Efficient genome-scale phylogenetic analysis under the duplication-loss and deep coalescence cost models

Mukul S Bansal[1,3], J Gordon Burleigh[2], Oliver Eulenstein*[3]

[1]School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel
[2]Department of Biology, University of Florida, Gainesville, FL 32611, USA
[3]Department of Computer Science, Iowa State University, Ames, IA 50011, USA

Email: Mukul S Bansal - bansal@tau.ac.il; J Gordon Burleigh - gburleigh@ufl.edu; Oliver Eulenstein - oeulenst@cs.iastate.edu;

*Corresponding author

Abstract

Background: Genomic data provide a wealth of new information for phylogenetic analysis. Yet making use of this data requires phylogenetic methods that can efficiently analyze extremely large data sets and account for processes of gene evolution, such as gene duplication and loss, incomplete lineage sorting (deep coalescence), or horizontal gene transfer, that cause incongruence among gene trees. One such approach is gene tree parsimony, which, given a set of gene trees, seeks a species tree that requires the smallest number of evolutionary events to explain the incongruence of the gene trees. However, the only existing algorithms for gene tree parsimony under the duplication-loss or deep coalescence reconciliation cost are prohibitively slow for large datasets.

Results: We describe novel algorithms for SPR and TBR based local search heuristics under the duplication-loss cost, and we show how they can be adapted for the deep coalescence cost. These algorithms improve upon the best existing algorithms for these problems by a factor of n, where n is the number of species in the collection of gene trees. We implemented our new SPR based local search algorithm for the duplication-loss cost and demonstrate the tremendous improvement in runtime and scalability it provides compared to existing implementations. We also evaluate the performance of our algorithm on three large-scale genomic data sets.

Conclusions: Our new algorithms enable, for the first time, gene tree parsimony analyses of thousands of genes from hundreds of taxa using the duplication-loss and deep coalescence reconciliation costs. Thus, this work expands both the size of data sets and the range of evolutionary models that can be incorporated into genome-scale phylogenetic analyses.

Introduction

The availability of large-scale genomic data sets provides an unprecedented wealth of information for phylogenetic analyses. However, genomic data sets also present a number of unique challenges for phylogenetics. Beyond the exceptional computational challenges involved in analyses of thousands of genes from many taxa, complex patterns of gene evolution within the genome, including incomplete lineage sorting (deep coalescence), gene duplications and losses, lateral gene transfer, and recombination, create tremendous heterogeneity in the topology of gene trees and obscure species relationships [1]. In fact, in some evolutionary scenarios, trees from the majority of genes in a genome can differ from the species phylogeny (e.g., [2]). Thus, methods for incorporating genomic data into phylogenetic analyses must be both computationally tractable for extremely large data sets and must account for heterogeneous processes of gene family evolution. In this paper, we introduce novel algorithms that enable for the first time estimates of phylogenetic trees from large genomic data sets based on gene duplications and losses as well as incomplete lineage sorting.

One approach for building phylogenetic trees from genomic data sets is gene tree parsimony (GTP). Given a collection of gene trees, GTP seeks a species tree that implies the minimum reconciliation cost, or number of events that cause conflict among the gene trees. However, available GTP algorithms either lack sufficient speed to be useful for large data sets or the flexibility to deal with a wide range of evolutionary processes that affect gene tree topologies. In particular, the *duplication-loss problem* [3–13], which is the GTP problem based on minimizing the number of gene duplications and losses, and the *deep-coalescence problem* [1,14–16], which is the GTP problem based on minimizing the number of deep coalescences, lack efficient heuristics.

Both the duplication-loss problem and the deep-coalescence problem are NP-hard [15,17]. Therefore, in practice, these problems are typically approached using local search heuristics. These heuristics start with some initial candidate species tree and find a minimum reconciliation cost tree in its neighborhood. This constitutes one local search step. The best tree thus found then becomes the starting point for the next local search step, and so on, until a local minima is reached. Thus, at each local search step, the heuristic solves a "local search problem". The time complexity of this local search problem depends on the tree edit operation used to define the neighborhood.

Rooted subtree pruning and regrafting (SPR) [18] and rooted tree bisection and reconnection (TBR) [19] are two of the most effective and most commonly used tree edit operations. For example, Page [20] and Maddison and Knowles [14] implemented heuristics based on the SPR local tree search to estimate the species tree that minimizes the number of deep coalescences. Similarly, SPR based local search heuristics were also developed for the duplication-loss problem [20, 21]. However, these heuristics estimate the reconciliation cost from scratch for each tree topology that is evaluated, and therefore, they are only useful for small data sets. Recently, Than and Nakhleh [16] developed an integer linear programming formulation and a dynamic programming algorithm to provide exact solutions for the deep coalescence problem. Although these exact algorithms can analyze data sets with hundreds of gene trees, they are limited to a very small number of taxa. Furthermore, while there do exist fast heuristics based on SPR [22] and TBR [23] local searches for GTP, these are based on minimizing the gene duplication cost only.

Several methods exist for inferring species trees from collections of conflicting genes in a probabilistic framework, but, these are similarly limited. Liu and Pearl [24] and Kubatko et al. [25] discuss likelihood-based approaches to estimate a species tree from gene trees based on a coalescent process. These methods, along with Ané et al.'s [26] Bayesian approach to estimating concordance among gene trees, require gene trees with a single gene per taxon, and they have only been tested on small data sets. In addition, probabilistic models of gene tree-species tree reconciliation that incorporate duplication and loss also exist [27, 28]. However, these models are computationally complex, and they have not been incorporated in any tree search heuristics.

Our contributions: A lack of efficient heuristics has limited the use of the GTP approach for phylogenetic analyses of large-scale genomic data sets, and there are no options for such GTP analyses based on the duplication-loss and deep-coalescence problems. In this paper, we address this issue by presenting efficient, novel algorithms for SPR and TBR based local searches for both of these problems. The currently best known (naïve) solutions for the SPR and TBR local search problems, for the duplication-loss as well as the deep-coalescence problem, require $\Theta(kn^3)$ and $\Theta(kn^4)$ time respectively, where k is the number of input gene trees and n is the size of the resulting species tree.[1] Our new algorithms solve these SPR and TBR local search problems in

[1]Here we assume, for convenience, that the size of the k given gene trees differs by a constant factor from the size of the resulting species tree.

$O(kn^2)$ and $O(kn^3)$ time respectively. Consequently, our algorithms provide a speedup of a factor of n over the best known SPR and TBR local search algorithms (like the ones implemented in [20, 21]) for the duplication-loss and deep-coalescence problems. This enables, for the first time, GTP analyses with hundreds of taxa and thousands of genes based on gene duplications and losses or incomplete lineage sorting.

We first develop our algorithms in the context of the duplication-loss problem, and then show how to apply them to the deep-coalescence problem. Then, we demonstrate the improvement in runtime and scalability provided by our algorithms over the best current solutions by using an implementation of our algorithm for the SPR local search.

Basic notation and preliminaries

Given a rooted tree T, we denote its node set, edge set, and leaf set by $V(T)$, $E(T)$, and $Le(T)$ respectively. The root node of T is denoted by $rt(T)$. Given a node $v \in V(T)$, we denote its parent by $pa_T(v)$, its set of children by $Ch_T(v)$, and the subtree of T rooted at v by T_v. If two nodes in T have the same parent, they are called *siblings*. The set of *internal nodes* of T, denoted $I(T)$, is defined to be $V(T) \setminus Le(T)$. We define $\leq_T$ to be the partial order on $V(T)$ where $x \leq_T y$ if y is a node on the path between $rt(T)$ and x. The *least common ancestor* of a non-empty subset $L \subseteq V(T)$ in tree T, denoted as $\mathrm{lca}_T(L)$, is the unique smallest upper bound of L under $\leq_T$. Given $x, y \in V(T)$, $x \to_T y$ denotes the unique path from x to y in T. We denote by $d_T(x, y)$ the number of edges on the path $x \to_T y$. T is fully *binary* if every node has either zero or two children. Throughout this paper, the term tree refers to a rooted fully binary tree.

Given T and a set $L \subseteq Le(T)$, let T' be the minimal rooted subtree of T with leaf set L. We define the leaf induced subtree $T[L]$ of T on leaf set L to be the tree obtained from T' by successively removing each non-root node of degree two and adjoining its two neighbors.

The Duplication-Loss 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 set of species. Thus, the nodes in a gene tree represent genes. We assume that each leaf of the gene trees is labeled with the species from which that gene was sampled. In order to compare a gene tree G with a species tree S, we require a mapping from each gene $g \in V(G)$ to the most recent species in S that could have contained g.

Definition 1 (Mapping)**.** *The* leaf-mapping $\mathcal{L}_{G,S}\colon Le(G) \to Le(S)$ *maps a leaf node* $g \in Le(G)$ *to that unique leaf node* $s \in Le(S)$ *which has the same label as* g*. The extension* $\mathcal{M}_{G,S}\colon V(G) \to V(S)$ *of* $\mathcal{L}_{G,S}$ *is the* mapping *defined by* $\mathcal{M}_{G,S}(g) = \mathrm{lca}(\mathcal{L}_{G,S}(Le(G_g)))$.

For any node $s \in V(S)$, we use $\mathcal{M}^{-1}_{G,S}(s)$ to denote the set of nodes in G that map to node $s \in V(S)$ under the mapping $\mathcal{M}_{G,S}$.

Definition 2 (Comparability)**.** *Given trees G and S, we say that G is* comparable *to S if, for each $g \in Le(G)$, the leaf-mapping $\mathcal{L}_{G,S}(g)$ is well defined. A set of gene trees $\mathcal{G}$ is* comparable *to S if each gene tree in $\mathcal{G}$ is comparable to S.*

Throughout this paper we use the following terminology: $\mathcal{G}$ is a set of gene trees that is comparable to a species tree S, and $G \in \mathcal{G}$.

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

Following [8], we define the number of losses as follows.

Definition 4 (Losses)**.** *The number of* losses *$Loss(G, S, g)$ at a node $g \in I(G)$, is defined to be:*

- 0, *if* $\mathcal{M}_{G,S'}(g) = \mathcal{M}_{G,S'}(g')\ \forall g' \in Ch(g)$, *and*
- $\sum_{g' \in Ch(g)} |d_{S'}(\mathcal{M}_{G,S'}(g), \mathcal{M}_{G,S'}(g')) - 1|$, *otherwise;*

where $S' = S[Le(G)]$. We define $Loss(G, S) = \sum_{g \in I(G)} Loss(G, S, g)$ to be the number of losses in G.

Under the duplication-loss model, the reconciliation cost of G with S is simply the duplication-loss cost; i.e., the number of duplications and losses.

Definition 5 (Reconciliation cost)**.** *We define reconciliation costs for gene and species trees as follows:*

1. $\Delta(G, S) = |\,\mathit{Dup}(G, S)| + \mathit{Loss}(G, S)$ *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 are 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-Loss)**.** *Given a set* $\mathcal{G}$ *of gene trees, the* Duplication-Loss *problem is to find a species tree* S^* *comparable with* $\mathcal{G}$*, such that* $\Delta(\mathcal{G}, S^*) = \Delta(\mathcal{G})$.

Local search problems

Here we first provide the definition of an SPR edit operation [18] and then formulate the related local search problems. The definition and associated local search problems for the TBR edit operation are considered later.

For technical reasons, before we can define the SPR operation, we need the following definition.

Definition 6 (Planted tree)**.** *Given a tree* T*, the* planted tree $\Phi(T)$ *is the tree obtained by adding a* root edge $\{p, \mathit{rt}(T)\}$*, where* $p \notin V(T)$*, to* T.

Definition 7 (SPR operation)**.** *Let* T *be a tree,* $e = \{u, v\} \in E(T)$*, where* $u = \mathit{pa}(v)$*, and* X, Y *be the connected components that are obtained by removing edge* e *from* T *such that* $v \in X$ *and* $u \in Y$*. We define* $\mathsf{SPR}_T(v, y)$ *for* $y \in Y$ *to be the tree that is obtained from* $\Phi(T)$ *by first removing edge* e*, and then adjoining a new edge* f *between* v *and* Y *as follows:*

1. *Create a new node* y' *that subdivides the edge* $\{\mathit{pa}(y), y\}$.
2. *Add edge* f *between nodes* v *and* y'.
3. *Suppress the node* u*, and rename* y' *as* u.
4. *Contract the root edge.*

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.

Notation. We define the following:

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

Throughout the remainder of this manuscript, S denotes a species tree such that $\mathit{Le}(S) = \bigcup_{G \in \mathcal{G}} \bigcup_{g \in \mathit{Le}(G)} \mathcal{L}_{G,S}(g)$, and v is a non-root node in $V(S)$.

We now define the relevant local search problems based on the SPR operation.

Problem 2 (SPR-Scoring (SPR-S))**.** *Given* $\mathcal{G}$ *and* S*, find a tree* $T^* \in \mathsf{SPR}_S$ *such that* $\Delta(\mathcal{G}, T^*) = \min_{T \in \mathsf{SPR}_S} \Delta(\mathcal{G}, T)$.

Our goal is to solve the SPR-S problem efficiently. To that end, we first define a restricted version of the SPR-S problem, called the SPR-*Restricted Scoring* problem.

Problem 3 (SPR-Restricted Scoring (SPR-RS))**.** *Given* $\mathcal{G}$*,* S*, and* v*, find a tree* $T^* \in \mathsf{SPR}_S(v)$ *such that* $\Delta(\mathcal{G}, T^*) = \min_{T \in \mathsf{SPR}_S(v)} \Delta(\mathcal{G}, T)$.

Let $n = |\,\mathit{Le}(S)|$, $m = |\,\mathit{Le}(S)| + |\,\mathit{Le}(G)|$ and $k = |\mathcal{G}|$, and let us assume, for convenience, that all $G \in \mathcal{G}$ have approximately the same size. In the following, we show how to solve the SPR-RS problem in $O(km)$ time. Since $\mathsf{SPR}_S = \bigcup_{\{\mathit{pa}(v), v\} \in E(S)} \mathsf{SPR}_S(v)$, it is easy to see that the SPR-S problem can be solved by solving the SPR-RS problem $O(n)$ times. This yields an $O(kmn)$ time algorithm for the SPR-S problem. Later, we show that the local search problem corresponding to the TBR operation reduces to solving $O(n^2)$ SPR-RS problems; which yields an $O(kmn^2)$ time algorithm for the TBR local search problem. In the interest of brevity, all lemmas and theorems in this paper appear with proofs omitted; however, all proofs are available in [29].

Solving the SPR-RS problem

Throughout this section, we limit our attention to one gene tree G; in particular, we show how to solve the SPR-RS problem for G in $O(m)$ time. Our algorithm extends trivially to solve the SPR-RS problem on the set of gene trees $\mathcal{G}$ in $O(km)$ time. For simplicity, we will assume that $\mathit{Le}(G) = \mathit{Le}(S)$.[2]

In order to solve the SPR-RS problem for G, it is sufficient to compute the values $|\,\mathit{Dup}(G, S')|$ and $\mathit{Loss}(G, S')$ for each $S' \in \mathsf{SPR}_S(v)$. Bansal et al. [22] showed how to compute the value $|\,\mathit{Dup}(G, S')|$ for each $S' \in \mathsf{SPR}_S(v)$, in $O(m)$ time. Losses, however, behave in a very different and much more complex manner compared to gene duplications and it has

[2]Note: if $\mathit{Le}(G) \neq \mathit{Le}(S)$ then we can simply set the species tree to be $S[\mathit{Le}(G)]$. This takes $O(n)$ time and, consequently, does not affect the time complexity of our algorithm.

remained unclear if their computation could be similarly optimized. In this paper we show that it is indeed possible to compute the value $\mathit{Loss}(G, S')$ for each $S' \in \mathsf{SPR}_S(v)$ in $O(m)$ time as well. Altogether, this implies that the SPR-RS problem for G can be solved in $O(m)$ time. Next, we introduce some of the basic structural properties that are helpful in the current setting.

Basic structural properties

Consider the tree $N^{S,v} = \mathsf{SPR}_S(v, rt(S))$. Observe that, since $\mathsf{SPR}_{N^{S,v}}(v) = \mathsf{SPR}_S(v)$, solving the SPR-RS problem on instance $\langle \{G\}, S, v\rangle$ is equivalent to solving it on the instance $\langle \{G\}, N^{S,v}, v\rangle$. Thus, in the remainder of this section, we will work with tree $N^{S,v}$ instead of tree S; the reason for this choice becomes clear in light of Lemmas 3 and 4.

Since S and v are fixed in the current context, we will, in the interest of clarity, abbreviate $N^{S,v}$ simply to N. Similarly, in the remainder of this section, we abbreviate $\mathcal{M}_{G,T}$ to $\mathcal{M}_T$, for any species tree T.

Throughout the remainder of this work, let u denote the sibling of v in N. We color the nodes of N as follows: (i) All nodes in the subtree N_v are colored red, (ii) the root node of N is colored blue, and (iii) all the remaining nodes, i.e. all nodes in N_u, are colored green. Correspondingly, we color the nodes of G by assigning to each $g \in V(G)$ the color of the node $\mathcal{M}_N(g)$.

Definition 8 (Γ). *We define Γ to be the tree obtained from G by removing all red nodes (along with any edges incident on these red nodes). Observe that while Γ must be binary, it might not be* fully *binary.*

The significance of the tree Γ stems from the following two Lemmas which, together, completely characterize the mappings from nodes in $V(G)$ for each $S' \in \mathsf{SPR}_S(v)$. This characterization is the basis of Lemmas 3 through 8.

Lemma 1. *Given G and N, if $g \in V(G)$ is either red or green, then $\mathcal{M}_{S'}(g) = \mathcal{M}_N(g)$ for all $S' \in \mathsf{SPR}_N(v)$.*

Lemma 2. *Given G and N, if $g \in V(G)$ is a blue node, then $\mathcal{M}_{S'}(g) = \mathrm{lca}_{S'}(v, \mathcal{M}_{\Gamma,N}(g))$ for any $S' \in \mathsf{SPR}_N(v)$.*

Characterizing losses

To solve the SPR-RS problem efficiently we rely on the following six lemmas, which make it possible to efficiently infer the value of $\mathit{Loss}(G, S', g)$ for any $S' \in \mathsf{SPR}_N(v)$ and any $g \in V(G)$.

Consider any $g \in I(G)$, and let g' and g'' be its two children. Let $a = \mathcal{M}_N(g)$, $b = \mathcal{M}_N(g')$ and $c = \mathcal{M}_N(g'')$. Without loss of generality, node g must correspond to one of the following six categories: 1) g is red, 2) g is green, 3) g, g', and g'' are all blue, 4) g and g' are blue, and g'' is green, 5) g and g' are blue, and g'' is red, or, 6) g is blue, g' is red, and g'' is green.

Lemmas 3 through 8 characterize the behavior of the loss cost $\mathit{Loss}(G, S', g)$, for each $S' \in \mathsf{SPR}_N(v)$, for each of these six cases. At this point, it would help to observe that $\mathsf{SPR}_N(v) = \{\mathsf{SPR}_N(v, s) \colon s \in V(N_u)\}$.

Lemma 3. *If g is red then $\mathit{Loss}(G, S', g) = \mathit{Loss}(G, N, g)$ for all $S' \in \mathsf{SPR}_N(v)$.*

Lemma 4. *If g is green then $\mathit{Loss}(G, S', g) = \mathit{Loss}(G, N, g) + 1$ if $S' = \mathsf{SPR}_N(v, x)$ where $b \leq_N x <_N a$ or $c \leq_N x <_N a$, and $\mathit{Loss}(G, S', g) = \mathit{Loss}(G, N, g)$ otherwise.*

Lemma 5. *Let g, g' and g'' all be blue nodes, $x \in V(N_u)$, and let $a' = \mathcal{M}_{\Gamma,N}(g)$, $b' = \mathcal{M}_{\Gamma,N}(g')$ and $c' = \mathcal{M}_{\Gamma,N}(g'')$.*

1. *If $S' = \mathsf{SPR}_N(v, x)$ where $x \not<_N a'$, then $\mathit{Loss}(G, S', g) = \mathit{Loss}(G, N, g)$.*
2. *If $S' = \mathsf{SPR}_N(v, x)$ where $x <_N a'$, and $S'' = \mathsf{SPR}_N(v, pa(x))$, then,*
   - (a) *$\mathit{Loss}(G, S', g) = \mathit{Loss}(G, S'', g) + 1$ if $b' \leq_N x <_N a'$ or $c' \leq_N x <_N a'$, and,*
   - (b) *$\mathit{Loss}(G, S', g) = \mathit{Loss}(G, S'', g)$ otherwise.*

Lemma 6. *Let g and g' be blue nodes and g'' be a green node, $x \in V(N_u) \setminus \{u\}$, and let $a' = \mathcal{M}_{\Gamma,N}(g)$, $b' = \mathcal{M}_{\Gamma,N}(g')$ and $c' = \mathcal{M}_{\Gamma,N}(g'')$.*

1. *If $S' = \mathsf{SPR}_N(v, x)$ where $x \not<_N a'$, and $S'' = \mathsf{SPR}_N(v, pa(x))$, then,*
   - (a) *$\mathit{Loss}(G, S', g) = \mathit{Loss}(G, S'', g) - 1$ if $a' \leq_N x <_N u$,*
   - (b) *$\mathit{Loss}(G, S', g) = \mathit{Loss}(G, S'', g) - 1$ if $a' \leq_N pa(x) <_N u$ but x is not such that $a' \leq_N x <_N u$, and,*

(c) *Loss*$(G, S', g) =$ *Loss*(G, S'', g) *otherwise.*

2. *Let* $S' = \mathsf{SPR}_N(v, x)$ *where* $x <_N a'$ *and* $S'' = \mathsf{SPR}_N(v, pa(x))$.

   (a) *If* $a' \neq b'$ *and* b'' *denotes the child of* a' *along the path* $a' \rightarrow_N b'$, *then,*

      i. *Loss*$(G, \mathsf{SPR}_N(v, b''), g) =$ *Loss*$(G, \mathsf{SPR}_N(v, a'), g) - 2$ *if* $a' \neq c'$. *And,* *Loss*$(G, \mathsf{SPR}_N(v, b''), g) =$ *Loss*$(G, \mathsf{SPR}_N(v, a'), g)$ *if* $a' = c'$,

      ii. *Loss*$(G, S', g) =$ *Loss*$(G, S'', g) + 1$ *if* $b' \leq_N x <_N b''$,

      iii. *Loss*$(G, S', g) =$ *Loss*(G, S'', g) *if* x *is such that* $x \in V(N_{b''})$ *but not such that* $b' \leq_N x <_N a'$,

      iv. *Loss*$(G, S', g) =$ *Loss*$(G, \mathsf{SPR}_N(v, a'), g)$ *if* $c' \leq_N x <_N a'$, *and,*

      v. *Loss*$(G, S', g) =$ *Loss*$(G, \mathsf{SPR}_N(v, a'), g)$ -1 *otherwise.*

   (b) *If* $a' = b'$, *then,*

      i. *Loss*$(G, S', g) =$ *Loss*$(G, \mathsf{SPR}_N(v, a'), g)$ *if* $c' \leq_N x <_N a'$, *and,*

      ii. *Loss*$(G, S', g) =$ *Loss*$(G, \mathsf{SPR}_N(v, a'), g)$ -1 *otherwise.*

Lemma 7. *Let* g *and* g' *be blue nodes and* c *be a red node,* $x \in V(N_u)$, *and let* $a' = \mathcal{M}_{\Gamma,N}(g)$.

1. *If* $S' = \mathsf{SPR}_N(v, x)$ *where* $x <_N a'$, *and* $S'' = \mathsf{SPR}_N(v, pa(x))$, *then* *Loss*$(G, S', g) =$ *Loss*$(G, S'', g) + 1$.

2. *If* $S' = \mathsf{SPR}_N(v, x)$ *where* $x \not<_N a'$, *then,*

   (a) *Loss*$(G, S', g) =$ *Loss*(G, N, g) *if* $a' \leq_N x \leq_N u$, *and,*

   (b) *Loss*$(G, S', g) =$ *Loss*$(G, S'', g) + 1$ *for* $S'' = \mathsf{SPR}_N(v, pa(x))$ *otherwise.*

Lemma 8. *Let* g *be blue,* g' *be red, and* g'' *be green. Let* $x \in V(N_u) \setminus \{u\}$ *and* $a' = \mathcal{M}_{\Gamma,N}(g)$.

1. *If* $S' = \mathsf{SPR}_N(v, x)$ *where* $x \not<_N a'$, *and* $S'' = \mathsf{SPR}_N(v, pa(x))$, *then,*

   (a) *Loss*$(G, S', g) =$ *Loss*$(G, S'', g) - 1$ *if* $a' \leq_N x <_N u$,

   (b) *Loss*$(G, S', g) =$ *Loss*(G, S'', g) *if* $a' \leq_N pa(x) <_N u$ *but* x *is not such that* $a' \leq_N x <_N u$, *and,*

   (c) *Loss*$(G, S', g) =$ *Loss*$(G, S'', g) + 1$ *otherwise.*

2. *If* $S' = \mathsf{SPR}_N(v, x)$ *where* $x <_N a'$, *and* $S'' = \mathsf{SPR}_N(v, pa(x))$, *then,*

   (a) *Loss*$(G, S', g) =$ *Loss*$(G, \mathsf{SPR}_N(v, a'), g) + 2$ *if* $x \in Ch_N(a')$, *and,*

   (b) *Loss*$(G, S', g) =$ *Loss*$(G, S'', g) + 1$ *otherwise.*

The algorithm

Observe that $\mathsf{SPR}_N(v) = \{\mathsf{SPR}_N(v, s) \colon s \in V(N_u)\}$. Therefore, the goal of our algorithm is to compute at each node $s \in V(N_u)$ the value *Loss*(G, S'), where $S' = \mathsf{SPR}_N(v, s)$. To do this efficiently, we rely on the characterization of losses given in Lemmas 3 through 8.

The first step of the algorithm is to compute the value *Loss*(G, N). This "loss value" is assigned to the node u. To compute the loss value for the rest of the nodes our algorithm makes use of six different types of *counters* at each node in N_u; we refer to these counters as counter-i, for $i \in \{1, \ldots, 6\}$. The reason for using these six counters is that the behavior of the loss values can be explained by using six types of patterns (captured by the six counters). These counters make it possible to efficiently compute the difference between the values *Loss*(G, N) and *Loss*(G, S'), where $S' = \mathsf{SPR}_N(v, s)$, for each $s \in V(N_u)$. Next, we describe each of these six counters; throughout our description, s represents some node in N_u.

counter-1 If the value of counter-1 is x at node s then this implies that the tree $\mathsf{SPR}_N(v, s)$ incurs x additional losses over the value *Loss*(G, N).

counter-2 If the value of counter-2 is x at node s, then this implies that for each $t \leq_N s$ the tree $\mathsf{SPR}_N(v, t)$ incurs an additional x losses over *Loss*(G, N).

counter-3 If the value of counter-3 is x at node s, then this implies that for each $t \leq_N s$ the tree $\mathsf{SPR}_N(v, t)$ loses x losses from *Loss*(G, N).

counter-4 If the value of counter-4 is x at node s, then this implies that for each $t \leq_N s$ the tree $\mathsf{SPR}_N(v, t)$ incurs $\alpha_t \cdot x$ additional losses over *Loss*$(G, N$, where $\alpha_t = d_N(pa(s), t)$.

counter-5 If the value of counter-5 is x at node s, then it is equivalent to incrementing counter-4 at the sibling of each node on the path $u \rightarrow_N s$, except at u, by x.

counter-6 If the value of counter-6 is x at node s, then it is equivalent to incrementing counter-4 at both children (if they exist) of the sibling of each node along the path $u \rightarrow_N s$, except u, and incrementing counter-3 at each node along the path $u \rightarrow_N s$, except at u, by x.

In the remainder of this section we first show how to compute the values of these counters, and then the final loss values, at each node in N_u.

Computing the counters

We now describe how the values of the six counters are computed. Initially, each counter at each node in N_u is set to 0. Consider any $g \in I(G)$, and let g' and g'' be its two children. Recall that node g must fall under one of the following six categories: 1) g is red, 2) g is green, 3) g, g', and g'' are all blue, 4) g and g' are blue, and g'' is green, 5) g and g' are blue, and g'' is red, or, 6) g is blue, g' is red, and g'' is green.

Let $a = \mathcal{M}_N(g)$, $b = \mathcal{M}_N(g')$ and $c = \mathcal{M}_N(g'')$. Also, whenever properly defined, let $a' = \mathcal{M}_{\Gamma,N}(g)$, $b' = \mathcal{M}_{\Gamma,N}(g')$ and $c' = \mathcal{M}_{\Gamma,N}(g'')$. Based on Lemmas 3 through 8, we now study how the six counters can be updated so as to capture the behavior of losses in each of these cases.

Case 1. By Lemma 3, we do nothing in this case.

Case 2. Based on Lemma 4, the contribution of any node g that satisfies the condition of case 2 can be captured by incrementing the value of counter-1 by one at each node on paths $a \rightarrow_N b$ and $a \rightarrow_N b$, except at node a.

Case 3. From Lemma 5 it follows that in this case the contribution of g to the loss value changes in a way that is captured by incrementing counter-2 by 1, at each node, except a', on the paths $a' \rightarrow_N b'$ and $a' \rightarrow_N c'$.

Case 4. According to Lemma 6, if N_v is regrafted on an edge of N_u that is not in $N_{a'}$, then the contribution of g to the loss cost is captured by incrementing counter-3 by 1 at each node except u along the path $u \rightarrow_N a'$, and at their siblings. If N_v is regrafted on an edge of N_u that is in $N_{a'}$ then there are two possible cases:

$a' \neq b'$: Recall that b'' represents the child of a' along the path $a' \rightarrow_N b'$. Here, the contribution of g to the loss cost is captured by (i) incrementing counter-3 by two at node b'', (ii) incrementing counter-2 by one at each node except b'' along the path $b'' \rightarrow_N b'$, (iii) incrementing counter-3 by one at the sibling of b'', and (iv) incrementing counter-1 by one at each node except a' on the path $a' \rightarrow_N c'$.

$a' = b'$: In this case, the contribution of g to the loss cost is captured by (i) incrementing counter-3 by one at both children of a', and (ii) incrementing counter-1 by one at each node except a' on the path $a' \rightarrow_N c'$.

Case 5. By Lemma 7, for this case, the change in the loss contribution of g is captured by incrementing counter-5 by 1 at node a', and by incrementing counter-4 by 1 at both children of a' in N.

Case 6. By Lemma 8, for this case, the change in the loss contribution of g is captured by incrementing counter-6 by 1 at node a', and by incrementing counter-4 and counter-2 by 1 each at both children of a' in N.

Computing the final loss values

Our algorithm considers each internal node of gene tree G, one at a time, and updates the relevant counters at the relevant nodes in N_u, as shown in the previous subsection. Then, based on the counters, it computes, at each node $s \in V(N_u)$ the value $\alpha(s) = \mathit{Loss}(G, S') - \mathit{Loss}(G, N)$, where $S' = \mathsf{SPR}_N(v, s)$; this can be achieved by performing a constant number of pre- and post-order traversals of N_u. A final traversal of the tree now allows us to compute the value $\mathit{Loss}(G, S') = \alpha(s) + \mathit{Loss}(G, N)$ at each $s \in V(N_u)$. Due to space limitations, a more complete description of the algorithm is omitted from this manuscript (but is available in [29]).

[3]We point out that the $\Theta(n)$ speed-up obtained by our algorithm does not depend on this simplifying assumption.

For simplicity in stating the time complexity of our algorithm, we assume that all $G \in \mathcal{G}$ have approximately the same size.[3] Recall that $n = |\,Le(S)|$, $m = |\,Le(S)| + |\,Le(G)|$ and $k = |\mathcal{G}|$.

Lemma 9. *The* SPR*-RS problem on the input instance* $\langle\{G\}, S, v\rangle$ *can be solved in* $O(m)$ *time.*

Remark on Lemma 9: Observe that in cases 2, 3 and 4 (from the previous subsection), handling each g might require updating the counters at $\Theta(n)$ nodes, yielding a total time complexity of $O(nm)$ for these cases. However, it is still possible to obtain the $O(m)$ time bound.
Thus, we have the following theorem.

Theorem 1. *The* SPR*-RS and* SPR*-S problems can be solved in* $O(km)$ *and* $O(kmn)$ *time respectively.*

The time complexity of the best known (naïve) solution for the SPR-S problem is $\Theta(kmn^2)$. Our algorithm improves on this by a factor of n.

Speeding-up the TBR local search problem

Intuitively, a (rooted) TBR operation may be viewed as being like an SPR operation except that the TBR operation allows the pruned subtree to be arbitrarily rerooted before being regrafted. The TBR-S problem is defined analogously to the SPR-S problem.

Observe that there are $\Theta(n)$ different ways to select a subtree of S to be pruned. Furthermore, there are $O(n)$ different ways to reroot the pruned subtree. The idea is to directly use the solution to the SPR-RS problem to compute the duplication and loss costs for the $O(n)$-cardinality subset of TBR_S defined by any fixed pruned subtree and its fixed rooting. This yields the following theorem.

Theorem 2. *The* TBR*-S problem can be solved in* $O(kmn^2)$ *time.*

This improves on the best known solution for the TBR-S problem by a factor of n.

The deep coalescence cost model

Our algorithms for efficient SPR and TBR local searches for the duplication-loss model apply directly to the corresponding SPR and TBR local search problems for the deep coalescence model. This can be achieved using any of the following two methods. The first method is to make use of the result of Zhang [15] who showed showed that if G is a uniquely leaf-labeled gene tree and S is a species tree such that $Le(G) = Le(S)$, then the deep coalescence cost of G and S is equal to $Loss(G, S) - 2|\,Dup(G, S)|$. Thus, our algorithms imply that we can compute the deep coalescence cost of each tree in the SPR (resp. TBR) neighborhood of S in $O(n^2)$ (resp. $O(n^3)$) time. The second method is to use the algorithm for computing losses, presented in this paper, and slightly modifying it to directly compute the deep coalescence cost. Owing to the similarity in the definition of the loss cost and the deep coalescence cost, this can be done in a straightforward manner (details omitted for brevity). Overall our algorithms yield speed-ups of a factor of n over the fastest current approaches for SPR and TBR local searches for the deep coalescence cost model.

Experimental analysis

To evaluate the performance of our novel local search algorithms, we implemented our algorithm for the SPR-S problem as part of a standard search heuristic for the duplication-loss problem. We refer to our program as *DupLoss*. We first evaluated its performance on randomly generated gene trees. These data sets represent extreme examples of incongruence among gene trees, and thus, they provide a challenging way to test the run-time performance of a gene tree reconciliation method. The input gene trees for each run consisted of 20 trees, each with the same set of taxa and with random binary topologies and random assignment of leaf labels. We conducted runs with 50, 100, 200, 400, and 1000 taxa in each gene tree. All analyses were performed on a 3 Ghz Intel Pentium 4 CPU based PC with Windows XP operating system. We compared the performance of DupLoss with the program GeneTree [20], which, like Mesquite [21], implements similar local search heuristics based on the best known (naïve) algorithm for the SPR-S problem. Our implementation shows a tremendous improvement in runtime and scalability compared to GeneTree (Table 1). For example, on the 200 taxon data set, our implementation finished in less than five minutes, while GeneTree ran for almost six days (Table 1). Furthermore, we could not run GeneTree when the input trees had more than 200 taxa.

We also tested the performance of DupLoss using several empirical data sets. First, we ran our implementation on the 8-taxon, 106-gene yeast data set of Rokas et al. [30], and the 8-taxon, 268-gene Apicomplexan data set of Kuo et al. [31]. For the yeast data set, we made the gene trees using maximum likelihood implemented in RAxML [32], and for the Apicomplexan data set, we used the gene trees included as supplemental data [31]. Tree searches for both data sets finished within one second, and the topologies were consistent with the unrooted species trees presented in each study. Finally, we ran our implementation on a plant data set consisting of $18,896$ gene trees from 136 taxa, previously used in a gene tree reconciliation analysis based on duplications only [33]. Although this is among the largest data sets used in a gene tree reconciliation analysis, our heuristic finished in approximately 24 hours, and the resulting species tree was consistent with the general consensus of plant relationships. DupLoss is freely available upon request.

Discussion

The novel algorithms presented in this paper enable GTP analyses that incorporate gene duplications and losses as well as deep coalescence on a scale that is impossible with previous implementations. Whereas most phylogenetic analyses only use data from putative orthologs, the duplication-loss model allows one to incorporate data from large gene families into phylogenetic analysis, and it does not depend on the accuracy of orthology estimates. Incorporating deep coalescence events may be critical when there is a history of rapid speciation, and these evolutionary scenarios represent many of the most difficult, and interesting, phylogenetic problems.

Previous advances in GTP heuristics for the duplication cost problem [22,23,34,35], that is, calculating the species tree based on the number of duplications only without considering losses, have enabled promising phylogenetic analyses of extremely large data sets (e.g., 136 taxa and $18,896$ genes in [33]). With partial sequence data, it is difficult, if not impossible, to distinguish losses from missing sequence data. Thus, it has been argued that the duplication only reconciliation cost is more appropriate than the duplication-loss cost when analyzing incomplete data sets (e.g., [36]). However, with the rapid accumulation of complete genome sequences, the duplication-loss problem provides a more complete model of evolution than the duplication problem. Similarly, there has been much recent interest in the deep coalescence problem (e.g., [14]), but it has not been applied to large data sets due to a lack of efficient heuristics.

While GTP has been effective on small data sets (see review in [36]), its performance in general is relatively uncharacterized, largely due to the lack of fast implementations. The parsimony approach minimizes the number of evolutionary events that are counted, and therefore, it may not be appropriate when genes exhibit high rates of duplication and loss or deep coalescence events. In such cases, inferring the species tree based on a likelihood model of gene evolution may be more appropriate. Still, likelihood methods are often computationally burdensome, and the computational difficulties are compounded by the extremely large data sets that are an inherent part of genome-scale analysis. The algorithms in this paper provide a pragmatic solution to the problem of addressing complex processes of evolution on enormous data sets in a phylogenetic analysis.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

MSB was responsible for algorithm design and program implementation, and wrote major parts of the paper. JGB performed the experimental evaluation and the analysis of the results, and contributed to the writing of the manuscript. OE supervised the project and contributed to the writing of the paper. All authors read and approved the final manuscript.

Acknowledgements

This work was supported in part by NSF grants 0334832 and 0830012. In addition, MSB was supported in part by a postdoctoral fellowship from the Edmond J. Safra Bioinformatics program at Tel-Aviv university.

References

1. Maddison WP: **Gene Trees in Species Trees**. *Systematic Biology* 1997, **46**:523–536.

2. Degnan JH, Rosenberg NA: **Discordance of Species Trees with Their Most Likely Gene Trees**. *PLoS Genetics* 2006, **2**(5):e68.
3. Goodman M, Czelusniak J, Moore GW, Romero-Herrera AE, Matsuda G: **Fitting the gene lineage into its species lineage. A parsimony strategy illustrated by cladograms constructed from globin sequences**. *Systematic Zoology* 1979, **28**:132–163.
4. Page RDM: **Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas**. *Systematic Biology* 1994, **43**:58–77.
5. Guigó R, Muchnik I, Smith TF: **Reconstruction of Ancient Molecular Phylogeny**. *Molecular Phylogenetics and Evolution* 1996, **6**(2):189–213.
6. Mirkin B, Muchnik I, Smith TF: **A Biologically Consistent Model for Comparing Molecular Phylogenies**. *Journal of Computational Biology.* 1995, **2**(4):493–507.
7. Eulenstein O, Vingron M: **On the equivalence of two tree mapping measures**. *Discrete Applied Mathematics* 1998, **88**:101–126.
8. Hallett MT, Lagergren J: **New algorithms for the duplication-loss model**. In *RECOMB* 2000:138–146.
9. Bonizzoni P, Vedova GD, Dondi R: **Reconciling a gene tree to a species tree under the duplication cost model**. *Theor. Comput. Sci.* 2005, **347**(1-2):36–53.
10. Górecki P, Tiuryn J: **DLS-trees: A model of evolutionary scenarios**. *Theor. Comput. Sci.* 2006, **359**(1-3):378–399.
11. Durand D, Halldórsson BV, Vernot B: **A Hybrid Micro-Macroevolutionary Approach to Gene Tree Reconstruction**. *Journal of Computational Biology.* 2006, **13**(2):320–335.
12. Chauve C, Doyon JP, El-Mabrouk N: **Gene Family Evolution by Duplication, Speciation, and Loss**. *Journal of Computational Biology.* 2008, **15**(8):1043–1062.
13. Chauve C, El-Mabrouk N: **New Perspectives on Gene Family Evolution: Losses in Reconciliation and a Link with Supertrees**. In *RECOMB* 2009:46–58.
14. Maddison WP, Knowles LL: **Inferring Phylogeny Despite Incomplete Lineage Sorting**. *Systematic Biology* 2006, **55**:21–30.
15. Zhang L: **Inferring a Species Tree from Gene Trees under the Deep Coalescence Cost**. In *RECOMB* 2000:192–193.
16. Than C, Nakhleh L: **Species tree inference by minimizing deep coalescences**. *PLoS Computational Biology* 2009, **5**(9):e1000501.
17. Ma B, Li M, Zhang L: **From Gene Trees to Species Trees**. *SIAM J. Comput.* 2000, **30**(3):729–752.
18. Bordewich M, Semple C: **On the computational complexity of the rooted subtree prune and regraft distance**. *Annals of Combinatorics* 2004, **8**:409–423.
19. Chen D, Eulenstein O, Fernández-Baca D, Burleigh JG: **Improved Heuristics for Minimum-Flip Supertree Construction**. *Evolutionary Bioinformatics* 2006, **2**:347–356.
20. Page RDM: **GeneTree: comparing gene and species phylogenies using reconciled trees**. *Bioinformatics* 1998, **14**(9):819–820.
21. Maddison WP, Maddison D: **Mesquite: a modular system for evolutionary analysis. Version 2.6. http://mesquiteproject.org** 2009.
22. Bansal MS, Burleigh JG, Eulenstein O, Wehe A: **Heuristics for the Gene-Duplication Problem: A $\Theta(n)$ Speed-Up for the Local Search**. In *RECOMB* 2007:238–252.
23. Bansal MS, Eulenstein O: **An $\Omega(n^2/\log n)$ Speed-Up of TBR Heuristics for the Gene-Duplication Problem**. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 2008, **5**(4):514–524.
24. Liu L, Pearl DK: **Species Trees from Gene Trees: Reconstructing Bayesian Posterior Distributions of a Species Phylogeny Using Estimated Gene Tree Distributions**. *Systematic Biology* 2007, **56**(3):504–514.
25. Kubatko LS, Carstens BC, Knowles LL: **STEM: species tree estimation using maximum likelihood for gene trees under coalescence**. *Bioinformatics* 2009, **25**(7):971–973.
26. Ané C, Larget B, Baum DA, Smith SD, Rokas A: **Bayesian Estimation of Concordance Among Gene Trees**. *Mol. Biol. Evol.* 2007, **24**(7):1575.
27. Arvestad L, Berglund AC, Lagergren J, Sennblad B: **Bayesian gene/species tree reconciliation and orthology analysis using MCMC**. In *ISMB (Supplement of Bioinformatics)* 2003:7–15.
28. Äkerborg O, Sennblad B, Arvestad L, Lagergren J: **Simultaneous Bayesian gene tree reconstruction and reconciliation analysis**. *Proceedings of the National Academy of Sciences* 2009, **106**(14):5714–5719.
29. Bansal MS: **Algorithms for efficient phylogenetic tree construction**. *PhD thesis*, Iowa State Univ. 2009.
30. Rokas A, Williams BL, King N, Carroll SB: **Genome-scale approaches to resolving incongruence in molecular phylogenies**. *Nature* 2003, **425**:798–804.
31. Kuo CH, Wares JP, Kissinger JC: **The Apicomplexan Whole-Genome Phylogeny: An Analysis of Incongruence among Gene Trees**. *Mol. Biol. Evol.* 2008, **25**(12):2689–2698.
32. Stamatakis A: **RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models**. *Bioinformatics* 2006, **22**(21):2688–2690.
33. Burleigh JG, Bansal MS, Eulenstein O, Hartmann S, Wehe A, Vision TJ: **Genome-scale phylogenetics: inferring the plant tree of life from 18,896 discordant gene trees**. *Systematic Biology, In press.*
34. Bansal MS, Eulenstein O, Wehe A: **The Gene-Duplication Problem: Near-Linear Time Algorithms for NNI-Based Local Searches**. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 2009, **6**(2):221–231.
35. Wehe A, Bansal MS, Burleigh JG, Eulenstein O: **DupTree: a program for large-scale phylogenetic analyses using gene tree parsimony**. *Bioinformatics* 2008, **24**(13).

36. Cotton JA, Page RDM: **Tangled tales from multiple markers: reconciling conflict between phylogenies to build molecular supertrees**. In *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*. Edited by Bininda-Emonds ORP, Springer-Verlag 2004:107–125.

Tables

Table 1 - GeneTree vs. DupLoss

Comparison of the runtimes of GeneTree and DupLoss on the same randomly generated datasets. Times are given in days(d), hours(h), minutes(m), and seconds(s).

Taxa size	GeneTree	DupLoss
50	11m:42s	5s
100	3h:57m	33s
200	5d:19h:49m	4m:24s
400	–	43m:08s
1000	–	19h:27m