Heuristics for the Gene-duplication Problem: A $\Theta(n)$ Speed-up for the Local Search *

M. S. Bansal[1], J. G. Burleigh[2], O. Eulenstein[1], and A. Wehe[3]

[1] Department of Computer Science, Iowa State University, Ames, IA, USA
{bansal,oeulenst}@cs.iastate.edu
[2] National Evolutionary Synthesis Center, Durham, NC, USA jgb12@duke.edu
[3] Department of Electrical and Computer Engineering, Iowa State University, Ames, IA, USA awehe@iastate.edu

Abstract. The gene-duplication problem is to infer a species supertree from a collection of 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 show how this local search problem can be solved efficiently by reusing previously computed information. This improves the running time of the current solution by a factor of n, where n is the number of species in the resulting supertree solution, and makes the gene-duplication problem more tractable for large-scale phylogenetic analyses. We verify the exceptional performance of our solution in a comparison study using sets of large randomly generated gene trees. Furthermore, we demonstrate the utility of our solution by incorporating large genomic data sets from GenBank into a supertree analysis of plants.

1 Introduction

The rapidly increasing amount of available genomic sequence data provides an abundance of potential information for phylogenetic analyses. Most phylogenetic analyses combine genes from presumably orthologous loci, or loci whose homology is the result of speciation. These analyses largely neglect the vast amounts of sequence data from gene families, in which complex evolutionary processes such as gene duplication and loss, recombination, and horizontal transfer generate gene trees that differ from species trees. One approach to utilize the data from gene families in phylogenetics is to reconcile their gene trees with species trees based on an optimality criterion, such as the gene-duplication model introduced by Goodman et al. [1]. This problem is a type of *supertree problem*, that is, assembling from a set of input gene trees a species 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

* During this research, O.E. and M.S.B. were supported in part by NSF grant no. 0334832 and J.G.B. by NESCent NSF grant no. EF-0423641.

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 trees for snakes [4], vertebrates [5, 6], *Drosophia* [7], and plants [8]. Yet, the run time performance of existing heuristics has limited the size of such studies. We improve on the best existing solution for the local search problem asymptotically by a factor of n, where n is the number of species from which sequences in the gene trees were sampled (that is the number of nodes in a resulting supertree). To show the applicability of our improved solution for the local search problem, we implemented it as part of standard heuristics for the gene-duplication problem. We demonstrate that the implementation of our method greatly improves the speed of standard heuristics for the gene-duplication problem and makes it possible to infer large supertrees that were previously difficult, if not impossible, to compute.

For convenience, the term "tree" refers to a rooted and full binary tree, and the terms "leaf-gene" and "leaf-species" refer to 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. 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.

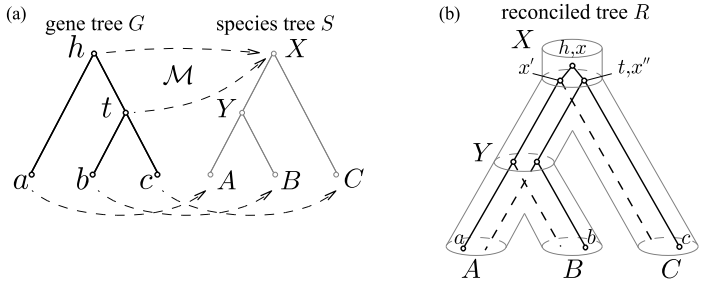

Fig. 1. (a) Gene tree G and species tree S are comparable, as the mapping from the leaf-genes to the leaf-species indicates. $\mathcal{M}$ is the lca-mapping from G to S. (b) R is the reconciled tree for G and S. In species X of R gene x duplicates into the genes x' and x''. The solid lines in R represent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

Gene-duplication model: The gene-duplication (GD) model [9–16] 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 *s-mapping*, exists that maps every leaf-gene to the leaf-species from which it was sampled. Fig. 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 s-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 Fig. 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 the gene duplication 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 *lca-mapping* $\mathcal{M}$, which is an extension of the given s-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. To make the definition precise, $\mathcal{M}$ maps each gene to the least common ancestor of the species from which the leaf-genes of the subtree rooted at the gene were sampled (given by the s-mapping). A gene in the gene tree is said to be a *gene duplication* if it has a child with the same lca-mapping. In Fig. 1 gene h and its child t map under the lca-mapping to the same species X. The *reconciliation cost* for a gene tree and a comparable species tree is measured in the number of gene 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 cost for every gene tree in the set and the species tree. The lca-mapping is linear time computable on a PRAM [13] through a reduction from the least common ancestor problem [17, 18]. Hence, the reconciliation cost for a set of gene trees and a species tree is computable in linear time.

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]. Therefore, in practice, heuristics are commonly used for the gene-duplication problem even if they are unable to guarantee an optimal solution. However, GeneTree [22], an implementation of a standard local search heuristic for the gene-duplication problem, demonstrated that the gene-duplication problem can be an effective approach for inferring species phylogenies. 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*, representing the tree space, is defined for the given set of gene trees and some fixed tree edit operation. The nodes in the tree graph are the species trees which are comparable with every given gene tree. An edge is drawn between 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 of a given node. The time complexity of the local search problem depends on the tree edit operation used. An edit operation of interest is the rooted subtree pruning and regrafting (rSPR) operation [23, 24]. Given a tree S, an rSPR operation can be performed in three steps: (i) prune some subtree P from S, (ii) add a root edge to the remaining tree S, (iii) regraft P into an edge of the remaining tree S. The resulting tree graph is connected and every node has a degree of $\Theta(n^2)$, where n is the size of a species tree comparable to the given gene trees. Assuming, for convenience, similar gene tree and species tree sizes, the local search problem for the rSPR edit operation can be solved naively in $\Theta(n^3)$ time per gene tree. If there are k gene trees then this gives a total time bound of $O(kn^3)$. This is the best-known algorithm to solve the local search problem for rSPR operations. In practice, the cubic run time typically allows only the computation of smaller supertrees [3]. A common approach to overcome this limitation is to consider only an $O(n)$ cardinality subset of the rSPR neighborhood at each node by using the rooted nearest neighbor interchange (rNNI) edit operation. The local search problem for the rNNI edit operation can be solved in $O(kn^2)$ time. We solve the local search problem for the rSPR edit operations within the same $O(kn^2)$ time bound.

Contribution of the Manuscript: First we introduce an algorithm that, irrespective of the sizes of the gene trees, improves the run time of the current solution by $\Theta(n)$, where n is the size of any species tree resulting from the given gene trees. To support typical input gene trees, our algorithm also allows multiple leaf-genes from the same gene tree to map to a single leaf-species. This algorithm was implemented as part of a standard heuristic for the gene-duplication problem, and we compared the run times of our implementation and the program GeneTree, which can infer species trees using the same local search heuristic. Finally, we demonstrate the ability of our heuristic to utilize gene-family sequences to construct large subtrees of the Tree of Life.

Organization of the Manuscript: Section 2 introduces basic terminology and problem definitions. In Sect. 3 we formally introduce the local search problem for the rSPR tree edit operation and our approach for solving it. To solve this refined local search problem we study gene duplication properties when a tree is modified using rSPR operations in Sect. 4. In Sect. 5 we introduce our algorithm for the (refined) local search problem, show its correctness and analyze its run time. Experimental results are presented in Sect. 6 and concluding remarks appear in Sect. 7. In the interest of brevity we shall omit the formal proofs for the lemmas and theorems presented herein.

2 Basic Notation and Preliminaries

Recall that throughout this work the term *tree* refers to a rooted full binary tree, unless otherwise stated. Given a tree T, let $V(T)$ and $E(T)$ denote the node and

edge sets of T respectively. $\mathsf{Root}(T)$ denotes the root node of T, and $\mathsf{Le}(T)$ the leaf set of T. Given a node $v \in V(T)$: (i) $\mathsf{Pa}_T(v)$ is the parent node of v, (ii) $\mathsf{Ch}_T(v)$ denotes the set of children of v, (iii) T_v denotes the complete subtree of T rooted at node v, and (iv) a node $u \in V(T_v)\backslash\{v\}$ is a *(proper) descendant* of v. Two nodes with the same parent are called *siblings* of each other. The *least common ancestor* of a set $L \subseteq \mathsf{Le}(T)$ in T is defined to be the node $v \in V(T)$ such that $L \subseteq \mathsf{Le}(T_v)$ and $L \not\subseteq \mathsf{Le}(T_u)$ for any descendant u of $v \in V(T)$.

A *species tree* and a *gene tree* are full binary trees that represent the evolutionary relationships between species and genes (of a gene family) respectively. In the following we define the gene-duplication problem and the terms necessary for its definition. Let G be a gene tree and S be a species tree.

Comparability: The trees G and S are *comparable* if $\mathsf{Le}(G) \subseteq \mathsf{Le}(S)$. A set of gene trees $\mathcal{G}$ and S are *comparable* if $\mathsf{Le}(S) = \bigcup_{G\in\mathcal{G}} \mathsf{Le}(G)$.

Gene duplication: The *(lca-)mapping* $\mathcal{M}_{G,S}\colon V(G) \to V(S)$ is defined for comparable trees G and S such that $\mathcal{M}_{G,S}(v)$ is the least common ancestor of $\mathsf{Le}(G_v)$ in S. A node $v \in V(G)$ is a *gene duplication* if there exists a child u of $v \in V(G)$ such that $\mathcal{M}_{G,S}(v) = \mathcal{M}_{G,S}(u)$.

Reconciliation cost: (i) The *reconciliation cost for* G *and* S is $\Delta(G,S) = |\{v\colon v \in V(G)$ and v is a gene duplication $\}|$. (ii) The *reconciliation cost for a set of gene trees* $\mathcal{G}$ *and* S is $\Delta(\mathcal{G},S) = \sum_{G\in\mathcal{G}} \Delta(G,S)$. (iii) The *reconciliation cost for a set of gene trees* $\mathcal{G}$ is $\Delta(\mathcal{G}) = \min_{S\in\mathcal{S}} \Delta(\mathcal{G},S)$, where $\mathcal{S}$ is the set of all species trees that are comparable with $\mathcal{G}$.

The GENE-DUPLICATION *problem*
Instance: A set $\mathcal{G}$ of gene trees.
Find: A species tree S_{OPT} such that $\Delta(\mathcal{G}) = \Delta(\mathcal{G}, S_{OPT})$.

3 Refining the Local Search Problem

The gene-duplication problem is heuristically approached by repeatedly solving the local search problem for the rSPR edit operation. In this section we first give definitions for the rSPR operation and the local search problem that were motivated in the introduction. Then we observe that the local search problem can be solved by dividing it into problem instances of the restricted local search problem, which we will introduce here. Finally, we present our central idea for solving the restricted local search problem efficiently.

The rSPR operation: The rSPR operation for a tree S is defined as cutting any edge, say $\{u,v\}$, where $u = \mathsf{Pa}_S(v)$, and thereby pruning a subtree, S_v, and then regrafting the subtree by the same cut edge in one of the following ways:

1. *Regrafting* S_v *into an edge* $e \in S\backslash S_v$: Creating a new node u' which subdivides e and regrafting the subtree by the cut edge at node u'. Then, either suppressing the degree-two node u or, if u is the root of S, deleting u and the edge incident with u, making the other end-node of this edge the new root.

2. *Regrafting S_v above* $\mathtt{Root}(S)$: Creating a new root node u' and a new edge between u' and the original root. Then regrafting the subtree by the cut edge at node u' and suppressing the degree-two node u.

Note that the rSPR operation involves deleting a node in the original tree and creating a new one where the subtree is regrafted. Throughout this text we assume that the new node created is given the same label as the node removed. This forms a new tree whose leaf set is the same as the original tree.

Consider an rSPR operation on the tree S that prunes the subtree S_v. We define $\mathtt{rSPR}(S, v, u)$ to be the tree obtained by regrafing S_v in one of the following two ways: (i) if $u \neq \mathtt{Root}(S)$, then V_s is regrafted into the edge $(u, \mathtt{Pa}_S(u))$, and (ii) if $u \neq \mathtt{Root}(S)$, then S_v is regrafted above $\mathtt{Root}(S)$. The set of trees into which S can be transformed by regrafting only S_v is $\mathtt{rSPR}(S, v) = \bigcup_{w \in V(S) \setminus V(S_v)} \mathtt{rSPR}(S, v, w)$. The set of trees into which S can be transformed by one rSPR operation is $\mathtt{rSPR}(S) = \bigcup_{v \in V(S) \setminus \{\mathtt{Root}(S)\}} \mathtt{rSPR}(S, v)$.

The specific local search problem defined for rSPR operations is called the neighborhood search problem.

The NEIGHBORHOOD-SEARCH (NS) *problem*
Instance: A gene tree set $\mathcal{G}$, and a comparable species tree S.
Find: The reconciliation cost for every tree in $\mathtt{rSPR}(S)$.

Restricting the NS Problem: We will show that the NS problem can be solved without computing the reconciliation cost for every tree in the neighborhood of S separately. Therefore we divide the NS problem into subproblems, called restricted neighborhood search problems, that can be solved efficiently by reusing previously computed information.

The RESTRICTED-NEIGHBORHOOD-SEARCH (RNS) *problem*
Instance: A triple $(\mathcal{G}, S, P)$, where $\mathcal{G}$ is a set of gene trees, S a comparable species tree, and P is a subtree of S.
Find: The reconciliation cost for every tree in $\mathtt{rSPR}(S, \mathtt{Root}(P))$.

Observation 1 *The* NS *problem on S can be solved by solving the* RNS *problem for each subtree of S.*

Our Idea to solve the RNS Problem: To solve the RNS problem instance $(\mathcal{G}, S, P)$ we first determine the reconciliation cost $\Delta(\mathcal{G}, \Re)$ for a particular tree $\Re \in \mathtt{rSPR}(S, \mathtt{Root}(P))$. $\Re$ is the tree obtained after pruning and regrafting P to the root of S (see Fig. 2(a)). After this initial step the reconciliation cost $\Delta(\mathcal{G}, S')$ for each tree within $S' \in \mathtt{rSPR}(S, \mathtt{Root}(P))$ can be determined in amortized $O(|\mathcal{G}|)$ time by following a particular order. Beginning with $\Re$ the subtree P is stepwise "moved down" in the tree S using rSPR operations (see Fig. 2(a)). We define the *move-down* operation for the pruned subtree P of the tree S as the rSPR operation which produces a tree $\mathtt{rSPR}(S, \mathtt{Root}(P), w)$, where $w \in \mathtt{Ch}_S(v)$ for the sibling v of $\mathtt{Root}(P)$.

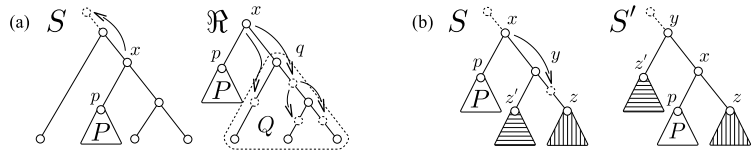

Fig. 2. (a): The tree $\Re$ is obtained from S after pruning and regrafting P to the root. Each tree in $\mathrm{rSPR}(S, \mathtt{Root}(P))$ can be obtained by starting from $\Re$ and successively performing move-down operations. (b): The subtree on the right, S', is obtained from S by moving x and P to the right subtree of y.

The set $movedown_S(P)$ consists of all species trees that can be obtained by performing successive move-down operations starting from $\Re$ with a fixed pruned subtree P.

Observation 2 $\mathrm{rSPR}(S, \mathtt{Root}(P)) = movedown_S(P) \bigcup \{\Re\}$.

In Sect. 4 we show how the reconciliation cost is affected by move-down operations. These properties allow the design of an efficient algorithm for the RNS problem as shown in Sect. 5.

Naming Convention for this work: We establish the following notation throughout this work. The pruned subtree under study is denoted by P, its root node by p, and the parent of p by x. In the tree $\Re$, the sibling of p is q, and the subtree rooted at q is Q (see Fig. 2(a)). Note, $\mathtt{Root}(\Re)$ is always x.

The sibling of p is always denoted by y. Note, q and y refer to the same node in the tree $\Re$. A general situation is dipicted in Fig. 2(b). In general, g is used to refer to a node in a gene tree, and s to refer to a node in the species tree.

4 Structural Properties

Let G be a gene tree and S a species tree. In this section we study first the effect of move-down operations on the mapping $\mathcal{M}_{G,S}$ and the gene duplication status of genes in G. Finally, we describe the effect on the mapping after a sequence of move-down operations for a fixed pruned subtree.

A Single Move-down Operation: Consider a move-down operation that changes tree S into tree $S' = \mathrm{rSPR}(S, p, z)$, where $z \in \mathrm{Ch}_S(y)$. Fig. 2(b) shows an example for a move-down operation. Further, let $\mathcal{M}^{-1}_{G,S}(v)$ denote the set of nodes in G that map to node $v \in V(S)$ under the mapping $\mathcal{M}_{G,S}$. In the following we study the effects of the move-down operation on the mapping $\mathcal{M}_{G,S}$ and the gene duplication status.

Relating $\mathcal{M}_{G,S}$ and $\mathcal{M}_{G,S'}$:

Lemma 1. $\mathcal{M}^{-1}_{G,S}(v) = \mathcal{M}^{-1}_{G,S'}(v)$, *for all* $v \in V(S) \backslash \{x, y\}$.

Lemma 2. *$\mathcal{M}^{-1}_{G,S'}(x) \subseteq \mathcal{M}^{-1}_{G,S}(x)$ and $\mathcal{M}^{-1}_{G,S}(y) \subseteq \mathcal{M}^{-1}_{G,S'}(y)$.*

Effects on the gene duplication status: Based on the observations from Lemmas 1 and 2, the following three lemmas characterize the possible change in the gene duplication status of nodes in G.

Lemma 3. *The gene duplication status for any node in G that does not map to x under mapping $\mathcal{M}_{G,S}$ remains unchanged.*

Lemma 4. *If a node $g \in \mathcal{M}^{-1}_{G,S}(x)$ is not a gene duplication under mapping $\mathcal{M}_{G,S}$, then it becomes a gene duplication under mapping $\mathcal{M}_{G,S'}$ if and only if one of the children of g maps to node y in S.*

Lemma 5. *Let $g \in \mathcal{M}^{-1}_{G,S}(x)$ be a gene duplication under mapping $\mathcal{M}_{G,S}$ and z' be the sibling of z in S. Under the mapping $\mathcal{M}_{G,S'}$ the node g will lose its gene duplication status if and only if both of the following hold:*

1. *Under $\mathcal{M}_{G,S}$ one of the two children $b \in \mathsf{Ch}_G(g)$ maps to x and the other child maps to a node in the subtree $S_{z'}$.*
2. *Under $\mathcal{M}_{G,S'}$ node b maps to x.*

A Sequence of Move-down operations: We describe changes in the mapping $\mathcal{M}_{G,\Re}$ when move-down operations for a subtree P rooted at a child of $\mathsf{Root}(\Re)$ are successively performed to obtain a tree $S' \in movedown_S(P)$.

The following proposition follows from Lemmas 1 and 2.

Proposition 1. *$\mathcal{M}_{G,\Re}(g)$ may only differ from $\mathcal{M}_{G,S'}(g)$, if $g \in \mathcal{M}^{-1}_{G,\Re}(x)$.*

Based on Proposition 1 we are left to characterize the differences between $\mathcal{M}_{G,\Re}(g)$ and $\mathcal{M}_{G,S'}(g)$ for all $g \in \mathcal{M}^{-1}_{G,\Re}(x)$. For this, we determine the nodes in G that can change in their mapping or that can be responsible for such a change.

The mapping of a node $g \in \mathcal{M}^{-1}_{G,\Re}(x)$ can change caused by a change in the mapping of its children. Such children would then be elements in $\mathcal{M}^{-1}_{G,\Re}(x)$ by Proposition 1. However, a change in the mapping can also be caused by other children, called supporting nodes, whose mapping does not change.

Definition 1 (Supporting nodes). *A node $g \in V(G)$ is a* supporting node*, if (i) $\mathcal{M}_{G,\Re}(g) \in V(Q)$ and (ii) $\mathsf{Pa}_G(g) \in M^{-1}_{G,\Re}(x)$.*

Definition 2 (Partial Gene Tree Γ). *The* partial gene tree *Γ is the subgraph of G induced by the set $\{g \in V(G) : g \in \mathcal{M}^{-1}_{G,\Re}(x)$ or g is a supporting node$\}$.*

Note, Γ is a binary tree (not necessarily full binary), and all its leaf nodes are supporting nodes.

The nodes in Γ induce a subtree in G and identify exactly the nodes whose mapping can change or that are responsible for such a change. The supporting nodes in Γ map to nodes in Q under mapping $\mathcal{M}_{G,\Re}$. Let this define an initial mapping from the leaves of Γ to the nodes in Q. This initial mapping can be extended to the (lca-)mapping $\mathcal{M}_{\Gamma,Q}$. All the internal nodes in Γ map to the

root node of $\Re$, because they have at least one descendant in G that maps to a node in P under mapping $\mathcal{M}_{G,\Re}$. The mapping $\mathcal{M}_{\Gamma,Q}$ shows where those nodes would map under mapping $\mathcal{M}_{G,\Re}$ if all nodes in $\mathcal{M}^{-1}_{G,\Re}(s)$, for all $s \in V(P)$, are removed from G.

Based on the mapping $\mathcal{M}_{\Gamma,Q}$, we have the following lemma.

Lemma 6. *Let $s = \mathcal{M}_{\Gamma,Q}(\gamma)$ where γ is an internal node of Γ, and let $S' \in movedown_{\Re}(P)$. The location of node $\mathcal{M}_{G,S'}(\gamma)$ depends on the edge e in $E(Q)$ into which P is regrafted, as follows:*

1. *$\mathcal{M}_{G,S'}(\gamma) = x$, if e is on the path from q to node s in $V(Q)$.*
2. *$\mathcal{M}_{G,S'}(\gamma) = s$, if e is an edge in $E(Q_s)$.*
3. *$\mathcal{M}_{G,S'}(\gamma)$ is a node on the path from q to s, but not q or s, otherwise.*

5 Solving the RNS Problem

Based on the results obtained in the previous section we will first design an efficient algorithm, called ReconciliationCostTree (RCT), which solves the RNS problem for one input gene tree. Algorithm FastRNS then makes use of RCT to solve the RNS problem. We then show the correctness and analyze the run time of FastRNS.

Algorithm RCT(G, S, P): The input for RCT is a gene tree G, a comparable species tree S, and subtree P to be pruned. The first step in the algorithm is to obtain the tree $\Re$ (see Fig. 2(a)). Recall that $x = \mathsf{Root}(\Re)$, $p = \mathsf{Root}(P)$, q denotes the sibling of p, and $Q = \Re_q$.

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

Initialization: Create $\Re$ and initialize two counters $\mathsf{g}(s)$ and $l(s)$ with 0, for each node $s \in V(Q)$. Then, compute the mapping $\mathcal{M}_{G,\Re}$, the tree Γ, and the mapping $\mathcal{M}_{\Gamma,Q}$.

Computing the values for g and l: For each leaf $\gamma \in \mathsf{Le}(\Gamma)$ that has no sibling we do the following: If $\mathcal{M}_{\Gamma,Q}(\gamma) = \mathcal{M}_{\Gamma,Q}(\mathsf{Pa}_\Gamma(\gamma))$, then we increment $\mathsf{g}(\mathcal{M}_{\Gamma,Q}(\gamma))$ by 1. Similarly, for each leaf $\gamma \in \mathsf{Le}(\Gamma)$ that has a sibling, we do the following: Let $\alpha = \mathsf{Pa}_\Gamma(\gamma)$, σ be the sibling of γ, $a = \mathcal{M}_{\Gamma,Q}(\alpha)$, and $\mathsf{Ch}_Q(a) = \{b, c\}$. If σ maps into Q_b and γ into Q_c under mapping $\mathcal{M}_{\Gamma,Q}$, then increment the counter $l(b)$ by 1. Further, if σ maps into Q_c and γ into Q_b, then increment the counter $l(c)$ by 1.

The value $\mathsf{g}(s)$, represents the number of additional nodes from G that will become gene duplications when P is regrafted onto the edge $\{s, t\}$, $t \in \mathsf{Ch}_Q(s)$, from the edge $\{\mathsf{Pa}_Q(s), s\}$. The value $l(s)$ represents the number of nodes from G that will lose their gene duplication status when P is regrafted onto the edge $\{\mathsf{Pa}_Q(s), s\}$ from the edge $\{\mathsf{Pa}_Q(\mathsf{Pa}_Q(s)), \mathsf{Pa}_Q(s)\}$.

Computing $\widetilde{Q}$: The tree $\widetilde{Q}$ is initialized to be Q and its node weights are set to 0. Set $d = \Delta(G, \Re)$. For each node s in a preorder traversal on the tree $\widetilde{Q}$,

we calculate the weight of that node as follows: If $s = \mathsf{Root}(Q)$ then $W(s) = d$. Otherwise, set $d = d + \mathsf{g}(\mathsf{Pa}_Q(s)) - l(s)$ and $W(s) = d$. Note, that the weight at the root node of $\widetilde{Q}$ represents the value $\Delta(G, \Re)$.

Algorithm FastRNS$(\mathcal{G}, S, P)$: Typically, the RNS problem needs to be solved for several input gene trees. In this case we execute RCT for each gene tree separately. We call this algorithm FastRNS. Note that the tree $\widetilde{Q}$ obtained for each gene tree is identical except for the weights on the nodes. The output of FastRNS is a tree Φ with topology identical to tree Q and the weight of each node equal to the sum of the weights at the corresponding node in each $\widetilde{Q}$ tree.

By Observation 1 the NS problem can be solved through solutions to the RNS problem. Each edge in the given species tree S, defines a subtree that can be pruned. To solve the NS problem we keep calling the algorithm described above for each of these subtrees that can be pruned in S. This produces a node weighted tree Φ for each pruned subtree, which solves the NS problem.

Correctness: Now we show the correctness of our algorithm for the NS problem. To do this it is sufficient to show that the RNS problem is correctly solved by FastRNS.

Lemma 7. RCT(G, S, P) *computes* $W(s) = \Delta(G, S')$ *for all* $S' \in \mathsf{rSPR}(S, p)$.

Proof (Sketch). A node in Γ is called *feasible* if it is the parent of a supporting node. The following statements hold.

- Values $\mathsf{g}(s)$ and $l(s)$ are only computed for $s = \mathcal{M}_{\Gamma,Q}(\gamma)$, if $\gamma \in V(\Gamma)$ is feasible . This is because all other internal nodes in Γ will maintain their gene duplication status as P is regrafted into edges in Q (see Lemma 1).
- Consider a feasible node $\gamma \in V(\Gamma)$ whose gene duplication status changes when subtree P is regrafted into an edge $\{s, \mathsf{Pa}_Q(s)\}$ in Q. If P is then regrafted into any edge in the subtree Q_s, the gene duplication status of γ is preserved.
- If a feasible node $\gamma \in V(\Gamma)$ has only one child and $\mathcal{M}_{\Gamma,Q}(\gamma) = s$, then it will gain gene duplication status if and only if P is regrafted into the subtree Q_s.
- If a feasible node $\gamma \in V(\Gamma)$ has two children, α and β, then one of them, say α, must be a non-supporting node. Suppose α maps to the subtree rooted at a child t of $\mathcal{M}_{\Gamma,Q}(s)$ in Q, and β maps to the subtree rooted at the sibling of t in Q. Then, if P is regrafted into Q_t, node γ loses its gene duplication status.

From the above statements and Lemmas 4, 5 and 6 it follows that the values for g and l, and hence the node weights of $\widetilde{Q}$, are computed correctly for each node.

Lemma 8 follows from Lemma 7 and the definition of reconciliation cost.

Lemma 8. *The weight of a node* s *in tree* Φ *is* $\Delta(\mathcal{G}, S')$ *where* $S' = rSPR(S, p, s)$.

Observation 3 *Each tree in $rSPR(S,p)$ can be obtained by starting at $\Re$ and regrafting P into an edge in the subtree Q.*

The node weights on the tree Φ thus provide all the information needed to solve the RNS problem.

Theorem 1. *The* RNS *problem is correctly solved by* FastRNS.

Time Complexity: The major component of our algorithm to solve the NS problem is FastRNS that solves the RNS problem. Therefore we first analyze the complexity of FastRNS. Note, to simplify our analysis we assume that all $G \in \mathcal{G}$ have approximately the same size. Even if this does not hold true, our algorithm shows the same improvement in complexity over the current solution.

The input for FastRNS is a set $\mathcal{G}$ of gene trees, a species tree S, and the pruned subtree P of S. Let $n = |\,\mathtt{Le}(S)|$, and $k = |\mathcal{G}|$. FastRNS calls RCT k times and then constructs the tree Φ.

Complexity of RCT(G,S,P): Let $m = |\,\mathtt{Le}(S)| + |\,\mathtt{Le}(G)|$. The overall time complexity of RCT(G,S,P) is bounded by $O(m)$. A step-by-step analysis of the complexity follows:

1. *Initialization in* $O(|V(S)| + |V(G)|)$: The initial tree $\Re$ and the counters $\mathtt{g}$ and l for each node in Q can be setup in $O(|V(Q)|)$ time. Computing the mapping $\mathcal{M}_{G,\Re}$ takes $O(|V(S)| + |V(G)|)$ time, and the tree Γ can then be constructed in $O(|V(\Gamma)|)$ time. The mapping $\mathcal{M}_{\Gamma,Q}$ can be computed in $O(|V(\Gamma)|+|V(S)|)$ time. Hence, the time for the initialization costs is bound by $O(|V(S)| + |V(G)|)$, which is $O(m)$.
2. *Computing* $\mathtt{g}$ *and* l *in* $O(|V(G)|) + O(|V(Q)|)$: The values for $\mathtt{g}$ and l can be computed by traversing through the tree Γ once. To update the values for l we have to check whether $\mathcal{M}_{\Gamma,Q}(\gamma) \in V(Q_a)$ and $\mathcal{M}_{\Gamma,Q}(\sigma) \in V(Q_b)$ or $\mathcal{M}_{\Gamma,Q}(\sigma) \in V(Q_a)$ and $\mathcal{M}_{\Gamma,Q}(\gamma) \in V(Q_b)$. This check can be done in $O(1)$ time as follows: Initially, we perform an inorder traversal of the tree Q and label the nodes with increasing integer values in the order in which they are traversed. This preprocessing step takes $O(|V(Q)|)$ time. Based on the resulting order we can check whether a given node is in $V(Q_a)$ or $V(Q_b)$ in $O(1)$. $O(|V(\Gamma)|)$ updates for $\mathtt{g}$ and l are necessary, and each update can be performed in $O(1)$ time. Hence, computing $\mathtt{g}$ and l can be done in time $O(|V(G)|) + O(|V(Q)|)$, which is $O(m)$.
3. *Computing* $\widetilde{Q}$ *in* $O(|V(G)|) + O(|V(Q)|)$ Computing W for each node in $\widetilde{Q}$ from the $\mathtt{g}$ and l values involves first computing the value $\Delta(G,\Re)$, then traversing the tree $\widetilde{Q}$ in preorder and spending $O(1)$ time at each node. The time complexity of this step is $O(|V(G)|) + O(|V(Q)|)$, which is $O(m)$.

Complexity of FastRNS$(\mathcal{G},S,P)$: Computing the final tree $\widetilde{Q}$ involves traversing each of the $\widetilde{Q}$ trees produced in preorder. This step takes $O(n)$ time per tree and hence $O(kn)$ time overall. Thus, the time complexity of FastRNS is

bounded by $O(km + kn)$, which is $O(km)$.

Complexity of the NS problem: The time complexity of our algorithm for the NS problem is thus $O(n) \times O(km) \equiv O(kmn)$ (based on Observation 1). The brute force algorithm to solve the NS problem requires $O(kmn^2)$ time. Our algorithm for the NS problem improves on this by a factor of n. Also observe that this speed up does not come at the expense of higher space complexity.

6 Experimental Results

In order to study the performance of our algorithm we implemented it as part of a standard local search heuristic for the GENE-DUPLICATION problem. This program is called FastGeneDup. We first analyzed the performance and scalability of FastGeneDup using simulated input data and then focused on an analysis of large empirical data sets.

Table 1. GeneTree vs. FastGeneDup

Taxa size	GeneTree	FastGeneDup	Taxa size	GeneTree	FastGeneDup
50	9m:23s	1s	400	–	9m:19s
100	3h:25m	6s	1000	–	3h:20m
200	108h:33m	58s	2000	–	38h:25m

Performance and Scalability: We first compared the run time performance of FastGeneDup against the program GeneTree [3]. GeneTree currently is the only publicly available program that can build species supertrees based on the same local search heuristic. We measured the run time of each program to compute its final species supertree for the same set of input gene trees and the same randomly generated starting species tree. The input gene trees for each run consisted of a set of 20 randomly generated gene trees, all with the same set of taxa. We conducted 6 such runs, each with a different number of taxa (50, 100, 200, 400, 1000, and 2000) in the input trees. All analyses were performed on a 3 Ghz Intel Pentium 4 CPU based PC with Windows XP operating system. FastGeneDup shows a vast improvement in run time and scalability compared to GeneTree(Table 1). We could not run GeneTree on input trees with more than 200 taxa. Also, the memory consumption of FastGeneDup was less than the memory consumption of GeneTree. Note that even though both FastGeneDup and GeneTree implement the same local search heuristic, they may produce different supertrees, which may also have different reconciliation costs. This happens because during a local search step, more than one neighboring node may have the smallest reconciliation cost. In this case the node to follow is chosen arbitrarily, and this may cause the programs to follow different

paths in the search space. In practice we noticed little or no difference in the final reconciliation costs, though FastGeneDup inferred supertrees with smaller reconciliation cost more often than GeneTree.

Empirical Example: The abundance of protein sequence data from many taxa makes it possible to perform large-scale analyses of the gene-duplication problem that require fast heuristics. We demonstrated the feasibility of such phylogenomic analyses using FastGeneDup on plant gene trees. The gene trees were derived from the set of all plant (Viridiplantae) sequences in GenBank (`http://www.ncbi.nlm.nih.gov`) downloaded on April 13, 2006. In total, this included 390,230 amino acid sequences. The amino acid sequences were clustered into sets of homologs, representing gene families, using the NCBI BLASTCLUST program [25], which performs single linkage clustering of the sequences based on pairwise BLAST scores. We used a 60% identity cutoff value for the single-linkage clustering and the BLASTCLUST default alignment length. We then identified a set of clusters containing at least 4 sequences from at least 3 taxa and containing only sequences from taxa that are found in 10 or more such clusters. We found 3,978 clusters containing sequences from 624 taxa (or technically 624 GenBank taxon ids, most of which represent distinct taxa) that met this criterion. From this set of clusters, we made three data sets that were used to produce the input trees for gene duplication analysis. The first set, the small data set, consisted of the 94 clusters (or gene families) that each had sequences from at least 40 different taxa. This set contained a total of 18,402 protein sequences. The second set, the medium data set, consisted of the 599 clusters that each had sequences from at least 10 different taxa and contained a total of 48,156 sequences. Finally, the large data set consisted of all 3,978 clusters and contained a total of 100,914 sequences, over 25% of the available plant protein sequences. To our knowledge, the large data set contains by far the most sequences ever incorporated into a single phylogenetic analysis of plants.

The sequences from each of the chosen clusters were aligned using the default options in ClustalW [26]. To obtain the gene trees from our data set, we built neighbor-joining trees [27] using PAUP* [28]. Since the gene-duplication problem requires binary, rooted gene trees, zero length branches were randomly resolved, and the trees were rooted with midpoint rooting. We tested the performance of FastGeneDup using the local search heuristic starting from a random species tree. The analyses of the small and medium data sets were performed on a Macintosh power PC laptop computer with a 1.5 GHz G4 processor and Mac OS X 10.4 operating system.

The small data set took 3 h. 15 m. 12 s. and found a species tree with a score of 13,393 gene duplications. The medium data set took 24 h. 55 m. 41 s. and found a species tree with a score of 36,080 gene duplications. The analysis of the large data set was performed on a 3 GHz Intel Pentium 4 based PC with Windows XP. It took 62 h. 35 m. 29 s. and found a species tree with 75,621 gene duplications. This purpose of this experiment was to demonstrate that large genomic data sets could be incorporated into phylogenetic analyses using

FastGeneDup. Like other attempts to build large plant trees from genome-scale data sets [29], the resulting species trees contain some anomalous relationships as well as some expected relationships. The presence of anomalous relationships is not surprising since the supertree analyses consisted only of a single run of the simple heuristic starting from a random tree. Also, the input trees were built using a simple neighbor-joining, and their quality can be improved with more thorough phylogenetic methods. Finally, mid-point rooting assumes that the sequences are evolving according to a molecular clock, which is a questionable assumption for many gene families.

7 Outlook and Conclusion

Despite the inherent complexity of the gene-duplication problem, it has been an effective approach for incorporating data from gene families into a phylogenetic inference [4–7]. Yet, existing local search heuristics for the problem are slow and thus cannot utilize the vast quantities of newly available genomic sequence data. We introduced an algorithm that speeds up the stepwise search procedure of local search heuristics for the gene-duplication problem. Our algorithm eliminates redundant calculations in computing the reconciliation cost for all trees resulting from pruning a given subtree and regrafting it to all possible positions. We implemented our algorithm as part of standard local search heuristics, and the resulting program, FastGeneDup, greatly improves upon the performance of GeneTree, a previous implementation to solve the gene-duplication problem. Furthermore, FastGeneDup made it possible to compute a supertree with 624 leaves from 3,978 input gene trees, representing over 25% of all available plant protein sequences, in less than three days on a desktop computer. This speed up also allows searching a much larger portion of the solution space within the same time, and hence can be used to obtain better solutions.

References

1. 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
2. Ma, B., Li, M., Zhang, L.: On reconcstructing 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**(3) (1997) 349–362
5. Page, R.D.M.: Extracting species trees from complex gene trees: reconciled trees and vertebrate phylogeny. Mol. 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., Page, R. In: Tangled tales from multiple markers: reconciling conflict between phylogenies to build molecular supertrees. 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 (In press)
9. 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
10. Guigó, R., Muchnik, I., Smith, T.F.: Reconstruction of ancient molecular phylogeny. Molecular Phylogenetics and Evolution **6**(2) (1996) 189–213
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**(3/4) (2000) 429–447
15. Bonizzoni, P., Vedova, G.D., Dondi, R.: Reconciling gene trees to a species tree. In: 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. 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), LNCS 1461. (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, LNCS 1663, Vancouver, Canada (1999)
21. Hallett, M.T., Lagergren, J.: New algorithms for the duplication-loss model. In: RECOMB. (2000) 138–146
22. Page, R.D.M.: Genetree. (`http://taxonomy.zoology.gla.ac.uk/rod/genetree/-genetree.html`)
23. Allen, B.L., Steel, M.: Subtree transfer operations and their induced metrics on evolutionary trees. Annals of Combintorics **5** (2001) 1–13
24. Bordewich, M., Semple, C.: On the computational complexity of the rooted subtree prune and regraft distance. Annals of Combintorics **8** (2004) 409–423
25. Dondoshansky, I.: Blastclust version 6.1 (2002)
26. Thompson, J., Higgins, D., Gibson, T.: ClustalW: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific penalties and weight matrix choice. Nucleic Acids Research **22** (1994) 4673–4680
27. Saitou, N., Nei, N.: The neighbour-joining method: a new method for reconstructing phylogenetic trees. Journal of Mol. Biology and Evolution **4** (1987) 406–425
28. Swofford, D.L.: PAUP*: Phylogenetic analysis using parsimony (*and other methods), version 4.0b10 (2002)
29. Driskell, A., An, C., Burleigh, J., McMahon, M., O'Meara, B., Sanderson, M.: Prospects for building the tree of life from large sequence databases. Science **306** (2004) 1172–1174