# Detecting Highways of Horizontal Gene Transfer

Mukul S. Bansal*[1], Guy Banay[1], J. Peter Gogarten[2], and Ron Shamir†[1]

[1]The Blavatnik School of Computer Science, Tel-Aviv University, Israel
mukul@csail.mit.edu, guy.banay@gmail.com, rshamir@tau.ac.il
[2]Department of Molecular and Cell Biology, University of Connecticut, Storrs, USA
gogarten@uconn.edu

**Abstract**

In a horizontal gene transfer (HGT) event, a gene is transferred between two species that do not have an ancestor-descendant relationship. Typically, no more than a few genes are horizontally transferred between any two species. However, several studies identified pairs of species between which many different genes were horizontally transferred. Such a pair is said to be linked by a *highway of gene sharing*. We present a method for inferring such highways. Our method is based on the fact that the evolutionary histories of horizontally transferred genes disagree with the corresponding species phylogeny. Specifically, given a set of gene trees and a trusted rooted species tree, each gene tree is first decomposed into its constituent quartet trees and the quartets that are inconsistent with the species tree are identified. Our method finds a pair of species such that a highway between them explains the largest (normalized) fraction of inconsistent quartets. For a problem on $n$ species and $m$ input quartet trees, we give an efficient $O(m + n^2)$-time algorithm for detecting highways, which is optimal with respect to the quartets input size. An application of our method to a dataset of 1128 genes from 11 cyanobacterial species, as well as to simulated datasets, illustrates the efficacy of our method.

**Keywords:** Horizontal gene transfer, microbial evolution, quartets, algorithms.

## 1 Introduction

*Horizontal gene transfer (HGT)* (also called lateral gene transfer) is an evolutionary process in which genes are transferred between two organisms that do not have an ancestor-descendant relationship. HGT plays an important role in bacterial evolution by allowing them to transfer genes across species boundaries. This transfer of genes between divergent organisms first became a research focus when the transfer of antibiotic resistance genes was discovered (Ochiai *et al.*, 1959; Gray and Fitch, 1983). Microbiologists soon realized that the sharing of genes between unrelated species resulted in evolutionary patterns very different from those found in multi-cellular animals. Gene transfer often was seen as preventing a natural taxonomy of prokaryotes, i.e., a classification based on shared ancestry (see Sapp 2005 for a review). Some went so far as to suggest that all prokaryotic microorganisms are a single species or super organism (Sonea, 1988;

*Currently with the Computer Science and Artificial Intelligence Laboratory at the Massachusetts Institute of Technology, Cambridge, USA.
†Corresponding author

Margulis and Sagan, 2002), because of their ability to share genes. However, the analysis of ribosomal RNAs has shown that at least some molecular systems follow the tree-like pattern of relationships that is expected under predominantly vertical inheritance (Woese and Fox, 1977; Woese, 1987).

The problem of detecting horizontally transferred genes has been extensively studied; see, for example, Zhaxybayeva 2009 for a review. An important problem in understanding microbial evolution is to infer the HGT events (i.e., the donor and recipient of each HGT) that occurred during the evolution of a set of species. This problem is generally solved in a comparative-genomics framework by employing a parsimony criterion, based on the observation that the evolutionary history of horizontally transferred genes does not agree with the evolutionary history of the corresponding set of species. This is illustrated in Fig. 1(b). More formally, given a gene tree and a species tree, the *HGT inference problem* is to find the minimum number of HGT events that can explain the incongruence of the gene tree with the species tree. The HGT inference problem is known to be NP-hard under most formulations (Bordewich and Semple, 2005; Hallett and Lagergren, 2001; Hickey *et al.*, 2008) and, along with some of its variants, has been extensively studied (Hallett and Lagergren, 2001; Boc and Makarenkov, 2003; Nakhleh *et al.*, 2004, 2005; Beiko *et al.*, 2005; Than *et al.*, 2007; Jin *et al.*, 2009; Boc *et al.*, 2010; Hill *et al.*, 2010).

In general, one expects at most a few genes to have been horizontally transferred between any given pair of species. However, Beiko *et al.* (2005) demonstrated that some pairs of species portray a multitude of horizontal gene transfer events. Such pairs are said to be connected by a *highway of gene sharing* (Beiko *et al.*, 2005). Highways of gene sharing point towards major events in evolutionary history; well corroborated examples of this phenomenon are the uptake of endosymbionts into the eukaryotic host, and the many genes transferred from the symbiont to the hosts nuclear genome (Gary, 1993). Recent proposals for evolutionary events that may be reflected in highways of gene sharing are the role of Chlamydiae in establishing the primary plastid in the Archaeplastida (red and green algae, plants and glaucocystophytes) (Huang and Gogarten, 2007), and the evolution of double membrane bacteria through an endosymbiosis between clostridia and actinobacteria (Lake, 2009). Detecting these highways of gene sharing is thus an important biological problem and is crucial for inferring past symbiotic and ecological associations that shaped the evolution of organisms.

Given a rooted species tree, any two species (nodes) in it that are not related by an ancestor-descendant relationship define a *horizontal edge* connecting those two nodes. Any HGT event must take place along a horizontal edge in one of its two directions (see Fig. 1(a)). A horizontal edge along which an unusually large number of HGT events have taken place (say 10% of the genes) will be called a *highway of gene sharing* or simply a *highway*. The only existing method for detecting highways is the one employed originally by Beiko *et al.* (2005). That method takes as input a species tree and a set of gene (protein) trees, and computes, for each gene tree, the HGT events affecting that gene on the species tree. This is done by solving the HGT inference problem for each gene tree. The HGT events that are inferred in the HGT scenarios for a significant fraction of the gene trees are postulated as the highways. However, this approach suffers from several drawbacks. First, the HGT inference problem is NP-hard under most formulations, and thus, difficult to solve exactly (and must often be solved using heuristics). Second, there may be multiple (in fact, exponentially many) alternative optimal solutions to the HGT inference problem (Than *et al.*, 2007). And third, when the rate of HGT is relatively high, there is little reason to expect that the number of HGT events should be parsimonious; i.e., the HGT inference problem, even if solved exactly and yielding only one optimal solution, may not infer the actual HGT events. In this work we propose an alternative approach to detecting highways that does not rely on inferring individual HGT events. Moreover, our formulation allows exact solution of the problem in polynomial time. Our method thus avoids all of the aforementioned pitfalls.

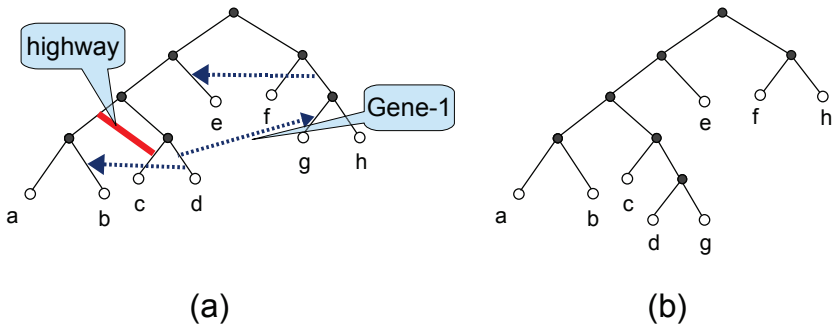



Figure 1: **Horizontal gene transfers and highways.** (a) A species tree depicting three HGT events (dotted arcs) and a highway (bold red horizontal edge). The highway represents many individual HGT events all occurring between the same two (present-day or ancestral) species. (b) The corresponding gene tree for Gene-1. Because of the HGT of Gene-1 from species *d* into species *g*, the copy of that gene in *g* is most closely related to the one in *d*. Therefore, in the tree for Gene-1, the species *g* appears next to *d*. (Here we assume that Gene-1 was not transferred on the highway as well.)

As in Beiko *et al.* 2005, the input to our method is a trusted rooted species tree for some set of species, and a set of gene trees on genes taken from those species. Since it is often difficult to accurately root gene trees, we assume that the input gene trees are unrooted. Our method is based on the observation that highways, by definition, affect the topologies of many gene trees. Thus, the idea is to combine the phylogenetic signals for HGT events from all the gene trees and use the combined signal to infer the highways, thereby avoiding the need to infer individual HGT events. We achieve this by employing a quartet decomposition of the gene trees. In particular, our method decomposes each gene tree into its constituent set of quartet trees and combines the quartet trees from all the gene trees to obtain a single weighted set of quartet trees. The intuition is that quartet trees that disagree with the species tree may indicate HGT events and thus the collective evidence from all quartet trees could pinpoint possible highways. The combined set of quartet trees is then analyzed against the given species tree to infer the highways of gene sharing. Decomposing the gene trees into quartet trees allows us to cleanly merge the phylogenetic signals for HGT events from all the different gene trees into a single summary signal, from which exact and efficient inference of the highways is possible.

To find highways, our method iteratively finds a horizontal edge that explains the largest fraction of inconsistent quartet trees. Essentially, for each (weighted) quartet tree inconsistent with the species tree, we identify the horizontal edges that can explain it by an HGT event (in either direction) along them. The horizontal edge that explains the most normalized inconsistency is proposed as a highway. (Normalization is needed since the structure of the species tree and the location of the horizontal edge in it influence the number of inconsistent quartet trees that edge may explain.) We give a dynamic programming algorithm that, given a weighted set $\Phi$ of quartet trees, computes the scores of all candidate highway edges (thereby finding the best highway) in $O(|\Phi| + n^2)$ time, where $n$ is the number of species in the species tree. Since there are $\Theta(n^2)$ candidate highway edges, our algorithm is asymptotically optimal with respect to the input and output size. In contrast, a naïve enumeration algorithm would require $O(|\Phi| \cdot n^2)$ time. Though $|\Phi|$ can be quite large (as large as $\Theta(n^4)$), our efficient algorithms allow our method to be applied to fairly large datasets; for example, we can analyze a dataset of 1000 gene trees with 200 taxa within a few hours on a personal computer (this includes the time required to compute the quartet decompositions for the 1000 gene trees). We demonstrate the utility of our method on simulated data as well as on a dataset of 1128 genes from

11 cyanobacterial species (Zhaxybayeva *et al.*, 2006), where its results match prior biological observations.

A preliminary version of this paper appeared in Bansal *et al.* 2010.

## 2 Basic Definitions and Preliminaries

Given a rooted or unrooted tree $T$, we denote its node set, edge set, and leaf set by $V(T)$, $E(T)$, and $\mathit{Le}(T)$ respectively. For the remainder of this paragraph, let $T$ denote a rooted tree. Given $T$, the root node of $T$ is denoted by $\mathit{rt}(T)$. Given a node $v \in V(T)$, we denote the parent of $v$ by $\mathit{pa}_T(v)$, its set of children by $\mathit{Ch}_T(v)$, and the (maximal) subtree of $T$ rooted at $v$ by $T(v)$. We define $\leq_T$ to be the partial order on $V(T)$ where $u \leq_T v$ if $v$ is a node on the path between $\mathit{rt}(T)$ and $u$. We say that $v$ is an *ancestor* of $u$, or that $u$ is a *descendant* of $v$, if $u \leq_T v$. Given a non-empty subset $L \subseteq \mathit{Le}(T)$, we denote by $\mathit{lca}_T(L)$ the least common ancestor (LCA) of all the leaves in $L$ in tree $T$; that is, $\mathit{lca}_T(L)$ is the unique smallest upper bound of $L$ under $\leq_T$. Throughout this work the term tree, rooted or unrooted, refers to a binary tree.

Given a rooted tree $T$, a *horizontal edge* on $T$ is a pair of nodes $\{u, v\}$, where $u, v \in V(T)$, such that $u, v \neq \mathit{rt}(T)$, $u \not\leq v$, $v \not\leq u$, and $\mathit{pa}_T(u) \neq \mathit{pa}_T(v)$. We denote by $H(T)$ the set of all horizontal edges on $T$. Horizontal edges represent potential horizontal gene transfer events; the (directed) horizontal edge $(u, v)$ represents the HGT event that transfers genetic material from the species represented by edge $(\mathit{pa}_T(u), u)$ to the species represented by edge $(\mathit{pa}_T(v), v)$. Thus, the horizontal edge $\{u, v\}$ represents the HGT events $(u, v)$ and $(v, u)$. Also note that, while any particular HGT event is directional, we address the problem in which horizontal edges are undirected because highways can be responsible for transfer of genetic material in both directions.

Our formulation and solution to the highway detection problem rely on the concept of quartets and quartet trees. A *quartet* is a four-element subset of some leaf set and a *quartet tree* is an unrooted tree whose leaf set is a quartet. The quartet tree with leaf set $\{a, b, c, d\}$ is denoted by $ab|cd$ if the path from $a$ to $b$ does not intersect the path from $c$ to $d$. Given a rooted or unrooted tree $T$, let $X$ be a subset of $\mathit{Le}(T)$ and let $T[X]$ denote the minimal subtree of $T$ having $X$ as its leaf set. We define the *restriction* of $T$ to $X$, denoted $T|X$, to be the unrooted tree obtained from $T[X]$ by suppressing all degree-two nodes (including the root, if $T$ is rooted). We say that a quartet tree $Q$ is *consistent* with a tree $T$ if $Q = T|\,\mathit{Le}(Q)$, otherwise $Q$ is *inconsistent* with $T$. Observe that, given any $T$ and any quartet $X = \{a, b, c, d\}$ from $\mathit{Le}(T)$, $X$ induces exactly one quartet tree in $T$, that is, the quartet tree $T|X$. Also observe that this quartet tree must have one of three possible topologies: $ab|cd$, $ac|bd$, or $ad|bc$.

## 3 Detecting Highways

Our goal is to detect the highways of gene sharing in the evolutionary history of a set of species $\mathbb{S}$. To that end, we are given a set of unrooted gene trees $\{T_1, \ldots, T_m\}$, and a rooted species tree $S$ showing the evolutionary history of $\mathbb{S}$. Thus, $\mathit{Le}(S) = \mathbb{S}$, and $\mathit{Le}(T_i) \subseteq \mathbb{S}$ for $1 \leq i \leq m$. The idea is to infer the highways by inspecting the differences in the topologies of the gene trees compared to the species tree. The *highway detection problem* can thus be stated as follows: Given a species tree $S$ and a collection of gene trees, find all the horizontal edges on $S$ that correspond to highways of gene sharing.

Throughout this manuscript, $S$ denotes the given species tree, and $n$ denotes the number of species in the analysis, i.e., $n = |\,\mathit{Le}(S)|$.

Our solution to the highway detection problem is based on decomposing each input gene tree $T$ into its constituent set of $\binom{|\mathit{Le}(T)|}{4}$ quartet trees, combining the quartet trees from the different gene trees into a single weighted set of quartet trees, and then comparing this set of quartet trees to the given species tree

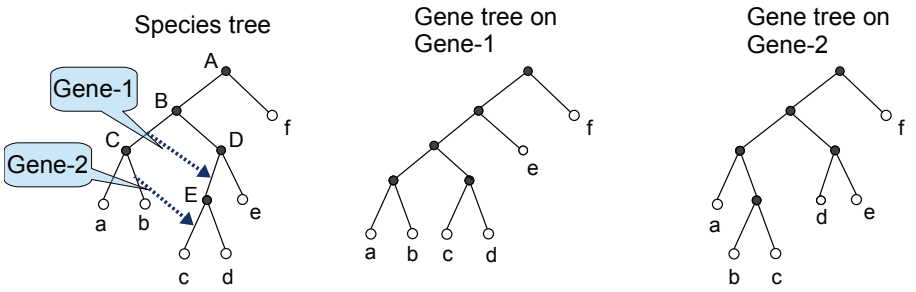



Figure 2: The tree on the left is a species tree showing the evolutionary history of a set of six species. Two HGT events $(C, E)$ and $(b, c)$, shown by the dotted arcs, are also depicted on this species tree. The two other trees show the evolutionary histories of Gene-1 and Gene-2.

to infer highways. To understand the intuition behind using quartet trees, consider the scenario depicted in Fig. 2. The figure on the left shows a species tree on six species, along with two HGT events of two different genes. Consider the HGT event $(C, E)$ that transfers Gene-1. This HGT event causes the topology of the gene tree constructed on Gene-1 to deviate from the topology of the species tree. Essentially, according to the standard subtree transfer model of horizontal gene transfer (see, e.g., Hein 1990; Beiko *et al.* 2005; Hill *et al.* 2010), this HGT event causes the subtree rooted at node $E$ to be pruned and then regrafted along the edge $(B, C)$, as shown in the figure. Let us decompose both trees into their constituent set of quartet trees: Each tree generates $\binom{6}{4} = 15$ quartet trees. Among these, most of the quartet trees are the same in both the species tree and the gene tree. In particular, only four of the fifteen quartets induce different quartet trees in the two trees; in the gene tree, these appear as $ac|ef$, $ad|ef$, $bc|ef$ and $bd|ef$. Different HGT events produce gene trees with different sets of inconsistent quartet trees. Thus, given the species tree, and the set of the four inconsistent quartet trees from the gene tree on Gene-1, we could have precisely inferred the HGT event $(C, E)$ that affected Gene-1.

Decomposing the gene trees into quartet trees allows us to cleanly combine the phylogenetic signal for HGT events from all the different gene trees into a single analysis, which is desirable since highways, by definition, affect the topologies of a large number of gene trees. Under our quartet based model, we say that a horizontal edge is a *highway* if its two HGT events together explain a disproportionately large number of inconsistent quartet trees. The highway detection problem thus becomes the problem of finding such horizontal edges.

### 3.1 The Method in Detail

Our method proceeds iteratively, inferring one highway per iteration, as follows.

**Step 1:** Decompose each input gene tree $T$ into its constituent set of $\binom{|Le(T)|}{4}$ quartet trees.

**Step 2:** Combine the quartet trees from the different gene trees into a single weighted set, $\Phi$, of quartet trees. Note that, since each quartet can have at most three different quartet trees, the number of quartet trees in this weighted set is at most $3 \cdot \binom{n}{4}$.

**Step 3:** Remove from $\Phi$ all those quartet trees that are consistent with $S$.

**Step 4:** Compute the HGT score of each edge in $H(S)$. This HGT score for an edge is computed based on $\Phi$, and is explained in detail below.

**Step 5:** Select the highest scoring horizontal edge as a highway.

**Step 6:** Remove from $\Phi$ all those quartet trees that are explained by the proposed horizontal edge.

**Step 7:** Go to Step 4 to start the next iteration.

The (raw) HGT score of a horizontal edge is simply the total weight of the quartets from $\Phi$ that are explained by a HGT along that edge (in either direction). Thus, this raw score of a horizontal edge captures the number of quartet trees from the input gene trees that support horizontal gene transfer along that edge. However, not all horizontal gene transfers affect the same number of quartets. Consider the example shown in Fig. 2. As seen previously, the HGT event $(C, E)$ causes four of the quartet trees in the corresponding gene tree to become inconsistent. Consider the HGT event $(b, c)$ that transfers Gene-2. This HGT event causes ten of the quartet trees in the gene tree built on Gene-2 (shown on the right in Fig. 2) to become inconsistent; these are $ad|bc$, $ae|bc$, $af|bc$, $ac|de$, $ac|df$, $ac|ef$, $bc|de$, $bc|df$, $bc|ef$ and $de|cf$. Thus, considering only the raw scores of the horizontal edges would lead to overestimation of the quantity of HGT along certain horizontal edges and underestimation of this quantity for other horizontal edges, leading to incorrect inference of highways.

To overcome this bias we modify the score of each horizontal edge by dividing its raw score by a normalization factor: The maximum number of distinct quartet trees that could be explained by a horizontal gene transfer (in either direction) along that edge. More precisely, let $\Psi$ be the set of all possible quartet trees on the leaf set $Le(S)$. Given a horizontal edge $\{u, v\}$, let $Q_1$ denote the set of quartet trees in $\Psi$ that become consistent due to the HGT event $(u, v)$, and let $Q_2$ denote the set of quartet trees in $\Psi$ that become consistent due to the HGT event $(v, u)$. The normalization factor for $\{u, v\}$ is defined to be $|Q_1 \cup Q_2|$. After normalization, the HGT scores of all horizontal edges can be directly compared to one another. In general, not all the gene trees will represent all the species considered in the analysis and this may lead to overestimation of the normalization factor for some horizontal edges. However, when analyzing HGTs, it is common to only include those gene trees that have genes from most of the taxa (say at least 75%) considered in the analysis, and this normalization scheme can be expected to yield accurate results on such datasets.

The number of iterations in the method can either be fixed at the beginning or, preferably, be decided on the fly, based on the distribution of the horizontal edge scores computed in the current iteration. In that case, the algorithm would terminate when none of the horizontal edges show a significant score. When using the algorithm iteratively, we should also check that the new suggested highway is time consistent with the previous ones.

### 3.2 The Basic Computational Problems

This iterative quartet based method involves four computational steps: (i) Computing the initial set of weighted quartet trees from the gene trees, (ii) removing the quartet trees that are consistent with $S$, (iii) computing the (normalized) HGT score of each edge in $H(S)$, and (iv) identifying and removing those quartet trees that are explained by the proposed highway. The main computational challenge here is (iii), in which we must compute the (normalized) HGT score of each horizontal edge. Next, we first briefly describe how to efficiently solve problems (i), (ii) and (iv), and then focus on the main problem.

**Computing the weighted set of quartet trees.** The goal here is to decompose each of the input gene trees into its constituent set of quartet trees, and then combine these sets into a single weighted set of quartet trees. We note that several quartet based phylogeny inference/analysis methods rely on decomposing a tree into its constituent set of quartet trees (see, e.g., Piaggio-Talice *et al.* 2004; Zhaxybayeva *et al.* 2006). Though

it is "folklore" that the problem of quartet decomposition can be solved in $O(n^4)$ time, this result has, to our knowledge, never been formally established. For the sake of completeness, here we fill this gap. In particular, we will show how to generate all the quartet trees for any gene tree $T$ in a predefined (e.g., lexicographical) order within $O(|Le(T)|^4)$ time. Since the quartet trees are generated in a predefined order, the quartet trees from different gene trees can all be combined together in linear time, yielding the $O(tn^4)$ overall time bound, where $t$ is the number of input gene trees. We rely on the following lemma.

**Lemma 3.1.** *Given an unrooted tree $T$ we can determine, after an initial $O(|\,Le(T)|)$ preprocessing step, whether any given quartet tree $Q$ on the leaf set of $T$ is consistent with $T$ within $O(1)$ time.*

*Proof.* Let $T'$ be the rooted tree obtained from $T$ by rooting it along any arbitrary edge. We first preprocess $T'$ so that, given any two nodes from $V(T')$, we can compute their LCA within $O(1)$ time (Bender and Farach-Colton, 2000). This preprocessing step takes $O(|Le(T)|)$ time, and also allows us to label the nodes of $T'$ in such a way that given any two nodes $u, v \in V(T')$ we can check if $v \in V(T'(u))$ in $O(1)$ time. To accomplish this, we first perform an in-order traversal of $T'$ and label all the nodes by increasing numbers in the order in which they are seen. Next, we perform a post-order traversal of $T'$ to associate a *start* and an *end* value to each node. These start and end values at a node are, respectively, the smallest and largest labels that occur in the subtree rooted at that node. It is easy to verify that, given any $u, v \in V(T')$, we can now check if $v$ is in the subtree rooted at $T'(u)$ simply by checking if the label of $v$ lies between the start and end values at $u$.

Given a quartet tree $Q = ab|cd$, let $X = lca_{T'}(a, b)$ and $Y = lca_{T'}(c, d)$. We claim that $Q$ is consistent with $T$ if and only if either $c, d \notin Le(T'(X))$ or $a, b \notin Le(T'(Y))$. To prove this claim, consider the following two cases.

**$Q$ is consistent with $T$:** Let $E$ and $F$ denote the internal nodes of $Q$ such that $E$ is on the path from $a$ to $b$, and $F$ is on the path from $c$ to $d$. Let $T'_Q$ denote the tree $T'[Le(Q)]$. Consider the embedding of $Q$ in $T'_Q$. The root of $T'_Q$ must appear along one of the following five paths: the path from (i) $a$ to $E$, (ii) $b$ to $E$, (iii) $E$ to $F$, (iv) $c$ to $F$, or (v) $d$ to $F$. In cases (i) and (ii), the node $Y$ will correspond to node $F$, and the condition $a, b \notin Le(T'(Y))$ must be satisfied. In case (iii), the nodes $X$ and $Y$ must correspond to nodes $E$ and $F$ respectively, and both the conditions $c, d \notin Le(T'(X))$ and $a, b \notin Le(T'(Y))$ are satisfied. In cases (iv) and (v), the node $X$ will correspond to node $E$, and the condition $c, d \notin Le(T'(X))$ must be satisfied. Thus, if $Q$ is consistent with $T$ then at least one of $c, d \notin Le(T'(X))$ or $a, b \notin Le(T'(Y))$ must hold true.

**$Q$ is not consistent with $T$:** Without any loss of generality, assume that $Q' = ac|bd$ is the corresponding quartet tree in $T$. Let $E$ and $F$ denote the internal nodes of $Q'$ such that $E$ is on the path from $a$ to $c$, and $F$ is on the path from $b$ to $d$. Let $T'_{Q'}$ denote the tree $T'[Le(Q')]$. Consider the embedding of $Q'$ in $T'_{Q'}$. The root of $T'_{Q'}$ must appear along one of the following five paths: the path from (i) $a$ to $E$, (ii) $c$ to $E$, (iii) $E$ to $F$, (iv) $b$ to $F$, or (v) $d$ to $F$. In case (i), we must have $b \in Le(T'(Y))$ and $c, d \in Le(T'(X))$. In case (ii), we must have $a, b \in Le(T'(Y))$ and $d \in Le(T'(X))$. In case (iii), we must have $a, b \in Le(T'(Y))$ and $c, d \in Le(T'(X))$. In case (iv), we must have $a \in Le(T'(Y))$ and $c, d \in Le(T'(X))$. In case (v), we must have $a, b \in Le(T'(Y))$ and $c \in Le(T'(X))$. Thus, if $Q$ is not consistent with $T$ then neither of the two conditions $c, d \notin Le(T'(X))$ or $a, b \notin Le(T'(Y))$ can hold true.

Since, after the preprocessing steps, both the conditions $c, d \notin Le(T'(X))$ and $a, b \notin Le(T'(Y))$ can be tested in $O(1)$ time, our proof is complete. □

Lemma 3.1 makes it easy to generate the set of constituent quartet trees for any gene tree $T$. First, we generate all the $3 \cdot \binom{|Le(T)|}{4}$ possible quartet trees on the leaf set of $T$ in some predefined order and then, after the $O(|Le(T)|)$-time preprocessing step, check which of these quartet trees are consistent with $T$. In this way, we can count for each quartet tree the number of gene trees it is consistent with in $O(mn^4)$ time.

**Removing consistent quartet trees.** This can be accomplished in $O(n^4)$ time by simply considering each quartet tree in $\Phi$ separately and using the result of Lemma 3.1 to check if it is consistent with $S$.

**Checking which quartet trees are explained by a proposed horizontal edge.** This step can be executed in $O(n^4)$ time as follows. Suppose the proposed horizontal edge is $\{u, v\}$. This edge represents the HGT events $(u, v)$ and $(v, u)$. Our goal is to identify those quartet trees in $\Phi$ that can be explained by at least one of these two HGT events. To accomplish this, construct a variant $S'$ of $S$ obtained by pruning the subtree rooted at $v$ and regrafting it at the edge $(pa(u), u)$ (this models the HGT event $(u, v)$). Now consider each quartet tree in $\Phi$ separately and use the result of Lemma 3.1 to check if it is consistent with $S'$. Remove all the consistent quartet trees from $\Phi$. Similarly, construct a second variant $S''$ of $S$ obtained by pruning the subtree rooted at $u$ and regrafting it at the edge $(pa(v), v)$ (this models the HGT event $(v, u)$). As before, consider each quartet tree in $\Phi$ separately, use the result of Lemma 3.1 to check if it is consistent with $S''$, and remove all the consistent quartet trees from $\Phi$.

Next, we study the main computational problem of our method, i.e., the highway scoring problem.

# 4 The Highway Scoring Problem

The highway scoring problem can be formally stated as follows:

**Problem 1.** *Given a rooted species tree $S$ and a set $\Phi$ of weighted quartet trees (that are inconsistent with $S$) on the leaf set $Le(S)$, the* Highway Scoring (HS) problem *is to find the (normalized) HGT score of each edge in $H(S)$.*

The naïve way to solve the HS problem would be to consider each edge in $H(S)$ one-at-a-time and to check which of the quartet trees from $\Phi$ are explained by that edge. As shown above, checking whether a quartet tree is explained by a horizonal edge can be accomplished in $O(1)$ time. Since there are $\Theta(n^2)$ candidate horizontal edges the complexity of computing just the raw score of each horizontal edge is still $O(|\Phi| \cdot n^2)$. In this section we show how to solve the HS problem in $O(|\Phi| + n^2)$ time. The time complexity of our algorithm is thus optimal.

**Notation.** Recall that each horizontal edge actually represents two HGT events. We denote the set of all these HGT events on $S$ by $\overrightarrow{H}(S)$. Thus, for any horizontal edge $\{u, v\} \in H(S)$, there are two HGT events $(u, v)$ and $(v, u)$ in $\overrightarrow{H}(S)$.

Given a horizontal edge $\{u, v\}$, if $Q_1$ and $Q_2$ denote the sets of quartet trees that are explained by the HGT events $(u, v)$ and $(v, u)$ respectively, then, the raw score of $\{u, v\}$ is $|Q_1 \cup Q_2|$, which is $|Q_1| + |Q_2| - |Q_1 \cap Q_2|$. First, in Section 4.1, we show how to compute the raw score of each horizontal event (i.e., how to compute $|Q_1|$ and $|Q_2|$), and then, in Section 4.2, we show how to compute $|Q_1 \cap Q_2|$ and thus obtain the raw scores of horizontal edges. Finally, in Section 4.3, we show how to efficiently compute the normalization factor for each horizontal edge.

## 4.1 Computing the Raw Scores of HGT Events

For any given quartet tree $Q \in \Phi$, there may be several HGT events from $\overrightarrow{H}(S)$ that could explain $Q$; we denote this set of HGT events by $\overrightarrow{H}(S, Q)$. Since $S$ is fixed, throughout the remainder of this work we will

abbreviate $H(S)$, $\overrightarrow{H}(S)$ and $\overrightarrow{H}(S, Q)$ to $H$, $\overrightarrow{H}$ and $\overrightarrow{H}(Q)$ respectively. Our algorithm relies on an efficient characterization of the HGT events that can explain a given quartet. This characterization appears in the next two lemmas; but first, we need some additional definitions and notation.

**Notation and Definitions.** We denote the raw score of an HGT event $(u, v) \in \overrightarrow{H}$ by $RS(u, v)$. Given any two nodes $p, q \in V(S)$, let $p \rightarrow q$ denote the path between them in $S$, and let $V(p \rightarrow q)$ denote the set of nodes on this path (including $p$ and $q$). A *subtree-path (SP) pair* on $S$ is a pair $\langle S(v), p \rightarrow q \rangle$, where $v, p, q \in V(S)$, such that the subtree $S(v)$ and the path $p \rightarrow q$ are node disjoint and none of the nodes in $p \rightarrow q$ is an ancestor or descendant of $v$. Given an SP pair $\sigma = \langle S(v), p \rightarrow q \rangle$, the set of all HGT events $(u, v)$ from $\overrightarrow{H}$ such that $u \in S(v)$ and $v \in V(p \rightarrow q)$ is denoted by $\overrightarrow{H}(\sigma)$. Similarly, a *subtree-complement-path (SCP) pair* on $S$ is a pair $\langle S(v), p \rightarrow q \rangle$, where $v, p, q \in V(S)$, such that $V(p \rightarrow q) \subseteq V(S(v))$. We define $\overline{V}(S(v))$ to be the set $[V(S) \setminus V(S(v))] \cup \{v\}$. Given an SCP pair $\sigma = \langle S(v), p \rightarrow q \rangle$, the set of all HGT events $(u, v)$ from $\overrightarrow{H}$ such that $u \in \overline{V}(S(v))$ and $v \in V(p \rightarrow q)$ is denoted, as before, by $\overrightarrow{H}(\sigma)$. If $\sigma$ is an SCP pair, then we say that $S(v)$ is the *subtree-complement* of $\sigma$, and it refers to the subtree of $S$ induced by $\overline{V}(S(v))$.

**Type I and Type II quartet trees.** Let $Q = ab|cd$ be any quartet tree from $\Phi$ and, without loss of generality, assume that the corresponding quartet tree in $S$ is $Q' = ac|bd$. We will label $Q$ as either a Type I or Type II quartet tree based on how $Q'$ in embedded in $S$. This is done as follows: Let $E$ and $F$ denote the internal nodes of $Q'$ such that $E$ is on the path from $a$ to $c$, and $F$ is on the path from $b$ to $d$. Let $S_{Q'}$ denote the tree $S[Le(Q')]$. Consider the embedding of $Q'$ in $S_{Q'}$. Note that the root of $S_{Q'}$ must appear along one of the following five paths: the path from (i) $a$ to $E$, (ii) $c$ to $E$, (iii) $E$ to $F$, (iv) $b$ to $F$, or (v) $d$ to $F$. If the root of $S_{Q'}$ appears along the path from $E$ to $F$ then we say that $Q$ is a *Type I* quartet tree with respect to $S$, otherwise we say that $Q$ is a *Type II* quartet tree with respect to $S$. An example is depicted in Fig. 3. Since $S$ is fixed, we can label each quartet tree from $\Phi$ directly as a Type I or Type II quartet tree.

**Lemma 4.1** (Characterization of HGT events for Type I quartet trees)**.** *Given any Type I quartet tree $Q \in \Phi$, there exist four SP pairs, denoted $\sigma_1, \sigma_2, \sigma_3, \sigma_4$, such that $\overrightarrow{H}(Q) = \overrightarrow{H}(\sigma_1) \cup \overrightarrow{H}(\sigma_2) \cup \overrightarrow{H}(\sigma_3) \cup \overrightarrow{H}(\sigma_4)$. Moreover, the four sets $\overrightarrow{H}(\sigma_1), \overrightarrow{H}(\sigma_2), \overrightarrow{H}(\sigma_3)$ and $\overrightarrow{H}(\sigma_4)$ are pairwise disjoint.*

*Proof.* We continue with the notation of the preceding paragraph. Suppose $Q = ab|cd$ is of Type I, i.e., the root of $S_{Q'}$ appears along the path from $E$ to $F$. Let $A$ denote the child of $E$ whose subtree contains $a$, $C$ denote the child of $E$ in $S_{Q'}$ whose subtree contains $c$, $B$ denote the child of $F$ whose subtree contains $b$, and $D$ denote the child of $F$ whose subtree contains $d$. See the first species tree in Fig. 3 for an example. We define the four SP pairs as follows: $\sigma_1 = \langle S(A), B \rightarrow b \rangle$, $\sigma_2 = \langle S(B), A \rightarrow a \rangle$, $\sigma_3 = \langle S(C), D \rightarrow d \rangle$, and $\sigma_4 = \langle S(D), C \rightarrow c \rangle$. It is straightforward to verify that the four sets of HGT events $\overrightarrow{H}(\sigma_1), \overrightarrow{H}(\sigma_2), \overrightarrow{H}(\sigma_3), \overrightarrow{H}(\sigma_4)$ are pairwise disjoint and that each of them is a subset of $\overrightarrow{H}(Q)$.

We will now show that there does not exist any HGT event in $\overrightarrow{H}(Q)$ that does not appear in any of these four sets. Consider any HGT event $(u, v) \in \overrightarrow{H}(Q)$. Observe that the node $v$ must be such that $|V(S(v)) \cap \{a, b, c, d\}| = 1$. This is because if $|V(S(v)) \cap \{a, b, c, d\}| = 0$ then this HGT event does not affect the embedding of the quartet tree $Q$ in the resulting gene tree at all, and if $|V(S(v)) \cap \{a, b, c, d\}| > 1$ then this HGT event yields a gene tree that remains consistent with $Q'$. Thus $v$ must be a node on one of the paths $A \rightarrow a$, $B \rightarrow b$, $C \rightarrow c$, or $D \rightarrow d$. Suppose $v \in A \rightarrow a$ (the other cases are symmetric). In order for the resulting gene tree to be consistent with $Q$, the path from $a$ to $b$ in this gene tree must not intersect the path from $c$ to $d$. This means that the node $u$ must be such that either $V(S(lca(b, u))) \cap \{c, d\} = \emptyset$ or $V(S(lca(c, d))) \cap \{b, u\} = \emptyset$ (or both). Since $b \in V(S(lca(c, d)))$, we must have $V(S(lca(b, u))) \cap \{c, d\} = \emptyset$. Thus, $u$ must lie in the subtree $S(B)$, i.e., $(u, v) \in \overrightarrow{H}(\sigma_2)$. In summary, any HGT event $(u, v) \in \overrightarrow{H}(Q)$ must be such that $(u, v) \in \overrightarrow{H}(\sigma_1) \cup \overrightarrow{H}(\sigma_2) \cup \overrightarrow{H}(\sigma_3) \cup \overrightarrow{H}(\sigma_4)$. □

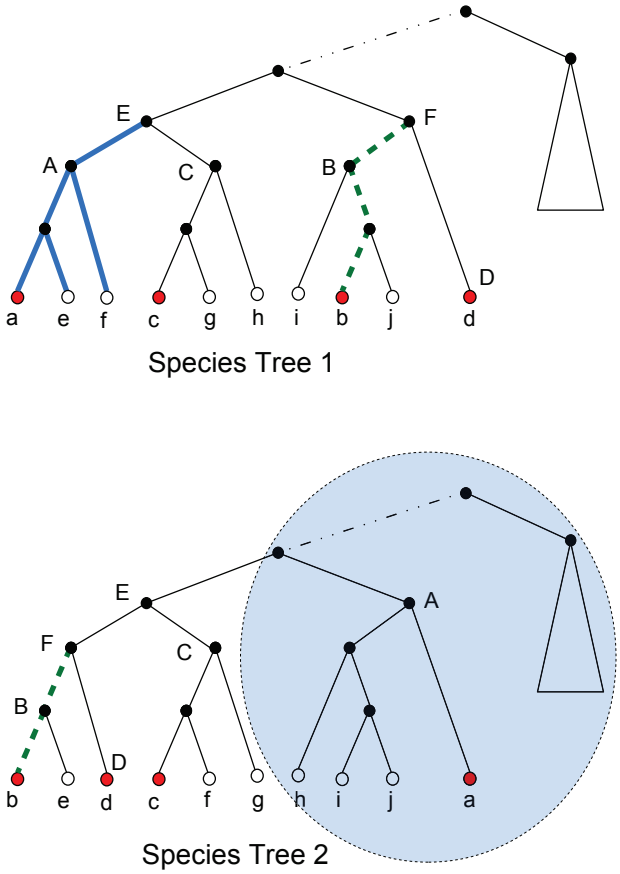



Figure 3: **Type I and Type II quartets, and SP and SCP pairs.** Consider a quartet tree $Q = ab|cd$ from $\Phi$. Then, $Q$ is a type I quartet tree with respect to the first species tree, and a type II quartet tree with respect to the second. In the first species tree, any HGT event that originates at a bold (blue) edge and ends at a dashed (bold green) edge can explain the quartet tree $Q$. This set of HGT events is represented by the SP pair $\langle S(A), B \to b\rangle$. The other three SP pairs for $Q$ on the first species tree are $\langle S(B), A \to a\rangle$, $\langle S(C), D \to d\rangle$, and $\langle S(D), C \to c\rangle$. Similarly, in the second species tree any HGT event that originates at any edge in the shaded region and ends at a dashed (bold green) edge can explain the quartet tree $Q$. This set of HGT events is represented by the SCP pair $\langle S(E), B \to b\rangle$. The three SP pairs for $Q$ on the second species tree are $\langle S(B), A \to a\rangle$, $\langle S(C), D \to d\rangle$, and $\langle S(D), C \to c\rangle$

**Lemma 4.2** (Characterization of HGT events for Type II quartet trees)**.** *Given any Type II quartet tree $Q \in \Phi$, there exist three SP pairs, denoted $\sigma_1, \sigma_2, \sigma_3$, and one SCP pair, denoted $\sigma_4$, such that $\overrightarrow{H}(Q) = \overrightarrow{H}(\sigma_1) \cup \overrightarrow{H}(\sigma_2) \cup \overrightarrow{H}(\sigma_3) \cup \overrightarrow{H}(\sigma_4)$. Moreover, the four sets $\overrightarrow{H}(\sigma_1), \overrightarrow{H}(\sigma_2), \overrightarrow{H}(\sigma_3)$ and $\overrightarrow{H}(\sigma_4)$ are pairwise disjoint.*

*Proof.* We reuse the notation from the paragraph preceding Lemma 4.1 and assume now that $Q = ab|cd$ is of Type II, i.e., the root of $S_{Q'}$ must appear along one of the following four paths: the path from (i) $a$ to $E$, (ii) $c$ to $E$, (iii) $b$ to $F$, or (iv) $d$ to $F$.

We prove the lemma for case (i); the proofs for the other cases are analogous. Let $A$ denote the child of the root of $S_{Q'}$ whose subtree contains $a$, $C$ denote the child of $E$ in $S_{Q'}$ whose subtree contains $c$, $B$ denote the child of $F$ whose subtree contains $b$, and $D$ denote the child of $F$ whose subtree contains $d$. We define the three SP pairs as follows: $\sigma_1 = \langle S(B), A \rightarrow a \rangle$, $\sigma_2 = \langle S(C), D \rightarrow d \rangle$, and $\sigma_3 = \langle S(D), C \rightarrow c \rangle$. The SCP pair $\sigma_4$ is defined to be $\langle S(E), B \rightarrow b \rangle$. It is straightforward to verify that the four sets of HGT events $\overrightarrow{H}(\sigma_1), \overrightarrow{H}(\sigma_2), \overrightarrow{H}(\sigma_3), \overrightarrow{H}(\sigma_4)$ are pairwise disjoint and that each of them is a subset of $\overrightarrow{H}(Q)$. See the second species tree in Fig. 3 for an example.

We will now show that there does not exist any HGT event in $\overrightarrow{H}(Q)$ that does not appear in any of these four sets. Consider any HGT event $(u, v) \in \overrightarrow{H}(Q)$. As in the proof of Lemma 4.1, node $v$ must be on one of the paths $A \rightarrow a$, $B \rightarrow b$, $C \rightarrow c$, or $D \rightarrow d$. Suppose $v \in A \rightarrow a$. In order for the resulting gene tree to be consistent with $Q$, the path from $a$ to $b$ in this gene tree must not intersect the path from $c$ to $d$. This means that the node $u$ must be such that $V(S(lca(b, u))) \cap \{c, d\} = \emptyset$. Thus, $u$ must lie in the subtree $S(B)$, i.e., $(u, v) \in \overrightarrow{H}(\sigma_1)$. The cases when $v \in C \rightarrow c$, and $v \in D \rightarrow d$ are analogous and correspond to $\sigma_2$ and $\sigma_3$ respectively. Now consider the case when $v \in B \rightarrow b$. To ensure that the path from $a$ to $b$ in the resulting gene tree does not intersect the path from $c$ to $d$, we must have either $V(S(lca(a, u))) \cap \{c, d\} = \emptyset$ or $V(S((lca(c, d)))) \cap \{a, u\} = \emptyset$ (or both). To get $V(S(lca(a, u))) \cap \{c, d\} = \emptyset$, $u$ must lie in the subtree $S(A)$, and to get $V(S((lca(c, d)))) \cap \{a, u\} = \emptyset$, we must have $u \in \overline{V}(S(E))$. In either case, $(u, v) \in \overrightarrow{H}(\sigma_4)$. In summary, any HGT event $(u, v) \in \overrightarrow{H}(Q)$ must be such that $(u, v) \in \overrightarrow{H}(\sigma_1) \cup \overrightarrow{H}(\sigma_2) \cup \overrightarrow{H}(\sigma_3) \cup \overrightarrow{H}(\sigma_4)$. □

Note that the path in any SP/SCP pair is monotone in the tree $S$, and in particular contains at most one node from any level of $S$. From the constructive proofs of Lemmas 4.1 and 4.2, the following corollary follows immediately.

**Corollary 4.1.** *For every quartet tree $Q \in \Phi$, (i) any two of the subtrees/subtree-complements from its four SP/SCP pairs are disjoint, and (ii) any two of the paths from its four SP/SCP pairs are disjoint.*

Our algorithm performs a nested tree traversal of $S$. Before we begin this nested tree traversal we (i) perform a pre-processing step, which precomputes certain values on the tree $S$, and (ii) perform a tree decoration step during which we decorate the nodes of $S$ with information about the four SP/SCP pairs for each quartet tree in $\Phi$. Next we describe these two steps in detail, and then proceed to describe the nested tree traversal procedure.

**The preprocessing step.** The first step in the algorithm is to preprocess the tree $S$ so that, given any two nodes from $V(S)$, we can compute their LCA within $O(1)$ time (Bender and Farach-Colton, 2000). This preprocessing step also allows us to label the nodes of $S$ in such a way that given any two nodes $u, v \in V(S)$ we can check if $v \in V(S(u))$ in $O(1)$ time (see the previous section for a description on how to do this efficiently). We also associate with each $v \in V(S)$ a counter, denoted by $counter_v$, initialized to zero, and a set $path_v$ initialized to be empty.

**Decorating the tree.** The tree decoration step marks, on the tree $S$, the endpoints of the four paths in the SP/SCP pairs of any quartet. This is done by executing the following procedure.

**Procedure** *Decorate*$(\Phi, S)$

**for** each quartet tree $Q \in \Phi$ **do**
  Using Lemmas 4.1 and 4.2, compute its four SP/SCP pairs $\sigma_1 = \langle S(v_1), p_1 \rightarrow q_1 \rangle$, $\sigma_2 = \langle S(v_2), p_2 \rightarrow q_2 \rangle$, $\sigma_3 = \langle S(v_3), p_3 \rightarrow q_3 \rangle$, and $\sigma_4 = \langle S(v_4), p_4 \rightarrow q_4 \rangle$. (Note that by convention the $q_i$s denote the leaf-node end points of the four paths.)
  **for** each $i \in \{1, 2, 3, 4\}$ **do**
    **if** $\sigma_i$ is an SP pair **then**
      Add the triple $(Q, v_i, SP)$ to the sets $path_{q_i}$ and $path_{pa(p_i)}$.
    **if** $\sigma_i$ is an SCP pair **then**
      Add the triple $(Q, v_i, SCP)$ to the sets $path_{q_i}$ and $path_{pa(p_i)}$.[1]

Note that, in the procedure above, each quartet appears in at most eight path sets on $S$. Our algorithm performs a post-order traversal of $S$ and, at each node $v$, calls the procedure *Augment*$(v)$ described below. This procedure marks the corresponding subtrees/subtree-complements for all the paths that appear in the set $path_v$, and computes a value $val_u$ at each $u \in V(S) \setminus \{rt(S)\}$. This value $val_u$ is the weight of all quartet trees $Q$ from $\Phi$ such that (i) $(Q, x, \Gamma) \in path_v$ and (ii) if $\Gamma$ is SP then $u \in V(S(x))$, and, if $\Gamma$ is SCP then $u \in \overline{V}(S(x))$. The reason for computing these $val_u$'s becomes clear in the context of Lemma 4.3.

**Procedure** *Augment*$(v)$ $\quad \{v \in V(S)\}$

**for** each $x \in V(S)$ **do**
  Set $counter_x$ to 0.
**for** each triple $(Q, y, \Gamma) \in path_v$ **do**
  **if** $\Gamma$ is SP **then**
    Increment $counter_y$ by the weight of $Q$.
  **if** $\Gamma$ is SCP **then**
    Increment $counter_{rt(S)}$ by the weight of $Q$.
    Decrement $counter_{y_1}$ and $counter_{y_2}$ by the weight of $Q$, where $\{y_1, y_2\} = Ch(y)$.
**for** each $u \in V(S) \setminus \{rt(S)\}$ **do**
  Set $val_u$ to $\sum_{x \in V(rt(S) \rightarrow u)} counter_x$.

**Observation 1.** *In the last "for" loop of procedure Augment*$(v)$*, a triple* $(Q, y, \Gamma)$ *from* $path_v$ *is counted in* $val_u$ *exactly when (i)* $\Gamma$ *is SP and* $u \in V(S(y))$*, or (ii)* $\Gamma$ *is SCP and* $u \in \overline{V}(S(y))$*.*

Our algorithm is based on the following key lemma.

**Lemma 4.3.** *Suppose* $S$ *has been decorated and procedure Augment*$(v)$ *has been executed for some* $v \in V(S)$*. Consider any* $(u, v) \in \overrightarrow{H}$*.*

1. *If* $v \in Le(S)$*, then* $RS(u, v) = val_u$*.*
2. *If* $v \notin Le(S)$*, then* $RS(u, v) = RS(u, v_1) + RS(u, v_2) - val_u$*, where* $\{v_1, v_2\} = Ch(v)$*.*

*Proof.* $v \in Le(S)$: Let $Q$ be any quartet tree that is explained by the HGT event $(u, v)$. Then, $Q$ must have an SP/SCP pair, say $\sigma = \langle S(x), p \rightarrow q \rangle$, such that $q = v$ and, if $\sigma$ is an SP pair then $u \in V(S(x))$ or if $\sigma$ is an SCP pair then $u \in \overline{V}(S(x))$. This implies that the set $path_v$ contains the triple $(Q, x, SP/SCP)$.

[1] We include SP/SCP in these triples to indicate whether the triple corresponds to an SP pair or to an SCP pair.

Thus, (the weight of) $Q$ is counted at least once in $val_u$. Now, from Corollary 4.1, we know that the subtrees/subtree-complements from any two SP/SCP pairs must be node disjoint. Thus, the quartet tree $Q$ is, in fact, counted exactly once in $val_u$. Finally, observe that if some quartet tree $Q'$ is counted in $val_u$, then $Q'$ must have an SP/SCP pair $\sigma = \langle S(x), p \rightarrow q \rangle$ such that $q = v$ and, if $\sigma$ is an SP pair then $u \in V(S(x))$ or if $\sigma$ is an SCP pair then $u \in \overline{V}(S(x))$; consequently, by Lemmas 4.1 and 4.2, the HGT event $(u, v)$ must indeed explain the quartet tree $Q'$. Thus, $RS(u, v) = val_u$.

$v \notin Le(S)$: Let $Q$ be a quartet tree from $\Phi$ and $\{v_1, v_2\} = Ch(v)$. Observe that at most one of the edges $(v, v_1)$ and $(v, v_2)$ may be present on the path of any single SP/SCP pair of $Q$ (since all paths are monotone). Corollary 4.1 therefore implies that $Q$ may be counted in at most one of $RS(u, v_1)$ and $RS(u, v_1)$ (since only one of the subtrees/subtree-complements from any two different SP/SCP pairs of $Q$ may contain the node $u$). Suppose $Q$ is explained by the HGT event $(u, v)$. Then, $Q$ must have an SP/SCP pair, say $\sigma = \langle S(x), p \rightarrow q \rangle$, such that $q \leq_S v \leq_S p$, and, if $\sigma$ is an SP pair then $u \in V(S(x))$ or if $\sigma$ is an SCP pair then $u \in \overline{V}(S(x))$. In other words, $Q$ must have been counted in one of $RS(u, v_1)$ or $RS(u, v_2)$. And, since $v \leq_S p$, the set $path_v$ does not contain the entry $(Q, x, SP/SCP)$. Also, from Corollary 4.1, we know that the node $u$ may occur in the subtree/subtree-complement of at most one SP/SCP pair of $Q$. Thus, $Q$ cannot be counted in the value $val_u$ and, consequently, $Q$ is counted exactly once in the value $RS(u, v_1) + RS(u, v_2) - val_u$.

Now, suppose that $Q$ is not explained by the HGT event $(u, v)$. There are two possible cases: (i) $Q$ is counted in one of $RS(u, v_1)$ or $RS(u, v_2)$, say $RS(u, v_1)$, or (ii) $Q$ is counted neither in $RS(u, v_1)$ nor in $RS(u, v_2)$. Consider case (i). Since $Q$ is satisfied by the HGT event $(u, v_1)$, but not by $(u, v)$, $Q$ must have an SP/SCP pair $\sigma = \langle S(x), p \rightarrow q \rangle$, such that $v_1 = p$, $q \in S(v_1)$ and, if $\sigma$ is an SP pair then $u \in V(S(x))$ or if $\sigma$ is an SCP pair then $u \in \overline{V}(S(x))$. Thus, since $v = pa(p)$, the set $path_v$ must contain the entry $(Q, x, SP/SCP)$. This implies that $Q$ is counted at least once in the value $val_u$. Now, from Corollary 4.1, we know that the subtrees/subtree-complements from any two SP/SCP pairs must be node disjoint. Thus, the quartet tree $Q$ is, in fact, counted exactly once in $val_u$. Consequently, $Q$ does not affect the value $RS(u, v_1) + RS(u, v_2) - val_u$. Consider case (ii). In this case, since the terms $RS(u, v_1)$ and $RS(u, v_2)$ do not count $Q$, it remains to show that $val_u$ does not count $Q$. If $path_v$ does not contain any entry of the form $(Q, y, SP/SCP)$, where $y \in V(S)$, then $Q$ could not have been counted in $val_u$ and the proof is complete. Therefore, suppose that $path_v$ contains an entry $(Q, y, SP/SCP)$. Let the SP/SCP pair corresponding to the entry $(Q, y, SP/SCP)$ be $\langle S(y), p \rightarrow q \rangle$. By construction of the $path$ sets, either $v_1$ or $v_2$ must be $p$. Without loss of generality, assume $v_1 = p$. Then, we must have $q \in V(S(v_1))$. Now, if $u$ is a node in the subtree/subtree-complement $S(y)$, then $Q$ must be counted in $RS(u, v_1)$, a contradiction. Thus, $u$ cannot be a node in the subtree/subtree-complement $S(y)$ and, consequently, $Q$ is not counted in $val_u$. This completes the proof for case (ii). Since the above analysis holds for any quartet tree $Q$, we must have $RS(u, v) = RS(u, v_1) + RS(u, v_2) - val_u$. □

**Nested tree traversal.** Once the preprocessing and tree decoration steps have been executed, the algorithm performs a nested tree traversal of $S$ and computes the raw score of each HGT event from $\overrightarrow{H}$ according to Lemma 4.3. More formally, the algorithm proceeds as follows:

**Algorithm** *ComputeScores*

1: **for** each $v \in V(S)$ in a post-order traversal of $S$ **do**
2: Perform procedure *Augment*$(v)$.
3: **for** each $u \in V(S) \setminus \{rt(S)\}$ **do**
4: **if** $(u, v)$ is a valid HGT event, i.e., $(u, v) \in \overrightarrow{H}$, **then**
5: **if** $v \in Le(S)$ **then**

6: Set $RS(u, v)$ to be $val_u$.
7: **else**
8: Set $RS(u, v)$ to be $RS(u, v_1) + RS(u, v_2) - val_u$, where $\{v_1, v_2\} = Ch(v)$.

**Lemma 4.4.** *The raw scores of all HGT events in $\overrightarrow{H}$ can be computed within $O(n^2 + |\Phi|)$ time.*

*Proof.* We will show that our algorithm correctly computes $RS(u, v)$, for each $(u, v) \in \overrightarrow{H}$ in $O(n^2 + |\Phi|)$ time.

*Correctness:* The correctness of the algorithm follows immediately from Lemma 4.3.

*Complexity:* As explained in the proof of Lemma 3.1, the preprocessing step can be executed in $O(n)$ time. For the tree decoration step, we can infer whether any given quartet is of Type I or Type II and its four SP/SCP pairs in $O(1)$ time by performing a constant number of LCA and subtree-inclusion queries. Once these four SP/SCP pairs are identified, updating the path sets on $S$ requires $O(1)$ time. Decorating the tree with information from all the quartet trees thus takes $O(|\Phi|)$ time. Next, we analyze the complexity of the nested tree traversal (Algorithm *ComputeScores*). Consider Step 2 of Algorithm *ComputeScores*: We perform the procedure *Augment*$(v)$ for each $v \in V(S)$. At any given $v$, the time complexity of *Augment*$(v)$ is $O(n)$ for the first "for" loop, $O(|path_v|)$ for the second "for" loop, and $O(n)$ for the third "for" loop (since all the $val_u$'s can be computed in a single pre-order traversal of $S$). The time complexity of Step 2 of Algorithm *ComputeScores* is thus $O(n + |path_v|)$. Now, by Lemma 4.1 we know that any $Q \in \Phi$ is represented in the $path$ sets of $S$ exactly eight times. Thus, over all $v \in V(S)$, the total time spent at Step 2 is $O(\sum_{v \in V(S)}(n + |path_v|))$, which is $O(n^2 + |\Phi|)$. Consider Steps 4 through 8 of Algorithm *ComputeScores*: Each of these steps requires $O(1)$ time per execution and are executed $O(n^2)$ times. The total time complexity of Steps 4 through 8 is thus $O(n^2)$. Thus, the total time complexity of our algorithm is $O(n^2 + |\Phi|)$. □

## 4.2 Computing the Raw Scores of Horizontal Edges

Our goal now is to compute the raw score of each horizontal edge in $H$. For any edge $\{u, v\} \in H$, let its raw score be denoted by $RS\{u, v\}$. Observe that $RS\{u, v\} = RS(u, v) + RS(v, u) - common\{u, v\}$, where $common\{u, v\}$ is the total weight of the quartet trees that are counted in both $RS(u, v)$ and $RS(v, u)$. We now show how to compute the value $common\{u, v\}$ for each horizontal edge $\{u, v\} \in H$ within $O(n^2+|\Phi|)$ time using a variant of the algorithm described above.

Let $common(u, v)$ denote the weight of the quartet trees that are counted in $RS(u, v)$, and that are also explained by the HGT event $(v, u)$. Note that, in fact, $common(u, v) = common(v, u) = common\{u, v\}$; but it will be conceptually simpler to compute the value $common(u, v)$ for each $(u, v) \in \overrightarrow{H}$.

For any given quartet tree $Q \in \Phi$, there may be several HGT events from $\overrightarrow{H}$ that could (i) explain $Q$, and (ii) their HGT events in the reverse direction also explain $Q$; we denote this set of horizontal edges by $\overleftrightarrow{H}(Q)$. Analogous to SP pairs we now define path-path pairs. A *path-path (PP) pair* on $S$ is a pair $\langle p_1 \to q_1, p_2 \to q_2 \rangle$, where $p_1, q_1, p_2, q_2 \in V(S)$, such that the paths $p_1 \to q_1$ and $p_2 \to q_2$ are monotone and node disjoint and if $v_1 \in V(p_1 \to q_1)$ and $v_2 \in V(p_2 \to q_2)$ then $v_1 \not\leq_S v_2$ and $v_2 \not\leq_S v_1$. Given a PP pair $\sigma = \langle p_1 \to q_1, p_2 \to q_2 \rangle$, the set of all HGT events $(u, v)$ from $\overrightarrow{H}$ such that $u \in V(p_1 \to q_1)$ and $v \in V(p_2 \to q_2)$ is denoted by $\overrightarrow{H}(\sigma)$. The following lemma and its proof are analogous to Lemma 4.1.

**Lemma 4.5.** *Given any quartet tree $Q \in \Phi$, there exist four PP pairs, denoted $\sigma_1, \sigma_2, \sigma_3, \sigma_4$, such that $\overleftrightarrow{H}(Q) = \overrightarrow{H}(\sigma_1) \cup \overrightarrow{H}(\sigma_2) \cup \overrightarrow{H}(\sigma_3) \cup \overrightarrow{H}(\sigma_4)$. Moreover, the four sets $\overrightarrow{H}(\sigma_1), \overrightarrow{H}(\sigma_2), \overrightarrow{H}(\sigma_3)$ and $\overrightarrow{H}(\sigma_4)$ are pairwise disjoint.*

*Proof.* Let $Q = ab|cd$ and, without loss of generality, assume that the corresponding quartet tree in $S$ is $Q' = ac|bd$. Let $E$ and $F$ denote the internal nodes of $Q'$ such that $E$ is on the path from $a$ to $c$, and $F$ is on the path from $b$ to $d$. Let $S_{Q'}$ denote the tree $S[Le(Q')]$. Consider the embedding of $Q'$ in $S_{Q'}$. The root of $S_{Q'}$ must appear along one of the following five paths: the path from (i) $a$ to $E$, (ii) $c$ to $E$, (iii) $E$ to $F$, (iv) $b$ to $F$, or (v) $d$ to $F$.

Consider case (i). Let $A$ denote the child of the root of $S_{Q'}$ whose subtree contains $a$, $C$ denote the child of $E$ in $S_{Q'}$ whose subtree contains $c$, $B$ denote the child of $F$ whose subtree contains $b$, and $D$ denote the child of $F$ whose subtree contains $d$. We define the four PP pairs as follows: $\sigma_1 = \langle A \to a, B \to b\rangle$, $\sigma_2 = \langle B \to b, A \to a\rangle$, $\sigma_3 = \langle C \to c, D \to d\rangle$, and $\sigma_4 = \langle D \to d, C \to c\rangle$. It is straightforward to verify that the four sets $H(\sigma_1)$, $H(\sigma_2)$, $H(\sigma_3)$ and $H(\sigma_4)$ are pairwise disjoint and that each of them is a subset of $H(Q)$.

We will now show that every HGT event in $\overleftrightarrow{H}(Q)$ must appear in one of these four sets. Consider any HGT event $(u, v) \in \overleftrightarrow{H}(Q)$. Observe that the node $v$ must be such that $|V(S(v)) \cap \{a, b, c, d\}| = 1$. This is because if $|V(S(v)) \cap \{a, b, c, d\}| = 0$ then this HGT event does not affect the embedding of the quartet tree $Q$ in the resulting gene tree at all, and if $|V(S(v)) \cap \{a, b, c, d\}| > 1$ then this HGT event yields a gene tree that remains consistent with $Q'$. Now observe that, if $(u, v) \in \overleftrightarrow{H}(Q)$ then, by definition, we must have $(v, u) \in \overleftrightarrow{H}(Q)$. Consequently, we must also have $|V(S(u)) \cap \{a, b, c, d\}| = 1$. Thus, $u$ and $v$ must be nodes on the paths $A \to a$, $B \to b$, $C \to c$, or $D \to d$ (but not on the same path). Suppose $v \in B \to b$. In order for the resulting gene tree to be consistent with $Q$, the path from $a$ to $b$ in this gene tree must not intersect the path from $c$ to $d$. This means that the node $u$ must lie on the path $A \to a$; i.e., $(u, v) \in \overrightarrow{H}(\sigma_1)$. The cases when $v \in A \to a$, $v \in C \to c$, and $v \in D \to d$ are analogous. In summary, any HGT event $(u, v) \in \overleftrightarrow{H}(Q)$ must be such that $(u, v) \in \overrightarrow{H}(\sigma_1) \cup \overrightarrow{H}(\sigma_2) \cup \overrightarrow{H}(\sigma_3) \cup \overrightarrow{H}(\sigma_4)$. This proves the correctness of the theorem for case (i). Cases (ii), (iv), and (v) are completely analogous to case (i).

Consider case (iii). Let $A$ denote the child of $E$ whose subtree contains $a$, $C$ denote the child of $E$ in $S_{Q'}$ whose subtree contains $c$, $B$ denote the child of $F$ whose subtree contains $b$, and $D$ denote the child of $F$ whose subtree contains $d$. As before we define the four PP pairs to be: $\sigma_1 = \langle A \to a, B \to b\rangle$, $\sigma_2 = \langle B \to b, A \to a\rangle$, $\sigma_3 = \langle C \to c, D \to d\rangle$, and $\sigma_4 = \langle D \to d, C \to c\rangle$. The remainder of the proof is identical to the proof for case (i) above. □

Note that the path in any PP pair is monotone in the tree $S$, and in particular contains at most one node from any level of $S$. From the constructive proof of Lemma 4.5, the following corollary follows immediately.

**Corollary 4.2.** *For every quartet tree $Q \in \Phi$, any two of its four PP pairs have disjoint first-paths and disjoint second-paths. Moreover, the four PP pairs only contain four distinct paths; Specifically, the four PP pairs must be such that $\sigma_1 = \langle p_1 \to q_1, p_2 \to q_2\rangle$, $\sigma_2 = \langle p_2 \to q_2, p_1 \to q_1\rangle$, $\sigma_3 = \langle p_3 \to q_3, p_4 \to q_4\rangle$, and $\sigma_4 = \langle p_4 \to q_4, p_3 \to q_3\rangle$.*

Our algorithm for computing *common*$(u, v)$ is essentially the same as the algorithm *ComputeScores* described above, but with a few key differences: First, as the reader may have already guessed, it is based on PP pairs instead of SP pairs. And second, the tree decoration step and the *Augment*$(v)$ procedure are different. Next we describe the modified *Decorate*$(\Phi, S)$ and *Augment*$(v)$ procedures.

**Procedure** *PP-Decorate*$(\Phi, S)$

1: **for** each quartet tree $Q \in \Phi$ **do**

2: Using Lemma 4.5, compute its four PP pairs $\sigma_1 = \langle p_1 \to q_1, p_2 \to q_2\rangle$, $\sigma_2 = \langle p_2 \to q_2, p_1 \to q_1\rangle$, $\sigma_3 = \langle p_3 \to q_3, p_4 \to q_4\rangle$, and $\sigma_4 = \langle p_4 \to q_4, p_3 \to q_3\rangle$. (Note: The $q_i$s denote the leaf-node end points of the four paths.)

3: Add the triple $(Q, p_1, q_1)$ to the set $path_{q_2}$ and to the set $path_{pa(p_2)}$.
4: Add the triple $(Q, p_2, q_2)$ to the set $path_{q_1}$ and to the set $path_{pa(p_1)}$.
5: Add the triple $(Q, p_3, q_3)$ to the set $path_{q_4}$ and to the set $path_{pa(p_4)}$.
6: Add the triple $(Q, p_4, q_4)$ to the set $path_{q_3}$ and to the set $path_{pa(p_3)}$.

**Procedure** *PP-Augment*$(v)$ $\quad \{v \in V(S)\}$
1: **for** each $x \in V(S)$ **do**
2: Set $counter_x$ to 0.
3: **for** each triple $(Q, p, q) \in path_v$ **do**
4: Increment $counter_q$ by the weight of $Q$.
5: Decrement $counter_{pa(p)}$ by the weight of $Q$.
6: **for** each $u \in V(S) \setminus \{rt(S)\}$ **do**
7: Set $val_u$ to $\sum_{x \in V(S(u))} counter_x$.

**Observation 2.** *In the last "for" loop of procedure PP-Augment$(v)$, a triple $(Q, p, q)$ from $path_v$ is counted in $val_u$ exactly when the path $p \to q$ is such that $u \in V(p \to q)$.*

Algorithm *ComputeScores* is also correspondingly modified as follows.

**Algorithm** *ComputeCommonScores*
1: **for** each $v \in V(S)$ in a post-order traversal of $S$ **do**
2: Perform procedure *PP-Augment*$(v)$.
3: **for** each $u \in V(S) \setminus \{rt(S)\}$ **do**
4: **if** $(u, v)$ is a valid HGT event, i.e., $(u, v) \in \overrightarrow{H}$ **then**
5: **if** $v \in Le(S)$ **then**
6: Set *common*$(u, v)$ to be $val_u$.
7: **else**
8: Set *common*$(u, v)$ to be *common*$(u, v_1)$ + *common*$(u, v_2)$ − $val_u$, where $\{v_1, v_2\} = Ch(v)$.

The correctness of our algorithm to compute the *common*$(u, v)$'s is based on the following lemma (analogous to Lemma 4.3).

**Lemma 4.6.** *Suppose $S$ has been decorated according to the modified tree decoration step and procedure PP-Augment$(v)$ has been executed for some $v \in V(S)$. Consider any $(u, v) \in \overrightarrow{H}$.*

1. *If $v \in Le(S)$, then common$(u, v) = val_u$.*
2. *If $v \notin Le(S)$, then common$(u, v)$ = common$(u, v_1)$+common$(u, v_2)-val_u$, where $\{v_1, v_2\} = Ch(v)$.*

*Proof.* $v \in Le(S)$: Let $Q$ be any quartet that is explained by both HGT events $(u, v)$ and $(v, u)$. Then, $Q$ must have a PP pair, say $\sigma = \langle p_1 \to q_1, p_2 \to q_2 \rangle$, such that $q_2 = v$ and $u \in V(p_1 \to q_1)$. This implies that the set $path_v$ contains the triple $(Q, p_1, q_1)$. Thus, (the weight of) $Q$ is counted at least once in $val_u$. Now, from Corollary 4.2, we know that there does not exist any other PP pair of $Q$, say $\sigma' = \langle p'_1 \to q'_1, p'_2 \to q'_2 \rangle$, such that $V(p_1 \to q_1) \cap V(p'_1 \to q'_1) \neq \emptyset$ (since the first-paths of any two PP pairs must be node disjoint). Thus, the quartet tree $Q$ is, in fact, counted exactly once in $val_u$. Finally, observe that if some quartet $Q'$ is counted in $val_u$, then $Q'$ must have a PP pair $\sigma = \langle p_1 \to q_1, p_2 \to q_2 \rangle$ such that $q_2 = v$ and $u \in V(p_1 \to q_1)$ (see Observation 2); consequently, by Lemma 4.5, the HGT event $(u, v)$ must indeed explain the quartet $Q'$. Thus, *common*$(u, v) = val_u$.

$v \notin Le(S)$: Let $Q$ be a quartet tree from $\Phi$ and $\{v_1, v_2\} = Ch(v)$. Observe that at most one of the edges $(v, v_1)$ and $(v, v_2)$ may be present on any single path from the PP pairs of $Q$ (since all paths are monotone). Corollary 4.2 therefore implies that $Q$ may be counted in at most one of $RS(u, v_1)$ and $RS(u, v_1)$ (since only one of the first-paths from any two different PP pairs of $Q$ may contain the node $u$). Suppose $Q$ is explained by both HGT events $(u, v)$ and $(v, u)$. Then, $Q$ must have a PP pair, say $\sigma = \langle p_1 \to q_1, p_2 \to q_2 \rangle$, such that $v \in V(p_2 \to q2)$ and $u \in V(p_1 \to q_1)$. In other words, $Q$ must have been counted in one of $RS(u, v_1)$ or $RS(u, v_2)$. And, since $v \leq_S p_2$, the set $path_v$ does not contain the entry $(Q, p_1, q_1)$. Also, from Corollary 4.2, we know that $path_v$ cannot contain any other entry $(Q, p'_1, p'_2)$ such that $V(p_1 \to q_1) \cap V(p'_1 \to q'_1) \neq \emptyset$. Thus, $Q$ is not counted in the value $val_u$, and consequently, $Q$ is counted exactly once in the value $RS(u, v_1) + RS(u, v_2) - val_u$.

Now, suppose that $Q$ is not explained by both HGT events $(u, v)$ and $(v, u)$. There are two possible cases: (i) $Q$ is counted in one of $common(u, v_1)$ or $common(u, v_2)$, say $common(u, v_1)$, or (ii) $Q$ is counted neither in $common(u, v_1)$ nor in $common(u, v_2)$. Consider case (i). Since $Q$ is counted in $common(u, v_1)$, but not in $common(u, v)$, $Q$ must have a PP pair $\sigma = \langle p_1 \to q_1, p_2 \to q_2 \rangle$, such that $v_1 = p_2$, $q_2 \in S(v_1)$ and $u \in V(p_1 \to q_1)$. Thus, since $v = pa(p)$, the set $path_v$ must contain the entry $(Q, p_1, q_1)$. This implies that $Q$ is counted at least once in the value $val_u$. Now, from Corollary 4.2, we know that there does not exist any other PP pair, say $\sigma' = \langle p'_1 \to q'_1, p'_2 \to q'_2 \rangle$ of $Q$ such that $V(p_1 \to q_1) \cap V(p'_1 \to q'_1) \neq \emptyset$ (this is because the first-paths from any two SP pairs must be node disjoint). Thus, the quartet tree $Q$ is, in fact, counted exactly once in $val_u$. Consequently, $Q$ is not counted in the value $common(u, v_1) + common(u, v_2) - val_u$. Consider case (ii). In this case, since the terms $common(u, v_1)$ and $common(u, v_2)$ do not count $Q$, it remains to show that $val_u$ does not count $Q$. If $path_v$ does not contain any entry of the form $(Q, p, q)$, where $p, q \in V(S)$, then $Q$ could not have been counted in $val_u$ and the proof is complete. Therefore, suppose that $path_v$ contains an entry $(Q, p, q)$. Let the SP pair corresponding to the entry $(Q, p, q)$ be $\langle p \to q, p_2 \to q_2\ rangle$. Therefore, by construction of the $path$ sets, either $v_1$ or $v_2$ must be $p_2$. Without loss of generality, assume $v_1 = p_2$. Then, we must have $q_2 \in V(S(v_1))$. Now, if $u \in V(p \to q)$, then $Q$ must be counted in $common(u, v_1)$, a contradiction. Thus, $u \notin V(p \to q)$ and, consequently, $Q$ is not counted in $val_u$. This completes the proof for case (ii). Since the above analysis holds for any quartet tree $Q$, we must have $common(u, v) = common(u, v_1) + common(u, v_2) - val_u$. □

This yields the following lemma.

**Lemma 4.7.** *The raw score of each edge in $H(S)$ can be computed within $O(n^2 + |\Phi|)$ time.*

*Proof.* By Lemma 4.4 we know that the raw score of each HGT event in $\overrightarrow{H}$ can be computed within $O(n^2 + |\Phi|)$ time. By Lemma 4.6 we also know that our algorithm to compute the $common(u, v)$'s computes these values correctly. Since, for any $\{u, v\} \in H$, $RS\{u, v\} = RS(u, v) + RS(v, u) - common(u, v)$, it only remains to show that our algorithm to compute the $common(u, v)$'s terminates in $O(n^2 + |\Phi|)$ time.

The modified tree decoration step takes $O(|\Phi|)$ time, as before. We analyze the complexity of the nested tree traversal (Algorithm *ComputeCommonScores*). Consider Step 2 of Algorithm *ComputeCommonScores*: We perform the procedure *PP-Augment*$(v)$ for each $v \in V(S)$. At any given $v$, the time complexity of *PP-Augment*$(v)$ is $O(n)$ for the first "for" loop, $O(|path_v|)$ for the second "for" loop, and $O(n)$ for the third "for" loop (since all the $val_u$'s can be computed in a single post-order traversal of $S$). The time complexity of Step 2 of Algorithm *ComputeCommonScores* is thus $O(n + |path_v|)$. Now, by Lemma 4.5 we know that any $Q \in \Phi$ is represented in the $path$ sets of $S$ exactly eight times. Thus, over all $v \in V(S)$, the total time spent at Step 2 is $O(\sum_{v \in V(S)}(n + |path_v|))$, which is $O(n^2 + |\Phi|)$. Now consider Steps 4 through 8 of the Algorithm *ComputeCommonScores*: Each of these steps requires $O(1)$ time per execution and are executed

$O(n^2)$ times. The total time complexity of Steps 4 through 8 is thus $O(n^2)$. Thus, the total time complexity of our algorithm for computing the *common*$(u, v)$'s is $O(n^2 + |\Phi|)$. □

### 4.3 Computing the Normalization Factors

We wish to normalize the raw score of each horizontal edge by dividing it by its normalization factor, i.e., the maximum number of distinct quartet trees that could be explained by an HGT event along that edge. All the normalization factors can be computed by running the algorithm described in Sections 4.1and 4.2, which computes the raw scores of horizontal edges, on a dataset that contains all the possible $3 \times \binom{n}{4}$ quartet trees, each with weight 1. In this approach, the total time complexity of our algorithm for the highway scoring problem becomes $O(n^2 + n^4)$, which is $O(n^4)$. However, all the normalization factors can actually be computed in $O(n^2)$ time. This $O(n^2)$-time algorithm is based on the observation that the normalization factors can be obtained by computing quartet distances (i.e., the number of quartets that have different topologies) between certain pairs of trees. In each such pair, one of the trees is $S$ and the other is a slightly modified version of $S$. It can be shown, based on combinatorial arguments, that all the required quartet distances can be computed efficiently within $O(n^2)$ time. In the interest of brevity, further details are deferred to the appendix. This implies that the total time complexity of our algorithm for computing all the normalized scores remains $O(n^2 + |\Phi|)$. Thus, we have the following theorem.

**Theorem 4.1.** *The highway scoring problem can be solved in $O(n^2 + |\Phi|)$ time.*

*Proof.* Once all the raw scores and normalization factors are generated, the final normalized score of any horizontal edge is simply its raw score divided by its normalization factor. Computing this final score for every horizontal edge thus takes $O(|H(S)|)$, which is $O(n^2)$, additional time. The theorem now follows immediately from Lemmas 4.7 and A.11. □

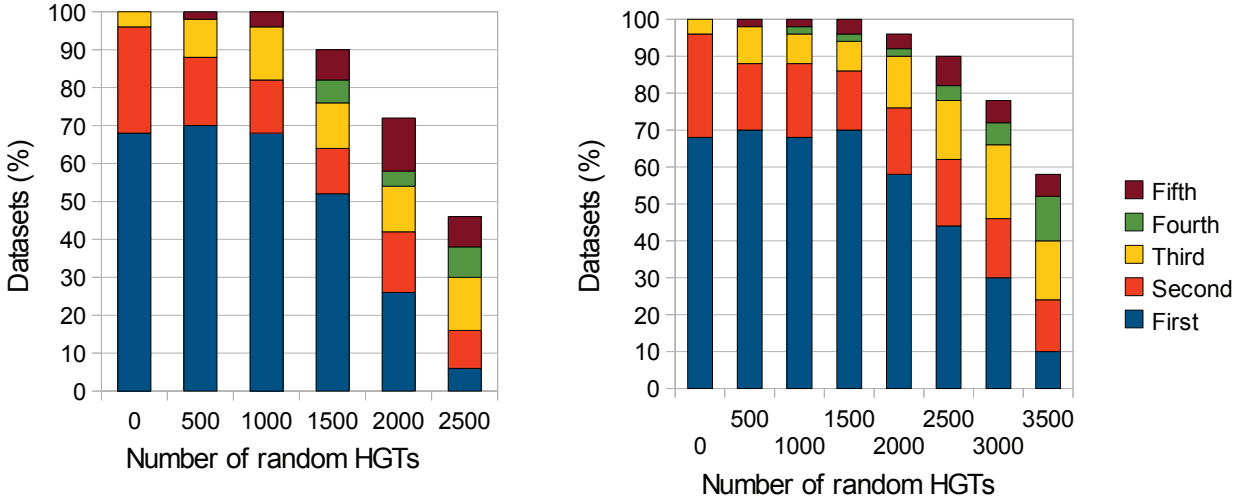
## 5 Experimental Analysis

**Runtime Analysis.** We implemented our $O(n^4)$ algorithm for the highway scoring problem (without the additional $O(n^2)$ speed-up for computing the normalization factors, mentioned above), and compared its running time against an implementation of the naïve $O(n^6)$ algorithm. We ran both implementations on five datasets of 25 taxa, each with an input consisting of the $3 \times \binom{25}{4}$ possible quartet trees, and on five datasets of 50 taxa, each with an input consisting of the $3 \times \binom{50}{4}$ possible quartet trees. Our fast algorithm averaged 0.27 seconds and 5.71 seconds on the 25 and 50 taxa datasets respectively. The corresponding times for the naïve algorithm were 118 seconds and 169 minutes, respectively. Note that these times do not include the time required to compute the quartet tree input. All of our timed experiments were run on a server with two quad-core Xeon 5410 CPUs running at at 2.33 GHz, and 16 GB of RAM, using a single core.

Due to the efficiency of our algorithm for solving the highway scoring problem, the bulk of the time in any analysis is spent on decomposing the input gene trees into their constituent quartet trees in order to generate the weighted set of quartet trees. Still, our fast algorithms make it possible to analyze datasets with hundreds of taxa and thousands of gene trees. For example, we can analyze datasets with 1000 gene trees each and 50, 100, and 200 taxa, in about 30 seconds, 15 minutes, and 5 hours respectively (including the time required to perform the quartet decompositions for each of the 1000 gene trees). Moreover, the memory requirements of our algorithms are actually very low, since they only need to generate and work with a small part of the weighted set of quartet trees at any particular time. We elaborate more on this in Section 6.

**Simulated datasets.** We performed two types of experiments on simulated data. The first tested the effect of HGT abundance on the ability to infer highways, and the second tested the ability of the method to detect multiple implanted highways. Each simulated dataset consisted of a random species tree on 50 taxa generated under a Yule process using the tool TreeSample (Hartmann *et al.*, 2010), and 1000 gene trees generated as described below.

For the first type of experiment, we randomly chose a highway on the species tree, and randomly assigned 10% of the 1000 genes as having been transferred along this highway, with equal probability for each transfer direction. Next, we simulated "noise" as additional single-gene HGT events. For each event, the horizontal edge and direction were selected randomly and independently, and the affected gene was selected at random. Selection was done with replacement, from the set of all gene trees (including those genes that were transferred on the chosen highway). We simulated noise at six different levels: 0 (i.e., no noise), 500, 1000, 1500, 2000, and 2500 HGTs. For each noise level, we created 50 different datasets (different species trees) and in each set computed the scores of all horizontal edges and the rank of the implanted highway among them. As shown in Fig. 4(a), our method tends to identify the implanted highway, even in datasets with high levels of noise; for instance, when there are 1500 random HGTs (15 times the number of highway transfers), the implanted highways were included among the top five edges in 90% of the simulations, and the implanted highway is the top-scoring edge in more than half the cases. By 2500 HGTs, performance has deteriorated. In general, the average ranks of the implanted highways for each of the six noise levels were 1.36, 1.46, 1.58, 2.56, 5.26, and 19.20 respectively. Note that there are over 4000 candidate horizontal edges, so even in the highest noise simulations the implanted edge is ranked in the top 0.5%.




(a) (b)

Figure 4: **Detecting implanted highways.** (a) Results when the implanted highways affect 10% of the genes. (b) Results when the implanted highways affect 15% of the genes. For each level of "noise" (random single-gene HGTs), we ran the algorithm on 50 simulated datasets. Both plots depict the fraction of simulations, for each noise level, in which the implanted highway edge is detected as one of the five highest-scoring edges.

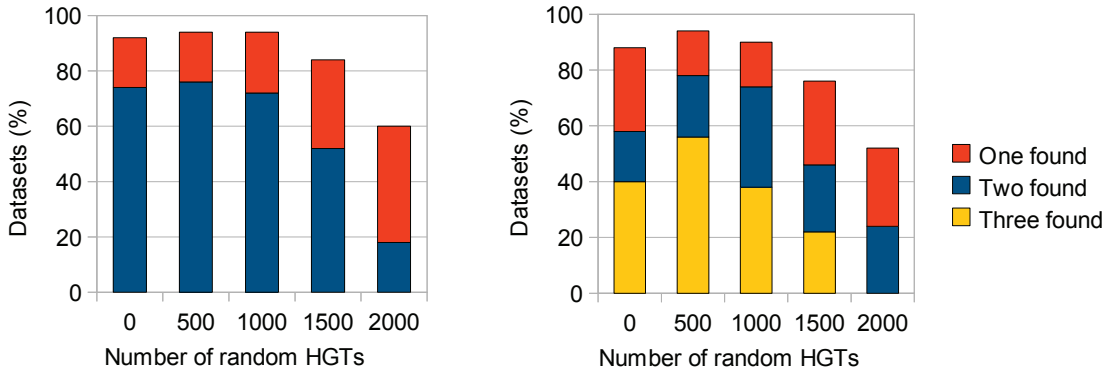
To study the effect of highway size on its detection frequency, we repeated the above experiment with

larger highways. This time, we randomly chose 15% of the 1000 genes to be transferred in each highway. The results are depicted in Fig. 4(b). Compared to the previous experiment (Fig. 4(a)), there is marked improvement in noise tolerance. For instance, even when there are 2500 random HGTs, 90% of the implanted highways are among the top five edges and almost 80% are among the top three. Even for datasets with 3500 random HGTs, almost 60% of the implanted highways are among the top five edges. The average ranks of the implanted highways for each of the eight noise levels (0 through 3500, in increments of 500) were 1.36, 1.46, 1.50, 1.54, 1.90, 2.60, 4.08, and 9.24 respectively.

Interestingly, even when there is no noise in the data, the method does not always identify the implanted highway as its top-scoring edge. In such cases, we generally observed that the top-scoring edge was a close neighbor of the implanted highway and had a marginally higher score. This probably happens because our normalization factors are independent of direction, while the actual HGT events that take place along the highway are directed. Still, as the experiment demonstrates, even with modestly sized highways (affecting 10% of the genes) and relatively high levels of noise, our algorithm usually brings to the top the correct highway, and further analysis of the top candidates can reveal the true highway. The experiment also suggests that the performance of the method can be further improved by using more sophisticated normalization.

Next, we tested the performance of our method on datasets with multiple implanted highways. The basic setup is identical to that used in the previous simulated experiments, except that we implanted multiple highways, each responsible for transferring 100 genes chosen randomly (a gene may be transferred on several highways). For each dataset we checked how many of the $x$ implanted highways were detected among the three highest scoring edges, in $x$ iterations of the algorithm. We detected at most one implanted highway during each iteration; thus, if more than one of the implanted edges were among the three highest scoring edges during an iteration, we only considered the one with the highest score amongst them as the detected highway. This detected highway was then removed to generate the input instance for the next iteration. If none of the implanted highways was discovered among the three highest scoring edges in an iteration, then we simply removed the top ranking edge. This procedure aims to imitate the real scenario when a biologist can identify the correct highway out of a handful of top ranking ones by other biological information. We performed two sets of experiments, one with two implanted highways per dataset and the other with three. The results, shown in Fig. 5, demonstrate the effectiveness of our method in detecting multiple highways. For instance, for the datasets with two implanted highways, both highways were discovered over 70% of the time for the datasets with 1000 random HGTs, and at least one highway was discovered about 85% of the time for the datasets with 1500 random HGTs. Similarly, for the three highway datasets, at least two of the implanted highways were discovered in almost 75% of the datasets with 1000 random HGTs and almost 45% of the time for datasets with 1500 random HGTs. Even on the 2000 HGT datasets, at least one highway was found among the three highest scoring edges in over 50% of the cases.

How does one distinguish between datasets that only contain many small-scale HGT events and those that actually contain a highway? If the highest-scoring edge for a dataset has a markedly higher score compared to the other edges, then this is a strong indication of the presence of a highway; this is, for example, what we see in the analysis of the real cyanobacterial dataset (see below). More generally, if we have two datasets with a similar degree of quartet incongruence (i.e. the total weight of inconsistent quartet trees is similar) such that one of these datasets has a highway and the other does not, then the dataset with the highway is expected to have a much larger highest-score, as compared to the dataset without the highway. We tested this in a simulation study in which we created 20 datasets, each with 50 taxa and 1000 gene trees, such that ten of these datasets had an implanted highway affecting 100 genes and 1000 random HGTs, and the remaining ten datasets had no highways but 1250 random HGTs. (Note that we add 1250 random




(a) (b)

Figure 5: **Detecting multiple highways.** (a) Results on datasets with two implanted highways. (b) Results on datasets with three implanted highways. For each level of noise we ran the algorithm on 50 simulated datasets. The histograms depict the fraction of simulations in which the correct highways were among the top three ranking edges in two (plot (a)) or three (plot (b)) iterations of the algorithm.

HGTs (instead of just 1100 HGTs) to the datasets without highways: This is to ensure that the total weight of inconsistent quartet trees is decidedly higher in the ten trees without a highway. Indeed, the average total weight of inconsistent quartet trees for the datasets with highways is $1.516 \times 10^7$ and for the datasets without highways is $1.677 \times 10^7$.) The same ten species trees were used in the datasets with and without the highways. Fig. 6 depicts the scores of the highest-scoring edges for each of these 20 datasets. Overall, the average highest-score for the datasets with a highway was 86.55 (min: 78.45, max: 92.74), while the average for the datasets without a highway was 68.55 (min: 64.16, max: 74.26). Thus, by testing a given dataset against a simulated dataset with similar gene tree size distribution and a similar degree of quartet incongruence, one can assess if the given dataset contains actual highways.

As we observed previously, even in datasets with highways, the top-scoring edge does not always represent a highway, and it may be necessary to consider a few of the top-scoring edges to discover the true highway(s). Still, in some cases, it seems possible to accurately infer if the top-scoring edge is in fact a highway simply by studying the differences between the scores of a few top-ranking edges. For example, if the score of the top-ranking edge is well separated from the score of the one ranked second, then that top-scoring edge is likely to represent a true highway. Furthermore, we observed that, even if there is no clear score separation, the gap between the scores of first and second ranked edges tends to be higher in the cases where the top-scoring edge is a highway. To quantify this observation, we performed the following experiment: We considered all the datasets from our first simulated experiment (with a highway size of 100 genes and noise varying from 0 to 2500 HGTs) and, for each dataset, computed a *gap score*, $g = \Delta_1/\Delta_2$, where $\Delta_1$ is the difference between the scores of the first and second ranking edges and $\Delta_2$ is the difference between the scores of the second and third ranking edges for that dataset. In general, we observed that datasets in which the top-scoring edge was the implanted highway had higher gap scores than in datasets where the top-scoring edge was not the highway. To test the predictive power of the gap score in deciding if the top-scoring edge is a highway or not, we viewed the problem as a classification problem: For a chosen gap threshold, we classified all input instances that had a gap value above this threshold as 'yes' instances

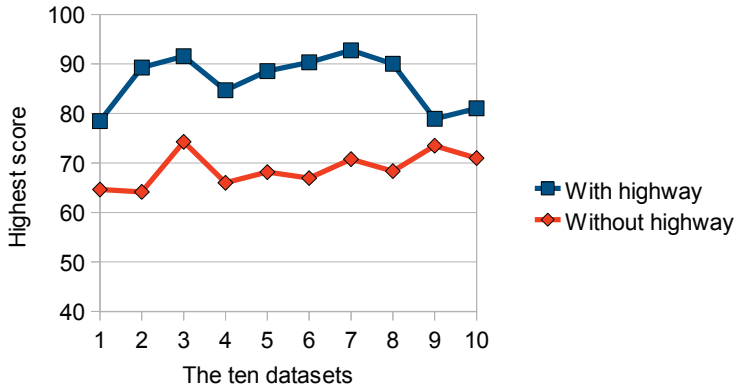



Figure 6: **High scores indicate the presence of highways.** The chart shows the scores of the top-ranking edges for ten datasets with implanted highways and ten datasets without implanted highways. The datasets with implanted highways each have 1000 gene trees on 50 taxa with 1000 random HGTs and one highway affecting 100 genes. The datasets without implanted highways are built on the same ten species trees but have 1250 random HGTs (and no highway). The highest-scores are much higher for the datasets with highways.

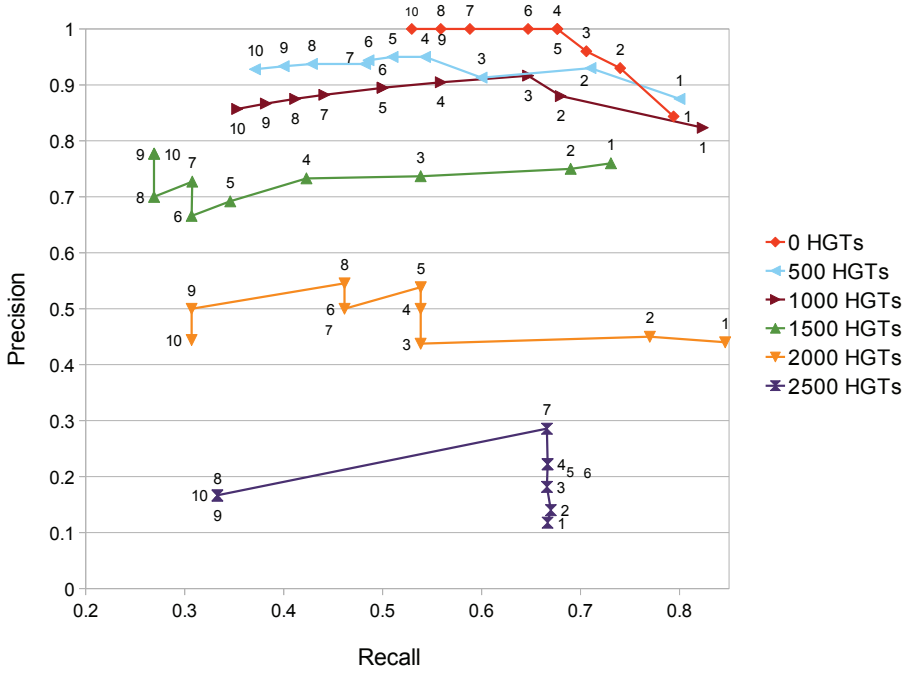



Figure 7: **Inferring if the top-scoring edge is a highway using gap scores.** The chart plots the precision and recall of our simple classification method based on gap scores. The noise level varies from 0 to 2500 HGTs and the gap threshold from 1 to 10. The results for different thresholds are marked along each connected colored curve with the threshold values noted on the curve. For each fixed noise level and gap threshold, the precision and recall are computed from classification results on 50 simulated datasets.

and all other instances as 'no' instances. Based on this classification we then computed the corresponding precision and recall values, separately for each noise level and each gap threshold. Precision measures the fraction of reported 'yes' instances that are true 'yes' instances, i.e., precision is the probability that an instance classified as being a 'yes' instance is indeed an instance in which the top-scoring edge is a highway. Recall is the fraction of true 'yes' instances that are correctly identified as being 'yes' instances in the classification, i.e., recall is the probability that an instance in which the top-scoring edge is a highway is correctly identified as being a 'yes' instance in the classification. Figure 7 summarizes our results. The results indicate that the predictive power of this simple method is quite good for datasets with low noise levels, when using relatively low gap thresholds. For example, using threshold of 1 for datasets with 500 HGTs one gets 85% precision and nearly 80% recall. Not surprisingly, precision decreases as noise increases, and recall decreases as the gap threshold increases. Interestingly, with the exception of the very extreme cases of 0 and 2500 HGTs, the curves are very flat. This implies that by using low threshold one gains in recall without losing much in precision. In fact, using a threshold of 1 gives a recall of at least 0.7 in all these cases.

**Cyanobacterial dataset.** We applied our method to a dataset of 1128 genes from 11 cyanobacterial species, taken from Zhaxybayeva *et al.* 2006. The existence of a highway on this set of species was postulated in Zhaxybayeva *et al.* 2006, 2009 and thus this dataset serves for method validation. Each of the 1128 gene trees had at least 9 of the 11 species (see Zhaxybayeva *et al.* 2006 for further details). As the trusted species tree, shown in Fig. 8, we used the rooted tree constructed on the 16S ribosomal RNA sequence from these species (Fournier and Gogarten, 2010). To account for uncertainty in the topologies of the gene trees, for each gene tree we used only those quartet trees that were present in at least 80% of the bootstrap replicates of that gene tree (Zhaxybayeva *et al.*, 2006). Our final weighted set had 799 different quartet trees with a total weight of 214,729. The total number of inconsistent quartet trees was 469 and their total weight was 23,042. There were 118 candidate horizontal edges. Fig. 9A shows the histogram of the normalized scores for these horizontal edges in the first iteration of the algorithm. The highest scoring edge (Fig. 9A) is extremely well separated from the next candidate in terms of the scores. It is marked in Fig. 8. A priori, it is surprising that this highway connects two different genera that are distinguished by different light harvesting machineries, but the high rate of transfer between marine Synecchococcus and Prochlorococcus has been previously observed and discussed (Zhaxybayeva *et al.*, 2006, 2009). The discovered highway thus matches perfectly with prior biological observations.

We performed further analysis of this dataset with the aim of discovering other novel highways. In the second iteration (Fig. 9B), our method proposes the second highway shown in Fig. 8. Though the normalized score of this highway is much smaller than that of the first highway (179.4 vs 508.6), it is well separated from the scores of all the other horizontal edges except the two that constitute a third candidate highway (see below). Like the first, this second highway also represents transfer between the small marine cyanobacteria, likely mediated by cyanophage. Further analysis (Fig. 9C) suggests the presence of a third highway (normalized score: 157.2, second-highest score: 97.7) along one of two possible horizontal edges, shown in Fig. 8. These two horizontal edges produce the same unrooted tree and are hence indistinguishable in our quartet-based model. By the fourth iteration (Fig. 9D), the top scoring edge is less well separated, so we focused on the first three iterations only.

# 6 Discussion

In this paper we addressed the problem of inferring highways of gene sharing, a fundamental problem in understanding the effects and dynamics of horizontal gene transfer, and a crucial step towards inferring past symbiotic associations that shaped the evolution of organisms. We formulated the problem, introduced

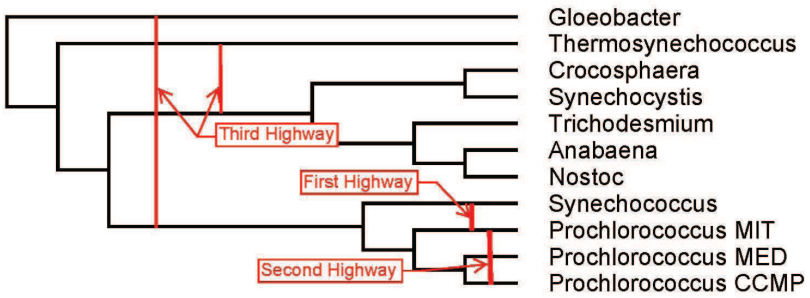



Figure 8: The 16SrRNA tree on the 11 cyanobacterial species, with detected highways marked.

a normalized score for evaluating candidate highways, and developed an algorithm that is linear in the number of input quartets and quadratic in the number of species. Our new systematic approach and efficient algorithms for the highway detection problem facilitate accurate and in-depth analysis of relatively large datasets. The method detects the fingerprints of highways by looking at combined data from all the input gene trees summarized as quartet tree counts. We thus avoid the computational burden and uncertainty of inferring individual HGT events for each gene. Our simulation results demonstrate that our method is effective at detecting highways and is robust to noise in the data: In these simulations, even in the presence of substantial noise, our method reports the true highway among the few top-scoring edges. We were also able to identify the established highway in the cyanobacterial dataset, and our analysis identified two additional putative highways.

Our approach is based on quartets. A second option would have been to use splits (also called bipartitions) instead of quartets (see, for example, Zhaxybayeva 2009). However, a quartet decomposition-based approach has several advantages in this setting: First, split decomposition is not as fine grained as quartet decomposition and the phylogenetic signal for a particular HGT event could be easily lost if the rate of HGT is high. Second, when using splits, the addition of more taxa can lead to shorter internal branches, and the support for a split depends on the amount of change along an individual branch. This leads to the unsatisfactory situation that the more taxa are added to an analysis, the shorter and less supported the internal branches tend to become. Our quartet decomposition based approach avoids this problem. However, further testing of the method in both simulations and on real datasets is needed, and it might be instructive to compare it to alternative non-quartet-based methods.

An important feature of our algorithms, both for creating the weighted quartet-tree set and for the highway scoring problem, is that they make it possible to generate and do the analysis on only a subset of the quartet trees at a time and then trivially combine the results. Thus, when the number of quartet trees becomes too large to fit into the main memory of the computer (e.g., when analyzing datasets with over 100 taxa), it is possible to partition the problem and combine the results efficiently. The memory requirements of our algorithm thus remain modest even when analyzing datasets with hundreds of taxa. This also makes it possible to efficiently and easily parallelize the analysis.

Our analysis is based on unrooted quartet trees, while our species tree is assumed to be rooted. Consequently, some of the valid HGT events are undetectable because the (unrooted) gene tree that they create is identical to the unrooted version of the species tree. This also leads to situations where two different HGT events may create the same unrooted gene tree and are hence indistinguishable (e.g., see the third highway

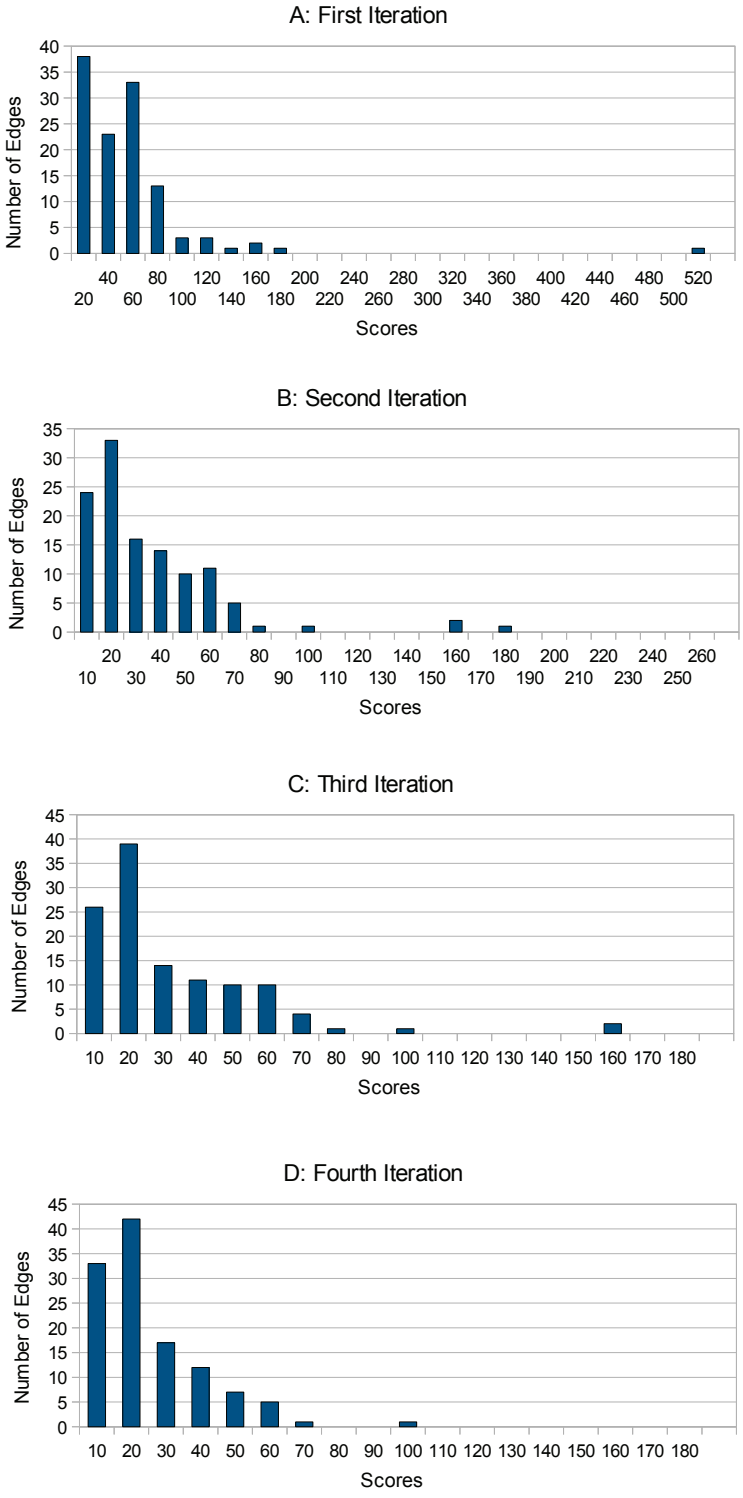



Figure 9: Histograms of edge scores for four iterations of the algorithm on the cyanobacterial dataset.

in Fig. 8). Our method and algorithms can be easily adapted to infer highways based on rooted triplet trees, in order to better handle those cases where the input gene trees are reliably rooted.

At present, our modeling and evaluation assume that transfers occur only between lineages that are represented on the species tree. HGT events from donor lineages that are not represented on the species tree, either due to under-sampling or to extinction, may result in inferred highways of gene sharing that originate at the point in the reference tree where the unsampled lineage bifurcates from the sampled lineages. In such cases, a consideration of branch lengths may help to distinguish true highways of gene sharing from artifacts due to under-sampling of lineages.

When seeking multiple highways, our approach is iterative, detecting one highway per iteration. Mathematically, a better approach is to find simultaneously a set of the $k$ highway edges (biologically consistent with each other) that together explain the most quartet trees. However, the time complexity of this alternative formulation seems to grow exponentially in $k$.

Our method, though effective, still has some limitations. In general, while the normalized scoring of the horizontal edges that we propose corrects for the variation in the number of candidate quartets of different edges, the top scoring edge is not always the correct highway.

Using more sophisticated normalization may help to improve the highway detection accuracy of our method, and we intend to investigate this possibility further. For example, we could run our algorithm on the quartet set of each gene tree separately, compute the raw score of each HGT event, normalize these scores by the maximum number of quartet trees that could be explained by that (directed) HGT event, and then, to each horizontal edge, assign the maximum of the normalized scores of its two HGT events as its score. The final normalized score of any horizontal edge would then be the average of these scores over all gene trees. Intuitively, this would make the normalization more sensitive to the direction of the HGT events affecting individual genes. Such normalization, though more demanding computationally, should also make the method more robust to gene trees that have missing taxa.

Our approach indirectly relies on the parsimony principle; for instance, if a dataset contains two highways that are closely related to one another then the method may only detect one of them (since many of the inconsistent quartet trees from one highway may also support the other highway). Formulating the highway detection problem in an explicit probabilistic framework could help improve the accuracy of highway detection in some datasets.

The runtime of our method is dominated by the time required to generate the weighted set of quartet trees from the input gene trees (this step requires $O(t \times n^4)$ time, where $t$ is the number of input gene trees, compared to $O(n^4)$ for solving the highway scoring problem). It may be possible to bypass the generation of this weighted set of quartet trees by employing an approach based on computing quartet distances between trees. It may also be possible to generate fewer quartet trees by employing sampling. Still, our method can be easily applied to fairly large datasets, and we were able to analyze datasets with 200 taxa and 1000 gene trees within five hours using a single core on a 2.33 GHz quad-core Xeon 5410 CPU.

In many cases, it might be desirable to have accurate estimates of the number of gene transfers supporting a detected highway. While the normalized scores that we compute could be used as rough estimates of the magnitude of gene transfer along the detected highways, more sophisticated normalization is needed to obtain accurate counts.

Finally, the assignment of a statistical significance to a detected highway is of interest. As we demonstrate on the cyanobacterial dataset and in simulations (Fig. 6), score separation and score distribution is indicative, but a model-based statistical analysis could help.

## Acknowledgements

MSB was supported in part by a postdoctoral fellowship from the Edmond J. Safra Bioinformatics program at Tel-Aviv University. JPG was supported in part by NSF grant DEB 0830024, the Edmond J. Safra Bioinformatics Program, and a fellowship from the Fulbright Program. RS was supported in part by the Israel Science Foundation (Grant 802/08) and by the Raymond and Beverly Sackler Chair in Bioinformatics.

## Author Disclosure Statement

No competing financial interests exist.

## References

Bansal, M. S., Gogarten, J. P., and Shamir, R. (2010). Detecting highways of horizontal gene transfer. In E. Tannier, editor, *RECOMB-CG*, volume 6398 of *Lecture Notes in Computer Science*, pages 109–120. Springer.

Beiko, R. G., Harlow, T. J., and Ragan, M. A. (2005). Highways of gene sharing in prokaryotes. *Proceedings of the National Academy of Sciences of the United States of America*, **102**(40), 14332–14337.

Bender, M. A. and Farach-Colton, M. (2000). The LCA problem revisited. In G. H. Gonnet, D. Panario, and A. Viola, editors, *LATIN*, volume 1776 of *Lecture Notes in Computer Science*, pages 88–94. Springer.

Boc, A. and Makarenkov, V. (2003). New efficient algorithm for detection of horizontal gene transfer events. In G. Benson and R. D. M. Page, editors, *WABI*, volume 2812 of *Lecture Notes in Computer Science*, pages 190–201. Springer.

Boc, A., Philippe, H., and Makarenkov, V. (2010). Inferring and validating horizontal gene transfer events using bipartition dissimilarity. *Systematic Biology*, **59**(2), 195–211.

Bordewich, M. and Semple, C. (2005). On the computational complexity of the rooted subtree prune and regraft distance. *Annals of combinatorics*, **8**(4), 409–423.

Fournier, G. P. and Gogarten, J. P. (2010). Rooting the ribosomal tree of life. *Mol Biol Evol*, **27**(8), 1792–1801.

Gary, M. W. (1993). Origin and evolution of organelle genomes. *Curr Opin Genet Dev*, **3**, 884–890.

Gray, G. and Fitch, W. (1983). Evolution of antibiotic resistance genes: The DNA sequence of a kanamycin resistance gene from *Staphylococcus aureus*. *Mol Biol Evol*, **1**(1), 57–66.

Hallett, M. T. and Lagergren, J. (2001). Efficient algorithms for lateral gene transfer problems. In *RECOMB*, pages 149–156.

Hartmann, K., Wong, D., and Stadler, T. (2010). Sampling trees from evolutionary models. *Systematic Biology*, **59**(4), 465–476.

Hein, J. (1990). Reconstructing evolution of sequences subject to recombination using parsimony. *Mathematical biosciences*, **98**(2), 185–200.

Hickey, G., Dehne, F., Rau-Chaplin, A., and Blouin, C. (2008). SPR distance computation for unrooted trees. *Evolutionary Bioinformatics*, **4**, 17–27.

Hill, T., Nordstrom, K., Thollesson, M., Safstrom, T., Vernersson, A., Fredriksson, R., and Schioth, H. (2010). Sprit: Identifying horizontal gene transfer in rooted phylogenetic trees. *BMC Evolutionary Biology*, **10**(1), 42.

Huang, J. and Gogarten, J. (2007). Did an ancient chlamydial endosymbiosis facilitate the establishment of primary plastids? *Genome Biology*, **8**(6), R99.

Jin, G., Nakhleh, L., Snir, S., and Tuller, T. (2009). Parsimony score of phylogenetic networks: Hardness results and a linear-time heuristic. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **6**(3), 495–505.

Lake, J. A. (2009). Evidence for an early prokaryotic endosymbiosis. *Nature*, **460**, 967–971.

Margulis, L. and Sagan, D. (2002). *Acquiring genomes: A theory of the origin of species*. Basic Books, New York.

Nakhleh, L., Warnow, T., and Linder, C. R. (2004). Reconstructing reticulate evolution in species: theory and practice. In P. E. Bourne and D. Gusfield, editors, *RECOMB*, pages 337–346. ACM.

Nakhleh, L., Ruths, D. A., and Wang, L.-S. (2005). RIATA-HGT: A fast and accurate heuristic for reconstructing horizontal gene transfer. In L. Wang, editor, *COCOON*, volume 3595 of *Lecture Notes in Computer Science*, pages 84–93. Springer.

Ochiai, K., Yamanaka, T., Kimura, K., and Sawada, O. (1959). Inheritance of drug resistance (and its transfer) between Shigella strains and between Shigella and E.coli strains [In Japanese]. *Hihon Iji Shimpor*, **1861**, 34–46.

Piaggio-Talice, R., Burleigh, G., and Eulenstein, O. (2004). *Phylogenetic supertrees: Combining information to reveal the Tree of Life*, chapter Quartet supertrees, pages 173–192. Springer.

Sapp, J. (2005). The prokaryote-eukaryote dichotomy: Meanings and mythology. *Microbiol Mol Biol Rev*, **69**(2), 292–305.

Sonea, S. (1988). The global organism: A new view of bacteria. *The Sciences*, **28**, 38–45.

Than, C., Ruths, D. A., Innan, H., and Nakhleh, L. (2007). Confounding factors in HGT detection: Statistical error, coalescent effects, and multiple solutions. *Journal of Computational Biology*, **14**(4), 517–535.

Woese, C. R. (1987). Bacterial evolution. *Microbiol Mol Biol Rev*, **51**(2), 221–271.

Woese, C. R. and Fox, G. E. (1977). Phylogenetic structure of the prokaryotic domain: the primary kingdoms. *Proceedings of the National Academy of Sciences of the United States of America*, **74**(11), 5088–5090.

Zhaxybayeva, O. (2009). *Horizontal gene transfer: Genomes in flux*, volume 532 of *Methods in Molecular Biology*, chapter Detection and quantitative assessment of horizontal gene transfer, pages 195–213. Humana Press.

Zhaxybayeva, O., Gogarten, J. P., Charlebois, R. L., Doolittle, W. F., and Papke, R. T. (2006). Phylogenetic analyses of cyanobacterial genomes: Quantification of horizontal gene transfer events. *Genome Research*, **16**(9), 1099–1108.

Zhaxybayeva, O., Doolittle, W. F., Papke, R. T., and Gogarten, J. P. (2009). Intertwined evolutionary histories of marine *Synechococcus* and *Prochlorococcus marinus*. *Genome Biol Evol*, **1**, 325–339.

# Appendix

Here we show how to compute all the normalization factors within $O(n^2)$ time. As before, we will first compute the normalization factors for each (directed) HGT event, and then show how to compute the normalization factors for the horizontal edges. Given any rooted tree $T$, denote the normalization factor for an HGT event $(u, v) \in \overrightarrow{H}(T)$ by $NF_T(u, v)$.

**Notation:** Given any rooted tree $T$ and two nodes $u, v \in V(T) \setminus \{rt(T)\}$ such that $v \notin V(T(u))$, let $SPR_T(u, v)$ be the tree obtained from $T$ by pruning the subtree rooted at $u$ and regrafting it at the edge $(pa(v), v)$. For example, in Fig. 2, if $T$ denotes the tree on the left, then the tree in the middle is the tree $SPR_T(E, C)$ and the tree on the right is the tree $SPR_T(c, b)$. Given two trees $T$ and $T'$ with the same leaf set, we denote the set of quartets that induce different topologies in the two trees by $Diff(T, T')$.

Then, we have the following observation.

**Observation 3.** *Given a rooted tree $T$ and an HGT event $(u, v) \in \overrightarrow{H}(T)$, we must have $NF_T(u, v) = |Diff(T, T')|$, where $T' = SPR_T(v, u)$.*

Thus, to compute $NF_T(u, v)$, it is sufficient to compute $|Diff(T, T')|$, where $T' = SPR_T(v, u)$. Note that though the tree $S$ is rooted, the rooting does not affect the topology of any quartet tree in $S$. Thus, when computing the value $|Diff(S, S')|$, for $S' = SPR_S(v, u)$, the rooting of $S$ does not affect the final value. We will use this insight in the following lemma.

**Lemma A.1.** *Consider any HGT event $(u, v) \in \overrightarrow{H}(S)$. Then, $NF_S(u, v) = NF_R(u, v)$, where $R$ denotes the tree $S$ rerooted on the edge $(v, pa(v))$ of $S$.*

*Proof.* Observe that the unrooted versions of $S$ and $R$, and of $SPR_S(v, u)$ and $SPR_R(v, u)$ are identical. Consequently, we must have $Diff(S, SPR_S(v, u)) = Diff(R, SPR_R(v, u))$. Furthermore, if $(u, v) \in \overrightarrow{H}(S)$ then $(u, v) \in \overrightarrow{H}(R)$ as well. Thus, by Obs. 3, we must have $NF_S(u, v) = |Diff(S, SPR_S(v, u))| = |Diff(R, SPR_R(v, u))| = NF_R(u, v)$. □

Let $\overrightarrow{H}(T, v) = \{(u, v) : u \in V(T) \text{ and } (u, v) \in \overrightarrow{H}(T)\}$. Given any $v \in V(S) \setminus \{rt(S)\}$, we will show how to efficiently compute the value $|Diff(R, SPR_R(v, u))|$ (i.e., $NF_R(u, v)$) for each $(u, v) \in \overrightarrow{H}(R, v)$, where $R$ denotes the tree $S$ rerooted on the edge $(v, pa(v))$. In light of Lemma A.1, this immediately yields the value of $NF_S(u, v)$, for each $(u, v) \in \overrightarrow{H}(S, v)$.

Throughout the remainder of this section, unless otherwise stated, let $v \in V(S) \setminus \{rt(S)\}$ be fixed and let $R$ denote the tree $S$ rerooted on the edge $(v, pa(v))$. The following lemma characterizes the quartets that constitute $Diff(R, SPR_R(v, u))$.

**Lemma A.2.** *Suppose we are given an HGT event $(u, v) \in \overrightarrow{H}(R, v)$ and a quartet $Q$ on the leafset of $R$. Then, $Q \in Diff(R, SPR_R(v, u))$ if and only if (i) $Q$ has exactly one leaf from $R(v)$ and (ii) there exist three other nodes $x, y, z \in V(R) \setminus V(rt(R) \to pa(u))$ such that $pa(x), pa(y), pa(z) \in V(rt(R) \to pa(u))$ and $Q$ has one leaf each from $R(x)$, $R(y)$, and $R(z)$.*

*Proof.* Let $R'$ denote $SPR_R(v, u)$.

Suppose $Q \in Diff(R, R')$. Let $\Lambda = \{\lambda \in V(R) \setminus V(rt(R) \to pa(u)) : pa(\lambda) \in V(rt(R) \to pa(u))\}$. Then, observe that (i) for any $\lambda_1, \lambda_2 \in \Lambda$, we must have $Le(R(\lambda_1)) \cap Le(R(\lambda_2)) = \emptyset$, and (ii) $\bigcup_{\lambda \in \Lambda} Le(R(\lambda)) = Le(R)$. Also observe that the subtree $R(\lambda)$, for any $\lambda \in \Lambda$, appears in the tree $R'$ as well. This means that, if for any $\lambda \in \Lambda$, the subtree $R(\lambda)$ contains two or more leaves from $Q$, then we must have

$R|Q = R'|Q$. Thus, for any $\lambda \in \Lambda$, $R(\lambda)$ may contain at most one leaf from $Q$. Furthermore, if $Q$ has no leaves from $R(v)$, then $R|Q$ and $R'|Q$ must be identical. Therefore, $R(v)$ must have exactly one leaf from $Q$. This proves that if $Q \in \mathit{Diff}(R, R')$, then the two conditions stated in the lemma must be satisfied.

Suppose conversely that quartet $Q = \{a, b, c, d\}$ satisfies conditions (i) and (ii) of the lemma. Without any loss of generality assume that $a \in R(v)$, $b \in R(x)$, $c \in R(y)$, and $d \in R(z)$ and that among $x$, $y$, and $z$ the node closest to $rt(R)$ is $x$. Thus, we must have $R|Q = ab|cd$. We now consider two cases: (1) Suppose one of $y$ or $z$, say $z$, is the node $u$. Then, in the tree $R'$, we must have $R'|Q = ad|bc$. (2) Suppose none of $y$ or $z$ is the node $u$, then one of these two nodes, say $z$, must be closer to $u$ than the other. Then, in the tree $R'$, we again have $R'|Q = ad|bc$. Thus, in either case we have $R|Q \neq R'|Q$, and therefore $Q \in \mathit{Diff}(R, R')$. □

The above lemma gives an $O(n)$-time algorithm to compute $\mathit{Diff}(R, R')$. This algorithm is stated formally below.

**Algorithm** *ComputeDiff*$(R, (u, v))$ $\{(u, v) \in \overrightarrow{H}(R, v)\}$

1: Perform a post-order traversal of $R$ to compute the value $|\mathit{Le}(R(x))|$ for each $x \in V(R)$.
2: Compute the set $\Lambda = \{\lambda \in V(R) \setminus V(rt(R) \to pa(u)) \colon pa(\lambda) \in V(rt(R) \to pa(u))$. Let $k$ denote the size of $\Lambda$ and $\lambda_1, \ldots, \lambda_k$ denote its elements. Let $\lambda_1$ denote node $v$.
3: Compute the value of $|\mathit{Diff}(R, R')|$, where $R' = \mathit{SPR}_R(v, u))$, as follows:

$$|\mathit{Diff}(R, R')| = |\mathit{Le}(R(v))| \cdot \binom{n - |\mathit{Le}(R(v))|}{3} - A - B, \tag{1}$$

where,

$$A = |\mathit{Le}(R(v))| \cdot \sum_{i=2}^{k} \binom{|\mathit{Le}(R(\lambda_i))|}{3},$$

and

$$B = |\mathit{Le}(R(v))| \cdot \sum_{i=2}^{k} \binom{|\mathit{Le}(R(\lambda_i))|}{2} \cdot (n - |\mathit{Le}(R(v))| - |\mathit{Le}(R(\lambda_i))|).$$

**Lemma A.3.** *Given any HGT event $(u, v) \in \overrightarrow{H}(R)$, Algorithm ComputeDiff computes the value $|\mathit{Diff}(R, R')|$, for $R' = \mathit{SPR}_R(v, u)$, correctly in $O(n)$ time.*

*Proof. Correctness:* From Lemma A.2 we know that $|\mathit{Diff}(R, R')|$ is the number of quartets that have exactly one leaf from $\mathit{Le}(R(v))$ and the remaining three leaves from three distinct subtrees of $R$ rooted at nodes from $\Lambda \setminus \{v\}$. Consider Eqn. (1) from the algorithm. The first term on the RHS counts the number of quartets that have exactly one leaf from $\mathit{Le}(R(v))$ and the remaining three leaves from the rest of the leafset of $R$, i.e., from $\bigcup_{i=2}^{k} \mathit{Le}(R(\lambda_i))$. There are three possible cases for these three remaining leaves: They may be such that (1) each leaf is from a distinct subtree of $R$, each rooted at a different node from $\Omega = \{\lambda_2, \ldots, \lambda_k\}$, (2) two of these leaves are from the same subtree of $R$ rooted at a node from $\Omega$ and the third is from a different subtree, or (3) all of these three leaves are from the same subtree of $R$ rooted at a node from $\Omega$. The term $A$ counts all those quartets that have one leaf from $\mathit{Le}(R(v))$ and the remaining three leaves are as described in case (3) above, and the term $B$ counts all those quartets that have one leaf from $\mathit{Le}(R(v))$ and the remaining three leaves are as described in case (2) above. Thus, the RHS of Eqn. (1) counts exactly those quartets that have exactly one leaf from $\mathit{Le}(R(v))$ and the remaining three leaves are as described in case (1) above. This is exactly the value $|\mathit{Diff}(R, R')|$.

*Complexity:* It is straightforward to verify that each of the first two steps in the algorithm can be executed within $O(n)$ time. For the fourth step, since the value of $|\mathit{Le}(R(x))|$ has already been precomputed for each $x \in V(R)$, the values of terms $A$ and $B$ can each be computed in $O(k)$, which is $O(n)$, time. Thus, the fourth step can also be executed within $O(n)$ time. □

Note that, based on Lemma A.3 and Observation 3, we now have an $O(n^2)$-time algorithm for computing all the values $\mathit{NF}_R(u, v)$ for each $(u, v) \in \overrightarrow{H}(R, v)$. However, we can do even better; we shall show how to compute all these values within $O(n)$ time.

The additional speed-up relies on the observation that if the value $|\mathit{Diff}(R, \mathit{SPR}_R(v, u))|$ has already been computed then the value $|\mathit{Diff}(R, \mathit{SPR}_R(v, u'))|$, for any $u' \in \mathit{Ch}(u)$, can be obtained within $O(1)$ time. This idea is developed more fully in the following algorithm.

**Algorithm** *ComputeAllDiff*$(R, v)$

1: Perform a post-order traversal of $R$ to compute the value $|\mathit{Le}(R(x))|$ for each $x \in V(R)$.
2: Let $x$ denote the sibling of $v$ in $R$. Let $X$ denote the set of grandchildren of $x$ in $R$, i.e., $X = \{x' \in V(R)\colon\ \mathit{pa}(\mathit{pa}(x')) = x\}$.
3: **for** each $x' \in X$ **do**
4: Compute $|\mathit{Diff}(R, \mathit{SPR}_R(v, x'))|$, as shown in Eqn. (1).
5: **for** each node $y$ (except $x'$) in a preorder traversal of $R(x')$ **do**
6: Let $y'$ denote the sibling of $y$. Compute the value of $|\mathit{Diff}(R, \mathit{SPR}_R(v, y))|$ as follows:

$$|\mathit{Diff}(R, \mathit{SPR}_R(v, y))| = |\mathit{Diff}(R, \mathit{SPR}_R(v, \mathit{pa}(y)))| + C + D, \tag{2}$$

where,

$$\begin{aligned} C \;=\; & |\mathit{Le}(R(v))| \cdot \binom{|\mathit{Le}(R(\mathit{pa}(y)))|}{3} - |\mathit{Le}(R(v))| \cdot \binom{|\mathit{Le}(R(y))|}{3} \\ & - |\mathit{Le}(R(v))| \cdot \binom{|\mathit{Le}(R(y'))|}{3}, \end{aligned}$$

and

$$\begin{aligned} D \;=\; & |\mathit{Le}(R(v))| \cdot \binom{|\mathit{Le}(R(\mathit{pa}(y)))|}{2} \cdot (n - |\mathit{Le}(R(v))| - |\mathit{Le}(R(\mathit{pa}(y)))|) \\ & - |\mathit{Le}(R(v))| \cdot \binom{|\mathit{Le}(R(y))|}{2} \cdot (n - |\mathit{Le}(R(v))| - |\mathit{Le}(R(y))|) \\ & - |\mathit{Le}(R(v))| \cdot \binom{|\mathit{Le}(R(y'))|}{2} \cdot (n - |\mathit{Le}(R(v))| - |\mathit{Le}(R(y'))|). \end{aligned}$$

**Remark:** Note that in Step 3 of the above algorithm we define $X$ to be the set of *grandchildren* of $x$. This is simply because $\mathit{SPR}_R(v, x')) = R$ for any $x' \in \mathit{Ch}(x)$.

**Lemma A.4.** *Let* $\Upsilon = \{u \in V(R)\colon (u, v) \in \overrightarrow{H}(R, v)\}$. *The values* $|\mathit{Diff}(R, \mathit{SPR}_R(v, u))|$, *for every* $u \in \Upsilon$, *can all be computed in* $O(n)$ *time overall.*

*Proof.* Let $x$ and $X$ be defined as in Algorithm *ComputeAllDiff*. Observe that $\Upsilon \subseteq \bigcup_{x' \in X} V(R(x'))$. Thus, we must show that the algorithm correctly computes the values $|\mathit{Diff}(R, \mathit{SPR}_R(v, y))|$ for every $y \in \bigcup_{x' \in X} V(R(x'))$ within $O(n)$ time.

*Correctness:* Consider some node $y \in V(R(x') \setminus \{x'\}$, for some $x' \in X$. As in the algorithm, let $y'$ denote the sibling of $y$ in $R$. Consider the expressions for $|\mathit{Diff}(R, \mathit{SPR}_R(v, y))|$ and $|\mathit{Diff}(R, \mathit{SPR}_R(v, pa(y)))|$ as given in Eqn. (1). In particular, let $|\mathit{Diff}(R, \mathit{SPR}_R(v, y))| = |\mathit{Le}(R(v))| \cdot \binom{n-|\mathit{Le}(R(v))|}{3} - A_1 - B_1$ and $|\mathit{Diff}(R, \mathit{SPR}_R(v, pa(y)))| = |\mathit{Le}(R(v))| \cdot \binom{n-|\mathit{Le}(R(v))|}{3} - A_2 - B_2$, where $A_1$, $A_2$ correspond to the term $A$, and $B_1$, $B_2$ correspond to the term $B$ from Eqn. (1). Now consider the terms $C$ and $D$ from Eqn. (2). By comparing the expressions for $A_1$ and $A_2$ it is straightforward to verify that $C = A_2 - A_1$. Similarly, $D$ is simply $B_2 - B_1$. Thus, we must have $|\mathit{Diff}(R, \mathit{SPR}_R(v, y))| = |\mathit{Diff}(R, \mathit{SPR}_R(v, pa(y)))| + C + D$, which is exactly Eqn. 2.

Thus, since the initial values $|\mathit{Diff}(R, \mathit{SPR}_R(v, x'))|$, for each $x' \in X$, are computed correctly (see Lemma A.3), Eqn. (2) ensures that the values of $|\mathit{Diff}(R, \mathit{SPR}_R(v, y))|$ for every $y \in \bigcup_{x' \in X} V(R(x'))$ are computed correctly as well.

*Complexity:* It is easy to verify that each of the first two steps in the algorithm can be executed within $O(n)$ time. The "for" loop of Step 3 is executed at most four times, which implies that the total time spent on Step 4 is $O(n)$ (by Lemma A.3). Step 6 is executed a total of $|\bigcup_{x' \in X} V(R(x'))|$ times, which is $O(n)$. By Eqn. 2, each of these executions requires $O(1)$ time, since all the values $|\mathit{Le}(R(x))|$, for each $x \in V(R)$, have already been precomputed, and when computing the value $|\mathit{Diff}(R, \mathit{SPR}_R(v, y))|$ the value of $|\mathit{Diff}(R, \mathit{SPR}_R(v, pa(y)))|$ is already available. The total time complexity of Algorithm *ComputeAllDiff* is thus $O(n)$. □

**Lemma A.5.** *The normalization factors $\mathit{NF}_S(u, w)$, for every HGT event $(u, w) \in \overrightarrow{H}(S)$ can be computed in $O(n^2)$ time overall.*

*Proof.* Since there are $O(n)$ candidates for $v$, Lemma A.4 implies that the values of $|\mathit{Diff}(R, \mathit{SPR}_R(w, u))|$ can be computed for all $(u, w) \in \overrightarrow{H}(R)$ within $O(n^2)$ time. Thus, by Observation 3, the values of $\mathit{NF}_R(u, w)$, for every $(u, w) \in \overrightarrow{H}(R)$ can be obtained within $O(n^2)$ time. Lemma A.1 now completes the proof. □

**Computing the normalization factors for horizontal edges.** Our goal is to compute the normalization factor of each horizontal edge in $H(S)$. Given a rooted tree $T$, for any edge $\{u, w\} \in H(T)$, let its normalization factor be denoted by $\mathit{NF}_T\{u, w\}$. By definition, $NF_T\{u, w\} = \mathit{NF}_T(u, w) + NF_T(w, u) - \mathit{comNF}_T\{u, w\}$, where $\mathit{comNF}_T\{u, w\}$ is the number of quartet trees that are counted in both $\mathit{NF}_T(u, w)$ and $\mathit{NF}_T(w, u)$.

We will show how the above algorithm can be modified to compute the value $\mathit{comNF}_S\{u, w\}$, for all $\{u, w\} \in H(S)$, in $O(n^2)$ time overall. The following observation and lemma are analogous to Obs. 3 and Lemma A.1 respectively.

**Observation 4.** *Given a rooted tree $T$ and a horizontal edge $\{u, w\} \in H(T)$, we must have $\mathit{comNF}_T\{u, w\} = |\mathit{Diff}(T, \mathit{SPR}_T(w, u)) \cap \mathit{Diff}(T, \mathit{SPR}_T(u, w))|$.*

**Lemma A.6.** *Consider any horizontal edge $\{u, w\} \in H(S)$. Then, $\mathit{comNF}_S\{u, w\} = \mathit{comNF}_R\{u, w\}$, where $R$ denotes the tree $S$ rerooted on the edge $(w, pa(w))$ of $S$.*

*Proof.* Observe that the unrooted versions of $S$ and $R$, of $\mathit{SPR}_S(w, u)$ and $\mathit{SPR}_R(w, u)$, and of $\mathit{SPR}_S(u, w)$ and $\mathit{SPR}_R(u, w)$ are identical. Consequently, we must have $\mathit{Diff}(S, \mathit{SPR}_S(w, u)) = \mathit{Diff}(R, \mathit{SPR}_R(w, u))$ and $\mathit{Diff}(S, \mathit{SPR}_S(u, w)) = \mathit{Diff}(R, \mathit{SPR}_R(u, w))$. Furthermore, if $\{u, w\} \in H(S)$ then $\{u, w\} \in H(R)$ as well. Thus, by Obs. 3, we must have $\mathit{comNF}_S\{u, w\} = |\mathit{Diff}(S, \mathit{SPR}_S(w, u)) \cap \mathit{Diff}(S, \mathit{SPR}_S(u, w))| = |\mathit{Diff}(R, \mathit{SPR}_R(w, u)) \cap \mathit{Diff}(R, \mathit{SPR}_R(u, w))| = \mathit{comNF}_R\{u, w\}$. □

Let $H(T, v) = \{\{u, v\} : u \in V(T) \text{ and } \{u, v\} \in H(T)\}$. We follow an approach similar to the one we employed earlier: In particular, given any $v \in V(S) \setminus \{rt(S)\}$, we will show how to efficiently compute the value $|\, \mathit{Diff}(R, \mathit{SPR}_R(v, u)) \cap \mathit{Diff}(R, \mathit{SPR}_R(u, v))|$ (i.e., $\mathit{comNF}_R\{u, v\}$) for each $\{u, v\} \in H(R, v)$, where $R$ denotes the tree $S$ rerooted on the edge $(v, pa(v))$. In light of Lemma A.6, this immediately yields the value of $\mathit{comNF}_S\{u, v\}$, for each $\{u, v\} \in H(S, v)$.

As before, and unless otherwise stated, let $v \in V(S) \setminus \{rt(S)\}$ be fixed and let $R$ denote the tree $S$ rerooted on the edge $(v, pa(v))$. In the interest of brevity, throughout the remainder of this section, we denote $\mathit{Diff}(R, \mathit{SPR}_R(v, u)) \cap \mathit{Diff}(R, \mathit{SPR}_R(u, v))$ by $\mathit{comDiff}(R, \mathit{SPR}_R\{v, u\})$ for any $\{u, v\} \in H(R, v)$.

The following lemma characterizes the quartets that constitute $\mathit{comDiff}(R, \mathit{SPR}_R\{v, u\})$.

**Lemma A.7.** *Suppose we are given a horizontal edge $\{u, v\} \in H(R, v)$ and a quartet $Q$ on the leafset of $R$. Then, $Q \in \mathit{comDiff}(R, \mathit{SPR}_R\, v, u)$ if and only if (i) $Q$ has exactly one leaf each from $R(v)$ and $R(u)$ and (ii) there exist two other nodes $x, y \in V(R) \setminus V(rt(R) \to u))$ such that $pa(x), pa(y) \in V(rt(R) \to pa(u))$ and $Q$ has one leaf each from $R(x)$ and $R(y)$.*

*Proof.* Let the tree $R'$ denote the tree $R$ rerooted on the edge $(u, pa(u))$. Consider the sets $\Lambda = \{\lambda \in V(R) \setminus V(rt(R) \to pa(u))\colon\ pa(\lambda) \in V(rt(R) \to pa(u))$, and $\Lambda' = \{\lambda \in V(R') \setminus V(rt(R') \to pa(u))\colon\ pa(\lambda) \in V(rt(R') \to pa(u))$. Observe that $\Lambda = \Lambda'$ and, for any $\lambda \in \Lambda$, $R(\lambda) = R'(\lambda)$. Now, it follows from Lemma A.2 that if $Q \in \mathit{comDiff}(R, \mathit{SPR}_R\{v, u\})$ then $Q$ must have exactly one leaf each from $R(v)$ and $R'(u)$, i.e., from $R(v)$ and $R(u)$. Lemma A.2 also implies that there must exist two nodes $x, y \in \Lambda \setminus \{u, v\}$ such that $Q$ has one leaf each from $R(x)$ and $R(y)$. Thus, if $Q \in \mathit{comDiff}(R, \mathit{SPR}_R\{v, u\})$ then both conditions of the lemma must be fulfilled.

To prove the reverse direction, observe that Lemma A.2 immediately implies that any $Q$ that satisfies these two conditions must be in $\mathit{Diff}(R, \mathit{SPR}_R(v, u))$. Furthermore, since $\Lambda = \Lambda'$ and for any $\lambda \in \Lambda$ we must have $R(\lambda) = R'(\lambda)$, it immediately follows from Lemma A.2 that any $Q$ that satisfies the two conditions of the lemma must be in $\mathit{Diff}(R, \mathit{SPR}_R(u, v))$ as well. □

Based on Lemma A.7, we can compute the value of $|\, \mathit{comDiff}(R, \mathit{SPR}_R\{v, u\})|$ by using a variant of Algorithm *ComputeDiff*. This updated algorithm is given below.

**Algorithm** *ComputeCommonDiff*$(R, \{u, v\})$ $\{\{u, v\} \in H(R, v)\}$

1: Perform a post-order traversal of $R$ to compute the value $|\, \mathit{Le}(R(x))|$ for each $x \in V(R)$.
2: Compute the set $\Lambda = \{\lambda \in V(R) \setminus V(rt(R) \to pa(u))\colon\ pa(\lambda) \in V(rt(R) \to pa(u))$. Let $k$ denote the size of $\Lambda$, $\lambda_1, \ldots, \lambda_k$ denote its elements, and let $\lambda_1$ and $\lambda_2$ denote the nodes $v$ and $u$ respectively.
3: Compute the value of $|\, \mathit{comDiff}(R, \mathit{SPR}_R\{v, u\})|$ as follows:

$$|\, \mathit{comDiff}(R, \mathit{SPR}_R\{v, u\})| = |\, \mathit{Le}(R(v))| \cdot |\, \mathit{Le}(R(u))| \cdot \binom{n - |\, \mathit{Le}(R(v))| - |\, \mathit{Le}(R(u))|}{2} - A, \quad (3)$$

where,

$$A = |\, \mathit{Le}(R(v))| \cdot |\, \mathit{Le}(R(u)| \cdot \sum_{i=3}^{k} \binom{|\, \mathit{Le}(R(\lambda_i))|}{2}.$$

**Lemma A.8.** *Given any horizontal edge $\{u, v\} \in H(R, v)$, Algorithm ComputeCommonDiff computes the value $|\, \mathit{comDiff}(R, \mathit{SPR}_R\{v, u\})|$ in $O(n)$ time.*

*Proof.* The proof is based on Lemma A.7 and is completely analogous to the proof of Lemma A.3. □

To show how to compute all the values $\mathit{NF}_R\{u, v\}$ for every $\{u, v\} \in H(R, v)$ within $O(n)$ time, we modify Algorithm *ComputeAllDiff* as follows.

**Algorithm** *ComputeAllCommonDiff*$(R, v)$

1: Perform a post-order traversal of $R$ to compute the value $|\,\mathit{Le}(R(x))|$ for each $x \in V(R)$.
2: Let $x$ denote the sibling of $v$ in $R$. Let $X$ denote the set of grandchildren of $x$ in $R$, i.e., $X = \{x' \in V(R)\colon\ \mathit{pa}(\mathit{pa}(x')) = x\}$.
3: **for** each $x' \in X$ **do**
4: Compute $|\,\mathit{comDiff}(R, \mathit{SPR}_R\{v, x'\})|$, as shown in Eqn. (3).
5: **for** each node $y$ (except $x'$) in a preorder traversal of $R(x')$ **do**
6: Let $y'$ denote the sibling of $y$. Consider the value $\psi = |\,\mathit{comDiff}(R, \mathit{SPR}_R\{v, \mathit{pa}(y)\})|$ and let $A'$ denote the term corresponding to $A$ in the expression for $\psi$ according to Eqn. (3). Compute the value of $|\,\mathit{comDiff}(R, \mathit{SPR}_R\{v, y\})|$ as follows:

$$\begin{aligned} |\,\mathit{comDiff}(R, \mathit{SPR}_R\{v, y\})| \;\;=\;\; & |\,\mathit{Le}(R(v))| \cdot |\,\mathit{Le}(R(y)| \cdot \binom{n - |\,\mathit{Le}(R(v))| - |\,\mathit{Le}(R(y))|}{2} \\ & - |\,\mathit{Le}(R(v))| \cdot |\,\mathit{Le}(R(y)| \cdot \left( \frac{A'}{|\,\mathit{Le}(R(v))| \cdot |\,\mathit{Le}(R(\mathit{pa}(y)))|} + \binom{|\,\mathit{Le}(R(y'))|}{2} \right). \end{aligned} \tag{4}$$

**Lemma A.9.** *Let $\Upsilon = \{u \in V(R)\colon \{u, v\} \in H(R, v)\}$. All the values $|\,\mathit{comDiff}(R, \mathit{SPR}_R\{v, u\})|$, for every $u \in \Upsilon$, can be computed in $O(n)$ time overall.*

*Proof.* Let $x$ and $X$ be defined as in Algorithm *ComputeAllCommonDiff*. Observe that $\Upsilon \subseteq \bigcup_{x' \in X} V(R(x'))$. Thus, we must show that the algorithm correctly computes the values $|\,\mathit{comDiff}(R, \mathit{SPR}_R\{v, y\})|$ for every $y \in \bigcup_{x' \in X} V(R(x'))$ within $O(n)$ time.

*Correctness:* Consider some node $y \in V(R(x') \setminus \{x'\}$, for some $x' \in X$. As in the algorithm, let $y'$ denote the sibling of $y$ in $R$. Also, let $A'$ be as defined in the algorithm. Consider the expressions for $|\,\mathit{comDiff}(R, \mathit{SPR}_R\{v, y\})|$ and $A'$ as given by Eqn. (3). The first term on the RHS of this expression for $|\,\mathit{comDiff}(R, \mathit{SPR}_R\{v, y\})|$ is identical to the first term on the RHS of Eqn. (4). And, once the expression for $A'$ is substituted in the second term of the RHS of Eqn. (4), we get precisely the second term of the RHS of the expression for $|\,\mathit{comDiff}(R, \mathit{SPR}_R\{v, y\})|$ from Eqn. (3).

Thus, since the initial values $|\,\mathit{Diff}(R, \mathit{SPR}_R(v, x'))|$, for each $x' \in X$, are computed correctly (see Lemma A.8), Eqn. (4) ensures that the value of $|\,\mathit{comDiff}(R, \mathit{SPR}_R\{v, y\})|$, for each $y \in \bigcup_{x' \in X} V(R(x'))$, is computed correctly as well.

*Complexity:* It is easy to verify that each of the first two steps in the algorithm can be executed within $O(n)$ time. The "for" loop of Step 3 is executed at most four times, which implies that the total time spent on Step 4 is $O(n)$ (by Lemma A.8). Step 6 is executed a total of $|\bigcup_{x' \in X} V(R(x'))|$ times, which is $O(n)$. By Eqn. (4), each of these executions requires $O(1)$ time (since all the values $|\,\mathit{Le}(R(x))|$, for each $x \in V(R)$, have already been precomputed, and when computing the value $|\,\mathit{Diff}(R, \mathit{SPR}_R(v, y))|$ the value of $A'$ is already available). The total time complexity of Algorithm *ComputeAllCommonDiff* is thus $O(n)$. □

This immediately yields the following.

**Lemma A.10.** *The values $\mathit{comNF}_S\{u, w\}$, for every $\{u, w\} \in H(S)$, can be computed in $O(n^2)$ time overall.*

*Proof.* Since there are $O(n)$ candidates for $v$, Lemma A.9 implies that the values of $|\,\mathit{comDiff}(R, \mathit{SPR}_R\{w, u\})|$ can be computed for all $(u, w) \in H(R)$ within $O(n^2)$ time. Thus, by Observation 4, the values of $\mathit{comNF}_R\{u, w\}$, for every $(u, w) \in H(R)$ can be obtained within $O(n^2)$ time. Lemma A.6 now completes the proof. □

**Lemma A.11.** *The normalization factors for all edges in $H(S)$ can be generated within $O(n^2)$ time.*

*Proof.* Since $NF_S\{u,w\} = \mathit{NF}_S(u,w) + NF_S(w,u) - \mathit{comNF}_S\{u,w\}$, for each $\{u,w\} \in H(S)$, the claim follows from Lemmas A.5 and A.10. □