve the BR problem as shown in Section 4. A solution to the BR problem leads naturally to a solution for the TBR-S problem (see Observation 1, Theorems 1 and 2, and, Corollaries 1 and 2).

5.3 Correctness and Complexity

To establish the correctness of our algorithm for the TBR-S problem, it is sufficient to show that the ROOTING problem is correctly solved by Algorithm Alg-RCT. The correctness of algorithm Alg-RCT is based on Lemmas 1-5. For brevity, a detailed proof is omitted herein.

We first state the time complexity of Alg-RCT, and then derive the time complexity of algorithm Alg-TBR which solves the TBR-S problem. Note, to simplify our analysis we assume that all $G \in \mathcal{G}$ have approximately the same size. The input for BR problem is a gene tree G, a species tree S, and the pruned subtree P of S. Let $n = |\mathsf{Le}(S)|$, and $k = |\mathcal{G}|$.

Complexity of Alg-RCT(G, S, P): Let $m = |\mathsf{Le}(S)| + |\mathsf{Le}(G)|$. The overall time complexity of Alg-RCT(G, S, P) is bounded by $O(m \log m)$ (proof omitted for brevity). This implies that the complexity of the BR problem is $O(km \log m)$.

Complexity of Alg-TBR$(\mathcal{G}, S, P)$: By Corollary 2 the time complexity of the TBR-RS problem is $O(km) + O(km \log m)$ which is $O(km \log m)$. The time complexity of Alg-TBR is thus, $O(n) \times O(km \log m)$, which is $O(knm \log m)$. The time complexity of the existing naive solution for the TBR-S problem is $O(kn^3m)$. Thus, our algorithm improves on the current solution by a factor of $n^2/\log m$.

6 Outlook and Conclusion

Despite the inherent complexity of the duplication problem, it has been an effective approach for incorporating data from gene families into a phylogenetic inference [4–7]. The duplication problem is typically approached by using local search heuristics. Among these, TBR heuristics are especially desirable for large-scale phylogenetic analyses, but current solutions have prohibitively large run times. Our algorithm offers a vast reduction in run time, which makes TBR heuristics applicable for such large-scale analyses.

The ideas developed in this paper could possibly be applied to other problems related to the reconciliation of gene and species trees. For example, our solution for the ROOTING problem can be used to efficiently find an optimal rooting for any species tree, with respect to the given gene trees.

References

1. Guigó, R., Muchnik, I., Smith, T.F.: Reconstruction of ancient molecular phylogeny. Molecular Phylogenetics and Evolution **6**(2) (1996) 189–213
2. Ma, B., Li, M., Zhang, L.: On reconstructing species trees from gene trees in term of duplications and losses. In: RECOMB. (1998) 182–191
3. Page, R.D.M.: GeneTree: comparing gene and species phylogenies using reconciled trees. Bioinformatics **14**(9) (1998) 819–820
4. Slowinski, J.B., Knight, A., Rooney, A.P.: Inferring species trees from gene trees: A phylogenetic analysis of the elapidae (serpentes) based on the amino acid sequences of venom proteins. Molecular Phylogenetics and Evolution **8** (1997) 349–362

5. Page, R.D.M.: Extracting species trees from complex gene trees: reconciled trees and vertebrate phylogeny. Molecular Phylogenetics and Evolution **14** (2000) 89–106
6. Cotton, J., Page, R.D.M.: Vertebrate phylogenomics: reconciled trees and gene duplications. In: Pacific Symposium on Biocomputing. (2002) 536–547
7. Cotton, J.A., Page, R.D.M.: Tangled tales from multiple markers: reconciling conflict between phylogenies to build molecular supertrees. In: Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life. Springer-Verlag (2004) 107–125
8. Sanderson, M.J., McMahon, M.M.: Inferring angiosperm phylogeny from EST data with widespread gene duplication. BMC Evolutionary Biology **7**(suppl 1:S3) (2007)
9. 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
10. 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
11. Mirkin, B., Muchnik, I., Smith, T.F.: A biology consistent model for comparing molecular phylogenies. Journal of Computational Biology **2**(4) (1995) 493–507
12. Eulenstein, O.: Predictions of gene-duplications and their phylogenetic development. PhD thesis, University of Bonn, Germany (1998) GMD Research Series No. 20 / 1998, ISSN: 1435-2699.
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: a program for dating gene duplications and optimizing gene family trees. Journal of Computational Biology **7** (2000) 429–447
15. Bonizzoni, P., Vedova, G.D., Dondi, R.: Reconciling gene trees to a species tree. In: CIAC2003 - Italian Conference on Algorithms and Complexity. (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. In: Latin American Theoretical INformatics. (2000) 88–94
18. Harel, D., Tarjan, R.E.: Fast algorithms for finding nearest common ancestors. SIAM Journal on Computing **13**(2) (1984) 338–355
19. Fellows, M., Hallett, M., Korostensky, C., Stege., U.: Analogs & duals of the mast problem for sequences & trees. In: European Symposium on Algorithms (ESA). (1998) 103–114
20. Stege, U.: Gene trees and species trees: The gene-duplication problem is fixed-parameter tractable. In: Proceedings of the 6th International Workshop on Algorithms and Data Structures. (1999)
21. Hallett, M.T., Lagergren, J.: New algorithms for the duplication-loss model. In: RECOMB. (2000) 138–146
22. Swofford, D.L., Olsen, G.J.: Phylogeny reconstruction. In: Molecular Systematics. Sinauer Associates (1996) 411 – 501
23. Allen, B.L., Steel, M.: Subtree transfer operations and their induced metrics on evolutionary trees. Annals of Combinatorics **5** (2001) 1–13
24. Bordewich, M., Semple, C.: On the computational complexity of the rooted subtree prune and regraft distance. Annals of Combinatorics **8** (2004) 409–423

25. Chen, D., Eulenstein, O., Fernández-Baca, D., Burleigh, J.G.: Improved heuristics for minimum-flip supertree construction. Evolutionary Bioinformatics (2006)
26. Bansal, M.S., Burleigh, J.G., Eulenstein, O., Wehe, A.: Heuristics for the gene-duplication problem: A $\theta(n)$ speed-up for the local search. In: RECOMB. (2007) 238–252