VOL. 3, NO. 7, July 2012 ISSN 2079-8407
Journal of Emerging Trends in Computing and Information Sciences

http://www.cisjournal.org

A New Heuristic Algorithm for MRTC Problem

Hadi Poormohammadi
Department of Mathematics, Shahid Beheshti University, G.C. Tehran, Iran

ABSTRACT

A rooted phylogenetic tree is a rooted tree which represents the evolutionary history of currently living species. A rooted binary tree on three leaves is a rooted triplet. The problem of determining whether there exists a rooted phylogenetic tree that contains all of the rooted triplets is polynomial solvable, while the problem of finding a rooted phylogenetic tree that contains the maximum number of rooted triplets is known to be NP-hard. This maximization problem is known as the Maximum Rooted Triplets Consistency (MRTC) problem. In this paper we present a new heuristic algorithm for this problem based on the concept of the height function of a tree. We study the performance of our algorithm from both simulation and theoretical viewpoints.

Keywords: *Rooted phylogenetic tree, Rooted triplet, Heuristic algorithm, Height function, Consistency*.

1. INTRODUCTION

Phylogenetics is a field that studies and models the relation between currently living species. The simplest possible model is a phylogenetic tree, a tree in which its leaves are distinctly labeled by species and represents the evolutionary relationship of species and the way that they have evolved. In general, the biological sequences information of species are available. There are many approaches to convert these sequences data into the evolutionary relationships [1]. In the simplest approach, sequences are directly considered as the input and there are methods that construct phylogenetic trees directly from these sequences. In the second approach, the information of sequences convert into a distance matrix and then a phylogenetic tree is constructed from the distance matrix. In the third approach, a quartet which is an unrooted binary tree on four leaves, is assigned to each subset of four sequences and then a phylogenetic tree is obtained from these quartets. If we consider sequence-based as a one-species trees approach, distance-based as a two-species trees approach, and quartet based as a four species trees approach, one can recognize that methods incorporating three species trees (rooted triplets) as a potentially fruitful method. Rooted triplets are easy to construct using the input sequences: Maximum Parsimony or Maximum Likelihood are existing methods for constructing rooted triplets [2]. In some applications, the data obtained experimentally may already have the form of rooted triplets; for example, Sibley-Ahlquist-style DNA-DNA hybridization experiments can yield triplets directly [3].

Mathematicians are interested in developing methods that infer a rooted phylogenetic tree from a set of rooted triplets [4]. Using rooted triplets as input first appeared in the context of database theory. In 1981, Aho et al., studied the problem of constructing a rooted tree from a set of rooted triplets [5]. They showed that, given a set of rooted triplets, it is possible to construct a rooted tree which contains all the input rooted triplets in polynomial time, or decide that no such tree exists [5]. Their algorithm called BUILD algorithm. Their method does not construct anything when there is no rooted tree which contains all the input rooted triplets.

When there is no rooted phylogenetic tree for a given set of rooted triplets, one may try to produce a rooted phylogenetic tree that contains the maximum number of rooted triplets. This problem is known as the Maximum Rooted Triplets Consistency problem (MRTC) or the Maximum Inferred Local Consensus Tree problem (MILCT) [6]. MRTC is proved to be NP-hard ([7]-[9]). In [10] the authors showed that in general this problem is APX-hard for non-dense data. A set of rooted triplets is called dense, if for each subset of three taxa there is at least one rooted triplet in the input set. In [11] the authors proved that even for the dense set of rooted triplets, this problem is NP-hard too.

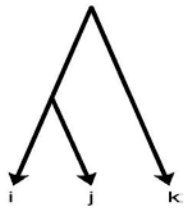
In [12] the authors proposed two algorithms by modifying BUILD algorithm. Their first and second algorithms are referred as One-Leaf-Split and Min-Cut-Split respectively One-Leaf-Split is guaranteed to be consistent with at least one third of the input rooted triplet set. In [8] Wu introduced a bottom up heuristic approach, called BPMF (Best Pair Merge First), which runs in $o(mn^3)$. Also he proposed an exact exponential algorithm which runs in $o((m+n^2)3^n)$ time and $o(2^n)$ space. The BPMF results show that its performance on randomly generated data is well in average [8]. In [6] the authors presented BPMR (Best Pair Merge with Reconstruction), which is a modified version of BPMF. BPMR runs in $o(mn^3)$.. In [10] the authors designed a modified version of BPMF to achieve an approximation ratio of at most three. They also investigated how Min RTI (Minimum Rooted Triplet Inconsistency) can be used to approximate MRTC and proved that MRTC admits a polynomial-time $(3-\frac{2}{n-2})$-approximation.

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. Then we restate BUILD algorithm based on these two new concepts. Then we present a heuristic algorithm for MRTC.

In section IV we discuss the runtime of the decudortnialgorithm. In section V the results are presented and we compare our results with the results of the best existing method, BPMR. Finally in section VI we discuss about the efficiency of our algorithm.

2. DEFINITONS 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 that is not a leaf has at least out degree two. A directed acyclic graph (DAG) is a directed graph that is free of directed cycles. A rooted triplet (triplet for short) is a binary rooted unordered tree with three leaves. We use ij|k to denote a triplet with taxa i and j on one side of the root and k on the other side of the root (Figeru. 1).



giF**1:** Triplet $ij|k$.

A triplet $ij|k$ is consistent with a tree T or equivalently T is consistent with ij|k if T contains a subdivision of ij|k, i.e. if T contains distinct nodes u and v and pair wise internally node-disjoint paths u → i, u → j, v → u and v → k.

Figure. 2 shows an example of a tree and a triplet consistent with it.

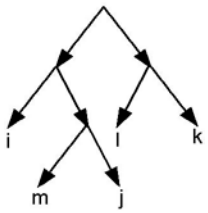


Fig 2: Triplet $ij|k$ is consistent with the above tree.

A set τ of triplets is consistent with a tree T if all the triplets in τ are consistent with T. We use the symbol L_T to represent the set of labels of its leaves. 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 $L(\tau) = X$.

3. METHODS

In this section we first introduce 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 new concepts. Then we present our heuristic algorithm for MRTC problem based on these concepts.

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 $E(G_\tau) = \{(ij,ik): ij|k \in \tau\} \cup \{(ij,jk): ij|k \in \tau\}$.

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 $h: \binom{X}{2} \to \mathbb{N}$ is called a height function on X.

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 $h_T(i,j) = l_T - d_T(c_{ij}, r)$ 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 without degree zero is nonempty. Assign $l_{G_\tau} + 1$ to the nodes with out degree zero and remove them from G_τ . Assign l_{G_τ} to the nodes with out degree 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.

VOL. 3, NO. 7, July 2012 ISSN 2079-8407
Journal of Emerging Trends in Computing and Information Sciences

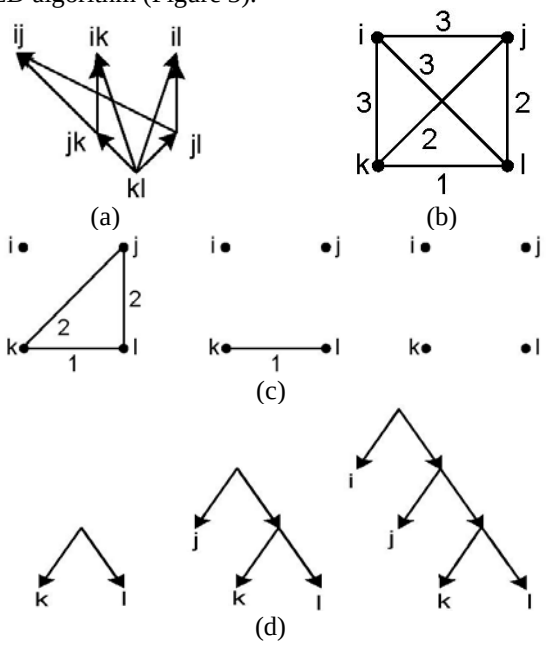
http://www.cisjournal.org

Let τ be a set of triplets. In [13] we 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. We refer to this algorithm by HBUILD [13].

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 results 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 (Figure 3).



giF **3:** The steps of constructing T_τ from the given set $\tau = \{kl|j, kl|i, jk|i, jl|i\}$. (a) The graph G_τ. (b) The graph (G, h). (c) Removing maximum weights from the graph (G, h). (d) Constructing T_τ using step c.

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

We denote $\mathbb{T}_\tau$, 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 $\mathbb{T}_\tau$ [13].

Now if there is no tree consistent with a set of triplets τ, then G_τ might not be a DAG. If G_τ is not a DAG, 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 [14]. Thus we use the heuristic algorithm GR [15], 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 three methods to disconnect C'.

a. Removing Maximum Edge Weights

In the first method, the process of removing edges with maximum weights from C' is continued until it becomes disconnected.

b. Min-Cut Algorithm

In the second method, Min-cut algorithm is applied and the edges are removed from C' in such a way that the sum of the weights of the removed edges are minimum and C' converted into two connected components [16].

c. Max-Cut Algorithm

In the third method, assume that in C' the weight of the edges with maximum weight is m. For an arbitrary edge in C' with weight w, its weight is updated to $m - w + 1$, and the Min-cut algorithm is performed on C' with these new edge weights.

For each of three methods continue its 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 each algorithm. The best tree which is obtained from the above three methods is reported as the output of our algorithm. We named this algorithm Tree Reconstruction with Height; Algorithm TRH for short.

4. RUNTIME

In this section we study the time complexity of TRH. The common part of the three methods in TRH is as what follows.

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 $O(|edges|)$

time, which is equivalent to $O(m)$. Now the nodes without degree zero are recognized and then Topological sort is performed on G_T'. 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 $O(n^2)$. It follows that the above steps run in time $O(m + n^2)$.

Now we are ready to perform the rest of the methods. For the method i, 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 $O(mn^2)$.

For the method ii, in each step, removing the edges with maximum weight is performed in $O(mn)$. Then, Min-cut is used in $O(mn + n^2 log(n))$. The overall run time is $O(mn + n^2 logn)$. Like the first method, the total runtime is $O(mn^2 + n^3 logn)$.

The runtime of the method iii, is exactly the same as the method ii.

So, the overall runtime of TRH is $O(mn^2 + n^3 logn)$.

5. RESULTS

In the following, we will present two simulations in order to show the performance of TRH.

First simulation

In the first simulation we randomly generated triplets with n=15, m=50, 100, 200, 300 like [6]

For each case we generated 2000 samples. Then we compared our results with the results of the best existing algorithm BPMR. The results are shown in Table 1.

Table 1: Simulation results for n=15 and m=50, 100, 200, 300 for both TRH and BPMR.

Number of triplets	50	100	200	300
The percent of trees in which TRH outperforms BPMR	20 %	17 %	12 %	7%
The percent of trees in which BPMR outperforms TRH	45 %	50 %	60 %	75 %
The average percent of triplets consistency for the TRH results	58 %	49 %	42 %	36 %
The average percent of triplets consistency for the BPMR results	60 %	50 %	43 %	40 %

The results show that in average BPMR outperforms TRH on randomly generated sets of triplets.

Second simulation

In the second simulation 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 un rooted binary trees based on the

Input sequences using Maximum Likelihood criterion. PhyML can be used to produce triplets. It is enough to add an out group 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. In this case we generated 2000 sets of sequences of size 15 under biological presumptions, and for each of them we obtained a set of triplets using described standard method. The results of TRH and BPMR are shown in Table 2.

elbaT**2:** Simulation results for n=15 and different thresholds for 2000 samples for both TRH and BPMR.

Threshold	0	0.2	0.4	0.6	0.8	0.9
The percent of trees in which TRH outperforms BPMR	25 %	29 %	35 %	45 %	47 %	50 %
The percent of trees in which BPMR outperforms TRH	60 %	55 %	53 %	48 %	42 %	41 %
The average percent of triplets consistency for the TRH results	85 %	89 %	90.5 %	93.5 %	94 %	96 %
The average percent of triplets consistency for the BPMR results	86 %	89 %	91 %	93 %	93 %	93.5 %

The results show that in average TRH outperforms BPMR on triplets which are obtained from (biological) sequences data.

6. DISCUSSION

In this paper we implemented TRH, a new heuristic algorithm for MRTC. To demonstrate the efficiency of TRH

we performed two simulations under two different scenarios. Then we compared our results with the best existing method BPMR [6].

In the first simulation we randomly generated different sets of triplets like [6].

The results show that in most cases BPMR outperforms TRH. Also the results show that in average the consistency of the BPMR trees with the input triplets is greater than TRH trees. It suggests that for randomly generated sets of triplets, in average BPMR outperform TRH.

In the second simulation we generated 2000 sets of sequences under biological presumptions. nehT by using standard methods, we converted each set of 15 sequences into a set of triplets. Like the first simulation we compared our results with BPMR results.

The results show that unlike the first simulation in average TRH outperform BPMR and the consistency of our resulting trees with the input triplets, is greater than BPMR trees. It suggests that for these kind of triplets data, our method outperforms.

Note that the overall runtime of TRH is $O(mn^2 + n^3 logn)$ and the BPMR runtime is $O(mn^3)$. Since in most cases m is greater than n, it follows that in most cases TRH outperforms BPMR in runtime. In more detail for large size data, when the number of taxa n increased, BPMR takes much time to give the output, but TRH constructs a tree at the appropriate time.

ACKNOWLEDGMENT

The author would like to thank Shahid Beheshti University for its supports.

REFERENCES

[1] J. Felsenstein, Introduction to algorithms, Sinauer Associates, Sunderland, MA, USA, 2004.

[2] L.V. Iersel and S. Kelk, ”Constructing the Simplest Possible Phylogenetic Network from Triplets,” Algorithmica, 2009.

[3] S. Kannan, E. Lawler and T. Warnow, ”Determining the evolutionary tree using experiments,” J. Algorithms, vol. 21, pp. 26-50, 1996.

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

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

[6] K. Maemura, J. Jansson, H. Ono, K. Sadakane, M. Yamashita, ”Approximation algorithms for constructing evolutionary trees from rooted triplets,” 10th Korea-Japan joint workshop on algorithms and computation, 2007.

[7] J. Jansson, ”On the complexity of inferring rooted evolutionary trees,” Electronic Notes in Discrete Mathematics vol. 7, pp. 50-53, 2001.

[8] B.Y. Wu, ”Constructing the maximum consensus tree from rooted triples,” Journal of Combinatorial Optimization, vol. 8, pp. 29-39, 2004.

[9] D. Bryant, ”Building trees, hunting for trees, and comparing trees, theory and methods in phylogenetic analysis,” Ph.D. thesis, University of Canterbury, 1997.

[10] J. Byrka, S. Guillemot, J. Jansson, ”New results on optimizing rooted triplets consistency,” In: S.H. Hong, H. Nagamochi, T. Fukunaga (eds.) Algorithms and Computation, Lecture Notes in Computer Science, vol. 5369, pp. 484-495. Springer Berlin / Heidelberg, 2008.

[11] L. Van Iersel, S. Kelk, M. Mnich, ”Uniqueness, intractability and exact algorithms: reflections on level-k phylogenetic networks,” Journal of Bioinformatics and Computational Biology (JBCB), vol. 7, pp. 597-623, 2009.

[12] L. Gasieniec, J. Jansson, A. Lingas, A. Ostlin, ”On the complexity of constructing evolutionary trees,” Journal of Combinatorial Optimization vol. 3(2-3), pp.183-197, 1999.

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

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

[15] P. Eades, X. Lin, W. F. Smyth, ”A fast and effective heuristic for the feedback arc set problem,” Information Processing Letters, vol. 47, pp. 319323, 1993.

[16] M. Stoer, F. Wagner, ”A simple min-cut algorithm,” Journal of the ACM (JACM), vol. 44(4), pp. 585 - 591, 1997.