AMB ALGORITHMS FOR MOLECULAR BIOLOGY

RESEARCH Open Access

Maximum Parsimony on Phylogenetic networks

Lavanya Kannan* and Ward C Wheeler

Abstract

Background: Phylogenetic networks are generalizations of phylogenetic trees, that are used to model evolutionary events in various contexts. Several different methods and criteria have been introduced for reconstructing phylogenetic trees. Maximum Parsimony is a character-based approach that infers a phylogenetic tree by minimizing the total number of evolutionary steps required to explain a given set of data assigned on the leaves. Exact solutions for optimizing parsimony scores on phylogenetic trees have been introduced in the past.

Results: In this paper, we define the parsimony score on networks as the sum of the substitution costs along all the edges of the network; and show that certain well-known algorithms that calculate the optimum parsimony score on trees, such as Sankoff and Fitch algorithms extend naturally for networks, barring conflicting assignments at the reticulate vertices. We provide heuristics for finding the optimum parsimony scores on networks. Our algorithms can be applied for any cost matrix that may contain unequal substitution costs of transforming between different characters along different edges of the network. We analyzed this for experimental data on 10 leaves or fewer with at most 2 reticulations and found that for almost all networks, the bounds returned by the heuristics matched with the exhaustively determined optimum parsimony scores.

Conclusion: The parsimony score we define here does not directly reflect the cost of the best tree in the network that displays the evolution of the character. However, when searching for the most parsimonious network that describes a collection of characters, it becomes necessary to add additional cost considerations to prefer simpler structures, such as trees over networks. The parsimony score on a network that we describe here takes into account the substitution costs along the additional edges incident on each reticulate vertex, in addition to the substitution costs along the other edges which are common to all the branching patterns introduced by the reticulate vertices. Thus the score contains an in-built cost for the number of reticulate vertices in the network, and would provide a criterion that is comparable among all networks. Although the problem of finding the parsimony score on the network is believed to be computationally hard to solve, heuristics such as the ones described here would be beneficial in our efforts to find a most parsimonious network.

Introduction

Phylogenetic trees, or evolutionary trees, are the basic structures necessary to examine the relationships among organisms. Phylogenetic networks are generalizations of phylogenetic trees that are used to model evolutionary events when they are not only passed via vertical descent, but also by events such as horizontal exchange or recombination that cannot be modeled on a tree. Several different methods and criteria have been used to construct phylogenetic trees. The parsimony method is one such approach for inferring phylogenies, whose general idea was given in [1-3]. In this paper, our focus is on extending this approach to phylogenetic networks.

The parsimony principle states that the simplest explanation that explains the greatest number of observations is preferred over more complex explanations. Most phylogeneticists recognize that inferring genealogy rests on the principle of parsimony, that is, choosing evolutionary trees so as to minimize requirements for ad hoc hypotheses of similarity of observed characters. See [4] for a discussion on some criticisms and relevance of parsimony in phylogenetic analysis.

The cost of each character change event from a parent to child along an edge is weighted as a substitution cost of the parental state to the child state on the edge. The parsimony approach seeks a phylogenetic tree/network

* Correspondence: lkannan@amnh.org
Division of Invertebrate Zoology and Richard Gilder Graduate School, American Museum of Natural History, New York, NY - 10024, USA

that, when we reconstruct the evolutionary events leading to the data on the leaves, minimizes the sum of the weights on the edges. We then face two important problems. First, we must be able to make a reconstruction of events on each vertex of the tree/network, such that the sum of the substitution costs on the edges is minimized (optimize the parsimony score on a network). Second, we must be able to search among all (or a subset of) possible phylogenetic networks for the one(s) that minimizes the parsimony score (find the network that has minimum parsimony score). This problem is NP-hard even for phylogenetic trees [5,6]; and heuristic methods have been developed to reconstruct the phylogenetic network with the given number of reticulation vertices [7,8]. In this paper, we restrict ourselves to the first of these issues, namely in establishing a parsimony criterion and to provide algorithms to achieve heuristics on finding the optimal score for any given phylogenetic network.

An often used structure to represent the evolution of sequences with reticulations is a family of trees each describing the evolution of a segment of the sequence [9,10]. In previous approaches [8,11-13], the parsimony criterion on a network has been defined as the sum total of the substitution costs on the edges of a tree (a subgraph of the network) that minimizes the parsimony score of the site. The parsimony scores for networks given in these definitions are NP-hard to compute. Moreover, a major problem with this extension is that it favors more complex evolutionary relationships by adding larger numbers of edges to trees over simpler ones that contain fewer additional edges, thus having the potential of overestimating the amount of reticulation (horizontal events) in the data. An ad hoc solution has been provided by the authors, namely to restrict blocks of contiguous sites to optimize on the same tree, rather than choosing site-specific most parsimonious tree. However, it is not clear how these blocks are chosen.

In this paper, we recall the definition of phylogenetic networks, and point out the specific class of networks for which a systematic deletion of sets of edges yields phylogenetic trees. To tackle the problem of overestimating the reticulate vertices, we define the parsimony problem simply as the sum of all mutations along all edges in the network. Thus the greater the number of edges in the network, the more the network will be penalized for having excessive number of substitutions and thus later efforts on searching the best network may identify a simpler structure with fewer reticulation events. It is also of interest to generalize the parsimony problems when the substitution costs between states are arbitrarily given. In such cases, the substitution costs along the edges of the network are stored as a cost matrix.

In the upcoming sections of the paper, we first recall the formal definition of a phylogenetic network introduced in [14], that is shown to be appropriate for various datasets. Then we will provide different parsimony criteria and some restrictions on the phylogenetic networks that offer lower complexity solutions to the previous definition. The main focus of this paper is to give a robust definition of the parsimony criterion using any given substitution cost matrix on phylogenetic networks. We also provide efficient upper and lower bounds for the optimum parsimony score on phylogenetic networks by extending the well-known Sankoff algorithm [15,16] for general cost matrix and Fitch algorithm [17] for counting the state changes along the edges of the phylogenetic trees. Our algorithm on general cost matrix works for all phylogenetic networks, thus providing a robust method to analyze any "weighted" parsimony score across the space of all phylogenetic networks. We also present an extension of the Fitch's algorithm for networks. This extension gives an upper bound on the number of state changes along the edges of the network. Additionally, for a restricted class of phylogenetic networks, defined later as phylogenetic networks with no sister reticulations, we present a method to calculate the lower bound on the number of state changes.

The parsimony criterion that we define here is simply the sum of the substitution costs along all the edges of the network. Although this total cost does not reflect the cost of the best tree in the network that displays the evolution of the site, having a overall cost of a network will be relevant while searching the space of all networks with the same number of reticulate vertices for the most parsimonious network. If needed, the tree-like evolutionary pattern of the site may later be extracted from such a parsimonious network that is found for the set of aligned DNA sequences that contains the site. This approach to search for a best network has the advantage of being much more direct than the somewhat ad hoc method that uses the criterion defined in [8]. Both our approach and the one explained in [8] would need additional cost considerations to find an appropriate number of reticulate vertices that reflects the evolutionary changes of a set of aligned DNA strings, for example. In this context, our approach has an advantage of having the score dependent on the number of reticulate edges. Later approaches to find the right number of reticulate vertices may just use some threshold on the score that proceed to consider an additional reticulate vertex if the score is above the threshold. Finding the most parsimonious network is beyond the scope of this paper, and we will only focus on efficiently computing the parsimony score defined here for a given network.

Parsimony on phylogenetic networks

Definition of phylogenetic networks

We follow the definition of the phylogenetic networks as given in [[14], Definition 4, page 16]. For all other graph-theoretical definitions that are not given here, we follow [18]. A rooted phylogenetic network, simply called here a *phylogenetic network*, is defined in [19] as a rooted, directed acyclic graph (DAG), whose root has indegree 0 and the leaves have outdegree 0. The vertices whose indegree is greater than 1 are called *reticulate vertices* and the edges with reticulate vertices as head vertices are called *reticulate edges*. All other edges are termed *tree edges*. The definition given in [14] takes care of the so-called "time-consistency" restraint, namely, that the tree edges take place in a positive time and the reticulate vertices have parents that can only "coexist in time". Hence, forward, we recall the formal definition of phylogenetic networks as given in [14].

Given any directed graph, we say two vertices u and v *cannot coexist in time* if there exists a sequence $P = (p_1, p_2, ..., p_k)$ of paths in N such that:

1. p_i is a directed path that contains at least one tree edge, for every $1 \le i \le k$,
2. u is the tail of p_1, and v is the head of p_k, and
3. for every $1 \le i \le k$ - 1, there exists a network vertex whose two parents are the head p_i and the tail of p_{i+1}.

A *phylogenetic network* N is a rooted DAG obeying the following constraints:

1. Every vertex has indegree and outdegree defined by one of the four combination (0, 2), (1, 0), (1, 2), or (2, 1) - corresponding to, respectively, *root, leaves, internal tree vertices*, and *reticulate vertices*. All vertices other than reticulate vertices are called *tree vertices*.
2. If two vertices u and v cannot coexist in time, then there does not exist a network vertex w with edges (u, w) and (v, w).
3. Given any edge of the network, at least one of its endpoints must be a tree vertex.

Another component of this definition is that for any edge in the phylogenetic network, at least one of its endpoints (either the head or tail) is a tree vertex. Here, we will use this definition. Wherever possible, we point out whether the conditions of the definition are necessary.

Phylogenetic networks can naïvely be thought of as a network that contain as subgraphs, the trees that explain the evolutionary histories of different segments of the input terminal sequences. Given a phylogenetic network, deleting one of each edge incident to a reticulate vertex does not guarantee a resulting phylogenetic tree with the same set of leaves as that of the network. This is an undesirable property, especially if the parsimony criterion is defined by finding a phylogenetic tree inside the network that is most parsimonious for the given site, as defined in [8,11-13]. In order to avoid this problem, it is necessary to assume that no internal vertex has two reticulate children. We call this class of phylogenetic networks as a *phylogenetic network with no sister reticulations*. See Figure 1 for some examples of phylogenetic networks.

Before we proceed to the definition of the parsimony problems, the following is a useful observation. For a phylogenetic network N with no sister reticulations, and having r reticulate vertices and with leaf set X, we denote $\mathcal{T}(N)$ as the set of all trees contained in N. Each such tree is obtained by following two steps: (1) for each reticulate vertex, remove one of the incoming edges, and then (2) for every vertex v of indegree and outdegree 1, whose parent is u and child is w, contract the edges (u, v) and (v, w) into a single edge (u, w). The condition that each edge in N has a tree vertex as an endpoint and that each tree vertex has at least one tree vertex as a child, ensures that the set of leaves of the resulting tree is the same as that of the network. Hence the set $\mathcal{T}(N)$ contains exactly 2^r phylogenetic trees whose leaf set is exactly X.

Maximum Parsimony

We refer the readers to [2,3] for a general description of the idea of parsimony and to the discussion of various parsimony algorithms. It has been pointed out in [9] that the parsimony method for trees can be extended to phylogenetic networks. In a series of papers [8,11,12], one such parsimony criterion is defined by finding a tree in the network that has the best parsimonious score, and efficient algorithms to optimize this criterion on a given phylogenetic network have been devised. Although these algorithms are shown to perform well in practice, they can perform correctly only for phylogenetic networks with no sister reticulations, since it is straightforward to search for an optimal tree in these restricted class of networks. In this section, we state an alternate version of the parsimony problem and in the following sections provide some heuristic solutions for optimizing the score on any phylogenetic network.

Let $[n] = \{1, 2, ..., n\}$ denote the set of leaf labels of a given phylogenetic network N. A function $\lambda: [n] \to \{0, 1, ...,|\Sigma| - 1\}$ is called a *state assignment function* over the alphabet Σ (a non-empty set) for N. We say that a function $\widehat{\lambda} : V(N) \to \{0, 1, \ldots, |\Sigma| - 1\}$ is an *extension of* λ *on* N if it agrees with λ on the leaves of N. For a vertex v in N, we call the $\widehat{\lambda}(v)$ as an *assignment* of $\widehat{\lambda}$ on v. A

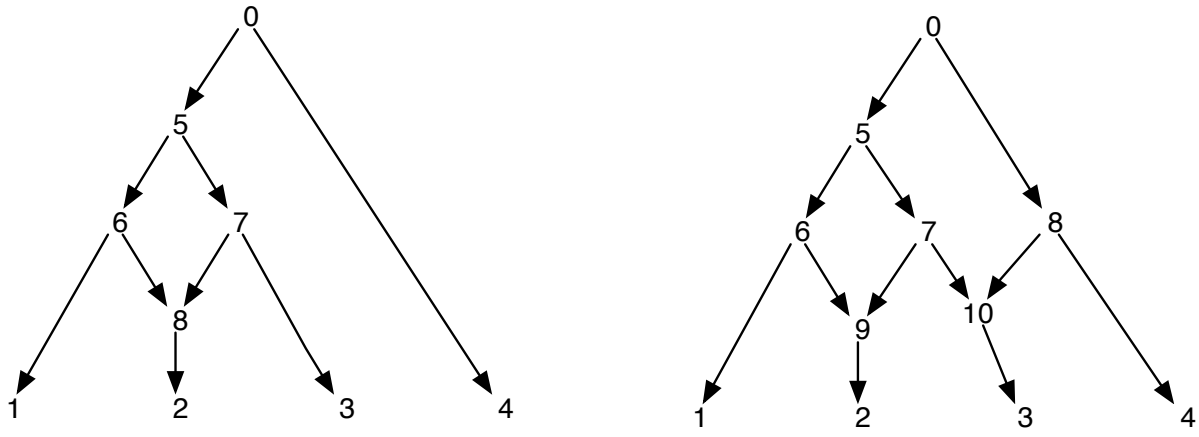

Figure 1 Phylogenetic Networks. The figure shows two phylogenetic networks; the one on the left has only a single reticulate vertex and the one on the right has two reticulate vertices that are sisters to each other. Note that removing the edges (7, 9) and (7, 10) from the network on the right does not result in a tree where vertex 7 is a leaf. This shows that removing one incoming edge each per reticulate vertex does not necessarily produce a tree with the same leaf set as the network. The post-ordering and the pre-ordering of vertices of the network on the left are 1, 2, 8, 6, 3, 7, 5, 4, 0 and 0, 5, 6, 1, 8, 2, 7, 3, 4 respectively and for the network on the right, they are 1, 2, 9, 6, 3, 10, 7, 5, 4, 8, 0 and 0, 5, 6, 1, 9, 2, 7, 10, 3, 8, 4 respectively.

fully assigned network is a network in which all the vertices have labels from {0, 1, ..., |Σ| - 1}. Let C be a cost matrix whose ij^{th} entry c_{ij} is the cost of transforming from state i to state j along any edge in N. If $e = (u, v)$ is an edge in N, where u is the parent of v, we denote $w_e(\hat{\lambda}) = c_{ij}$, where $i = \hat{\lambda}(u)$ and $j = \hat{\lambda}(v)$. For a graph G, we let $E(G)$ denote the edge set of G. Then the parsimony problem is defined as follows.

Input: A phylogenetic network N with leaf labels $[n]$ and a state assignment function λ over the alphabet Σ for N.

Parsimony criterion: For an extension $\hat{\lambda}$ of λ, let

$$P_1(\hat{\lambda}) = \min_{T \in \mathcal{T}(N)} \sum_{e \in E(T)} w_e(\hat{\lambda}),$$

and

$$P_2(\hat{\lambda}) = \sum_{e \in E(N)} w_e(\hat{\lambda}).$$

Output: Given $P \in \{P_1, P_2\}$, find $\hat{\lambda}$ that minimizes $P(\hat{\lambda})$.

We note that $P_1(\hat{\lambda})$ is introduced in [8] and $P_2(\hat{\lambda})$ is the definition we will use in this paper. A more general approach is to minimize $Q(\hat{\lambda}) = \sum_{e \in E(N)} d_e(w_e(\hat{\lambda}))$, where d_e is a non-negative weight function on the edges of N. For the purposes of this paper, we restrict ourselves to $P = P_2$, although the first of our approaches, the dynamic programming solution also holds for $P = Q$.

Parsimony algorithms on networks

Traversing a phylogenetic network

In a network, *vertex traversal* refers to the process of visiting each vertex, exactly once, in a systematic way. Such traversals are classified by the order in which the vertices are visited. We need two types of network vertex traversals to describe our algorithms. These are well-known for phylogenetic trees, and we present them here for phylogenetic networks. The algorithms for the traversals given below start from any given vertex v in the network. In this paper, we will always perform the traversals from the root vertex of the network.

Pre-order traversal of a phylogenetic network from a vertex v

1. Visit the vertex v.
2. Recursively perform pre-order traversal from each child that has not yet been visited.

Post-order traversal of a phylogenetic network from a vertex v

1. Recursively perform post-order traversal from each child that has not yet been visited.
2. Visit the vertex v.

Since a phylogenetic network is a DAG, such traversals will visit all the vertices of the network exactly once. (Refer to [18] for more details on existence on such traversals on DAGs). For the purposes of this

paper, we assume that the vertices of a network are uniquely labelled by integers. Note that the leaves are already labelled from the set $[n]$; and so we use other integers for other vertices. Whenever the child vertices of v are extracted, they are also arranged in increasing order of their integer labelings and the pre- and post-order traversals are performed in this order. This will ensure the following: if vertices v and v' are such that there is no directed path between them, then the vertex v is traversed prior to vertex v' in the pre-order if and only if the vertex v is traversed prior to the vertex v' in the post-order. See the Figure 1 for some examples. With this property, we notice that the pre- and post-order traversals from the root of a phylogenetic network each trace the same spanning tree, which we call here the *traversal tree*.

Dynamic Programming solution

Dynamic programming is used to provide efficient solutions for finding the exact parsimony score when the network is a phylogenetic tree [15,16]. In this section, we show that the same approach can be generalized to phylogenetic networks. Sankoff's algorithm on a tree traverses the vertices of the tree via post-order while computing the minimum costs of each state at each vertex from the leaves to the root, and then chooses the best assignments on each vertex by backtracking from the root to the leaves by traversing the tree vertices via pre-order. Both the phases are presented for networks in Algorithms 1 and 2 respectively. We describe them briefly below. It can be noted that if the network is a tree, then our algorithms match with the pre-order and post-order phases of Sankoff's method for trees.

Given a phylogenetic network N, with leaf vertices labeled $[n]$ and with state assignment function λ over the alphabet Σ, assign to each vertex $v \in V$ a quantity $S_v(i)$ for each $i \in \Sigma$. In phylogenetic trees, $S_v(i)$ denotes the minimum sum of costs of all the events from the vertex v to all the leaves that are reachable from v, given that v is assigned state i and all the descendant vertices from v are each assigned a state. In networks, there is no simple way to compute such a quantity. Instead, we allow $S_v(i)$ to be a lower-bound of the above exact score and it is calculated during the post-order traversal phase.

Post-order traversal phase: If v is a leaf of N, then $S_v(i)$ is assigned 0 if the observed state is state i, and infinite otherwise. Now all we need is a recursion relationship to calculate $S_v(i)$ for rest of the vertices. For each child w of v, we say w satisfies the *post-order traversal condition with respect to v*, or simply *traversal condition with respect to v* in view of the observation in the beginning of this section, if the following hold:

(i) The vertex w is a reticulate vertex and

(ii) if v' is the parent of w other than v, then the vertex v must be traversed prior to v' in the post-order traversal of N.

We now define recursively for each edge (v, w),

$$s_{(v,w)}(i) = \begin{cases} \min_j[c_{ij} + S_w(j)] & \text{if } w \text{ satisfies the traversal condition with respect to } v; \\ \min_j c_{ij} & \text{otherwise.} \end{cases}$$

For a phylogenetic tree, $s_{(v,\ w)}(i)$ always assumes the first of these quantities, and it thus gives the sum of the substitution costs along the edges of the tree that lie below the vertex v, provided the vertex v is assigned the state i. For phylogenetic networks, in order to account for the substitution costs along the edges that lie below a reticulate vertex w just a single time when vertex v is assigned the state i, we let the 'parent' v of w in the traversal tree account for all the substitution costs along all the edges that lie below v. On the other hand, if v is not a parent of w in the traversal tree, $s_{(v,\ w)}(i)$ simply denotes the substitution cost from state i at vertex v to another state at w that is least expensive.

We then define

$$S_v(i) = \sum_w s_{(v,w)}(i), \tag{1}$$

where the sum runs for all child(ren) vertex(s) w of v. As mentioned before, in phylogenetic trees, $S_v(i)$ denotes the minimum possible sum of substitution costs along all the edges from the vertex v to all the leaves that are reachable from v, given that v is assigned state i and all the vertices reachable from v are each assigned a state.

In phylogenetic networks, while calculating $s_{(v,\ w)}(i)$ where w is a reticulate vertex such that (v, w) is not an edge in the traversal tree, there is no prior knowledge of the state that will be later assigned at the reticulate vertex w. Thus $s_{(v,\ w)}(i)$ can only be a lower bound of the edges of the network that lie below the vertex v, if the vertex v is assigned the state i. The reasoning for this is that $s_{(v,\ w)}(i)$ is the substitution cost from state i at vertex v to another state at w that is least expensive, instead of the substitution cost from state i at v to the state at w that will be later assigned. Since the definition of $S_v(i)$ depends on the definition of $s_{(v,\ w)}(i)$, and they are defined recursively, we observe the following: $S_v(i)$ is a lower bound on the sum of substitution costs along the edges of the network that are reachable from the vertex v, provided that v is assigned state i and all the descendant tree vertices are assigned a unique state, and the reticulate vertices are assigned two states that are not necessarily the same. The assigned states of the reticulate vertex contributes to a *conflict* if the states are not the same. Let us suppose that state i is assigned to the root vertex r, and all tree vertices are assigned a unique state,

while the reticulate vertices are assigned two states. Then the cost $S_r(i)$ denotes the minimum possible sum of substitution costs along all the edges of a traversal tree with one of states assigned for reticulate vertices, plus the sum of the substitution costs along the remaining reticulate edges with the alternate assignment state at the reticulate vertices. Since we seek an assignment on the vertices of the network with no conflicts in the reticulate vertices, $S_r(i)$ is a lower bound on the cost of such assignment where the root vertex is assigned i and all vertices are assigned with a unique assignment.

During this phase, we also store the states

$$t_{(v,w)}(i) = \begin{cases} \text{argmin}_j[c_{ij} + S_w(j)] & \text{if } w \text{ satisfies the traversal} \\ & \text{condition with respect to } v; \\ \text{argmin}_j c_{ij} & \text{otherwise.} \end{cases} \quad (2)$$

to be able to backtrack the state of w that achieves the quantity $s_{(v,\ w)}(i)$ during the pre-order phase. See Algorithm 1.

Pre-order traversal phase: We first choose the minimum

$$S = \min_i S_r(i)$$

where r is the root vertex and assign the state that attains the minimum at the root vertex, *i.e.*, let $\widehat{\lambda}(r) = i_r$ such that $S_r(i_r) = S$. For any other vertex w that is not a reticulate vertex, whose parent v is already assigned with a state i, we assign the state $t_{(v,\ w)}(i)$. For a reticulate vertex w whose parent vertices are v and v', let us suppose that v and v' are assigned states i and i' respectively when traversing by the pre-order. The possible states $j = t_{(v,\ w)}(i)$ and $j' = t_{(v',\ w)}(i')$ of w that achieve $s_{(v,\ w)}(i)$ and $s_{(v',\ w)}(i')$ respectively, need not be the same. In other words, it is possible that $j \neq j'$. In this case, we have a *conflict* on the reticulate vertex w. Thus, the dynamic programming technique fails to give an extension for λ whose parsimony score is S. In this case, we simply choose between j and j' for $\lambda(w)$ according to which of the vertices among v and v' is traversed first in the pre-order. Thus, if the vertex w satisfies the traversal condition with respect to v we have $\widehat{\lambda}(w) = j$.

After completing the pre-order phase, we can get the score corresponding to the extension $\widehat{\lambda}$ by first setting $S' = S$ and updating S' at each reticulate vertex w as follows: The upper bound score S' is updated corresponding to the assignment j at vertex w as $S' - c_{i'\,j'} + c_{i'\,j}$. See Algorithm 2. Figure 2 shows an example of how the algorithm runs on a network. Since $S_r(i)$ is a lower bound on the optimum assignment where the root vertex is assigned i and all vertices are assigned with a unique assignment, and since $S = \min_i S_r(i)$, we conclude that S is a lower bound of the optimum we seek to find. See Lemma 1 for a formal proof.

Lemma 1. *The quantity S is a lower bound of the optimum parsimony score on the network N.*

Proof. By the construction of S, we have

$$S = \sum_{(v,w)\in E(N):w \text{ is a tree vertex}} c_{\widehat{\lambda}(v),\widehat{\lambda}(w)} + \sum_{(v,w),\ (v',w)\in E(N)} [c_{\widehat{\lambda}(v),\widehat{\lambda}(w)} + c_{\widehat{\lambda}(v'),t(v',w)(\widehat{\lambda}(v'))}], \quad (3)$$

where the second summand is for the reticulate vertex w with parents are v and v', such that v satisfies the traversal condition w.r.t. w. Thus the cost $c_{\widehat{\lambda}(v),\widehat{\lambda}(w)}$ is the substitution cost from the assigned state $\widehat{\lambda}(v)$ at v to the state $\widehat{\lambda}(w)$ at w. On the other hand, the cost $c_{\widehat{\lambda}(v'),t_{(v',w)}(\widehat{\lambda}(v'))}$ is the substitution cost from the assigned state $\widehat{\lambda}(v)$ at v to the state $t_{(v',w)}(\widehat{\lambda}(v'))$ at w. Note that the state $t_{(v',w)}(\widehat{\lambda}(v'))$ is not necessarily same as the state $\widehat{\lambda}(w)$, and S is the minimum among all assignments that may result in conflicts at the reticulate vertices.

Suppose $\widehat{S}$ is the optimum parsimony score on N with the function μ: $V(N) \to \{0, 1, ..., |\Sigma| - 1\}$ as the extension of λ we have

$$\widehat{S} = \sum_{(v,w)\in E(N):w \text{ is a tree vertex}} c_{\mu(v),\ \mu(w)} + \sum_{(v,w),\ (v',w)\in E(N)} [c_{\mu(v),\mu(w)} + c_{\mu(v'),\mu(w)}], \quad (4)$$

where in the second summand w is a reticulate vertex with parents v and v'. Since μ is a conflict-free assignment that is contained in the set of all assignments among whose costs S is the minimum (compare equation (3) and (4)) we have $S \leq \widehat{S}$. □

Now for the complexity of the algorithm. Suppose the network N has n leaves and r reticulate vertices. Then the number of vertices in N is $2(n + r) - 1$. At each vertex v and for each state i, the quantity S can be computed in $O(k^2)$ time, where $k = |\Sigma|$. The pre-order traversal step involves finding S in $O(k)$ complexity and assigning the best states for each vertex. Also, fixing conflicting reticulate vertex states takes $O(r)$ time. Thus the complexity of the algorithm (presented here) to find a lower and an upper bound is $O((n + r)k^2)$. An alternate upper bound can be obtained in $O(nk^2)$ by simply assigning during the post-order traversal phase, for each reticulate vertex the state that occurs the maximum number of times at the leaves reachable from the respective reticulate vertex; and proceeding via finding $S_v(i)$ for the remaining vertices. The exact optimum can also be obtained by restricting the possible states to a

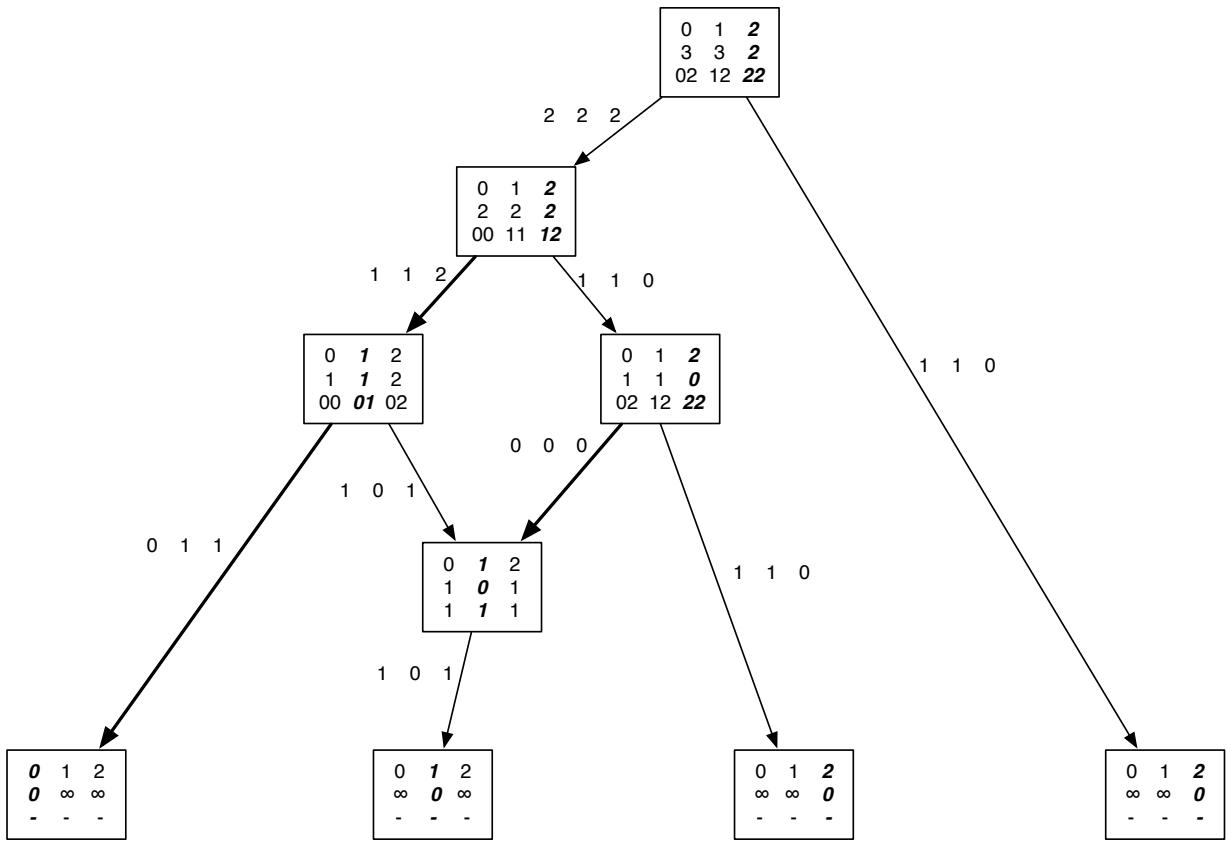

Figure 2 Dynamic programming solution. The dynamic programming solution applied to a phylogenetic network. The states are 0, 1 and 2. The cost matrix used has all 1s except the diagonal elements which are all 0s. The tables shown on each vertex v are the costs, S_v (i) (second row) of each state, i (first row) that are computed during the post-order traversal. Also shown at the vertices are the states of the child w, namely the $t_{(v,\ w)}$ (i) (third row) that correspond to the costs in the second row; when there are two children for a vertex, the entries in the third row are represented as a pair of states of the left child and the right child respectively. Each edge (v, w) is labelled with $s_{(v,\ w)}$ (i) for each state i. During the pre-order traversal, the states for each vertex are selected (shown in bold). The cost of 2 highlighted in bold at the root vertex gives a lower bound, S. The state assigned at each vertex is highlighted in bold. The algorithm finds a total of three substitutions (highlighted by bold edges). This is because the states assigned at the parent vertices of the reticulate vertex give conflicting assignments of 1 and 2 respectively, of which state 1 is assigned at the reticulate vertex. Thus with an extra cost of 1, we get the score of 3 (an upper bound of the optimal score) as the parsimony score corresponding to the assignment shown. Note that the optimum parsimony score on the network is 2 (equal to the lower bound), which can be found by exhaustive search and can be realized by changing the assignment from 1 to 2 for the left parent of the reticulate vertex and from 1 to 2 for the reticulate vertex. Thus the lower bound matches with the optimal score, although the assignment corresponding to the lower bound is not conflict-free and not the same as the assignment corresponding to the optimum.

single state for each reticulate vertex, by running the dynamic programming algorithm for each of the k^r combinations of states for the reticulate vertices, and choosing the minimum among all of them. The time-complexity of this process is $O(nk^{r+2})$.

Algorithm 1 Post-order traversal phase: Calculate the cost of each state at each vertex

1: Input: Network N and the observed states from Σ at the leaves of N, *i.e.*, a state assignment function λ over the alphabet Σ for N.
2: For each leaf v, let S_v (i) = 0 if $\lambda(v) = i$ and ∞ otherwise.
3: Repeat in post-order for each in internal vertex (root, internal tree vertex or reticulate vertex) v in N: For each state i, compute S_v (i) given in (1) and $t_{(v,\ w)}$ (i) for each child w of v, given in (2).
4: Output: $\{(S_v(i), [t_{(v,\ w)}(i)$: w is a child of $v]): v \in V(N), i \in \Sigma\}$.

Minimizing the number of mutations on a phylogenetic network

The Fitch algorithm [17] counts the number of changes in a bifurcating phylogenetic tree for any character set, where the states can change from any state to any other state. Thus, the cost matrix is such that its diagonal

elements are all zeros and the off-diagonal elements are all ones. In this section, we show how Fitch's algorithm extends to finding upper and lower bounds for the number of evolutionary changes in a given phylogenetic network. First, we show that the Fitch algorithm can be extended to give an upper bound for the optimum parsimony score. As before, the post-order and the pre-order traversal phases are given in Algorithms 3 and 4 below. See Figure 3 for an example run of the algorithm.

Algorithm 2 Pre-order traversal phase: Calculate lower and upper bounds of the optimum and the corresponding assignment of the upper bound

1: Input: $\{(S_v\ (i), [t_{(v,\ w)}\ (i)$: w is a child of $v]): v \in\ V\ (N), i \in\ \Sigma\}$.

2: Let $S = \min_i\ S_r\ (i)$, where r is the root vertex and let $\widehat{\lambda}(r) = arg\min_i S_r(i)$.

3: Let $S' = S$

4: For each vertex w in pre-order whose parent vertex v immediately preceeds w in the pre-order, let $\widehat{\lambda}(\omega) = t_{(v,w)}(i)$, where $i = \widehat{\lambda}(v)$.

5: Visit each reticulate vertex w with parents v and v' such that w satisfies the traversal condition with respect to v, with $i = \widehat{\lambda}(v)$, $i' = \widehat{\lambda}(v')$, $j' = t_{(v',w)}(i)$ and update S' as follows:

$$S' \leftarrow S' - c_{i'j'} + c_{i'j}.$$

6: Output: (Lower bound, Upper bound) = (S, S'); extension corresponding to the upper bound score $S' : \widehat{\lambda}$.

Algorithm 3 Post-order traversal phase: Calculate the optimum

1: Input: Phylogenetic network N and a state assignment function λ over the alphabet Σ for N.

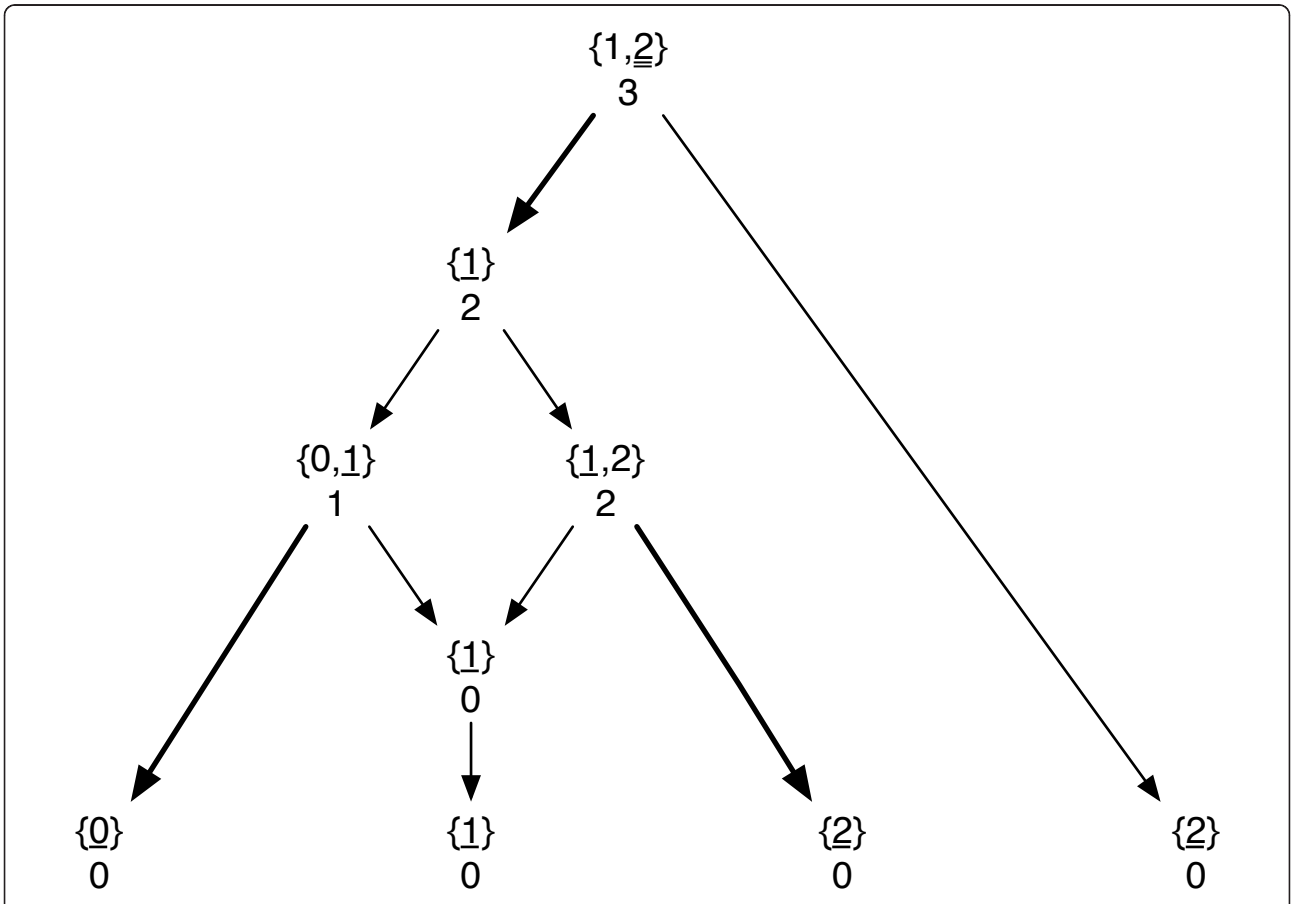

Figure 3 Fitch-type solution. The Fitch-type solution applied to the same phylogenetic network and the leaf data in Figure 2. Each vertex is assigned a set of all possible states, along with a score when the network vertices are traversed in post-order. The score at the root gives a upper bound for the optimal score. The state assignments are given during the pre-order traversal phase and the number of substitutions matches with the score at the root.

2: For every leaf v of N, we are given $A(v) = \{\lambda(v)\}$, a singleton set containing the observed state at the leaf.
3: Set $UB = 0$.
4: Recurse using post-order: For a vertex v of T with children w_1 and w_2, let

$$A(v) = \begin{cases} A(w_1) \cap A(w_2) \text{ if } A(w_1) \cap A(w_2) \neq \emptyset; \\ A(w_1) \cup A(w_2) \text{ otherwise.} \end{cases}$$

and

$$UB \leftarrow \begin{cases} UB & \text{if } A(v_1) \cap A(v_2) \neq \emptyset; \\ UB + 1 \text{ otherwise.} \end{cases}$$

If the vertex v has a single child w, then

$$A(v) = A(w),$$

and

$$UB \leftarrow UB.$$

5: $(\{A(v): v \in V(N)\}, UB)$

Since the pre-order traversal phase gives a conflict-free assignment on the vertices, UB is an upper bound. This is a special case of the dynamic algorithm presented for general cost-matrix. Suppose we restrict N to be a phylogenetic network with no sister reticulations, then any Fitch solution on any tree T in $\mathcal{T}(N)$ forms a lower bound for the optimal score on networks; and adding the cost on edges not in T gives an upper bound for the optimal score. Thus, it is possible to calculate our lower bound for counting the number of character changes only for phylogenetic networks with no sister reticulations, where it is straightforward to find a tree in $\mathcal{T}(N)$.

Algorithm 4 Pre-order traversal phase: Assigning the states

1: Input: Phylogenetic tree N and $(\{A(v): v \in V(N)\}, UB)$.
2: For every vertex v in the tree that is not already assigned, the algorithm computes $\widehat{\lambda}(v)$ as follows: For the root r, $\widehat{\lambda}(r) = \sigma$, where σ is an arbitrary element of $A(r)$. Assign recursively via pre-order: For a vertex v whose parent u is assigned,

$$\widehat{\lambda}(v) = \begin{cases} \widehat{\lambda}(u) & \text{if } \widehat{\lambda}(u) \in A(v); \\ \sigma \in A(v) \text{ otherwise.} \end{cases}$$

3: Fixing the score: for each reticulate vertex v, if u' is not the parent in pre-order, and if $\widehat{\lambda}(u') \in A(v)$, but $\widehat{\lambda}(u') \neq \widehat{\lambda}(v)$, then increment UB by 1.
4: Output: UB and extension function $\widehat{\lambda}$ of λ.

Discussion and conclusion

In the maximum parsimony problem, there are known character-states for a set of taxa (of the species) or Operational Taxonomic Units (OTUs). The problem is to find an order of branching and an ancestral configuration of character-states requiring the minimum number of character-state changes to account for the descent of the OTUs. Short of searching all possible networks, the problem is still in the early stage of being addressed. A more modest goal is to find maximum parsimony ancestral character-states for which both the current character-states and the network are known.

In this paper, we extend the parsimony score defined on phylogenetic trees to phylogenetic networks. This score is defined as the sum of all the substitution costs along all edges of the network. This approach provides an estimate on the amounts of substitutions along all edges, and hence later efforts to find networks with optimal score will fetch networks with fewer reticulations. Although the complexity of finding the exact score on a given network is unknown, we suspect that the problem will also be NP-hard as with the definition of the problem via previously defined criterion. We extended Sankoff and Fitch algorithms that are well-known for trees to heuristic algorithms on networks that compute upper and lower bounds for then optimal parsimony score. Sankoff's algorithm works for any general substitution cost matrix, and our extension also provides a robust method to calculate heuristic bounds for the optimal score on networks with non-homogeneous substitution costs.

We ran our algorithm for networks with fewer than 10 leaves with at most 2 reticulation events and found that for all these networks, the bounds matched with the exact optimum, which we were able to compute using our exact algorithm. Future efforts in this area of research will involve tightening these bounds for general phylogenetic networks. This will enable us to proceed to the next step of the parsimony problem, namely to find the networks with optimum parsimony score.

Acknowledgements
This research was supported by Defense Advanced Research Projects Agency (DARPA) grant W911NF-10-1-0339.

Authors' contributions
WW conceived the study, and participated in its design and coordination, LK implemented the algorithms in OCAML. LK wrote the paper and WW proofread all versions of the manuscript during its preparation. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Received: 19 October 2011 Accepted: 2 May 2012
Published: 2 May 2012

References

1. Edwards AWF, Cavalli-Sforza LL: **The reconstruction of evolution.** *Annals of Human Genetics (also published in Heredity 18: 553)* 1963, **27**:105-106.
2. Farris JS: **Estimation of Conservatism of Characters by Constancy Within Biological Populations.** *Evolution* 1966, **20(4)**:587-591.
3. Kluge AG, Farris JS: **Quantitative Phyletics and the Evolution of Anurans.** *Systematic Zoology* 1969, **18.**
4. Farris JS: In *The logical basis of phylogenetic analysis. Volume 2.* Bronx, New York: New York Botanical Garden; 1983:7-36.
5. Foulds L, Graham R: **The steiner problem in phylogeny is NP-complete.** *Advances in Applied Mathematics* 1982, **3**:43-49.
6. Day W: **Computationally difficult parsimony problems in phylogenetic systematics.** *Journal of Theoretical Biology* 1983, **103(3)**:429-438.
7. Hein J: **A heuristic method to reconstruct the history of sequences subject to recombination.** *Journal of Molecular Evolution* 1993, **36**:396-405.
8. Nakhleh L, Jin G, Zhao F, Mellor-Crummey J: **Reconstructing Phylogenetic Networks Using Maximum Parsimony.** *Proceedings of the 2005 IEEE Computational Systems Bioinformatics Conference* 2005, 93-102.
9. Hein J: **Reconstructing evolution of sequences subject to recombination using parsimony.** *Math Biosci* 1990, **98(2)**:185-200.
10. Nakhleh L, Warnow T, Linder CR, St John K: **Reconstructing reticulate evolution in species-theory and practice.** *J Comput Biol* 2005, **12(6)**:796-811.
11. Jin G, Nakhleh L, Snir S, Tuller T: **Efficient parsimony-based methods for phylogenetic network re-construction.** *Bioinformatics* 2007, **23(2)**:123-128.
12. Jin G, Nakhleh L, Snir S, Tuller T: **A new linear-time heuristic algorithm for computing the parsimony score of phylogenetic networks: theoretical bounds and empirical performance.** *Proceedings of the 3rd international conference on Bioinformatics research and applications* ISBRA'07, Berlin, Heidelberg: Springer-Verlag; 2007, 61-72.
13. Nguyen CT, Nguyen NB, Sung WK, Zhang L: **Reconstructing recombination network from sequence data: the small parsimony problem.** *IEEE/ACM Trans Comput Biol Bioinform* 2007, **4(3)**:394-402.
14. Moret BM, Nakhleh L, Warnow T, Linder CR, Tholse A, Padolina A, Sun J, Timme R: **Phylogenetic networks: modeling, reconstructibility, and accuracy.** *IEEE/ACM Trans Comput Biol Bioinform* 2004, **1**:13-23.
15. Sankoff D: **Minimal mutation trees of sequences.** *SIAM Journal of Applied Mathematics* 1975, **28**:3542.
16. Sankoff D, Rousseau P: **Locating the vertices of a Steiner tree in an arbitrary metric space.** *Math Progr* 1975, **9**:240-276.
17. Fitch WM: **Toward Defining the Course of Evolution: Minimum Change for a Specific Tree Topology.** *Systematic Zoology* 1971, **20(4)**:406-416.
18. Schrijver A: *Combinatorial Optimization - Polyhedra and Efficiency* Springer; 2003.
19. Huson D, Rupp R, Scornavacca C: *Phylogenetic Networks: Concepts, Algorithms and Applications* Cambridge University Press; 2011.

doi:10.1186/1748-7188-7-9
Cite this article as: Kannan and Wheeler: **Maximum Parsimony on Phylogenetic networks.** *Algorithms for Molecular Biology* 2012 **7**:9.