Simultaneous Identification of Duplications, Losses, and Lateral Gene Transfers

Zhi-Zhong Chen, Fei Deng, and Lusheng Wang

Abstract—We give a fixed-parameter algorithm for the problem of enumerating all minimum-cost LCA-reconciliations involving gene duplications, gene losses, and lateral gene transfers (LGTs) for a given species tree S and a given gene tree G. Our algorithm can work for the weighted version of the problem, where the costs of a gene duplication, a gene loss, and an LGT are left to the user's discretion. The algorithm runs in $O(m + 3^{k/c}n)$ time, where m is the number of vertices in S, n is the number of vertices in G, c is the smaller between a gene duplication cost and an LGT cost, and k is the minimum cost of an LCA-reconciliation between S and G. The time complexity is indeed better if the cost of a gene loss is greater than 0. In particular, when the cost of a gene loss is at least $0.614c$, the running time of the algorithm is $O(m + 2.78^{k/c}n)$.

Index Terms—Phylogenetic trees, reconciliations, fixed-parameter algorithms.

1 Introduction

Gene duplication, gene loss, and LGT are important evolutionary events in the study of evolutionary history among species [8], [18], [33]. Gene duplication is very common in many parts of life and its importance has long been recognized [7], [9], [17], [23], [24]. Contrary to duplications, the importance and prevalence of LGT are much controversial [10]. However, as more and more evidence for the existence of LGT is being revealed, there is growing awareness that LGT is much more common than previously thought [5], [22].

Phylogenetic trees have long been used to describe the evolutionary history of a set of species. Generally, the phylogenetic tree built from a single gene family (referred to as the gene tree) may disagree with the phylogenetic tree of the underlying species (referred to as the species tree) in the presence of evolutionary events, such as gene duplications, gene losses, and LGTs. Thus, we have the following problem: given a gene tree and the corresponding species tree, how to explain the discrepancy between them using these evolutionary events? Many approaches have been proposed to solve this problem [1], [2], [6], [21], [32]. Among them, computing a minimum-cost reconciliation between a species tree and a gene tree has been extensively studied in the literature [11], [18], [19], [25], [30]. A reconciliation is a mapping from the set of vertices of the gene tree into the set of vertices of the species tree based on some ancestor-descendant relationships. Once a reconciliation is set up, the evolutionary events, such as gene duplications, gene losses, and LGTs, can be identified accordingly. When computing a reconciliation between a gene tree and a species tree, most of the previous studies consider only a proper subset of the three events: gene duplications, gene losses, and LGTs, where the cost of each considered event is 1.

The concept of reconciliation between a species tree and a gene tree was first introduced by Goodman et al. [12]. Subsequently, reconciliations between a gene tree and a species tree involving only gene duplications and gene losses were studied in [14], [19], [25], and [30]. Hallett and Lagergren [20] study reconciliations involving LGT events only, where gene duplications and gene losses are disallowed. A fast and accurate heuristic for reconstructing LGTs for a set of gene trees and a species tree is given in [29]. Other algorithms dealing with LGTs solely can be found in [3], [4], and [21].

There have been few attempts to design algorithms for simultaneous identification of duplications and LGTs. It has been proved that finding a reconciliation that minimizes both events is NP-hard [18]. To solve this problem, many algorithms proposed often require extra information as part of the input. Górecki [13] proposes a model for reconciliations involving all the three events: gene duplication, gene loss, and LGT. However, in his model, it is assumed that all the involved LGTs are given in advance as part of the input. In this case, the problem becomes much easier from the algorithmic point of view. Zhang et al. [34] generalize reconciliations between a gene tree and a species tree, and propose a framework to reconcile a gene tree with a species LGT network. Under this framework, they give an efficient approach for inferring relative events of one type with assumptions on another. Doyon et al. [11] also consider reconciliations involving all the three events. In their model, a time stamp function f is also given together with a species tree S and a gene tree G. The output of their algorithm is a *single* minimum-cost reconciliation between S and G in which the LGT events are consistent with f. Similar work has also been done in coevolutionary studies in ecology [27], [28], where a "host" (respectively, "parasite") tree is the same as a species (respectively, gene) tree and "sorting" (respectively, "host switch") events are the same as gene losses (respectively, LGTs). In [28], Merkle et al. give a dynamic programming

- *Z.-Z. Chen is with the Division of Information System Design, Tokyo Denki University, Hatoyama, Saitama 359-0394, Japan. E-mail: zzchen@mail.dendai.ac.jp.*
- *F. Deng and L. Wang are with the Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong SAR, China. E-mail: fdeng4@student.cityu.edu.hk, cswangl@cityu.edu.hk.*

Manuscript received 22 Oct. 2011; revised 16 Apr. 2012; accepted 30 Apr. 2012; published online 23 May 2012.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBB-2011-10-0267.
Digital Object Identifier no. 10.1109/TCBB.2012.79.

algorithm to find a *single* minimum-cost reconciliation involving all the three events (i.e., duplications, sortings, and host switches). In their model, they adopt a parameter-adaptive approach and no costs have to be assigned to these evolutionary events in advance as part of the input. They also associate timing information with a host tree and a parasite tree and detect whether a reconciliation is valid with respect to the given timing information.

Tofigh et al. [18] study the problem of computing a minimum-cost reconciliation between a given gene tree and a given species tree involving all the three events, where a duplication and an LGT are associated with a cost of 1 but a loss is associated with no cost. They prove that the acyclic version of the problem is NP-hard. By dropping the requirement of acyclicity, they give a polynomial-time algorithm. They also design a fixed-parameter algorithm for enumerating all minimum-cost LCA-reconciliations between a given species tree S and a given gene tree G which runs in $O(m + 3^k n)$ time, where m is the size of S, n is the size of G, and k is the minimum total number of duplications and LGTs in an LCA-reconciliation between G and S. The work by Tofigh et al. [18] lays the foundation for dealing with the three evolutionary events simultaneously. They also mention that it would be interesting to consider weighting duplications and LGTs differently.

In this paper, we extend the work of Tofigh et al. [18] by allowing weighting gene losses together with gene duplications and LGTs. Besides, all the three events can be assigned arbitrary nonnegative integral costs. Note that assuming integral costs does not lose generality because fractional costs can be scaled up to integers. We extend the fixed-parameter algorithm by Tofigh et al. [18] and design an $O(m + 3^{k/c} n)$-time fixed-parameter algorithm for our more general problem, where m is the size of S, n is the size of G, c is the smaller between a gene duplication cost and an LGT cost, and k is the minimum cost of an LCA-reconciliation between S and G. Note that like the algorithm of Tofigh et al. [18], our algorithm does not take cycles into account. Indeed, our algorithm becomes identical to that of Tofigh et al. [18] when the cost of a gene loss is 0 and the cost of a gene duplication is the same as the cost of an LGT. The time complexity of our algorithm is indeed better than $O(m + 3^{k/c} n)$, if the cost of a gene loss is greater than 0. In particular, when the cost of a gene loss is at least $0.614c$, the running time of the algorithm is $O(m + 2.78^{k/c} n)$.

It is worth mentioning that Tofigh et al.'s algorithm [18] can be easily extended so that it also works in the case where a duplication is associated with an arbitrary integral cost and so is an LGT but a loss is associated with no cost. In this special case, the extended algorithm runs in $O(m + 3^{k/c} n)$-time, i.e., runs as fast as our algorithm. However, our algorithm runs faster when a gene loss is associated with a positive cost.

Tofigh et al. [18] propose to use the number of losses as a second criterion to choose between the minimum-cost LCA-reconciliations. In other words, they suggest the following two-phase approach: we first find all LCA-reconciliations with the minimum total cost of duplications and LGTs, among which we then choose the ones with the smallest total cost of losses. An obvious problem with this approach is that some LCA-reconciliations with the minimum total cost of duplications, LGTs, and losses may be missed, unless the cost of a gene loss is negligible compared to the cost of a duplication or LGT. In contrast, our algorithm can enumerate all the minimum-cost LCA-reconciliations no matter what the cost of a gene loss is. In this sense, our algorithm is more general than that of [18].

The remainder of this paper is organized as follows: in Section 2, we give basic definitions that will be used throughout this paper. In Section 3, we present our algorithm and state its time complexity. In Section 4, we test the performance of our algorithm on a real biological data set. Finally, in Section 5, we conclude this paper and point out several open questions and future directions.

2 Preliminaries

2.1 Basic Definitions

Let F be a rooted forest. We use $V(F)$ and $E(F)$ to denote the sets of vertices and edges in F, respectively. F is a *rooted tree* if it has only one root. F is a *rooted binary tree* if it is a rooted tree and the out-degree of every nonleaf vertex in F is 2. Two edges (u_1, v_1) and (u_2, v_2) of F are *siblings* if $u_1 = u_2$.

Let u and v be two vertices of F. For convenience, we view each vertex of F as both an ancestor and a descendant of itself in F. If u is an ancestor (respectively, descendant) of v in F, then we write $u \geq_F v$ (respectively, $u \leq_F v$). If $u \geq_F v$ and $u \neq v$, then u is a *proper* ancestor of v in F (denoted by $u >_F v$). If $u \leq_F v$ and $u \neq v$, then u is a *proper* descendant of v in F (denoted by $u <_F v$). If $u \geq_F v$ or $u \leq_F v$, then u and v are *comparable* in F; otherwise, u and v are *incomparable* in F. If u and v are comparable in F, then the *distance* between u and v in F (denoted by $\mathrm{Dist}_F(u, v)$) is the number of edges on the path between u and v in F; otherwise, the *distance* between u and v in F is ∞. The lowest common ancestor of a set U of vertices in F is denoted by $\mathrm{LCA}_F U$. If $w = \mathrm{LCA}_F\{u, v\}$ exists and $\mathrm{Dist}_F(w, v) < \mathrm{Dist}_F(w, u)$, then v is *higher than* u in F (or equivalently, u is *lower than* v in F).

If v has only one child in F, then v is *unifurcate*. If v is a root of F and is unifurcate, then *contracting v in F* is the operation that modifies F by deleting v. If v is a nonroot vertex of F and is unifurcate, then *contracting v in F* is the operation that modifies F by first adding an edge from the parent of v to the child of v and then deleting v.

For a rooted binary tree T and a subset R of $E(T)$, $T \setminus R$ denotes the rooted forest obtained from T by removing the edges in R, while $T \between R$ denotes the rooted forest obtained from $T \setminus R$ by repeatedly contracting a unifurcate vertex until none exists. Note that each nonleaf vertex of $T \between R$ has exactly two children in $T \between R$.

2.2 Transfer Sets and LCA-Reconciliations

Let X be a set of existing species. A *species tree on X* is a rooted binary tree S whose leaves one-to-one correspond to the species in X. A *gene tree on X* is a rooted binary tree G whose leaves (not necessarily one-to-one) correspond to the species in X. Since two leaves of G may correspond to the same species in X, G may have more than $|X|$ leaves. Fig. 1 gives an example of S and G.

For a subset R of $E(G)$ containing no sibling edges, the *LCA mapping* of G into S associated with R (denoted by M_R) is defined as follows:

- If u is a leaf of G, then $M_R(u)$ is the unique leaf of S that corresponds to the same species in X as u.
- Otherwise, $M_R(u)$ is $\mathrm{LCA}_S\{M_R(v_1), \ldots, M_R(v_k)\}$, where $v_1, \ldots, v_k$ are the leaf descendants of u in $G \setminus R$.

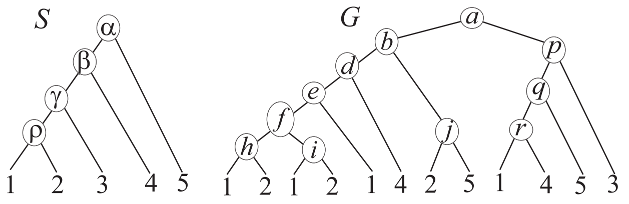


Fig. 1. A species tree S and a gene tree G on the same set $\{1, 2, 3, 4, 5\}$ of species.

A *transfer set* w.r.t. (S, G) is a subset R of $E(G)$ such that no two edges of R are siblings and $M_R(u)$ is incomparable to $M_R(v)$ in S for every edge $(u, v) \in R$. For two examples of R and M_R, see Fig. 2.

Let R be a transfer set w.r.t. (S, G). We call M_R an *LCA-reconciliation* between G and S. To get an intuitive idea on how a gene tree evolves along a species tree, the reader is referred to Fig. 3, where we use the LCA-reconciliation M_{R_2} in Fig. 2 to embed the gene tree G in Fig. 1 inside the species tree S in Fig. 1. With respect to (w.r.t. for short) R, we classify nonleaf vertices of G into three types as follows: for each nonleaf vertex u with children v and w in G,

- u is a *transfer vertex* if $(u, v) \in R$ or $(u, w) \in R$;
- u is a *speciation vertex* if it is not a transfer vertex and $M_R(v)$ and $M_R(w)$ are incomparable in S; and
- u is a *duplication vertex* if it is not a transfer vertex and $M_R(v)$ and $M_R(w)$ are comparable in S.

We use $\mathrm{DV}(R)$ to denote the set of all duplication vertices w.r.t. R. For example, in Fig. 2, $\mathrm{DV}(R_1) = \{a, e, f, p\}$ and $\mathrm{DV}(R_2) = \{a, b, e, f, p\}$. For each edge $e = (u, v) \in E(G)$, we define the number of *losses* inferred by e w.r.t. R (denoted by $\mathrm{Loss}_R(e)$ or $\mathrm{Loss}_R(u, v)$) as follows:

- If $e \in R$, then $\mathrm{Loss}_R(e) = 0$.
- If $u \in \mathrm{DV}(R)$ and $M_R(u) \neq M_R(v)$, then $\mathrm{Loss}_R(e) = \mathrm{Dist}_S(M_R(u), M_R(v))$,
- Otherwise,

$$\mathrm{Loss}_R(e) = \max\{0, \mathrm{Dist}_S(M_R(u), M_R(v)) - 1\}.$$

The *cost* of R (denoted by $\mathrm{Cost}(R)$) is $c_d \cdot |\mathrm{DV}(R)| + c_t \cdot |R| + c_\ell \cdot \sum_{e \in E(G)} \mathrm{Loss}_R(e)$, where c_d, c_t, and c_ℓ are fixed (nonnegative) integers and correspond to the costs of a gene duplication, an LGT, and a gene loss, respectively. For example, the costs of the transfer sets R_1 and R_2 in Fig. 2 are $3c_t + 4c_d + 7c_\ell$ and $c_t + 5c_d + 11c_\ell$, respectively.

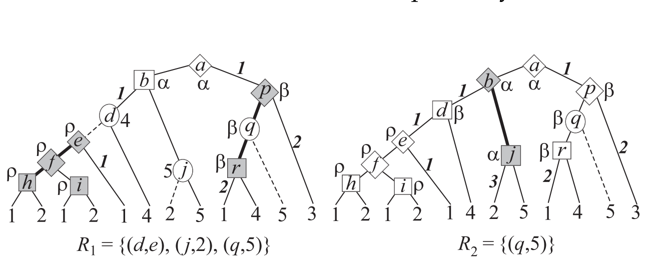


Fig. 2. The LCA mappings of the gene tree G in Fig. 1 into the species tree S in Fig. 1 associated with the transfer sets $R_1 = \{(d, e), (j, 2), (q, 5)\}$ and $R_2 = \{(q, 5)\}$, where (1) for each vertex v, the image of v under a mapping is placed near v; (2) speciation vertices and duplication vertices are enclosed by squares and diamonds, respectively; and (3) if the number of losses inferred by an edge e is positive, then it is placed near e in bold italic.

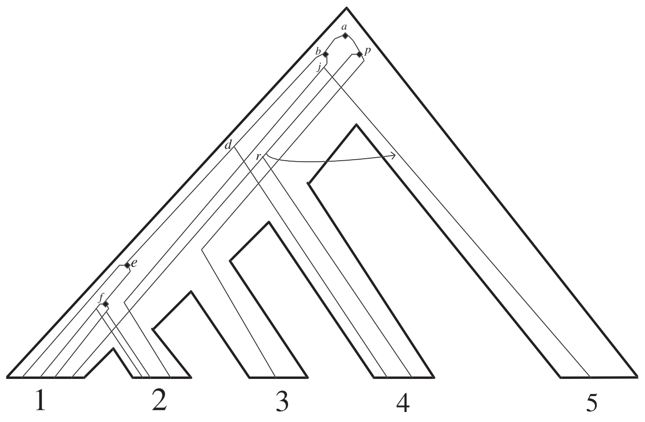


Fig. 3. Using the LCA-reconciliation M_{R_2} shown in Fig. 2 to embed G inside S, where G (respectively, S) is the gene (respectively, species) tree in Fig. 1, the tubes indicate the evolution of the species $\{1, 2, 3, 4, 5\}$, the thin lines indicate the evolution of the gene family represented by G, black diamonds indicate duplications, and directed edges indicate LGTs.

In this paper, we want to solve the following problem (called the *transfer set enumeration (TSE) problem*).

Input: A quintuple (S, G, c_t, c_d, c_ℓ), where S is a species tree on a set X of species, G is a gene tree on X, and c_t (respectively, c_d or c_ℓ) is a nonnegative integer corresponding to the cost of an LGT (respectively, duplication, or loss).

Output: All minimum-cost transfer sets R w.r.t. (S, G).

For a nonnegative integer k, a *k-transfer set* w.r.t. (S, G) is a transfer set R w.r.t. (S, G) such that $\mathrm{Cost}(R) = k$. To enumerate all minimum-cost transfer sets w.r.t. (S, G), we can proceed as follows:

1. Initialize $k = 0$.
2. Enumerate all k-transfer sets w.r.t. (S, G).
3. If at least one k-transfer set is found in Step 2, then stop; otherwise, increase k by $\mathrm{GCD}(c_t, c_d, c_\ell)$ and goto Step 2, where $\mathrm{GCD}(c_d, c_t, c_\ell)$ is the greatest common divisor of c_d, c_t, and c_ℓ.

So, our problem has become how to perform Step 2, i.e., how to solve the following problem (called the *bounded transfer set enumeration (BTSE) problem*).

Input: A sextuple $(S, G, c_t, c_d, c_\ell, k)$, where (S, G, c_t, c_d, c_ℓ) is the same as that in the TSE problem and k is an integer such that there is no k'-transfer set w.r.t. (S, G) for all nonnegative integers $k' < k$.

Output: All k-transfer sets w.r.t. (S, G).

For convenience, we refer to the number k in an input $(S, G, c_t, c_d, c_\ell, k)$ to the BTSE problem as the *given bound*.

3 Algorithm for the BTSE Problem

Throughout this section, fix an input $(S, G, c_t, c_d, c_\ell, k)$ to the BTSE problem. To avoid triviality, we assume that the cost c_t of an LGT is a positive integer. For a subset U of $V(G)$, $E[U]$ denotes the set of all edges (u, v) in G with $u \in U$.

3.1 Outline of the Algorithm

To enumerate all k-transfer sets w.r.t. (S, G), we start with an empty set R and try all possible ways to extend it to a k-transfer set w.r.t. (S, G) by *gradually* adding edges to R. While extending R by adding more edges to it, we can also find some vertices u such that both edges connecting u to its children have not been added to R and should not be considered for addition to R by the algorithm later on. Such vertices u will always remain to be duplication or speciation vertices w.r.t. R. By using a set U to memorize such vertices, we can speed up extending R. Note that U is initially empty. Another merit of maintaining U is that it enables us to estimate a lower bound on $\min_{R^*}\mathrm{Cost}(R^*)$, where R^* ranges over all transfer sets w.r.t. (S, G) such that $R \subseteq R^*$ and R^* contains no edge (u, v) with $u \in U$. Initially, the lower bound is 0 because both R and U are empty. However, gradually extending R (and also U) will gradually increase the lower bound. Once the lower bound becomes larger than k, we can stop extending R.

As outlined in the above, our algorithm needs to always maintain a pair (R, U). Indeed, we always want (R, U) to satisfy the following two conditions:

C1. R is a transfer set w.r.t. (S, G).
C2. U is a set of nonleaf vertices of G with $E[U] \cap R = \emptyset$.

For convenience, we refer to each (R, U) satisfying Conditions C1 and C2 as a *legal pair*. Clearly, $(\emptyset, \emptyset)$ is a legal pair.

Example 1. Let S and G be as in Fig. 1, and R_1 and R_2 be as in Fig. 2. Then, both (R_1, U_1) and (R_2, U_2) are legal pairs, where $U_1 = \{e, f, h, i, p, r\}$ and $U_2 = \{b, j\}$.

We say that a subset R' of $E(G)$ *expands* a legal pair (R, U) if $R \subseteq R'$ and $R' \cap E[U] = \emptyset$. At the heart of our algorithm will be a recursive subroutine (called *AllTransSets*) for the following problem:

Input: A legal pair (R, U).

Output: All k-transfer sets R^* w.r.t. (S, G) that expand (R, U).

Obviously, to solve the BTSE problem for $(S, G, c_t, c_d, c_\ell, k)$, it suffices to call *AllTransSets* on input $(\emptyset, \emptyset)$.

The remainder of this section is organized as follows: in Section 3.2, we show several useful properties of legal pairs that indeed hold for more general pairs. In Section 3.3, we use these properties to show, for a legal pair (R, U), how to estimate a lower bound on $\min_{R^*}\mathrm{Cost}(R^*)$, where R^* ranges over all transfer sets w.r.t. (S, G) that expand (R, U). Then, in Section 3.4, we describe a case where it is meaningless to extend a legal pair (R, U) any more. In Section 3.5, we show that if the case in Section 3.4 does not happen for a legal pair (R, U), then we can always extend (R, U). Finally, we detail *AllTransSets* in Section 3.6 and analyze the running time of the whole algorithm in Section 3.7.

3.2 Semilegal Pairs and Their Properties

For convenience, we relax Condition C1 as follows:

C1′. No two edges of R are siblings.

We refer to each pair (R, U) satisfying Conditions C1 and C2 as a *semilegal pair*. Throughout this subsection, fix a semilegal pair (R, U).

The next lemma has been implicitly proved in [18] (for a different proof, see the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2012.79).

Lemma 1. *Suppose that u is a speciation vertex w.r.t. R. Let R^* be a superset of R such that $E[\{u\}] \cap R^* = \emptyset$ and R^* contains no sibling edges. Then, $M_R(u) = M_{R^*}(u)$. Moreover, u is also a speciation vertex w.r.t. R^*.*

A speciation vertex u w.r.t. R is *final* w.r.t. (R, U) if $u \in U$. In Example 1 (cf. Fig. 2), h, i, and r are final speciation vertices w.r.t. (R_1, U_1), while j is a final speciation vertex w.r.t. (R_2, U_2). The next lemma follows from Lemma 1 immediately.

Lemma 2. *If u is a final speciation vertex w.r.t. (R, U) and R^* is a subset of $E(G)$ which expands (R, U) and contains no sibling edges, then $M_R(u) = M_{R^*}(u)$ and u is a speciation vertex w.r.t. R^*.*

A duplication vertex z w.r.t. R is *final* w.r.t. (R, U) if $G \between R$ contains a directed path Q from z to a proper descendant u such that the following conditions hold:

C3. Each vertex of Q other than u belongs to U.
C4. Either $u \in U$ or u is a leaf of G.
C5. For every vertex y of Q, $M_R(y) = M_R(u)$.
C6. If $u \in U$, then u is a final speciation vertex w.r.t. (R, U).

For convenience, we call the above path Q a *duplication-witness path* for z w.r.t. (R, U). In Example 1 (cf. Fig. 2), e, f, and p are final duplication vertices w.r.t. (R_1, U_1), while b is a final duplication vertex w.r.t. (R_2, U_2); the directed path (shown by bold lines in Fig. 2) from e to h in $G \between R_1$ is a duplication-witness path for e w.r.t. (R_1, U_1) and also contains a duplication-witness path for f w.r.t. (R_1, U_1), while the directed path (shown by bold lines in Fig. 2) from b to j in $G \between R_2$ is a duplication-witness path for b w.r.t. (R_2, U_2).

The next lemma has been implicitly proved in [18] (for a different proof, see the online supplementary material).

Lemma 3. *Suppose that z is a final duplication vertex w.r.t. (R, U). Let R^* be a subset of $E(G)$ which expands (R, U) and contains no sibling edges. Then, the following hold:*

1. *$z \in \mathrm{DV}(R^*)$ and $M_{R^*}(z) = M_R(z)$.*
2. *If x is a child of z in G with $M_R(z) >_S M_R(x)$, then $M_{R^*}(z) >_S M_{R^*}(x)$.*

Lemma 4. *Suppose that $e = (u, v)$ is an edge of G such that u is a final speciation or duplication vertex w.r.t. (R, U). Let R^* be a subset of $E(G)$ which expands (R, U) and contains no sibling edges. Then, $\mathrm{Loss}_R(e) \le \mathrm{Loss}_{R^*}(e)$.*

Proof. By Lemmas 2 and 3, $M_{R^*}(u) = M_R(u)$. Morever, since $R \subseteq R^*$, $M_{R^*}(v) \le_S M_R(v)$. Hence, $\mathrm{Loss}_R(e) \le \mathrm{Loss}_{R^*}(e)$. □

3.3 Estimating a Lower Bound on the Minimum Cost w.r.t. a Legal Pair

Throughout this section, fix a legal pair (R, U).

The next theorem follows from Lemmas 3 and 4 immediately.

Theorem 5. *Let n_{fd} be the number of final duplication vertices w.r.t. (R, U). Let $n_{fl} = \sum_e \mathrm{Loss}_R(e)$, where e ranges over all edges (u, v) of G such that u is a final speciation or duplication* *vertex w.r.t. (R, U). Then, $c_t \cdot |R| + c_d \cdot n_{fd} + c_\ell \cdot n_{fl}$ is a lower bound on $\min_{R^*}\mathrm{Cost}(R^*)$, where R^* ranges over all transfer sets w.r.t. (S, G) which expands (R, U).*

For convenience, we call the lower bound in Theorem 5 the *lower bound* w.r.t. (R, U). In Example 1 (cf. Fig. 2), the lower bounds w.r.t. (R_1, U_1) and (R_2, U_2) are $3c_t + 3c_d + 5c_\ell$ and $c_t + c_d + 4c_\ell$, respectively.

By Theorem 5, we have the following simple but important observation:

Corollary 6. *If (R, U) is a legal pair such that the lower bound w.r.t. (R, U) is larger than the given bound k, then AllTransSets can stop immediately on input (R, U) (without outputting any k-transfer set w.r.t. (S, G)).*

One may wonder whether it is always true that for a legal pair (R, U), every vertex $u \in U$ is a final speciation or duplication vertex w.r.t. (R, U). It is fairly easy to see that Conditions C1 and C2 are not sufficient to guarantee this to be true. Fortunately, we will design *AllTransSets* so that its input is always a legal pair (R', U') satisfying the following condition:

C7. Every vertex $u \in U'$ is a final speciation or duplication vertex w.r.t. (R', U').

For convenience, we refer to each legal pair (R', U') satisfying Condition C7 as a *valid pair*. Obviously, $(\emptyset, \emptyset)$ is a valid pair.

Basically, *AllTransSets* will start on input $(R, U) = (\emptyset, \emptyset)$, then extend (R, U) (in one or more ways) without losing its validity, and further recurse on each extension. Hereafter, an *extension* of (R, U) is a valid pair (R', U') such that $R \subseteq R'$, $U \subseteq U'$, and $R \neq R'$ or $U \neq U'$. *AllTransSets* will recurse only on those extensions (R', U') of (R, U) such that the lower bound w.r.t. (R', U') is larger than the lower bound w.r.t. (R, U).

3.4 A Case Not to Extend a Valid Pair

Throughout this section, fix a valid pair (R, U). Our goal is to find a sufficient condition on (R, U) under which it is meaningless to extend (R, U) any more. By Corollary 6, one such condition is that the lower bound w.r.t. (R, U) is larger than the given bound k. In this section, we will find another such condition.

Lemma 7. *Suppose that u is a vertex of G such that either every proper ancestor of u in $G \setminus R$ has only one child in $G \setminus R$ or the lowest proper ancestor of u in $G \setminus R$ that has two children in $G \setminus R$ belongs to U. Let R^* be a transfer set w.r.t. (S, G) which expands (R, U). Then, R' is also a transfer set w.r.t. (S, G), where R' is the set obtained from R^* by removing all edges (x, y) such that $x \leq_{G\setminus R} u$ and $(x, y) \notin R$.*

Proof. Let $G' = G \setminus R'$. Consider an arbitrary edge $(v, w) \in R'$. We want to show that $M_{R'}(v)$ and $M_{R'}(w)$ are incomparable in S. The simplest case is when u and v are incomparable in G'. In this case, $M_{R'}(v) = M_{R^*}(v)$, u and w are also incomparable in G', and $M_{R'}(w) = M_{R^*}(w)$. So, we are done in this case because R^* is a transfer set w.r.t. (S, G) and hence $M_{R^*}(v)$ and $M_{R^*}(w)$ are incomparable in S. Thus, we may assume that $v \leq_{G'} u$ or $u <_{G'} v$. Clearly, one of the following cases occurs.

Case 1: $v \leq_{G'} u$ (see Fig. 1 in the online supplementary material). In this case, $(v, w) \in R$ and $M_{R'}(v) = M_R(v)$. Moreover, since $R \subseteq R'$ and v and w are incomparable in $G \setminus R$, $M_{R'}(w) \leq_S M_R(w)$. Now, because R is a transfer set w.r.t. (S, G), $M_R(v)$, and $M_R(w)$ are incomparable in S and in turn so are $M_{R'}(v)$ and $M_{R'}(w)$.

Case 2: $u <_{G'} v$ and every proper ancestor of u in $G \setminus R$ that is also a descendant of v in $G \setminus R$ has only one child in $G \setminus R$ (see Fig. 2 in the online supplementary material). In this case, since $R \subseteq R^*$ and R^* is a transfer set w.r.t. (S, G), the path from v to u in $G \setminus R$ remains to be a path in $G \setminus R'$. Thus, $(v, w) \in R$ and $M_{R'}(v) = M_R(v) = M_R(u)$. So, we can proceed as in Case 1 to show that $M_{R'}(v)$ and $M_{R'}(w)$ are incomparable in S.

Case 3: $u <_{G'} v$ and some proper ancestor of u in $G \setminus R$ that is also a descendant of v in $G \setminus R$ has two children in $G \setminus R$. In this case, let p be the lowest proper ancestor of u in $G \setminus R$ that has two children in $G \setminus R$. Obviously, $p \leq_{G\setminus R} v$ and $p \in U$. Indeed, $p \neq v$ because $p \in U$ and $E[\{v\}] \cap R^* \neq \emptyset$. Hence, one of the following two cases occurs.

Case 3.1: $p \leq_{G\setminus R} w$ (see Fig. 3 in the online supplementary material). In this case, v and u are incomparable in $G \setminus R$ and hence $M_{R'}(v) = M_{R^*}(v)$. Moreover, if w and p are incomparable in $G \setminus R^*$, then clearly $M_{R'}(w) = M_{R^*}(w)$; otherwise, by Lemmas 2 and 3, $M_{R'}(p) = M_{R^*}(p) = M_R(p)$ and in turn $M_{R'}(w) = M_{R^*}(w)$. So, we always have $M_{R'}(w) = M_{R^*}(w)$ and $M_{R'}(v) = M_{R^*}(v)$. Therefore, $M_{R'}(v)$ and $M_{R'}(w)$ are incomparable in S because R^* is a transfer set w.r.t. (S, G).

Case 3.2: p and w are incomparable in $G \setminus R$ (see Fig. 4 in the online supplementary material). In this case, w and u are incomparable in $G \setminus R$ and hence $M_{R'}(w) = M_{R^*}(w)$. Moreover, if v and p are incomparable in $G \setminus R^*$, then clearly $M_{R'}(v) = M_{R^*}(v)$; otherwise, by Lemmas 2 and 3, $M_{R'}(p) = M_{R^*}(p) = M_R(p)$ and in turn $M_{R'}(v) = M_{R^*}(v)$. So, we always have $M_{R'}(w) = M_{R^*}(w)$ and $M_{R'}(v) = M_{R^*}(v)$. Hence, $M_{R'}(v)$ and $M_{R'}(w)$ are incomparable in S because R^* is a transfer set w.r.t. (S, G). □

We classify speciation vertices w.r.t. R into three types as follows: let u be a speciation vertex w.r.t. R, and v and w be the children of u in G.

- If $c_\ell \cdot (\mathrm{Loss}_R(u, v) + \mathrm{Loss}_R(u, w)) < c_t$, then u is a *tight* speciation vertex w.r.t. R.
- If $c_t \leq c_\ell \cdot (\mathrm{Loss}_R(u, v) + \mathrm{Loss}_R(u, w)) < c_t + c_\ell$, then u is a *mild* speciation vertex w.r.t. R.
- If $c_\ell \cdot (\mathrm{Loss}_R(u, v) + \mathrm{Loss}_R(u, w)) \geq c_t + c_\ell$, then u is a *loose* speciation vertex w.r.t. R.

In Example 1 (cf. Fig. 2), if $c_t = c_d = 2$ and $c_\ell = 1$, then d, h, and i are tight speciation vertices w.r.t. R_2, r is a mild speciation vertex w.r.t. R_2, and j is a loose speciation vertex w.r.t. R_2.

If a vertex u of G is a leaf of G, belongs to U, or is a transfer vertex w.r.t. R, then u is *marked* w.r.t. (R, U); otherwise, it is *unmarked* w.r.t. (R, U).

A vertex u of G is *settled* w.r.t. (R, U) if each unmarked descendant of u in $G \setminus R$ w.r.t. (R, U) is a tight speciation vertex w.r.t. R. In Example 1 (cf. Fig. 2), if $c_t = c_d = 2$ and $c_\ell = 1$, then only the root a of G is not settled w.r.t. (R_1, U_1), while only h, i, and j are settled w.r.t. (R_2, U_2). The next lemma justifies the naming of a settled vertex.

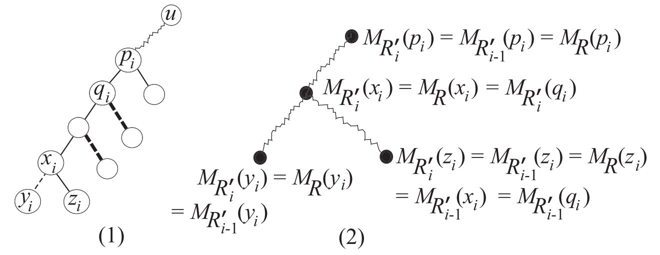

u
p_i
q_i
x_i
y_i
z_i
(1)
$M_{R'_i}(p_i) = M_{R'_{i-1}}(p_i) = M_R(p_i)$
$M_{R'_i}(x_i) = M_R(x_i) = M_{R'_i}(q_i)$
$M_{R'_i}(y_i) = M_R(y_i) = M_{R'_{i-1}}(y_i)$
$M_{R'_i}(z_i) = M_{R'_{i-1}}(z_i) = M_R(z_i) = M_{R'_{i-1}}(x_i) = M_{R'_{i-1}}(q_i)$
(2)

Fig. 4. Illustrations for Case 1' in the proof of Lemma 8. (1) A part of G and (2) a part of S, where zig-zag lines are paths in G or S, bold dashed lines are edges in R'_i, and the thin dashed line is the unique edge in $R'_{i-1} \setminus R'_i$.

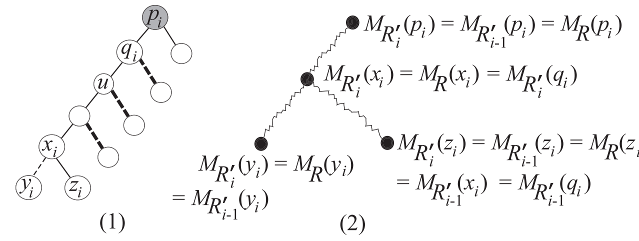

p_i
q_i
u
x_i
y_i
z_i
(1)
$M_{R'_i}(p_i) = M_{R'_{i-1}}(p_i) = M_R(p_i)$
$M_{R'_i}(x_i) = M_R(x_i) = M_{R'_i}(q_i)$
$M_{R'_i}(y_i) = M_R(y_i) = M_{R'_{i-1}}(y_i)$
$M_{R'_i}(z_i) = M_{R'_{i-1}}(z_i) = M_R(z_i) = M_{R'_{i-1}}(x_i) = M_{R'_{i-1}}(q_i)$
(2)

Fig. 5. Illustrations for Case 2' in the proof of Lemma 8. (1) A part of G and (2) a part of S, where zig-zag lines are paths in S, bold dashed lines are edges in R'_i, and the thin dashed line is the unique edge in $R'_{i-1} \setminus R'_i$.

Lemma 8. *Suppose that u is a settled vertex of G w.r.t. (R, U) such that either every proper ancestor of u in $G \setminus R$ has only one child in $G \setminus R$ or the lowest proper ancestor of u in $G \setminus R$ that has two children in $G \setminus R$ belongs to U. Let R^* be a k-transfer set w.r.t. (S, G) which expands (R, U). Then, G has no edge (v, w) such that $v \le_{G \setminus R} u$ and $(v, w) \in R^* \setminus R$. Consequently, u is still settled w.r.t. every valid pair (R^*, U^*) with $U \subseteq U^*$.*

Proof. Consider the vertex p of G defined as follows: if every proper ancestor of u in $G \setminus R$ has only one child in $G \setminus R$, then p is the root of $G \setminus R$ with $p \ge_{G \setminus R} u$; otherwise, p is the lowest proper ancestor of u in $G \setminus R$ that has two children in $G \setminus R$.

Toward a contradiction, assume that G has an edge (u_1, u_2) such that $u_1 \le_{G \setminus R} u$ and $(u_1, u_2) \in R^* \setminus R$. Then, u_1 is an unmarked tight speciation vertex w.r.t. (R, U). Let R' be the set obtained from R^* by removing all edges (x, y) such that $x \le_{G \setminus R} u$ and $(x, y) \notin R$. Then, $(u_1, u_2) \in R^* \setminus R'$ and in turn $|R'| < |R^*|$. For convenience, let $G' = G \setminus R'$. By Lemma 7, R' is a transfer set w.r.t. (S, G). So, in order to obtain a contradiction, it suffices to prove that $\mathrm{Cost}(R') < \mathrm{Cost}(R^*)$.

We claim that $\mathrm{DV}(R') \subseteq \mathrm{DV}(R^*)$. To see this claim, consider an arbitrary vertex $x \in \mathrm{DV}(R')$. Let y and z be the children of x in G. Then, $(x, y) \notin R'$ and $(x, z) \notin R'$. So, $(x, y) \notin R$ and $(x, z) \notin R$ for $R \subseteq R'$. Moreover, $M_{R'}(y)$ and $M_{R'}(z)$ are comparable in S. In order to prove that $x \in \mathrm{DV}(R^*)$, it suffices to prove that $M_{R^*}(y)$ and $M_{R^*}(z)$ are comparable in S. If x and u are incomparable in G', then $M_{R'}(y) = M_{R^*}(y)$ and $M_{R'}(z) = M_{R^*}(z)$, implying that $M_{R^*}(y)$ and $M_{R^*}(z)$ are comparable in S. So, we may assume that $x \le_{G'} u$ or $x >_{G'} u$. Clearly, one of the following three cases occurs.

Case 1: $x \le_{G'} u$. In this case, $x \in \mathrm{DV}(R)$ because $M_{R'}(v) = M_R(v)$ for every vertex v of G' with $v \le_{G'} u$. This together with the fact that u is settled w.r.t. (R, U) implies that $x \in U$ and in turn x is a final duplication vertex w.r.t. (R, U). Thus, by Lemma 3, $x \in \mathrm{DV}(R^*)$.

Case 2: $x >_{G'} p$. In this case, $p \in U$ and in turn p is a final speciation or duplication vertex w.r.t. (R, U). Moreover, $y \ge_{G'} p$ or $z \ge_{G'} p$. Without loss of generality, we may assume that $y \ge_{G'} p$. Then, z and u are incomparable in G and in turn $M_{R'}(z) = M_{R^*}(z)$. Furthermore, if y and u are incomparable in $G \setminus R^*$, then clearly $M_{R'}(y) = M_{R^*}(y)$; otherwise, by Lemmas 2 and 3, $M_R(p) = M_{R'}(p) = M_{R^*}(p)$ and in turn $M_{R'}(y) = M_{R^*}(y)$. Thus, we always have $M_{R'}(y) = M_{R^*}(y)$ and $M_{R'}(z) = M_{R^*}(z)$. Hence, $M_{R^*}(y)$ and $M_{R^*}(z)$ are comparable in S because so are $M_{R'}(y)$ and $M_{R'}(z)$.

Case 3: $x = p >_{G'} u$. In this case, $p \in U$ and in turn x is a final speciation or duplication vertex w.r.t. (R, U). If x were a final speciation vertex w.r.t. (R, U), then by Lemma 2, x would not be a duplication vertex w.r.t. $M_{R'}$. Thus, x is a final duplication vertex w.r.t. (R, U). Hence, by Lemma 3, x is a duplication vertex w.r.t. M_{R^*}.

By the above case-analysis, the claim holds. So, in order to prove that $\mathrm{Cost}(R') < \mathrm{Cost}(R^*)$, it suffices to prove the following inequality:

$$c_\ell \cdot \sum_{(v,w) \in E(G)} (\mathrm{Loss}_{R'}(v, w) - \mathrm{Loss}_{R^*}(v, w)) < c_t \cdot |R^* \setminus R'|. \tag{1}$$

To prove Inequality 1, let $(x_1, y_1), (x_2, y_2), \ldots, (x_d, y_d)$ be the edges in $R^* \setminus R'$, where we assume that for every pair (i, j) of integers with $1 \le i < j \le d$, x_i is not an ancestor of x_j in G. For each $1 \le i \le d$, let R'_i be the set obtained from R^* by deleting all (x_j, y_j) with $1 \le j \le i$. Note that $R' = R'_d$. For convenience, let $R'_0 = R^*$.

We first claim that for each $1 \le i \le d$, R'_i is a transfer set w.r.t. (S, G). We prove the claim by induction on $d - i$. In the base case, $d - i = 0$ and hence the claim is clearly true because $R'_d = R'$ is a transfer set w.r.t. (S, G). Assume that $d - i \ge 0$ and R'_i is a transfer set w.r.t. (S, G). Since R^* expands (R, U) and $(x_i, y_i) \in R^* \setminus R$, we know that x_i is unmarked w.r.t. (R, U). This together with the assumption that u is settled w.r.t. (R, U) implies that x_i is a speciation vertex w.r.t. R. In addition, we clearly have that the subtree of $G \setminus R'_i$ rooted at x_i is the same as the subtree of $G \setminus R$. Hence, x_i is also a speciation vertex w.r.t. R'_i. Therefore, $R'_i \cup \{(x_i, y_i)\}$ (i.e., R'_{i-1}) is a transfer set w.r.t. (S, G). This completes the proof of the claim.

We then claim that for every $1 \le i \le d$, the following inequality holds:

$$c_\ell \cdot \sum_{(v,w) \in E(G)} \Big(\mathrm{Loss}_{R'_i}(v, w) - \mathrm{Loss}_{R'_{i-1}}(v, w)\Big) < c_t. \tag{2}$$

Note that this claim implies Inequality 1 immediately. So, it remains to prove this claim. To this end, consider an arbitrary integer i with $1 \le i \le d$. Let z_i be the child of x_i in G other than y_i. Obviously, $M_{R'_i}(x_i) = M_R(x_i)$, $M_{R'_{i-1}}(y_i) = M_{R'_i}(y_i) = M_R(y_i)$, and $M_{R'_{i-1}}(x_i) = M_{R'_{i-1}}(z_i) = M_{R'_i}(z_i) = M_R(z_i)$. See Figs. 4 and 5. As aforementioned in this proof, x_i is a speciation vertex w.r.t. R'_i. Hence, we have

- $$\begin{aligned}\mathrm{Loss}_{R'_i}(x_i, y_i) &= \mathrm{Dist}_S(M_{R'_i}(x_i), M_{R'_i}(y_i)) - 1\\ &= \mathrm{Dist}_S(M_R(x_i), M_R(y_i)) - 1\\ &= \mathrm{Loss}_R(x_i, y_i) \text{ and}\end{aligned}$$

- $$\begin{aligned}\mathrm{Loss}_{R'_i}(x_i, z_i) &= \mathrm{Dist}_S(M_{R'_i}(x_i), M_{R'_i}(z_i)) - 1\\ &= \mathrm{Dist}_S(M_R(x_i), M_R(z_i)) - 1\\ &= \mathrm{Loss}_R(x_i, z_i).\end{aligned}$$

Now, if every proper ancestor of x_i in $G \setminus R'_i$ has only one child in $G \setminus R'_i$, then the left-hand side of Inequality 2 is $c_\ell \cdot (\mathrm{Loss}_{R'_i}(x_i, y_i) + \mathrm{Loss}_{R'_i}(x_i, z_i)) = c_\ell \cdot (\mathrm{Loss}_R(x_i, y_i) + \mathrm{Loss}_R(x_i, z_i))$, which is less than c_t because x_i is a tight speciation vertex w.r.t. R. Thus, we may assume that some proper ancestor of x_i in $G \setminus R'_i$ has two children in $G \setminus R'_i$. Let p_i be the lowest proper ancestor of x_i in $G \setminus R'_i$ that has two children in $G \setminus R'_i$. We distinguish two cases as follows:

Case 1′: $p_i \le_{G\setminus R} u$ (see Fig. 4). If p_i is marked w.r.t. (R, U), then $p_i \in U$ and in turn $M_{R'_i}(p_i) = M_{R'_{i-1}}(p_i) = M_R(p_i)$ by Lemmas 2 and 3. On the other hand, if p_i is unmarked w.r.t. (R, U), then the fact that u is settled w.r.t. (R, U) implies that p_i is an unmarked speciation vertex w.r.t. (R, U) and in turn $M_{R'_i}(p_i) = M_{R'_{i-1}}(p_i) = M_R(p_i)$ by Lemma 1. Hence, we always have $M_{R'_i}(p_i) = M_{R'_{i-1}}(p_i) = M_R(p_i)$. Now, letting q_i be the child of p_i in G with $q_i \ge_G x_i$, we know that the left-hand side of Inequality 2 is

$$\begin{aligned}c_\ell \cdot \big(&\mathrm{Loss}_{R'_i}(p_i, q_i) + \mathrm{Loss}_{R'_i}(x_i, y_i) + \mathrm{Loss}_{R'_i}(x_i, z_i)\\ &- \mathrm{Loss}_{R'_{i-1}}(p_i, q_i)\big)\end{aligned}$$

Obviously,

$$\begin{aligned}\mathrm{Loss}_{R'_i}(p_i, q_i) &\le \mathrm{Dist}_S(M_{R'_i}(p_i), M_{R'_i}(q_i))\\ &= \mathrm{Dist}_S(M_R(p_i), M_R(x_i))\end{aligned}$$

and $\mathrm{Loss}_{R'_{i-1}}(p_i, q_i) \ge \mathrm{Dist}_S(M_{R'_{i-1}}(p_i), M_{R'_{i-1}}(q_i)) - 1 = \mathrm{Dist}_S(M_R(p_i), M_R(z_i)) - 1$. Also recall that $\mathrm{Loss}_{R'_i}(x_i, y_i) = \mathrm{Loss}_R(x_i, y_i)$ and $\mathrm{Loss}_{R'_i}(x_i, z_i) = \mathrm{Dist}_S(M_R(x_i), M_R(z_i)) - 1$. Thus, the left-hand side of Inequality 2 is at most

$$\begin{aligned}c_\ell \cdot \big(&\mathrm{Dist}_S(M_R(p_i), M_R(x_i)) + \mathrm{Loss}_R(x_i, y_i)\\ &+ \mathrm{Dist}_S(M_R(x_i), M_R(z_i)) - \mathrm{Dist}_S(M_R(p_i), M_R(z_i))\big)\\ = c_\ell &\cdot \mathrm{Loss}_R(x_i, y_i) < c_t,\end{aligned}$$

where the last inequality holds because x_i is a tight speciation vertex w.r.t. R.

Case 2′: $p_i >_{G\setminus R} u$ (see Fig. 5). In this case, $p_i = p$ and $p \in U$. So, $M_{R'_i}(p_i) = M_{R'_{i-1}}(p_i) = M_R(p_i)$. Thus, as in Case 1′, we can see that the left-hand side of Inequality 2 is at most $c_\ell \cdot \mathrm{Loss}_R(x_i, y_i) < c_t$, which is less than c_t because x_i is a tight speciation vertex w.r.t. R. This finishes the proof. □

By Lemma 8, we have the following corollary immediately.

Corollary 9. *If every vertex of G is settled w.r.t. (R, U), then no k-transfer set R^* w.r.t. (S, G) expands (R, U).*

By Corollary 9, it is meaningless to extend (R, U) once every vertex of G becomes settled w.r.t. (R, U).

3.5 Extending a Valid Pair

Throughout this section, fix a valid pair (R, U) such that at least one vertex of G is not settled w.r.t. (R, U). We will show that we can always extend (R, U) so that we get closer to the case in Section 3.4.

A *complete extension set* of (R, U) is a set $\mathcal{P}$ of extensions of (R, U) such that every k-transfer set w.r.t. (S, G) expanding (R, U) also expands at least one pair in $\mathcal{P}$. Obviously, if $\mathcal{P}$ is a complete extension set of (R, U), then we can find all k-transfer sets w.r.t. (S, G) expanding (R, U) as follows: for all $(R', U') \in \mathcal{P}$, find all k-transfer sets w.r.t. (S, G) expanding (R', U').

Basically, *AllTransSets* will start on input $(R, U) = (\emptyset, \emptyset)$, then construct a complete extension set $\mathcal{P}$ of (R, U), and further recurse on each pair in $\mathcal{P}$. As mentioned before, we will always require that the lower bound w.r.t. each pair in $\mathcal{P}$ is larger than the lower bound w.r.t. (R, U). To make *AllTransSets* run fast, we want to make $\mathcal{P}$ as small as possible and make the lower bound w.r.t. each pair in $\mathcal{P}$ as large as possible.

The next lemma has been proved in [18] (for a different proof, see the online supplementary material).

Lemma 10. *Suppose that u is an unmarked speciation vertex w.r.t. (R, U). Let v and w be the children of u in G. Then, the following statements hold:*

1. *$(R \cup \{(u, v)\}, U)$, $(R \cup \{(u, w)\}, U)$, and $(R, U \cup \{u\})$ are valid pairs and they together form a complete extension set of (R, U).*
2. *If u is a mild or loose speciation vertex w.r.t. R, then the lower bound w.r.t. each of $(R \cup \{(u, v)\}, U)$, $(R \cup \{(u, w)\}, U)$, and $(R, U \cup \{u\})$ is larger than the lower bound w.r.t. (R, U) by at least c_t.*

The complete extension set of (R, U) suggested by Statement 1 in Lemma 10 is not good enough to achieve the desired time complexity of our algorithm. So, we next consider how to find better complete extension sets of (R, U).

A *d-move* w.r.t. (R, U) is an unmarked vertex u w.r.t. (R, U) such that for some $v \in U$ or some leaf v of G, $(u, v) \in E(G \between R)$ and $M_R(u) = M_R(v)$. In Example 1 (cf. Fig. 2), the root a of G is a d-move w.r.t. (R_2, U_2).

The next lemma has been proved in [18] (for a different proof, see the online supplementary material).

Lemma 11. *Suppose that u is a d-move w.r.t. (R, U). Let R^* be a transfer set w.r.t. (S, G) which expands (R, U). Then, the following statements hold:*

1. *R^* contains no edge in $E[\{u\}]$.*
2. *$(R, U \cup \{u\})$ is a valid pair and it alone forms a complete extension set of (R, U).*
3. *The lower bound w.r.t. $(R, U \cup \{u\})$ is larger than the lower bound w.r.t. (R, U) by at least c_d.*

To define the next type of moves, we classify speciation vertices w.r.t. R into two types as follows: for a speciation vertex u w.r.t. R, if the children v and w of u in G satisfy that $M_R(u)$ is the parent of both $M_R(v)$ and $M_R(w)$ in S, then u is a *special* speciation vertex w.r.t. R; otherwise, u is a *nonspecial* speciation vertex w.r.t. R. In Example 1 (cf. Fig. 2), h and i are special speciation vertices w.r.t. R_2, while d, j, and r are nonspecial speciation vertices w.r.t. R_2. Note that if u is a special speciation vertex w.r.t. R and its children in G are v and w, then $\mathrm{Loss}_R(u,v) = 0$ and $\mathrm{Loss}_R(u,w) = 0$, and in turn u is a tight speciation vertex w.r.t. R because we assume $c_t > 0$.

A *singleton s-move* w.r.t. (R,U) is an unmarked speciation vertex u w.r.t. (R,U) such that the following conditions hold:

C8. u is not a root of $G \between R$.
C9. The parent p of u in $G \between R$ does not belong to U, and $M_R(p) = M_R(u)$.
C10. u is a nonspecial speciation vertex w.r.t. R, or the sibling u' of u in $G \between R$ satisfies that $M_R(p) \neq M_R(u')$.

In Example 1 (cf. Fig. 2), b is a singleton s-move w.r.t. (R_1, U_1), and r is a singleton s-move w.r.t. (R_2, U_2).

The next lemma has been implicitly proved in [18] (for a different proof, see the online supplementary material):

Lemma 12. *Suppose that u is a singleton s-move w.r.t. (R,U). Let v and w be the children of u in G, p be the parent of u in $G \between R$, and R^* be a transfer set w.r.t. (S,G) that expands (R,U). Then, the following statements hold:*

1. *Either $E[\{p,u\}] \cap R^* = \emptyset$, or $E[\{u\}] \cap R^*$ is equal to $\{(u,v)\}$ or $\{(u,w)\}$.*
2. *$(R \cup \{(u,v)\}, U)$, $(R \cup \{(u,w)\}, U)$, and $(R, U \cup \{p,u\})$ are valid pairs and they together form a complete extension set of (R,U).*
3. *The lower bounds w.r.t. $(R \cup \{(u,v)\}, U)$, $(R \cup \{(u,w)\}, U)$, and $(R, U \cup \{p,u\})$ are larger than the lower bound w.r.t. (R,U) by at least c_t, c_t, and $c_d + c_\ell$, respectively.*

A *twin s-move* w.r.t. (R,U) is an unordered pair $\{u,u'\}$ of unmarked speciation vertices w.r.t. (R,U) satisfying the following conditions:

C11. u and u' are siblings in $G \between R$ and both are tight speciation vertices w.r.t. R.
C12. The parent p of u and u' in $G \between R$ is unmarked w.r.t (R,U).
C13. $M_R(p) = M_R(u) = M_R(u')$.
C14. The children of u in G are settled w.r.t. (R,U) and so are the children of u' in G.

In Example 1 (cf. Fig. 2), if $c_t = c_d = 2$ and $c_\ell = 1$, then $\{h,i\}$ is a twin s-move w.r.t. (R_2, U_2).

Lemma 13. *Suppose that $\{u,u'\}$ is a twin s-move w.r.t. (R,U). Let v and w be the children of u in G, v' and w' be the children of u' in G, and p be the parent of u and u' in $G \between R$. Assume that R^* is a k-transfer set w.r.t. (S,G) which expands (R,U). Then, the following statements hold:*

1. *Either $E[\{p,u,u'\}] \cap R^* = \emptyset$, or $E[\{u,u'\}] \cap R^*$ is equal to $\{(u,v),(u',v')\}$, $\{(u,v),(u',w')\}$, $\{(u,w),(u',v')\}$, or $\{(u,w),(u',w')\}$.*
2. *$(R \cup \{(u,v),(u',v')\}, U)$, $(R \cup \{(u,v),(u',w')\}, U)$, $(R \cup \{(u,w),(u',v')\}, U)$, $(R \cup \{(u,w),(u',w')\}, U)$, and $(R, U \cup \{p,u,u'\})$ are valid pairs and they together form a complete extension set of (R,U).*
3. *The lower bounds w.r.t. $(R \cup \{(u,v),(u',v')\}, U)$, $(R \cup \{(u,v),(u',w')\}, U)$, $(R \cup \{(u,w),(u',v')\}, U)$, $(R \cup \{(u,w),(u',w')\}, U)$, and $(R, U \cup \{p,u,u'\})$ are larger than the lower bound w.r.t. (R,U) by at least $2c_t$, $2c_t$, $2c_t$, $2c_t$, and c_d, respectively.*

Proof. To prove Statement 1, we distinguish three cases as follows:

Case 1: $E[\{u'\}] \cap R^* = \emptyset$. By Statement 1 in Lemma 10, $(R, U \cup \{u'\})$ is a valid pair. Moreover, since u' is a final speciation vertex w.r.t. $(R, U \cup \{u'\})$, p is a d-move w.r.t. $(R, U \cup \{u'\})$. Now, since $E[\{u'\}] \cap R^* = \emptyset$, we know that $R^* \cap E[U \cup \{u'\}] = \emptyset$ and in turn $E[\{p\}] \cap R^* = \emptyset$ by Statement 1 in Lemma 11. We further claim that $E[\{u\}] \cap R^* = \emptyset$. To see this claim, first note that $(R, U \cup \{u',p\})$ is a valid pair, because of Statement 2 in Lemma 11 and the fact that p is a d-move w.r.t. $(R, U \cup \{u'\})$. Moreover, since the children of u are settled w.r.t. (R,U), u is settled w.r.t. $(R, U \cup \{u',p\})$. So, by Lemma 8, $E[\{u\}] \cap R^* = \emptyset$. In summary, $E[\{u,u',p\}] \cap R^* = \emptyset$.

Case 2: $E[\{u\}] \cap R^* = \emptyset$. This case is symmetric to Case 1.

Case 3: $E[\{u\}] \cap R^* \neq \emptyset$ and $E[\{u'\}] \cap R^* \neq \emptyset$. Then, since R^* is a transfer set w.r.t (S,G), $E[\{u,u'\}] \cap R^*$ is equal to $\{(u,v),(u',v')\}$, $\{(u,v),(u',w')\}$, $\{(u,w),(u',v')\}$, or $\{(u,w),(u',w')\}$.

By the above case-analysis, Statement 1 holds.

By Statement 1 in Lemma 10, $(R \cup \{(u,v)\}, U)$, $(R \cup \{(u,w)\}, U)$, and $(R, U \cup \{u\})$ together form a complete extension set $\mathcal{P}$ of (R,U). Obviously, u' is an unmarked speciation vertex w.r.t. each pair in $\mathcal{P}$. So, again by Statement 1 in Lemma 10, $(R \cup \{(u,v), (u',v')\}, U)$, $(R \cup \{(u,v),(u',w')\}, U)$, $(R \cup \{(u,w),(u',v')\}, U)$, $(R \cup \{(u,w),(u',w')\}, U)$, and $(R, U \cup \{u,u'\})$ are extensions of (R,U). Since p is clearly a d-move w.r.t. $(R, U \cup \{u,u'\})$, $(R, U \cup \{p,u,u'\})$ is an extension of (R,U) by Statement 2 in Lemma 11. Hence, $(R \cup \{(u,v),(u',v')\}, U)$, $(R \cup \{(u,v),(u',w')\}, U)$, $(R \cup \{(u,w),(u',v')\}, U)$, $(R \cup \{(u,w),(u',w')\}, U)$, and $(R, U \cup \{p,u,u'\})$ are extensions of (R,U) and in turn they together form a complete extension set of (R,U) by Statement 1. This completes the proof of Statement 2. Statement 3 follows from Statement 2 immediately. □

Now, we define the last type of moves. An *e-move* w.r.t. (R,U) is an unmarked mild or loose speciation vertex u w.r.t. (R,U) such that

- the children of u in G are settled w.r.t. (R,U) and
- either u is a root of $G \between R$, or the parent of u in $G \between R$ belongs to U or is an unmarked speciation vertex w.r.t. R.

Example 2. Let S and G be as in Fig. 1. Then, as shown in Fig. 6, $(R_3, U_3) = (\emptyset, \{b,e,f,h,i,j\})$ is a valid pair. Moreover, if $c_t = c_d = 2$ and $c_\ell = 1$, then only r is an e-move w.r.t. (R_3, U_3). On the other hand, if $c_t = c_d = c_\ell = 1$, then both d and r are e-moves w.r.t. (R_3, U_3).

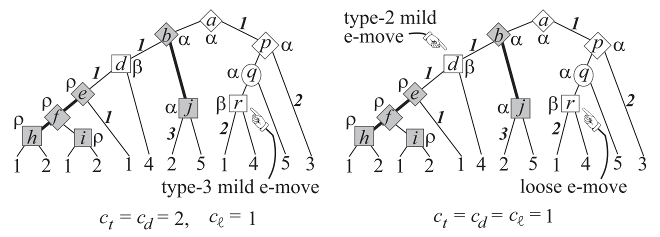


Fig. 6. The LCA mapping of the gene tree G in Fig. 1 into the species tree S in Fig. 1 associated with the transfer set $R_3 = \emptyset$, where (1) for each vertex v, the image of v under a mapping is placed near v; (2) speciation vertices and duplication vertices are enclosed by squares and diamonds, respectively; and (3) if the number of losses inferred by an edge e is positive, then it is placed near e in bold italic.

Statement 1 in Lemma 10 suggests a way to use an e-move w.r.t. (R, U) to find a complete extension set of (R, U). In order to find a better complete extension set of (R, U), we classify e-moves w.r.t. (R, U) into two types as follows: if u is a mild (respectively, loose) speciation vertex w.r.t. R, then u is a *mild* (respectively, *loose*) e-move w.r.t. (R, U). In Example 2, r is a mild e-move w.r.t. (R_3, U_3) if $c_t = c_d = 2$ and $c_\ell = 1$, while d is a mild e-move w.r.t. (R_3, U_3) and r is a loose e-move w.r.t. (R_3, U_3) if $c_t = c_d = c_\ell = 1$.

The next lemma is obvious and shows that loose e-moves are better than mild e-moves.

Lemma 14. *If u is a loose e-move w.r.t. (R, U), then the lower bound w.r.t. $(R, U \cup \{u\})$ is larger than the lower bound w.r.t. (R, U) by at least $c_t + c_\ell$.*

We further classify mild e-moves u w.r.t. (R, U) into three types as follows:

- u is of *type-1* if u is a root of $G \between R$.
- u is of *type-2* if u is not a root of $G \between R$ and its parent in $G \between R$ belongs to U.
- u is of *type-3* if u is not a root of $G \between R$ and its parent in $G \between R$ is an unmarked speciation vertex w.r.t. R.

In Example 2 (cf. Fig. 6), r is a type-3 mild e-move w.r.t. (R_3, U_3) if $c_t = c_d = 2$ and $c_\ell = 1$, while d is a type-2 mild e-move w.r.t. (R_3, U_3) if $c_t = c_d = c_\ell = 1$.

Lemma 15. *Suppose that u is a type-2 mild e-move w.r.t. (R, U). Let v and w be the children of u in G. Then, the following statements hold:*

1. *For every k-transfer set R^* w.r.t. (S, G) expanding (R, U), R^* contains neither (u, v) nor (u, w).*
2. *$(R, U \cup \{u\})$ is a valid pair and it alone forms a complete extension set of (R, U).*

Proof. (See Fig. 7.) Statement 2 follows from Statement 1 immediately. To prove Statement 1, consider an arbitrary k-transfer set R^* w.r.t. (S, G) which expands (R, U). Toward a contradiction, assume that $R^* \cap E[\{u\}] = \{(u, v)\}$ or $R^* \cap E[\{u\}] = \{(u, w)\}$. We assume that $R^* \cap E[\{u\}] = \{(u, v)\}$; the other case is similar. Let $R'' = R \cup \{(u, v)\}$ and p be the parent of u in $G \between R$. By Statement 1 in Lemma 10, (R'', U) is a valid pair. Moreover, u is settled w.r.t. (R'', U), $p \in U$ is the lowest ancestor of u in $G \setminus R''$ that has two children in $G \setminus R''$, and R^* expands (R'', U). So, by Lemma 8, G has no edge (x, y) such that $x \leq_{G \setminus R''} u$ and $(x, y) \in R^* \setminus R''$. Similarly, G has no edge

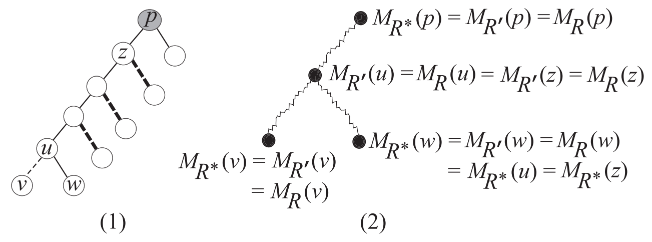

$M_{R^*}(p) = M_{R'}(p) = M_R(p)$
$M_{R'}(u) = M_R(u) = M_{R'}(z) = M_R(z)$
$M_{R^*}(v) = M_{R'}(v) = M_R(v)$
$M_{R^*}(w) = M_{R'}(w) = M_R(w) = M_{R^*}(u) = M_{R^*}(z)$
(1)
(2)

Fig. 7. (1) A part of G and (2) a part of S, where zig-zag lines are paths in S, bold dashed lines are edges in R, and the thin dashed line is the unique edge in $R^* \setminus R'$.

(x, y) such that $x \leq_{G \setminus R''} v$ and $(x, y) \in R^* \setminus R''$ because v is settled w.r.t. (R'', U) and is a root of $G \setminus R''$. Hence, there is no edge (x, y) other than (u, v) such that $x \leq_{G \setminus R} u$ and $(x, y) \in R^* \setminus R$. Consequently, $R' = R^* \setminus \{(u, v)\}$ is the same as the set obtained from R^* by removing all edges (x, y) with $x \leq_{G \setminus R} u$ and $(x, y) \notin R$. Therefore, by Lemma 7, R' is a transfer set w.r.t. (S, G). It is also clearly true that R' expands (R, U). So, to obtain a contradiction, it remains to prove that $\mathrm{Cost}(R') < \mathrm{Cost}(R^*)$.

Let z be the child of p in G with $z \geq_G u$. Since $p \in U$, Lemmas 2 and 3 imply that $M_{R^*}(p) = M_{R'}(p) = M_R(p)$. Now, since $M_{R^*}(p) = M_{R'}(p)$ and $R' = R^* \setminus \{(u, v)\}$, we have $M_{R'}(x) = M_{R^*}(x)$ for all vertices x of G such that x is not both an ancestor of u and a proper descendant of p in $G \setminus R$. Thus,

$$\begin{aligned}\mathrm{Cost}(R^*) - \mathrm{Cost}(R') = c_t + c_\ell \cdot (&\mathrm{Loss}_{R^*}(p, z) \\ &- \mathrm{Loss}_{R'}(p, z) - \mathrm{Loss}_{R'}(u, w) - \mathrm{Loss}_{R'}(u, v)).\end{aligned}$$

For convenience, let $\Delta = (\mathrm{Loss}_{R^*}(p, z) - \mathrm{Loss}_{R'}(p, z)) - \big(\mathrm{Dist}_S(M_{R^*}(p),\ M_{R^*}(z)) - \mathrm{Dist}_S(M_{R'}(p),\ M_{R'}(z))\big)$. We claim that $\Delta \geq 0$. To see this, we distinguish two cases as follows.

Case 1: p is a final speciation vertex w.r.t. (R, U). In this case, p is a speciation vertex w.r.t. both R' and R^* by Lemma 2, implying that $M_{R^*}(p) \neq M_{R^*}(z)$ and $M_{R'}(p) \neq M_{R'}(z)$. Thus, $\mathrm{Loss}_{R^*}(p, z) = \mathrm{Dist}_S(M_{R^*}(p), M_{R^*}(z)) - 1$ and $\mathrm{Loss}_{R'}(p, z) = \mathrm{Dist}_S(M_{R'}(p),\ M_{R'}(z)) - 1$. Hence, $\Delta = 0$.

Case 2: p is a final duplication vertex w.r.t. (R, U). In this case, $M_{R^*}(p) = M_{R'}(p)$ and p is a duplication vertex w.r.t. both R^* and R' by Lemma 3. Moreover, $M_{R^*}(z) \leq_S M_{R'}(z)$ because $R' \subseteq R^*$. Thus, if $M_{R'}(p) \neq M_{R'}(z)$, then $M_{R^*}(p) \neq M_{R^*}(z)$ and in turn $\mathrm{Loss}_{R^*}(p, z) = \mathrm{Dist}_S(M_{R^*}(p), M_{R^*}(z))$ and $\mathrm{Loss}_{R'}(p, z) = \mathrm{Dist}_S(M_{R'}(p), M_{R'}(z))$, implying $\Delta = 0$. So, we may assume that $M_{R'}(p) = M_{R'}(z)$. Then,

$$\mathrm{Loss}_{R'}(p, z) = 0 = \mathrm{Dist}_S(M_{R'}(p), M_{R'}(z)).$$

Now, if $M_{R^*}(p) = M_{R^*}(z)$, then

$$\mathrm{Loss}_{R^*}(p, z) = 0 = \mathrm{Dist}_S(M_{R^*}(p), M_{R'}(z))$$

and in turn $\Delta = 0$; otherwise,

$$\mathrm{Loss}_{R^*}(p, z) = \mathrm{Dist}_S(M_{R^*}(p), M_{R^*}(z)) - 1 \geq 0$$

and in turn $\Delta \geq 0$. This completes the proof of the claim.

By the above claim, we always have $\Delta \geq 0$ and hence $\mathrm{Cost}(R^*) - \mathrm{Cost}(R') \geq c_t + c_\ell \cdot \big(\mathrm{Dist}_S(M_{R^*}(p),\ M_{R^*}(z)) - \mathrm{Dist}_S(M_{R'}(p), M_{R'}(z)) - \mathrm{Loss}_{R'}(u, w) - \mathrm{Loss}_{R'}(u, v)\big)$.

Recall that $M_{R^*}(p) = M_{R'}(p) = M_R(p)$. Obviously, $M_R(u) = M_R(z)$, $M_{R'}(u) = M_{R'}(z)$, and $M_{R^*}(z) = M_{R^*}(u) = M_{R^*}(w)$. Moreover, since there is no edge (x, y) other than (u, v) such that $x \leq_{G \setminus R} u$ and $(x, y) \in R^* \setminus R$, we know that $M_{R^*}(w) = M_{R'}(w) = M_R(w)$, $M_{R^*}(v) = M_{R'}(v) = M_R(v)$, $M_R(u) = M_{R'}(u)$, and u is a speciation vertex w.r.t. R' (which implies $\mathrm{Loss}_{R'}(u, w) = \mathrm{Dist}_S(M_{R'}(u), M_{R'}(w)) - 1$). By all these facts, we finally have

$$\begin{aligned}\mathrm{Cost}(R^*) - \mathrm{Cost}(R') &\geq c_t + c_\ell \cdot \big(\mathrm{Dist}_S(M_R(p), M_R(w)) \\ &- \mathrm{Dist}_S(M_R(p), M_R(u)) - \mathrm{Dist}_S(M_R(u), M_R(w)) \\ &+ 1 - \mathrm{Loss}_R(u, v)\big) = c_t + c_\ell \cdot (1 - \mathrm{Loss}_R(u, v)) > 0,\end{aligned}$$

where the last inequality holds because u is a mild speciation vertex w.r.t. R. □

Lemma 16. *Suppose that u is a type-3 mild e-move w.r.t. (R, U). Let v and w be the children of u in G, p be the parent of u in $G \between R$, and x and y be the children of p in G. Then, the following statements hold:*

1. *For every k-transfer set R^* w.r.t. (S, G) expanding (R, U), either $E[\{u\}] \cap R^* = \emptyset$ or $R^* \cap E[\{p, u\}]$ is equal to $\{(u, v), (p, x)\}$, $\{(u, v), (p, y)\}$, $\{(u, w), (p, x)\}$, or $\{(u, w), (p, y)\}$.*
2. *$(R \cup \{(u, v), (p, x)\}, U)$, $(R \cup \{(u, v), (p, y)\}, U)$, $(R \cup \{(u, w), (p, x)\}, U)$, $(R \cup \{(u, w), (p, y)\}, U)$, and $(R, U \cup \{u\})$ are valid pairs and they together form a complete extension set of (R, U).*
3. *The lower bounds w.r.t. $(R \cup \{(u, v), (p, x)\}, U)$, $(R \cup \{(u, v), (p, y)\}, U)$, $(R \cup \{(u, w), (p, x)\}, U)$, $(R \cup \{(u, w), (p, y)\}, U)$, and $(R, U \cup \{u\})$ are larger than the lower bound w.r.t. (R, U) by at least $2c_t$, $2c_t$, $2c_t$, $2c_t$, and c_t, respectively.*

Proof. Let R^* be a k-transfer set w.r.t. (S, G) which expands (R, U). Since u is a type-3 mild e-move w.r.t. (R, U), u is a mild speciation vertex w.r.t. R and p is an unmarked speciation vertex w.r.t. (R, U). So, u is a type-2 mild e-move w.r.t. $(R, U \cup \{p\})$. Now, if $E[\{p\}] \cap R^* = \emptyset$, then $E[\{u\}] \cap R^* = \emptyset$ by Statement 1 in Lemma 15. Otherwise, it is clear that either $E[\{u\}] \cap R^* = \emptyset$ or $R^* \cap E[\{p, u\}]$ is equal to $\{(u, v), (p, x)\}$, $\{(u, v), (p, y)\}$, $\{(u, w), (p, x)\}$, or $\{(u, w), (p, y)\}$. This completes the proof of Statement 1.

By Statement 1 in Lemma 10, $(R \cup \{(u, v)\}, U)$, $(R \cup \{(u, w)\}, U)$, and $(R, U \cup \{u\})$ together form a complete extension set $\mathcal{P}$ of (R, U). Obviously, p is an unmarked speciation vertex w.r.t. each pair in $\mathcal{P}$. So, again by Statement 1 in Lemma 10, $(R \cup \{(u, v), (p, x)\}, U)$, $(R \cup \{(u, v), (p, y)\}, U)$, $(R \cup \{(u, w), (p, x)\}, U)$, and $(R \cup \{(u, w), (p, y)\}, U)$ are extensions of (R, U). Moreover, by Statement 1, $(R \cup \{(u, v), (p, x)\}, U)$, $(R \cup \{(u, v), (p, y)\}, U)$, $(R \cup \{(u, w), (p, x)\}, U)$, $(R \cup \{(u, w), (p, y)\}, U)$, and $(R, U \cup \{u\})$ together form a complete extension set of (R, U). This completes the proof of Statement 2. Statement 3 follows from Statement 2 immediately. □

We are unable to use a type-1 mild e-move u w.r.t. (R, U) to find a complete extension set of (R, U) that is better than the one suggested by Statement 1 in Lemma 10. Our idea is to handle type-1 mild e-moves at last. In other words, we handle type-1 mild e-moves only when no other moves exist. Hereafter, a *move* w.r.t. (R, U) means a d-move, singleton or twin s-move, or e-move w.r.t. (R, U).

The next lemma says that we cannot have many type-1 mild e-moves when no other moves exist.

Lemma 17. *Suppose that there are h type-1 mild e-moves w.r.t. (R, U) and no other moves w.r.t. (R, U) exist. Let u be an arbitrary type-1 mild e-move w.r.t. (R, U), and v and w be the children of u in G. Then, the following statements hold:*

1. *R contains at least $h - 1$ edges.*
2. *For each pair $(R', U') \in \{(R \cup \{(u, v)\}, U), (R \cup \{(u, w)\}, U), (R, U \cup \{u\})\}$, there are $h - 1$ type-1 mild e-moves w.r.t. (R', U') and no other moves w.r.t. (R', U') exist.*
3. *If $b + hc_t > k$, then no k-transfer set R^* w.r.t. (S, G) expands (R, U), where b is the lower bound w.r.t. (R, U).*

Proof. Since there are h type-1 mild e-moves w.r.t. (R, U), $G \setminus R$ has at least h connected components. Now, since G is a tree and removing one edge from a forest increases the number of connected components in the forest by 1, R contains at least $h - 1$ edges. Hence, Statement 1 holds. Statement 2 is obvious. Statement 3 follows from Statement 2 in this lemma and Statement 2 in Lemma 10 immediately. □

Finally, the next lemma says that if at least one vertex of G is not settled w.r.t. (R, U), then there is a move w.r.t. (R, U).

Lemma 18. *Suppose that at least one vertex of G is not settled w.r.t. (R, U). Then, there is a move w.r.t. (R, U).*

Proof. Since at least one vertex of G is not settled w.r.t. (R, U), at least one unmarked vertex w.r.t. (R, U) is not settled w.r.t. (R, U). Recall that a vertex u is *lower than* than another vertex v in $G \between R$ if they belong to the same tree T in $G \between R$ and u is farther from the root of T than v. Consider an unmarked vertex p w.r.t. (R, U) such that p is not settled w.r.t. (R, U) but every vertex lower than p in $G \between R$ is. By this choice, p is not a tight speciation vertex w.r.t. R. So, one of the following three cases occurs:

Case 1: p is a duplication vertex w.r.t. R. In this case, $M_R(p) = M_R(u)$ or $M_R(p) = M_R(u')$, where u and u' are the children of p in $G \between R$. We assume that $M_R(p) = M_R(u)$; the other case is similar. If u is either a nonleaf in U or a leaf, then p is a d-move w.r.t. (R, U) and we are done. So, we may assume that u is neither a nonleaf in U nor a leaf. Then, by the choice of p, u is a tight speciation vertex w.r.t. R. Thus, in case $M_R(p) >_S M_R(u')$, u is clearly a singleton s-move w.r.t. (R, U) and we are done. Hence, we may further assume that $M_R(p) = M_R(u')$. Now, if $u' \in U$, then p is a d-move w.r.t. (R, U). Otherwise, by the choice of p, u' is a tight speciation vertex w.r.t. R and in turn $\{u, u'\}$ is a twin s-move w.r.t. (R, U).

Case 2: p is a mild or loose speciation vertex w.r.t. R and is a root of $G \between R$. In this case, p is obviously an e-move w.r.t. (R, U).

Case 3: p is a mild or loose speciation vertex w.r.t. R and is not a root of $G \obslash R$. In this case, let x and p' be the parent and the sibling of p in $G \obslash R$, respectively. If x belongs to U or is an unmarked speciation vertex w.r.t. (R, U), then p is an e-move w.r.t. (R, U). Thus, we may assume that x is neither a speciation vertex w.r.t. R nor a vertex in U. Then, x is a duplication vertex w.r.t. R and in turn $M_R(x) = M_R(p)$ or $M_R(x) = M_R(p')$. If $M_R(x) = M_R(p)$, then p is a singleton s-move w.r.t. (R, U) because p is an unmarked nonspecial speciation vertex w.r.t. (R, U). So, we may further assume that $M_R(x) \neq M_R(p)$. Then, $M_R(x) = M_R(p')$. By the choice of p, we also know that the children of p' in $G \obslash R$ are settled w.r.t. (R, U). Hence, if p' is an unmarked speciation vertex w.r.t. (R, U), then p' is a singleton s-move w.r.t. (R, U). Therefore, we may also assume that p' is not an unmarked speciation vertex w.r.t. (R, U).

Case 3.1: p' is marked w.r.t. (R, U). In this case, p' is a leaf of G, belongs to U, or is a transfer vertex w.r.t. R. Since p' is a vertex of $G \obslash R$, p' cannot be a transfer vertex w.r.t. R. Moreover, since $M_R(x) \neq M_R(p)$ and $M_R(x) = M_R(p')$, p' cannot be a leaf of G. So, $p' \in U$ and in turn x is a d-move w.r.t. (R, U).

Case 3.2: p' is unmarked w.r.t. (R, U). In this case, p' is not a speciation vertex w.r.t. (R, U) and in turn p' is a duplication vertex w.r.t. (R, U). So, as in Case 1, we can prove that there is a d-move, singleton s-move, or twin s-move w.r.t. (R, U). □

3.6 Details of *AllTransSets*

Based on the lemmas in Sections 3.4 and 3.5, we are now ready to detail *AllTransSets*. Given a valid pair (R, U), *AllTransSets* proceeds as follows:

1. Calculate the lower bound b w.r.t. (R, U).
2. If either $b > k$, or $b = k$ and at least one vertex of G is not settled w.r.t. (R, U), then return.
3. If all vertices of G are settled w.r.t. (R, U), then perform the following steps:
   a. If R is a k-transfer set w.r.t. (S, G), output R.
   b. return.
4. If there is a d-move or type-2 mild e-move u w.r.t. (R, U), recursively call *AllTransSets* on input $(R, U \cup \{u\})$ and then return.
5. If there is a singleton s-move u w.r.t. (R, U), recursively call *AllTransSets* on input $(R \cup \{(u, v)\}, U)$, $(R \cup \{(u, w)\}, U)$, and $(R, U \cup \{p, u\})$ in any order and then return, where v and w are the children of u in G and p is the parent of u in $G \obslash R$.
6. If there is a twin s-move $\{u, u'\}$ w.r.t. (R, U), recursively call *AllTransSets* on input $(R \cup \{(u, v), (u', v')\}, U)$, $(R \cup \{(u, v), (u', w')\}, U)$, $(R \cup \{(u, w), (u', v')\}, U)$, $(R \cup \{(u, w), (u', w')\}, U)$, and $(R, U \cup \{p, u, u'\})$ in any order and then return, where p is the parent of u and u' in $G \obslash R$, v and w are the children of u in G, and v' and w' are the children of u' in G.
7. If there is a loose e-move u w.r.t. (R, U), recursively call *AllTransSets* on input $(R \cup \{(u, v)\}, U)$, $(R \cup \{(u, w)\}, U)$, and $(R, U \cup \{u\})$ in any order and then return, where v and w are the children of u in G.
8. If there is a type-3 mild e-move u w.r.t. (R, U), recursively call *AllTransSets* on input $(R \cup \{(u, v), (p, x)\}, U)$, $(R \cup \{(u, v), (p, y)\}, U)$, $(R \cup \{(u, w), (p, x)\}, U)$, $(R \cup \{(u, w), (p, x)\}, U)$, and $(R, U \cup \{u\})$ in any order and then return, where p is the parent of u in $G \obslash R$, v and w are the children of u in G, and x and y are the children of p in G.
9. Compute the number h of type-1 mild e-moves w.r.t. (R, U). If $b + hc_t > k$, then return. (*Comment:* when the algorithm starts performing this step, only type-1 mild e-moves w.r.t. (R, U) can exist. Moreover, by Statement 1 in Lemma 17 and the fact that $b \geq |R|c_t$, if the algorithm proceeds to the next step, then $b + hc_t \leq k$ and $b \geq (h-1)c_t$, implying that $h \leq \frac{k+c_t}{2c_t}$.)
10. If there is a type-1 mild e-move u w.r.t. (R, U), recursively call *AllTransSets* on input $(R \cup \{(u, v)\}, U)$, $(R \cup \{(u, w)\}, U)$, and $(R, U \cup \{u\})$ in any order and then return, where v and w are the children of u in G. (*Comment:* when the algorithm starts performing this step, only type-1 mild e-moves w.r.t. (R, U) are possible. Moreover, this remains true after processing a type-1 mild e-move.)

The next theorem shows that *AllTransSets* is correct.

Theorem 19. *Given a valid pair (R, U), AllTransSets outputs all k-transfer sets w.r.t. (S, G) expanding (R, U).*

Proof. The proof is by induction on the recursion depth of *AllTransSets* on input (R, U). In the base case, *AllTransSets* makes no recursive calls on input (R, U) and its correctness follows from Corollaries 6 and 9. So, assume that *AllTransSets* makes at least one recursive call on input (R, U). Obviously, *AllTransSets* makes recursive calls only when it finds a move u w.r.t. (R, U). No matter what kind of move u is, *AllTransSets* finds a complete extension set $\mathcal{P}$ of (R, U) and recurses on each pair in $\mathcal{P}$ in an arbitrary sequential order. Let R^* be a k-transfer set w.r.t. (S, G) which expands (R, U). Since $\mathcal{P}$ is a complete extension set of (R, U), R^* also expands at least one pair $(R', U') \in \mathcal{P}$. So, by the inductive hypothesis, the recursive call of *AllTransSets* on input (R', U') can output R^*. Consequently, *AllTransSets* can output R^* on input (R, U). This finishes the proof. □

3.7 Time Complexity of the Algorithm

As for the time complexity of *AllTransSets*, we can prove the next theorem.

Theorem 20. *Given an input $(S, G, c_t, c_d, c_\ell, k)$ to the BTSE problem, we can enumerate all k-transfer sets w.r.t. (G, S) in $O(m + \min\{3^{\frac{k}{c}}, k\beta^{\frac{k}{c}} + 3^{\frac{k}{2c_t}}\beta^{\frac{k}{2c}}\}n)$ time, where n is the number of nodes in G. Moreover, if $c_\ell \geq 0.614c$, then the time complexity is $O(m + 2.78^{k/c}n)$. In particular, if $c_t = c_d = c_\ell = 1$, then the time complexity is $O(m + 2.78^k n)$.*

The proof of Theorem 20 is quite involved and is given in Section 3 of the online supplementary material.

Corollary 21. *Given an input (S, G, c_t, c_d, c_ℓ) to the TSE problem, we can enumerate all minimum-cost transfer set w.r.t. (S, G) in $O(m + \min\{3^{\frac{k}{c}}, k\beta^{\frac{k}{c}} + 3^{\frac{k}{2c_t}}\beta^{\frac{k}{2c}}\}n)$ time, where n is the number of nodes in G and k is the minimum cost of a* *transfer set w.r.t. (S, G). Moreover, if $c_\ell \geq 0.614c$, then the time complexity is $O(m + 2.78^{k/c}n)$. In particular, if $c_t = c_d = c_\ell = 1$, then the time complexity is $O(m + 2.78^{k}n)$.*

Proof. Let (S, G) and k be as in the corollary. By Theorem 20, we can enumerate all minimum-cost transfer set w.r.t. (S, G) in time $O(m + \sum_{i=1}^{k} \min\{3^{\frac{i}{c}},\ i\beta^{\frac{i}{c}} + 3^{\frac{i}{2c_t}}\beta^{\frac{i}{2c}}\}n) = O(m + \min\{3^{\frac{k}{c}}, k\beta^{\frac{k}{c}} + 3^{\frac{k}{2c_t}}\beta^{\frac{k}{2c}}\}n)$. □

4 Experiments

We have tested our algorithm on a real biological data set. The main purpose of the test is to evaluate how different weighting schemes affect the minimum-cost reconciliations outputted by the algorithm.

The data set we use is the same as that used in Section 9 of Tofigh et al. [18], which is originally from [26]. The data set consists of 1) two different species trees S_{RP} and S_{rRNA} on the same set of 14 archaeal organisms and 2) one unrooted gene tree obtained from the archaeal *rpl12e* sequences. S_{RP} and S_{rRNA} are, respectively, shown in Figs. 9a and 9b in the online supplementary material, where the directed edges (representing LGTs) should be ignored. The gene tree is shown in Fig. 10 in the online supplementary material, where G_1 through G_{25} indicate 25 possible root positions. Therefore, there are 2×25 species-gene tree pairs.

Recall that c_d, c_t, and c_ℓ represent the costs of a gene duplication, an LGT, and a gene loss, respectively. Now, we try to use different weighting schemes $c_d : c_t : c_\ell$ for all the 50 species-gene tree pairs. First, we report the results for the following different weighting schemes: $1:1:0$, $2:2:1$, and $2:6:1$. These weighting schemes are chosen for the following reasons.

The first scheme $1:1:0$ corresponds to the case where we only count the number of duplications and LGTs but ignore gene losses when computing the total cost of a reconciliation. This special case has been considered by Tofigh et al. [18] and we want to compare their results with ours.

The second scheme $2:2:1$ is a typical choice when we want to weight a gene duplication and an LGT equally but slightly more than a gene loss. This choice is reasonable because gene losses usually occur more frequently than the other two events.

The third scheme $2:6:1$ is supposed to be a refinement on the second scheme $2:2:1$ because it is known that in many species, gene duplications are more common than LGTs and hence LGTs should be given larger costs than duplications.

For each combination of a species-gene tree pair and a weighting scheme that is used in our experiments, our algorithm can report all the minimum-cost reconciliations in less than 1 second on a Windows-XP (32 bits) desktop PC with 3.16 GHz CPU and 4 GB RAM. For each of the two species trees, we report the minimum-cost reconciliations among all the 25 rooted gene trees. The results for different weighting schemes are summarized in Table 1, where those minimum-cost reconciliations containing cycles are ignored. Fig. 9 in the online supplementary material depicts the minimum-cost reconciliations listed in Table 1 and Section 4 in the online supplementary material gives a close inspection on the reconciliations.

TABLE 1
Summary of LCA-Reconciliations Found for the Data Set under Various Weighting Schemes

scheme	S_{RP}				S_{rRNA}			
$c_d : c_t : c_\ell$	Gene Tree	D	T	L	Gene Tree	D	T	L
$1:1:0$	G_{17}	4	1	16	G_{17}	3	1	15
	G_{19}	3	2	10	G_{19}	2	2	8
	G_9	2	3	7	G_{15}	1	3	5
	G_{19}	2	3	7	G_{21}	1	3	5
	G_9	0	5	1	G_{21}	0	4	2
$2:2:1$	G_9	0	5	1	G_{18}	0	5	0
					G_{21}	0	4	2
$2:6:1$	G_{19}	3	2	10	G_{19}	2	2	8

Note that for the scheme $1:1:0$, the set of reconciliations outputted by our algorithm is a superset of that outputted by the algorithm of Tofigh et al. [18]. The reason is as follows: recall that Tofigh et al. [18] propose to use a two-phase approach in their algorithm, i.e., they first find all reconciliations with the minimum total cost of duplications and LGTs, among which they then choose the ones with the smallest number of losses. As a result, for the scheme $1:1:0$, many reconciliations are discarded in the second phase of their algorithm, though the total number of duplications and LGTs in each discarded reconciliation is the minimum. Unlike their two-phase approach, our algorithm allows us to weight losses together with duplications and LGTs, and hence it can directly output all the minimum-cost reconciliations. So, for the scheme $1:1:0$, our algorithm can output all reconciliations with the minimum total number of duplications and LGTs, even if they have different numbers of losses. However, since there are lots of minimum-cost reconciliations for the scheme $1:1:0$, we only list a portion of them in Table 1.

For the scheme $2:2:1$, the minimum-cost reconciliation (with 5 LGTs only) is obtained between S_{RP} and G_{23}. However, since the root of G_{23} is placed in a very unlikely position (inside the *Pyrococcus* clade), we discard this optimal reconciliation and use the minimum-cost reconciliations among the other 24 gene trees instead. As shown in the table, the solutions found for the scheme $2:2:1$ contain no duplications. This looks reasonable, because the scheme $2:2:1$ assigns a relatively large cost to a gene loss and a gene duplication can lead to a number of gene losses. For the scheme $2:6:1$, only one solution is found for each of S_{RP} and S_{rRNA}. Since this scheme assigns a relatively small cost to a gene loss, each solution contains several gene duplications and gene losses.

The above experimental results show that associating a positive cost to a gene loss leads to a bias toward using more LGTs in the resulting reconciliations. This bias is especially obvious when the cost of an LGT is not relatively large when compared with those of a duplication and a loss. The bias can be explained by the fact that a gene duplication usually leads to a number of gene losses. However, the bias can be reduced by increasing the cost of an LGT. Accordingly, more duplications may occur in the resulting reconciliations. This can be seen from Table 2 in the online supplementary material.

5 Concluding Remarks

Defining a proper weighting scheme for the three evolutionary events (namely, duplications, losses, and LGTs) is an important but very difficult task. To the best of our knowledge, no realistic weighting scheme for the events has been published. Nonetheless, we have the following general principles for choosing a weighting scheme. First, since gene losses are usually more common than gene duplications and LGTs, the cost of a loss should be smaller than those of a duplication and an LGT. Moreover, different weighting schemes we should apply to different sets of species. For instance, since LGTs usually occur more frequently in prokaryotes than in eukaryotes, the cost of an LGT for prokaryotes should be smaller than that for eukaryotes.

In real applications, one may run the algorithm presented in this paper for various weighting schemes, and find the scheme that gives the best result. In this sense, we believe that our algorithm may help us find suitable weighting schemes in various applications.

In our tree-reconciliation problem, we have three parameters (namely, the costs of a duplication, loss, and LGT) but do not know how to assign fixed values to them. Similar problems include DNA sequence alignment [15], [16] and RNA secondary structure comparison [31]. It is known that parametric tools are useful for such problems [15], [16], [31]. Basically, one can use a parametric tool to avoid the problem of choosing fixed parameter settings by decomposing the parametric space into a set of regions, where each region admits optimal solutions with the same cost. It seems interesting to design a parametric tool for tree reconciliation. In the design of such a tool, the algorithm presented in this paper may be very useful.

Acknowledgments

The authors would like to thank the referees for very insightful comments. Zhi-Zhong Chen was supported in part by the Grant-in-Aid for Scientific Research of the Ministry of Education, Science, Sports, and Culture of Japan, under Grant No. 24500023. Lusheng Wang was fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China [Project No. CityU 121608].

References

[1] L. Arvestad, J. Lagergren, and B. Sennblad, "The Gene Evolution Model and Computing Its Associated Probabilities," *J. ACM*, vol. 56, no. 2, pp. 1-44, 2009.

[2] M. Bansal and O. Eulenstein, "The Multiple Gene Duplication Problem Revisited," *Bioinformatics*, vol. 24, pp. 132-138, 2008.

[3] H. Birin, Z. Gal-Or, I. Elias, and T. Tuller, "Inferring Horizontal Transfers in the Presence of Rearrangements by the Minimum Evolution Criterion," *Bioinformatics*, vol. 24, no. 6, pp. 826-832, 2008.

[4] A. Boc, H. Philippe, and V. Makarenkov, "Inferring and Validating Horizontal Gene Transfer Events Using Bipartition Dissimilarity," *Systematic Biology*, vol. 59, no. 2, pp. 195-211, 2010.

[5] R. Bock, "The Give-and-Take of DNA," *Trends in Plant Science*, vol. 15, no. 1, pp. 11-22, 2010.

[6] K. Chen, D. Durand, and M. Farach-Colton, "NOTUNG: A Program for Dating Gene Duplications and Optimizing Gene Family Trees," *J. Computational Biology*, vol. 7, pp. 429-447, 2000.

[7] J. Cotton and R. Page, "Rates and Patterns of Gene Duplication and Loss in the Human Genome," *Proc. Biological Sciences*, vol. 272, no. 1560, pp. 277-283, 2005.

[8] M. Csűrös and I. Miklós, "A Probabilistic Model for Gene Content Evolution with Duplication, Loss, and Horizontal Transfer," *Proc. 10th Ann. Int'l Conf. Research in Computational Molecular Biology (RECOMB)*, pp. 206-220, 2006.

[9] J. Demuth, T. De Bie, J. Stajich, N. Cristianini, and M. Hahn, "The Evolution of Mammalian Gene Families," *PLoS One*, vol. 1, no. 1, p. e85, 2006.

[10] W. Doolittle and E. Bapteste, "Pattern Pluralism and the Tree of Life Hypothesis," *Proc. Nat'l Academy of Sciences USA*, vol. 104, no. 7, pp. 2043-2049, 2007.

[11] J. Doyon, C. Scornavacca, K. Gorbunov, G. Szöllősi, V. Ranwez, and V. Berry, "An Efficient Algorithm Algorithm for Gene/Species Trees Parsimonious Reconciliation with Losses, Duplications and Transfers," *Proc. Int'l Conf. Comparative Genomics*, pp. 93-108, 2011.

[12] M. Goodman, J. Czelusniak, G. Moore, A. Romero-Herrera, and G. Matsuda, "Fitting the Gene Lineage into its Species Lineage, a Parsimony Strategy Illustrated by Cladograms Constructed from Globin Sequences," *Systematic Biology*, vol. 28, no. 2, pp. 132-163, 1979.

[13] P. Górecki, "Reconciliation Problems for Duplication, Loss and Horizontal Gene Transfer," *Proc. Eighth Ann. Int'l Conf. Research in Computational Molecular Biology (RECOMB)*, pp. 316-325, 2004.

[14] R. Guigó, I. Muchnik, and T. Smith, "Reconstruction of Ancient Molecular Phylogeny," *Molecular Phylogenetics and Evolution*, vol. 6, no. 2, pp. 189-213, 1996.

[15] D. Gusfield, K. Balasubramanian, and D. Naor, "Parametric Optimization of Sequence Alignment," *Algorithmica*, vol. 12, pp. 312-326, 1994.

[16] D. Gusfield and P. Stelling, "Parametric and Inverse-Parametric Sequence Alignment with XPARAL," *Methods in Enzymology*, vol. 266, pp. 481-491, 1996.

[17] M. Hahn, T. De Bie, J. Stajich, C. Nguyen, and N. Cristianini, "Estimating the Tempo and Mode of Gene Family Evolution from Comparative Genomic Data," *Genome Research*, vol. 15, no. 8, pp. 1153-1160, 2005.

[18] A. Tofigh, M. Hallett, and J. Lagergren, "Simultaneous Identification of Duplications and Lateral Gene Transfers," *IEEE/ACM Trans. Computational Biology and Bioinformatics*, vol. 8, no. 2, pp. 517-535, Mar./Apr. 2011.

[19] M. Hallett and J. Lagergren, "New Algorithms for the Duplication-Loss Model," *Proc. Fourth Ann. Int'l Conf. Research in Computational Molecular Biology (RECOMB)*, pp. 138-146, 2000.

[20] M. Hallett and J. Lagergren, "Efficient Algorithms for Lateral Gene Transfer Problems," *Proc. Fifth Ann. Int'l Conf. Research in Computational Molecular Biology (RECOMB)*, pp. 149-156, 2001.

[21] G. Jin, L. Nakhleh, S. Snir, and T. Tuller, "Maximum Likelihood of Phylogenetic Networks," *Bioinformatics*, vol. 22, no. 21, pp. 2604-2611, 2006.

[22] P. Keeling and J. Palmer, "Horizontal Gene Transfer in Eukaryotic Evolution," *Nature Rev. Genetics*, vol. 9, no. 8, pp. 605-618, 2008.

[23] M. Lynch and J. Conery, "The Evolutionary Fate and Consequences of Duplicate Genes," *Science*, vol. 290, no. 5494, pp. 1151-1155, 2000.

[24] M. Lynch and J. Conery, "The Evolutionary Demography of Duplicate Genes," *J. Structural and Functional Genomics*, vol. 3, no. 1, pp. 35-44, 2003.

[25] B. Ma, M. Li, and L. Zhang, "From Gene Trees to Species Trees," *SIAM J. Computing*, vol. 30, no. 3, pp. 729-752, 2000.

[26] O. Matte-Tailliez, C. Brochier, P. Forterre, and H. Philippe, "Archaeal Phylogeny Based on Ribosomal Proteins," *Molecular Biology and Evolution*, vol. 19, no. 5, pp. 631-639, 2002.

[27] D. Merkle and M. Middendorf, "Reconstruction of the Cophylogenetic History of Related Phylogenetic Trees with Divergence Timing Information," *Theory in Biosciences*, vol. 123, pp. 277-299, 2005.

[28] D. Merkle, M. Middendorf, and N. Wieseke, "A Parameter-Adaptive Dynamic Programming Approach for Inferring Cophylogenies," *BMC Bioinformatics*, vol. 11(Suppl 1), p. S60, 2010.

[29] L. Nakhleh, D. Ruths, and L. Wang, "RIATA-HGT: A Fast and Accurate Heuristic for Reconstructing Horizontal Gene Transfer," *Proc. 11th Int'l Computing and Combinatorics Conf. (COCOON '05)*, pp. 84-93, 2005.

[30] R. Page and M. Charleston, "From Gene to Organismal Phylogeny: Reconciled Trees and the Gene Tree/Species Tree Problem," *Molecular Phylogenetics and Evolution*, vol. 7, no. 2, pp. 231-240, 1997.

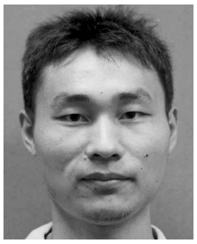
[31] L. Wang and J. Zhao, "Parametric Alignment of Ordered Trees," *Bioinformatics,* vol. 19, no. 17, pp. 2237-2245, 2003.

[32] I. Wapinski, A. Pfeffer, N. Friedman, and A. Regev, "Natural History and Evolutionary Principles of Gene Duplication in Fungi," *Nature,* vol. 449, no. 7158, pp. 54-61, 2007.

[33] L. Zhang, "On a Mirkin-Muchnik-Smith Conjecture for Comparing Molecular Phylogenies," *J. Computational Biology,* vol. 4, no. 2, pp. 177-187, 1997.

[34] L. Zhang, Y. Ng, T. Wu, and Y. Zheng, "Network Model and Efficient Method for Detecting Relative Duplications or Horizontal Gene Transfers," *Proc. IEEE First Int'l Conf. in Computational Advances in Bio and Medical Sciences (ISSABS),* pp. 214-219, 2011.

Zhi-Zhong Chen received the PhD degree from the University of Electro-Communications, Tokyo, Japan, in 1992. Currently, he is working as a professor in the Division of Information System Design, Tokyo Denki University. His research interests include algorithms, computational biology, and bioinformatics.



Fei Deng is currently working toward the PhD degree in the Department of Computer Science, City University of Hong Kong. His research interests include bioinformatics and computational biology.

Lusheng Wang received the PhD degree from McMaster University, Hamilton, Ontario, Canada, in 1995. Currently, he is working as a professor and the acting head in the Department of Computer Science, City University of Hong Kong. His research interests include algorithms, bioinformatics, and computational biology. He is a member of the IEEE.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.