International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, Volume 2, Issue 7, July 2012)

Constructing Rooted Phylogenetic Networks from Triplets based on Height Function

Hadi Poormohammadi[1], Changiz Eslahchi[2]

[1]Department of Mathematics, Shahid Beheshti University, G.C. Tehran, Iran
[2] School of Computer Science, Institute for research in Fundamental Sciences (IPM), Tehran, Iran
1poormohammadi@ipm.ir
[2]ch-eslahchi@sbu.ac.ir

***Abstract*— The problem of constructing an optimal rooted phylogenetic network from a set of rooted triplets is NP-hard. In this paper, we present a novel method called NCH, which tries to construct a rooted phylogenetic network with the minimum number of reticulation nodes from an arbitrary set of rooted triplets based on the concept of the height function of a tree and a network. We report the performance of this method on simulated data.**

***Keywords*— Rooted phylogenetic network, Triplet, Density, Consistency, Height function, Reticulation node**

I. Introduction

Phylogenetic networks are a generalization of phylogenetic trees that permit the representation of non-tree-like underlying histories. A rooted phylogenetic network is a rooted directed acyclic graph in which no node has indegree greater than 2 and the outdegree of each node with indegree 2 is 1. Such nodes are called reticulation nodes.

In rooted phylogenetic networks the nodes with indegree 1 and outdegree 0 are called leaves and are distinctly labeled by a set of given taxa. Mathematicians are interested in developing methods that infer a phylogenetic tree or network from basic building blocks. In the computation of a rooted tree or network, one group of the basic building blocks are rooted triplets, the rooted binary trees on three taxa [1]. In 1981, Aho et al., studied the problem of constructing a rooted tree from a set of rooted triplets [2]. They proposed an algorithm called BUILD algorithm which shows that, given a set of rooted triplets, it is possible to construct in polynomial time a rooted tree that all the input rooted triplets are contained in it or decides that no such tree exists. When there is no rooted tree for a given set of rooted triplets one may try to produce an optimal phylogenetic network.

In this context, the goal is to compute an optimal rooted phylogenetic network that contains all the rooted triplets. One possible optimality criterion is to minimize the *level* of the rooted phylogenetic network, which is defined as the maximum number of reticulation nodes contained in any biconnected component of the rooted phylogenetic network. The other optimality criterion is to minimize the number of reticulation nodes [1].

In [3] and [4] the authors considered the problem of deciding whether, given a set of rooted triplets as input, is it possible to construct a level-1 rooted phylogenetic network that contains all the input triplets? They showed that, in general, this problem is NP-hard. However, in [4] the authors showed that when the set of rooted triplets is dense, which means that for each set of three taxa there is at least one rooted triplet in the input set, the problem can be solved in polynomial time. After their results, all research in this new area has up to this point focused on constructing rooted phylogenetic networks from dense sets of rooted triplets. LEV1ATHAN is an algorithm for generating a level-1 rooted phylogenetic network from a set of rooted triplets [5]. Specifically, it attempts to find a level-1 rooted phylogenetic network consistent with as many of the input rooted triplets as possible. This problem is an NP-hard problem [5]. The algorithm by [6] can be used to find a level-1 or a level-2 rooted phylogenetic network which minimizes the number of reticulation nodes, if such a network exists. In [6] the authors also showed that for a dense set of rooted triplets τ, if τ is precisely equal to the set of rooted triplets consistent with some rooted phylogenetic network, then they can construct such a rooted phylogenetic network with smallest possible level in time $O(|\tau|^{k+1})$, where k is a fixed upper bound on the level of the network. In addition based on the ideas described in [6], for a given dense set of rooted triplets τ, the authors proposed SIMPLISTIC algorithm which always returns some rooted phylogenetic network consistent with τ. But it does not give any minimality guarantees.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, Volume 2, Issue 7, July 2012)

In [7] the authors presented TripNet algorithm which tries to construct a rooted phylogenetic network with the minimum number of reticulation nodes from an arbitrary set of rooted triplets. In this paper we present a new heuristic algorithm called NCH for constructing rooted phylogenetic networks with the minimum number of reticulation nodes from an arbitrary set of rooted triplets. Similar to TripNet, NCH is applicable on any set of rooted triplets which is not necessarily dense. In this paper we follow the same definitions and notation of [7].

This paper is organized as follows. In section II we present some definitions and notation. In section III, first the concepts of the directed graph G_τ related to a set of triplets τ and the height function of a tree are introduced and we restate BUILD algorithm based on these two concepts. Then we generalize the concept of the height function from trees to networks and present NCH.

In section IV we discuss the runtime of NCH. In section V the results are presented and we compare our results with the SIMPLISTIC results and discuss the performance of NCH.

II. Definitions And Notation

Let X be a set of taxa. A *rooted phylogenetic tree* (*tree* for short) on X is a rooted unordered leaf labeled tree whose leaves are distinctly labeled by X and every node which is not a leaf has at least outdegree two.

A *directed acyclic graph* (DAG) is a directed graph that is free of directed cycles. A DAG G is *connected* if there is an undirected path between any two nodes of G. It is *biconnected* if it contains no node whose removal disconnects G. A biconnected component of a graph G is a maximal biconnected subgraph of G. A *rooted phylogenetic network* (*network* for short) on X is a rooted DAG in which *root* has indegree 0 and outdegree 2 and every node except the root satisfies one of the following conditions:

a) It has indegree 2 and outdegree 1. These nodes are called *reticulation nodes*.

b) It has indegree 1 and outdegree 2.

c) It has indegree 1 and outdegree 0. These nodes are called *leaves* and are distinctly labeled by X.

A network is said to be a level-k network if each of its biconnected component contains at most k reticulation nodes. A tree can be considered as a level-0 network.

A *rooted triplet* (*triplet* for short) is a rooted binary unordered tree with three leaves. We use $ij|k$ to denote a triplet with taxa i and j on one side and k on the other side of the root (Fig.1(a)).

A set of triplets τ is called *dense* if for each subset of three taxa, there is at least one triplet in τ .A triplet $ij|k$ is c*onsistent* with a network N or equivalently N is consistent with $ij|k$ if the leaf set of $ij|k$ is the subset of the leaf set of N, and N contains a subdivision of $ij|k$, i.e. if N contains distinct nodes u and v and pairwise internally node-disjoint paths $u \to i$, $u \to j$, $v \to u$ and $v \to k$. Fig. 1(b) shows an example of a network which is consistent with $ij|k$. A set τ of triplets is consistent with a network N if all the triplets in τ are consistent with N. We use the symbols $\tau(N)$ and L_N to represent the set of all triplets that are consistent with N and the set of labels of its leaves respectively. For any set τ of triplets define $L(\tau) = \cup_{t\in\tau} L_t$. The set τ is called a set of triplets on X if $L(\tau) = X$.

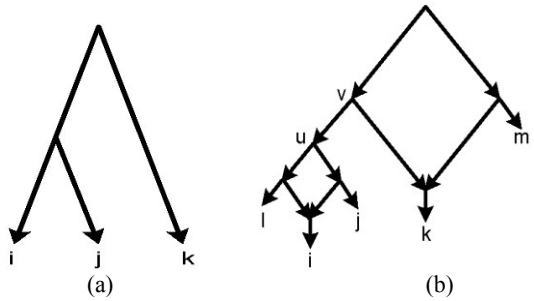

Figure 1: a) Triplet $ij|k$, b) $ij|k$ is consistent with the network.

III. Methods

In this section we first review the concept of the directed graph related to a set of triplets, the height function of a tree, and restate BUILD algorithm based on these concepts which is introduced in [7]. In [7] the authors introduced a generalization of the concept of the height function to the networks. In this section based on this generalization we present our new heuristic algorithm.

Definition 1. *Let* τ *be a set of triplets. Define* G_τ *, the directed graph related to* τ *, by* $V(G_\tau) = \{\{i, j\} : i, j \in L(\tau), i \neq j\}$ *(we denote* $\{i, j\}$ *by* ij *for short) and* $E(G_\tau) = \{(ij, ik) : ij \mid k \in \tau\} \cup \{(ij, jk) : ij \mid k \in \tau\}$.

The graph G_τ has an important role in remaining of the paper. Let $\binom{X}{2}$ denotes the set of all subsets of X of size 2.

Definition 2. *Let* X *be an arbitrary finite set. A function* $h : \binom{X}{2} \to \square$ *is called a height function on* X.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, Volume 2, Issue 7, July 2012)

Let T be a rooted tree with the root r, c_{ij} be the lowest common ancestor of the leaves i and j, and l_T denotes the length of the longest path started from r. For any two nodes x and y, let $d_T(x,y)$ denotes the number of edges of the path between x and y.

Definition 3. *The height function of T, h_T is defined as $h_T(i,j)=l_T-d_T(r,c_{ij})$ where i and j are two distinct leaves of T.*

Let τ be a set of triplets, G_τ be a DAG and l_{G_τ} denotes the length of the longest path in G_τ. Since G_τ is a DAG, the set of nodes with outdegree zero is nonempty. Assign $l_{G_\tau}+1$ to the nodes with outdegree zero and remove them from G_τ. Assign l_{G_τ} to the nodes with outdegree zero in the resulting graph and continue this procedure until all nodes are removed.

Definition 4. *For any two distinct $i, j \in L(\tau)$, define $h_{G_\tau}(i,j)$ as the value that is assigned by the above procedure to the node ij and call it the height function related to G_τ.*

Let τ be a set of triplets. In [7] the authors showed that if τ is consistent with a tree then G_τ is a DAG and h_{G_τ} is well-defined. Now we restate BUILD algorithm, using height function. which is referred by *HBUILD* [7].

Let h be a height function on X. Define a weighted complete graph (G,h) where $V(G)=X$ and edge $\{i,j\}$ has weight $h(i,j)$. Remove the edges with maximum weight from G. If removing these edges result in a connected graph the algorithm stops. Otherwise, the process of removing the edges with maximum weight is continued in each connected component until each connected component contains only one node. At the end of this procedure one can reconstruct the tree by reversing the steps of the algorithm similar to BUILD algorithm.

The algorithm above decides in polynomial time whether a tree with height function h exists.

We denote T_τ, the unique tree that is produced by HBUILD. Now if τ is a set of triplets which is consistent with a tree then G_τ is a DAG, $h_{G_\tau}=h_{T_\tau}=h$, and HBUILD constructs T_τ [7].

Now we are going to generalize the concept of the height function form trees to networks.

This generalization is not straightforward because the concept of (lowest) common ancestor of two leaves of a network is not well-defined.

Let N be a network with the root r and l_N be the length of the longest directed path from r to the leaves. For each node u consider $d(r,u)$ as the length of the longest directed path from r to u. For any two nodes u and v, we call u an *ancestor* of v, if there exists a directed path from u to v. If u is an ancestor of v then we say that v is *lower* than u.

Let i and j be two leaves of N. c is called a *lowest common ancestor* of i and j in N, if c is a common ancestor of i and j and there is no common ancestor of i and j lower than c. For any two leaves i and j, let C_{ij} denote the set of all lowest common ancestors of i and j.

Definition 5. *For each pair of leaves i and j, define $h_N(i,j)=\min\{l_N-d(r,c):c\in C_{ij}\}$ and call it the height function of N.*

Obviously, every network N indicates a unique height function h_N. But two different networks may have the same height function. In [7] the authors proved that for a given height function h there is a network N such that $h_N=h$. In [7] the authors introduced a computational method for computing h_N using Integer Programming and proved that a triplet $ij|k$ is consistent with a tree T if and only if $h_T(i,j)<h_T(i,k)$ or $h_T(i,j)<h_T(j,k)$. Also in [7] it was proved that for a given network N and its three distinct leaves i, j, and k, if $h_N(i,j)<h_N(i,k)$ or $h_N(i,j)<h_N(j,k)$ then $ij|k$ is consistent with N.

Based on these concepts the authors introduced the following Integer Programming $IP(\tau,s)$ for a given set of triplets τ with $|L(\tau)|=n$.

Maximize $\sum_{1\le i,j\le n} h(i,j)$,

Subject to: $h(i,k)-h(i,j)>0 \quad ij|k\in\tau$,
$h(j,k)-h(i,j)>0 \quad ij|k\in\tau$,
$0<h(i,j)\le s \quad 1\le i,j\le n$.

It was proved that G_τ is a DAG if and only if for some integer s, $IP(\tau,s)$ has a feasible solution. In this case the minimum number s, for which $IP(\tau,s)$ has a feasible solution, is $l_{G_\tau}+1$ Also it was proved that if τ is consistent with a tree then h_{T_τ} is the unique optimal solution to the $IP(\tau,l_{G_\tau}+1)$. The above contents imply that the solution of the above IP is a good approximation of the height function of a network N which is consistent with τ [7].

Now we are ready to present our method. If there is a tree consistence with a set of triplet τ, using HBUILD, we construct this tree.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, Volume 2, Issue 7, July 2012)

If there is no tree consistent with a set of triplets τ, one possibility is that G_τ is not a DAG. In this case we remove some edges from G_τ in such a way that the resulting graph G'_τ is a DAG. Removing minimum number of edges from a directed graph to make it a DAG is known as the *minimum Feedback Arc Set* problem which is NP-hard [8]. Thus we use the heuristic algorithm GR [9], and try to remove as minimum number of edges as possible from G_τ in order to lose minimum information.

For simplicity from here we denote G'_τ by G_τ. Now similar to HBUILD, we remove the edges with maximum weight from (G,h). If in one step, removing the edges with maximum weight from a connected component C, results in a connected component C', we use the following method to disconnect C'. The process of removing edges with maximum weights from C' is continued until it becomes disconnected. Continue this process for each connected component until each connected component contains only one node.At the end, like HBUILD one can reconstruct a unique tree by reversing the steps of the above method.

Now our goal is to add some edges to this tree in order to obtain a network consistent with the given set of triplets τ based on the concept of the height function of networks.

Let r be the root of T, i be one of its leaves, and l_i be the length of the path between r and i. The method of constructing T shows that for each leaf i, $l_i \le l_{G_\tau}+1$. We add $l_{G_\tau}+1-l_i$ nodes to the edge for which one of its two ends is i. In this new tree for each two leaves i and j if $h(i,j)$ is the same as the value which is obtained from the above IP we do nothing. Else let $h_{IP}(i,j)$ be its IP value which is not the same as $h(i,j)$.

We add a new edge such that one of its ends is the node of the path between r and i which has distance $l_{G_\tau}+1-h_{IP}(i,j)$ from r and the other ends is the node *of* the path between r and j which has distance $l_{G_\tau}+1-h_{IP}(i,j)$ from r and randomly assign a direction to it. Suppose that the triplet $ij|k \in \tau$ is not consistent with this network. Connect the node which its child is i to the node which its child is j. Now $ij|k$ is consistent with this new network. We continue this process until the final network be consistent with τ. We named this algorithm Network Construction with Height; Algorithm NCH for short.

I. RUNTIME

In this section we study the time complexity of NCH. Let $|L(\tau)| = n$ and $|\tau| = m$. At the beginning, G_τ should be computed. Its time complexity is $O(m)$. Then, if G_τ is not a DAG, the algorithm GR is applied in $O(|\text{edges}|)$ time, which is equivalent to $O(m)$. Now the nodes with outdegree zero are recognized and then Topological sort is performed on G'_τ Its time complexity is O(|nodes|+|edges|) or equivalently $O(m+n^2)$. The next step is assigning the height to each node of DAG in $O(n^2)$. After these steps, the graph (G,h) is constructed in $O(n^2)$. So the runtime of the above steps is $O(m+n^2)$. Now we are ready to perform the tree construction method. In each step, removing the edges with maximum weight is done for each connected component in $O(m)$. In addition, in each step, it is necessary to compare the number of connected components with the previous step. Thus, *DFS* algorithm is performed in $O(n)$. The overall run time is $O(mn)$. Since there are n nodes the total runtime is $O(mn^2)$. So the tree construction runtime is $O(mn^2)$.

The network construction procedure from tree is done in $O(mlogn)$. So the overall runtime of NCH is $O(mn^2)$.

IV. RESULTS AND DISCUSSION

In order to study the performance of NCH we perform the following scenario. We use standard methods to obtain triplets from (biological) sequences data [2]. Triplets are easy to construct using the input sequences: Maximum Parsimony or Maximum Likelihood are existing methods for constructing triplets [2]. PhyML is a software that construct weighted unrooted binary trees based on the input sequences using Maximum Likelihood criterion. PhyML can be used to produce triplets. It is enough to add an outgroup to all the sets consisting of three sequences in the input and construct a quartet using the current methods. At the end, we can extract the triplet corresponding to each of these groups by removing the outlier sequence. Finally, note that this simple and intuitive method works with a certainty threshold where we have the option to adjust this threshold. In fact the unique inner edge weight in the corresponding quartet is considered as the *threshold* and the triplets with the threshold at most zero are not considered. We generate 1000 sets of sequences of size at least 10 and at most 30 under biological presumptions, and for each of them we obtained a set of triplets using Maximum Likelihood criterion. Then we compare the NCH results with the SIMPLISTIC results on these sets of triplets. SIMPLISTIC is a method which constructs networks from dense sets of triplets.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, Volume 2, Issue 7, July 2012)

Since for a set of triplets which is obtained from the above method there might be triplets with the threshold at most zero, then a set of triplets might not be dense.In order to make it dense, some triplets are randomly added to it.

The results show that when the level of the Simplistic network is at most 6, SIMPLISTIC outperforms NCH in both the number of reticulation nodes and the level of the resulting networks. In these cases on average the difference between these numbers is at most 4 and in average is 3.

In the 82 % of these cases, the runtime of both algorithms is nearly the same. In the remaining 18 % of the cases, the runtime of NCH is at most 8 seconds, but the Simplistic runtime is at least 55 seconds.For the network with level greater than 6, the runtime of Simplistic is very high and in most cases after one hour, Simplistic does not give any output. But the NCH runtime for all these cases is at most 3 minutes and the resulting network contains at most 22 reticulation nodes. For a dense set of triplets τ, SIMPLISTIC checks all possible solutions starting from trees until it finds an answer with the minimum level. When the resulting network is more complex, the search space grows rapidly. So SIMPLISTIC take much times to give a solution. The results show that SIMPLISTIC is appropriate for dense sets of triplets which are consistent with the networks with level at most 6. For more complex networks with the level greater than 6, Simplistic does not work well and it takes much time to give output. It means that for complex networks with high levels, the Simplistic runtime increased exponentially. But in all cases NCH output a network in an appropriate time. It is remarkable that in contrast with Simplistic which just works for dense sets of triplets, NCH works for any arbitrary given set of triplets.

ACKNOWLEDGMENT

We would like to thank Shahid Beheshti University for its supports. This work was partially supported by a grant from IPM.

References

[1] Huson, D.H., Rupp, R., and Scornavacca,.C. 2010. Phylogenetic Networks Concepts, Algorithms and Applications, Cambridge University Press.

[2] Aho, A.V., Sagiv, Y., Szymanski, T.G., and Ullman, J.D. 1981. Inferring atree from lowest common ancestors with an application to the optimization of relational expressions. SIAM J. Comp., vol. 10, pp. 405-421.

[3] Jansson, J., Nguyen, N.B., and Sung, W.K. 2006. Algorithms for combining rooted triplets into a galled phylogenetic network. SIAM Journal on Computing, vol. 35, pp. 1098-1121.

[4] Jansson, J., and Sung, W.K. 2006. Inferring a Level-1 Phylogenetic Network from a Dense Set of Rooted Triplets. Theoretical Computer Science, vol. 363, pp. 60-68.

[5] Huber, K., Iersel, L. van., Kelk, S., and Suchecki, R. 2010. A Practical Algorithm for Reconstructing Level-1 Phylogenetic Networks. IEEE/ACM Transactions on Computational Biology and Bioinformatics.

[6] Iersel, L. van., and Kelk, S. 2009. Constructing the Simplest Possible Phylogenetic Network from Triplets," Algorithmica, DOI 10.1007/s00453-009-9333-0.

[7] Poormohammadi, H., Eslahchi, Ch., and Tusserkani R. 2012. TripNet: A Heuristic Algorithm for Constructing Rooted Phylogenetic Networks from Triplets, *arXiv:1201.3722v1*.

[8] Karp, R. 1972. Reducibility among combinatorial problems. Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., pp. 85-103.

[9] Eades, P., Lin, X., and. Smyth, W. F. 1993. A fast and effective heuristic for the feedback arc set problem. Information Processing Letters, vol. 47, pp. 319-323.