Fast algorithms for computing the Tripartition-based Distance between Phylogenetic Networks

Nguyen Bao Nguyen, C. Thach Nguyen, and Wing-Kin Sung

National University of Singapore, 3 Science Drive 2, Singapore 117543. E-mail: {baonguyen, thachnguyen, dcsswk}@nus.edu.sg

Abstract. Consider two phylogenetic networks N and N' of size n. The tripartition-based distance finds the proportion of tripartitions which are not shared by N and N'. This distance is proposed by Moret et al (2004) and is a generalization of Robinson-Foulds distance, which is orginally used to compare two phylogenetic trees. This paper gives an $O(\min\{kn \log n, n \log n + hn\})$-time algorithm to compute this distance, where h is the number of hybrid nodes in N and N' while k is the maximum number of hybrid nodes among all biconnected components in N and N'. Note that $k << h << n$ in a phylogenetic network. In addition, we propose algorithms for comparing galled-trees, which are an important, biological meaningful special case of phylogenetic network. We give an $O(n)$-time algorithm for comparing two galled-trees. We also give an $O(n + kh)$-time algorithm for comparing a galled-tree with another general network, where h and k are the number of hybrid nodes in the latter network and its biggest biconnected component respectively.

1 Introduction

Phylogenetic trees are traditionally used to describe evolutionary relationships among a set of objects. However, evolutionary events such as horizontal gene transfer or hybrid speciation (often referred to as *recombination events*) which suggest convergence between objects cannot be adequately represented in a single tree structure. In the famous Science paper [3] by Doolittle, he also pointed out that the phylogenetic tree is inadequated to represent the "true" evolution history. To solve the shortcoming, phylogenetic networks were introduced. A *phylogenetic network* is a distinctly leaf-labeled directed acyclic graph where the in-degree and out-degree of all nodes are bounded above by 2. The nodes with in-degree 2 are called hybrid nodes and they are used to model the recombination events. Fig. 1(a) shows an example of a phylogenetic network.

Recently, a lot of works have been proposed to reconstruct phylogenetic networks [1, 4–6, 8–11, 15–17, 19]. To access the topological accuracy of different construction methods, two measurements were proposed for comparing networks. They are Maximum Agreement Phylogenetic Subnetwork (MASN)[2, 13], and Tripartition-based distance[14]. This paper focuses on the latter measurement.

The tripartition-based distance is a generalization of the Robinson-Foulds measure [18], which is a well-known method for comparing phylogenetic trees. Given two phylogenetic networks N and N' which have the same leaf set, the tripartition-based distance $tri(N, N')$ computes the proportion of tripartitions (defined in Section 2.3) which are not shared by N and N'. The tripartition-based distance is shown to be a distance metric[14]. More importantly, when both N and N' are trees, $tri(N, N')$ equals the Robinson-Foulds distance between them.

In this paper, we compute $tri(N, N')$ in $O(\min\{kn \log n, n \log n + hn\})$ time where $n = \max\{V(N), V(N')\}$, h is the maximum number of hybrid nodes in N and N', and k is the maximum number of hybrid nodes among all the biconnected components in N and N'. As the number of hybrid nodes in a network is relatively rare (recombination events do not happen frequently), $k << h << n$. Thus, in practice, the running time of our algorithm achieves $O(n \log n)$.

We also consider comparing an important, biologically motivated special case of phylogenetic networks, known as galled-trees. A galled-tree [2, 5, 10–13, 16, 19] (also referred to in the literature as a *a level-1 network* [2, 12], a *gt-network* [16], or a *topology with independent recombination events* [19]) is a phylogenetic network in which all cycles in the underlying undirected graph are node-disjoint (see the network N in Fig. 2 for an example). When both N and N' are galled-trees, we show that $tri(N, N')$ can be computed in $O(n)$ time. If only N is known to be a galled-tree, $tri(N, N')$ can be computed in $O(n + kh)$ where h and k are the number of hybrid nodes in N' and in the biggest biconnected component of N' respectively.

Our improvement is stemmed from a novel labeling technique which labels the nodes of the networks to facilite efficient identification of common tripartitions between two networks.

The rest of the paper is organized as follows. We first present the preliminaries in Section 2. Section 3 details the results for computing $tri(N, N')$ when at least one of N and N' is a galled-tree. Finally, we present the result for computing the tripartition-based distance for two general networks in Section 4.

2 Preliminaries

2.1 Phylogenetic Network

A *phylogenetic tree* is a binary, rooted, unordered tree whose leaves are distinctly labeled. A *phylogenetic network* is a generalization of a phylogenetic tree formally defined as a rooted, connected, directed acyclic graph in which: (1) exactly one node has indegree 0 (the *root*), and all other nodes have indegree 1 or 2; (2) all nodes with indegree 2 (referred to as *hybrid nodes*) have outdegree 1, and all other nodes have outdegree 0 or 2; and (3) all nodes with outdegree 0 (the *leaves*) are distinctly labeled.

For each hybrid node h, there are several nodes from which there are two disjoint paths to h. Such nodes are called *split nodes* of h. These disjoint paths are called the *merge paths* of h. Two merge paths of h starting from the same split node u form a simple cycle, called the *recombinant cycle* of u.

For any phylogenetic network N, let $\mathcal{U}(N)$ be the undirected graph obtained from N by replacing each directed edge by an undirected edge. The level of N [2] is the maximum number of hybrid nodes among all biconnected components of $\mathcal{U}(N)$. A *galled-tree* is a network of level 1.

In the following, we will use $V(N)$, $E(N)$, $L(N)$, $h(N)$ and $k(N)$ to denote the set of nodes, edges, leaves, the number of hybrid node and the level of the network N.

2.2 Component Tree of a Network

We decompose $\mathcal{U}(N)$ into biconnected components (or simply components). In each biconnected component C in $\mathcal{U}(N)$, there is a node being an ancestor of all the others, called the component's sub-root and denoted by $r(C)$. For a node $x \in C$, a child u of x is called its *internal child* if $u \in C$; otherwise, u is x's *external child*. *Internal descendants* and *external descendants* of a node are similarly defined.

Given a component C, its *reduced component* C^r is obtained by contracting all nodes which have one external and one internal children (by "contracting" a node, we mean deleting it and letting all its children become its parent's children). The *reduced network* N^r is obtained by replacing each component of N by its reduced component. Fig. 1(a) and (c) show a network N and its reduced network N^r.

If we consider every biconnected component in N as a node, the resulting graph is a tree and we denoted it as biconnected component tree $\mathcal{T}(N)$ (see Fig. 1(b) for an example). One property of $\mathcal{T}(N)$ is that edges from one component to another have the sub-roots of the latter as their heads.

We will use the term *children* to indicate both children of a node in a network N as well as children of a component in $\mathcal{T}(N)$. *Parents*, *ancestors* and *descendants* will also be used in this way. Thus, children, parents, descendants and ancestors of a node are nodes in N whereas those of a component are components in $\mathcal{T}(N)$.

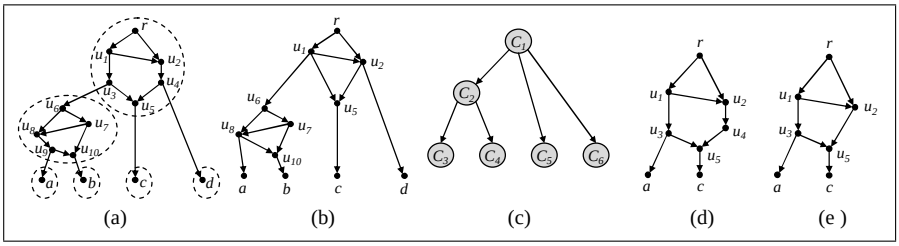


Fig. 1. (a) A network N. (b) The reduced network N^r. (c) The component tree $\mathcal{T}(N)$. (d) $N^*_{\{a,c\}}$. (e) $N_{\{a,c\}}$.

The following lemma states an important property of the reduced components.

Lemma 1. *Each reduced component contains $O(t)$ nodes and $O(t)$ internal edges where t is the number of its hybrid nodes.*

Proof. A node in a reduced component belongs to one of 4 types A, B, C and D whose internal in and out degrees are 0 and 2, 1 and 2, 2 and 1 and 2 and 0 respectively. Let a, b, c, d be the numbers of nodes belong to each type respectively. We have $c + d = t$ and $2a + 2b + c = b + 2c + 2d$. This implies $a + b \leq 2a + b = c + 2d \leq 2(c + d) = 2t$. Thus $a + b + c + d \leq 3t$. □

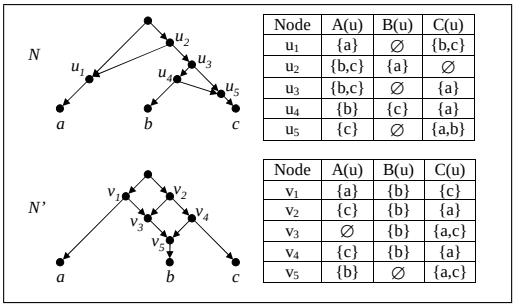
2.3 Tripartition-based Measure

Consider a phylogenetic network N leaf-labeled by S. For a node $u \in V(N)$, an ancestor v of u is called its *strict ancestor* if all paths from the root of N to u contain v. Otherwise, v is called a *non-strict ancestor* of u. The tripartition of u is $(A(u), B(u), C(u))$ where $A(u) = \{s \in S | u \text{ is a strict ancestor of } s\}$; $B(u) = \{s \in S | u \text{ is a non-strict ancestor of } s\}$; and $C(u) = \{s \in S | u \text{ is not an ancestor of } s\}$. We also denote $A(u) \cup B(u)$ as $D(u)$.

Given two networks N and N' having the same leaf set, a node u in one network is *unmatched* if and only if there is no node v in the other network such that $A(u) = A(v)$, $B(u) = B(v)$ and $C(u) = C(v)$. An edge (u, v) is unmatched if and only if v is unmatched. The tripartition-based distance $tri(N, N')$ between N and N' is defined by:

$$\left(\frac{|\{e \in E(N) | e \text{ is unmatched}\}|}{|E(N)|} + \frac{|\{e \in E(N') | e \text{ is unmatched }\}|}{|E(N')|}\right)/2$$

Fig. 2 shows an example of how to compute tripartition of all nodes in N and N'. All the nodes are unmatched. Hence, all the edges whose heads are of these nodes are unmatched and $tri(N, N') = (7/10 + 7/10)/2 = 0.7$.



Node	A(u)	B(u)	C(u)
u_1	{a}	∅	{b,c}
u_2	{b,c}	{a}	∅
u_3	{b,c}	∅	{a}
u_4	{b}	{c}	{a}
u_5	{c}	∅	{a,b}

Node	A(u)	B(u)	C(u)
v_1	{a}	{b}	{c}
v_2	{c}	{b}	{a}
v_3	∅	{b}	{a,c}
v_4	{c}	{b}	{a}
v_5	{b}	∅	{a,c}

Fig. 2. Two networks N and N' whose tripartition-based distance $tri(N, N') = 0.7$.

3 Comparing a Galled-tree and a General Network

Given a galled-tree N and a general network N' having the same leaf set, this section describes an algorithm to identify their unmatched nodes. Once all such nodes are identified, $tri(N, N')$ can be readily computed. The algorithm processes in 3 steps as in Fig. 3.

Algorithm *UnmatchedNode*
Input: A galled-tree N and a general network N' of the same leaf set
Output: The unmatched nodes in the two networks
1 Label the nodes of N and N' such that two nodes $u \in V(N)$ and $u' \in V(N')$ have the same label if and only if they induce the same tripartition.
2 Sort the nodes of both networks by their labels.
3 Compare the sorted lists of labels to identify unmatched nodes in the two networks.
End *UnmatchedNode*

Fig. 3. Algorithm to identify all unmatched nodes in two phylogenetic networks

The first step of Fig. 3 is achieved in two phases. Phase 1 reindexes the leaves by numbers from 1 to $|L(N)|$ so that for each node u of the galled-tree N, both $D(u)$ and $B(u)$ form sub-intervals of $[1..|L(N)|]$. Phase 2 labels each non-leaf node u by a 6-tuple of integers $(m_d(u),\ M_d(u),\ n_d(u),\ m_b(u),\ M_b(u),\ n_b(u))$ whose meaning is:

- $m_d(u), M_d(u), n_d(u)$ indicate the minimum leaf index, maximum leaf index and the number of leaves, respectively, in $D(u)$.
- $m_b(u), M_b(u), n_b(u)$ indicate the minimum leaf index, maximum leaf index and the number of leaves, respectively, in $B(u)$.

The labeling satisfies the following property.

Lemma 2. *For $u \in V(N)$ and $u' \in V(N')$, $(m_d(u),\ M_d(u),\ n_d(u)) = (m_b(v),\ M_b(v),\ n_b(v))$ if and only if u and v induce the same tripartition.*

Given the labeling, Steps 2 and 3 can identify all unmatched nodes. Below, we detail the reindexing and the labeling.

3.1 Reindexing the Leaves of a Galled-tree

We now describe how to index the leaves of a galled-tree so that for each internal node u, both $D(u)$ and $B(u)$ are sub-intervals of $[1, l]$. First, we state a property of the biconnected components of a galled-tree.

Lemma 3. *Each biconnected component C of a galled-tree consists of either a single node or the (only) recombinant cycle of its sub-root.*

Based on the lemma, we design the reindexing algorithm as in Fig. 4. The correctness and time complexity of this algorithm is stated below.

Algorithm *Reindex*
Input: A galled-tree N of l leaves and an integer i
Output: A new indexing of N's leaves so that for each node $u \in V(N)$, both $D(u)$ and $B(u)$ are sub-intervals of $[i, i+l-1]$.
1 if N consists of a single leaf **then**
1.1 reindex the leaf as i
elseif the root of N is a tree node **then**
1.2 let N_l and N_r be the left and right subnetworks attached to the left and right children of the root of N
1.3 *Reindex*(N_l, i)
1.4 *Reindex*(N_r, $i + |L(N_l)|$)
elseif the root of N is a split node **then**
1.5 let $N_1, N_2, \ldots N_x$ be the list of subnetworks attached to the recombinant cycle of the root of N in counter clockwise order
1.6 for $j = 1$ to x **do**
1.6.1 *Reindex*(N_j, $i + |L(N_1)| + |L(N_2)| + \cdots |L(N_{j-1})|$)
endfor
endif
End *Reindex*

Fig. 4. Algorithm to reindexing the leaves of a galled-tree

Lemma 4. *Reindex(N, 1) runs in $O(|E(N)|)$ time and it reindexes the leaves of a galled-tree N such that, for each $u \in V(N)$, both $D(u)$ and $B(u)$ are sub-intervals of $[1, |L(N)|]$.*

3.2 Labeling the Nodes of a Network

Given a network N, we go bottom up on $\mathcal{T}(N)$ and label the nodes in each component visited. To reduce the time of labeling the nodes in each component, we divide the process into two steps. First, all the nodes in the reduced component are labeled. Then the remaining nodes are labeled based on the labeled nodes.

For every $x \in V(C^r)$, let $Ex(x)$ and $In(x)$ be the set of external children and internal descendants of x, respectively, i.e., $Ex(x) = \{v|(x,v) \in E(N^r)$ and $v \notin V(C^r)\}$ and $In(x) = \{v|v \in V(C^r)$ and v is a descendant of $x\}$. In addition, let $e(x) = \sum_{v \in Ex(x)} n_d(v)$ and $H(x) = \{v|v \in V(C^r)$ and x is a non-strict ancestor of $v\}$.

The following lemmas help us label the nodes of N. Lemma 5 and Lemma 6 compute the labels of nodes in a reduced component C^r whereas Lemma 7 computes the labels of the other nodes.

Lemma 5. *For each node $x \in V(C^r)$ whose children are u and v, we have:* $n_d(x) = e(x) + \sum_{w \in In(x)} e(w)$, $m_d(x) = \min\{m_d(u), m_d(v)\}$, *and* $M_d(x) = \max\{M_d(u), M_d(v)\}$.

Lemma 6. *For each node $x \in V(C^r)$,* $n_b(x) = \sum_{v \in H(x)} e(v)$, $m_b(x) = \min_{v \in H(x)} m_d(v)$ *and* $M_b(x) = \max_{v \in H(x)} M_d(v)$.

Lemma 7. *For each node x having an internal child u and an external child v.*

$$
\begin{aligned}
n_d(x) &= n_d(u) + n_d(v) \\
m_d(x) &= \min\{m_d(u), m_d(v)\} \\
M_d(x) &= \max\{M_d(u), M_d(v)\} \\
n_b(x) &= n_b(u) \text{ if } u \text{ is a tree node or } n_d(u) \text{ if } u \text{ is a hybrid node} \\
m_b(x) &= m_b(u) \text{ if } u \text{ is a tree node or } m_d(u) \text{ if } u \text{ is a hybrid node} \\
M_b(x) &= M_b(u) \text{ if } u \text{ is a tree node or } M_d(u) \text{ if } u \text{ is a hybrid node}
\end{aligned}
$$

Lemma 8. *The tripartition distance between a galled-tree N and a general network N' can be computed in $O(|E(N)| + |E(N')| + k(N')h(N'))$.*

Proof. (Sketch) The time needed to compute $e(x)$, $In(x)$ and $H(x)$ for all $x \in V(N^r)$ is $O(|E(N)| + \sum_i h(C_i)^2) = O(|E(N)| + k(N)h(N))$ where C_i for $i = 1, 2, \ldots$ is a non-singleton biconnected component of N and $h(C_i)$ is the number of hybrid nodes in C_i. Then, by Lemmas 5 and 6, the labels of all $x \in V(N^r)$ can be computed in $O(|V(N)| + k(N)h(N))$. Finally, the labels of other nodes are computed by Lemma 7 using $O(|E(N)|)$ time. The lemma then follows. □

Corollary 1. *The tripartition distance between two galled-tree N and N' can be computed in $O(|E(N)| + |E(N')|)$.*

4 Comparing Two General Networks

4.1 The Subnetwork Induced by a Set of Leaves

We first define the subnetwork induced by a set of leaves, which will play an important role in comparing two general networks.

Given a network N and a set of leaves $X = \{l_1, l_2, \ldots l_t\}$, we denote $\mathcal{T}_X(N)$ be a subtree of $\mathcal{T}(N)$ induced by X, which is a tree such that (1) whose nodes are X and their lowest common ancestors in $\mathcal{T}(N)$; and (2) whose edges preserve the ancestor-descendant relationship of $\mathcal{T}(N)$.

Let $E'_X = \{(u, v) | u \in C_1, v \in C_2, (C_1, C_2) \in V(\mathcal{T}_X(N))$ and there exists a path from u to v which does not pass through any nodes belonging to some components in $V(\mathcal{T}_X(N))\}$. Let N^*_X be a subnetwork of N such that (1) whose node set is $\bigcup_{C \in V(\mathcal{T}_X(N))} V(C)$ and (2) whose edge set is $E'_X \cup \bigcup_{C \in V(\mathcal{T}_X(N))} E(C)$. We denote N_X be the subnetwork of N induced by X, which is a network formed by contracting all nodes in N^*_X whose in-degree and out-degree are equal to 1. Fig. 1(d) and (e) show example of $N^*_{\{a,c\}}$ and $N_{\{a,c\}}$ where N is the network in Fig. 1(a).

Lemma 9. $|E(N_X)| = O(\min\{h(N) + |X|, k(N)|X|\})$.

Lemma 10. *Given t disjoint leaf sets $X_1, X_2, \ldots X_t$ such that $\bigcup X_i = L(N)$, the subnetworks $N_{X_1}, N_{X_2}, \ldots N_{X_t}$ can be computed in total $O(\sum |N_{X_i}|)$ time.*

4.2 *DB*-labeling

We also use *UnmatchedNode* (Fig. 3) to identify all unmatched nodes of two general networks N and N'. However, the non-leaf nodes of the networks are labeled in a different way. Each of them is assigned a *DB-label*, which is a pair of integers ($d(u)$,$b(u)$) such that (1) $d(u) = d(v)$ if and only if $D(u) = D(v)$; (2) $b(u) = b(v)$ if and only if $B(u) = B(v)$; and (3) $d(u) = b(v)$ if and only if $D(u) = B(v)$. Furthermore, $d(u) = 0$ if and only if $D(u) = \emptyset$ and $b(v) = 0$ if and only if $B(v) = \emptyset$.

It is clear that two nodes have the same *DB*-label if and only if they induce the same tripartition. The above labeling is called a *DB-labeling* of N and N'.

Algorithm *DBlabeling*
Input: two networks N and N' of the same leaf set S
Output: the *DB*-labeling of N and N'
1 Consider the singleton sets $X_1, X_2, \ldots X_{|S|}$, each containing a distinct leaf in S. For each i, find the *DB*-labeling for N_{X_i} and N'_{X_i}.
2 Repeat the following for $\log |S|$ rounds: Let $X_1, X_2, \ldots$ be the sets of leaves considered in last round. Pair up X_i's and let $X_{2i-1} = X_{2i-1} \cup X_{2i}$. Delete all X_{2i}'s and rename X_{2i-1}'s as X_i's. For each i, compute the *DB*-labeling of N_{X_i} and N'_{X_i} based on the result of last round.
End *DBlabeling*

Fig. 5. Algorithm to compute the *DB*-labeling of two networks of the same leaf set

We compute the *DB*-labeling of N and N' incrementally in a way similar to [7] as in Fig. 5. In step 2, given the *DB*-labeling of N_X and N_Y, we find a *DB*-labeling of $N_{X\cup Y}$ by the following relabeling procedure. This procedure utilizes the concept of Z-stamp of a node u where Z is a set of leaves, which is a pair of integers $(d_Z(u), b_Z(u))$ such that (1) $d_Z(u) = d_Z(v)$ if and only if $D(u) \cap Z = D(v) \cap Z$; and (2) $d_Z(u) = b_Z(v)$ if and only if $D(u) \cap Z = B(v) \cap Z$; and (3) $b_Z(u) = b_Z(v)$ if and only if $B(u) \cap Z = B(v) \cap Z$.

Relabeling procedure:

1. For each $u \in V(N_{X\cup Y})$, compute its X-stamp and Y-stamp as follow. If $u \in V(N_X)$, by Lemma 11, the X-stamp of u equals its label in N_X. Otherwise, its X-stamp is computed by Lemmas 12 and 13. The Y-stamps are calculated similarly.

2. Sort all the pairs $(d_X(u), d_Y(u))$ and $(b_X(u), b_Y(u))$ together and replace each pair by a number such that two pairs are replaced by the same number if and only if they are identical.

Lemma 11. *Let the tripartition of u in N_X be $(A_X(u), B_X(u), C_X(u))$. We have $A_X(u) = A(u) \cap X$, $B_X(u) = B(u) \cap X$ and $C_X(u) = C(u) \cap X$.*

Lemma 12. *Let u be a node in a component C which is in $V(\mathcal{T}_X(N))$. If u is in $V(N_{X\cup Y})$ but not in $V(N_X)$ then it must have an internal child v and an external child w. Furthermore, $d_X(w) = 0$, $d_X(u) = d_X(v)$ and $b_X(u) = d_X(v)$ if v is a hybrid node and $b_X(v)$ otherwise.*

Lemma 13. *Each component C in $V(\mathcal{T}_{X\cup Y}(N)) - V(\mathcal{T}_X(N))$ has at most one child C' in $\mathcal{T}_{X\cup Y}(N)$ such that $D(r(C')) \neq \emptyset$. If no such C' exists, $D_X(u) = \emptyset$ for all $u \in C$. Otherwise, for each $u \in V(C) \cap V(N_{X\cup Y})$, $d_X(u) = d_X(r(C'))$ if u is an ancestor of $r(C')$ and 0 otherwise and $b_X(u) = d_X(r(C'))$ if u is a non-strict ancestor of $r(C')$ and 0 otherwise.*

From the above two lemmas, we can compute the X-stamps and Y- stamps of nodes in $N_{X\cup Y}$ and $N'_{X\cup Y}$ separately. To achieve good running time, for each node in a reduced component, we store the sets of its ancestors and non-strict ancestors in that reduced component. These sets for all nodes in N can be pre-calculated in total $O(k(N)h(N))$ time.

Lemma 14. *The sets of ancestors and non-strict ancestors of any node $u \in V(C) \cap V(N_{X\cup Y})$ can be computed in $O(|V(C) \cap V(N_{X\cup Y})|)$.*

Let $k = \max\{k(N), k(N')\}$, $h = \max\{h(N), h(N')\}$ and $n = \max\{|V(N)|, |V(N')|\}$. The following lemmas state the time complexity of comparing two general networks.

Lemma 15. *Given the DB-labeling of N_X, N'_X, N_Y and N'_Y, the above procedure computes DB-labeling of $N_{X\cup Y}$ and $N'_{X\cup Y}$ using $O(\min\{h+|X\cup Y|, k|X\cup Y|\})$ time.*

Lemma 16. *The tripartition-based distance between two general networks N and N' can be computed in $O(\min\{hn + n\log n, kn\log n\})$ time.*

References

1. D. Bryant and V. Moulton. NeighborNet: an agglomerative method for the construction of planar phylogenetic networks. In *Proc. of the 2^{nd} Workshop on Algorithms in Bioinformatics* (WABI 2002), volume 2452 of *LNCS*, pages 375–391. Springer, 2002.
2. C. Choy, J. Jansson, K. Sadakane, and W.-K. Sung. Computing the maximum agreement of phylogenetic networks. *Theoretical Computer Science*, 335(1):93–107, 2005.
3. W. F. Doolittle. Phylogenetic classification and the universal tree. *Science*, 284:2124–2128, 1999.

4. D. Gusfield and V. Bansal. A fundamental decomposition theory for phylogenetic networks and incompatible characters. In *Proc. of the 9^{th} Annual International Conf. on Research in Computational Molecular Biology* (RECOMB 2005), pages 217–232, 2005.
5. D. Gusfield, S. Eddhu, and C. Langley. Efficient reconstruction of phylogenetic networks with constrained recombination. In *Proc. of the Computational Systems Bioinformatics Conference* (CSB2003), pages 363–374, 2003.
6. J. Hein. Reconstructing evolution of sequences subject to recombination using parsimony. *Mathematical Biosciences*, 98(2):185–200, 1990.
7. Wing-Kai Hon, Ming-Yang Kao, Tak Wah Lam, Wing-Kin Sung, and Siu-Ming Yiu. Non-shared edges and nearest neighbor interchanges revisited. *Inf. Process. Lett.*, 91(3):129–134, 2004.
8. D. H. Huson, T. Dezulian, T. Klöpper, and M. Steel. Phylogenetic super-networks from partial trees. In *Proc. of the 4^{th} Workshop on Algorithms in Bioinformatics* (WABI 2004), pages 388–399, 2004.
9. D. H. Huson, T. Klopper, P. J. Lockhart, and M. A. Steel. Reconstruction of reticulate networks from gene trees. In *Proc. of the 9^{th} Annual International Conf. on Research in Computational Molecular Biology* (RECOMB 2005), pages 233–249, 2005.
10. T. N. D. Huynh, J. Jansson, N. B. Nguyen, and W. K. Sung. Constructing a smallest refining galled phylogenetic network. In *Proc. of the 9^{th} Annual International Conf. on Research in Computational Molecular Biology* (RECOMB 2005), pages 265–280, 2005.
11. J. Jansson, N. B. Nguyen, and W. K. Sung. Algorithms for combining rooted triplets into a galled phylogenetic network. In *Proc. of the 16^{th} Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA 2005), pages 349–358, 2005.
12. J. Jansson and W.-K. Sung. Inferring a level-1 phylogenetic network from a dense set of rooted triplets. In *Proc. of the 10^{th} International Computing and Combinatorics Conference* (COCOON 2004), 2004.
13. J. Jansson and W. K. Sung. The maximum agreement of two nested phylogenetic networks. In *Proc. of the 15^{th} Annual International Symposium on Algorithms and Computation* (ISAAC 2004), pages 581–593, 2004.
14. B. M. E. Moret, L. Nakhleh, T. Warnow, C. R. Linder, A. Tholse, A. Padolina, J. Sun, and R. Timme. Phylogenetic networks: Modeling, reconstructibility, and accuracy. *IEEE Transactions on Computational Biology and Bioinformatics*, 1(1):1–12, 2004.
15. L. Nakhleh, J. Sun, T. Warnow, C. R. Linder, B. M. E. Moret, and A. Tholse. Towards the development of computational tools for evaluating phylogenetic reconstruction methods. In *Proc. of the 8^{th} Pacific Symposium on Biocomputing* (PSB 2003), pages 315–326, 2003.
16. L. Nakhleh, T. Warnow, and C. R. Linder. Reconstructing reticulate evolution in species – theory and practice. In *Proc. of the 8^{th} Annual International Conf. on Research in Computational Molecular Biology* (RECOMB 2004), pages 337–346, 2004.
17. D. Posada and K. A. Crandall. Intraspecific gene genealogies: trees grafting into networks. *TRENDS in Ecology & Evolution*, 16(1):37–45, 2001.
18. D. F. Robinson and L. R. Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53:131–147, 1981.
19. L. Wang, K. Zhang, and L. Zhang. Perfect phylogenetic networks with recombination. *Journal of Computational Biology*, 8(1):69–78, 2001.