Send Orders for Reprints to reprints@benthamscience.ae

2218 *The Open Cybernetics & Systemics Journal,* 2015, *9,* 2218-2223

Open Access

Algorithms for Computing Cluster Dissimilarity between Rooted Phylogenetic Trees

Li Shuguang[1,2,*] and Liu Zhihui[1,2]

[1]*Key Laboratory of Intelligent Information Processing in Universities of Shandong (Shandong Institute of Business and Technology), Yantai, 264005, China;* [2]*College of Computer Science and Technology, Shandong Institute of Business and Technology, Yantai, 264005, China*

Abstract: Phylogenetic trees represent the historical evolutionary relationships between different species or organisms. Creating and maintaining a repository of phylogenetic trees is one of the major objectives of molecular evolution studies. One way of mining phylogenetic information databases would be to compare the trees by using a tree comparison measure. Presented here are a new dissimilarity measure for comparing rooted trees and three algorithms to efficiently compute it. This new measure operates on clusters of compared trees as in the case of standard Robinson-Foulds distance, but extracts more subtle differences between clusters, and thus may offer better discrimination than the Robinson-Foulds distance.

Keywords: Cluster dissimilarity, combinatorial algorithms, phylogenetic trees, robinson-foulds distance, tree comparison.

1. INTRODUCTION

Phylogenetic trees are widely used in biology to represent evolutionary relationships of a collection of species. Typically, the extant species correspond to the leaves each assigned unique labels and the remaining vertices (the interior vertices) represent ancestral species. One interior vertex may be distinguished from the others as the root so that the tree becomes rooted, and among interior vertices only the root may have degree two. Each vertex of a rooted tree associates with a cluster, *i.e.*, the subset of leaf labels in the subtree rooted at this vertex. The set of all such clusters is called the cluster representation of the tree. A tree is called a binary tree if all its vertices have degree at most three [1].

With a large number of completely sequenced genomes and many more in progress, there is a large publicly available dataset that can be used to generate phylogenetic trees. Building phylogenetic trees is one of the primary objectives of phylogenetics [2]. This is typically done on the basis of molecular information (*e.g.*, DNA sequences) from these species, and there are many methods used for it: parsimony [3], maximum likelihood [4, 5], distance matrices [6-8], Bayesian approaches [9, 10], *etc*. The problem is *NP*-hard under most models [3, 5, 11, 12].

Since applying different reconstruction methods often results in different trees for the same input data, many phylogenetic trees included in the databases are actually alternative trees for the same sets of species. This variety makes it necessary to compare the trees for measuring their differences [3]. Moreover, even if a database is made available its usefulness will be measured by how it can be queried. Tree comparison is very useful in querying databases of phylogenetic information [13]. Tree comparison is also used for other purposes, *e.g.*, to assess the stability of the reconstruction algorithms [14], and in the comparative analysis of other hierarchical cluster structures [15, 16].

Tree comparison concerns three related problems: to construct a consensus tree for a given set of trees [17], to compute a consensus index for a given set of trees [18, 19], and to measure pairwise dissimilarity between trees. The third problem may form the basis of consensus tree or index methods [20]. A dissimilarity measure is used to determine how far the two compared trees are. The larger the value, the more different the two trees are considered to be.

There have been a number of dissimilarity measures proposed for comparing phylogenetic trees in the literature. Some measures are edit distances, such as the nearest-neighbor interchange distance [21] and the subtree prune-and-regrafting distance [22]. However, computing such edit distances is typically NP-hard [22-24]. Some measures are based on the comparison of phylogenetic trees through some consensus subtree, *e.g.*, the MAST (Maximum Agreement Subtree) distances defined in [25, 26]. Finally, many dissimilarity measures compare the encodings of the phylogenetic trees, such as the Robinson-Foulds distance [27], the quartet distance [8], the triples distance [28], the splitted nodal distances [29], the cophenetic metrics [30, 31], to name just a few.

The Robinson-Foulds distance is the most widespread measure for the comparative analysis of phylogenetic trees. For two rooted trees, it is defined as the normalized count of the symmetric difference of their cluster representations. It can be computed in linear time by using Day's algorithm [20]. For a set of phylogenetic trees, the distance matrix can be computed in sublinear time [32]. The main disadvantages of this measure come from its poor distribution and sensitiv-

1874-110X/15

ity. The Robinson-Foulds distance between two random binary trees has a highly skewed distribution, in which most values equal the largest possible value. It also lacks robustness in the face of small changes in the original tree: one leaf relocation may generate a tree at largest possible Robinson-Foulds distance [33, 34].

In this paper, we introduce a new pairwise dissimilarity measure for comparing rooted phylogenetic trees. Similarly to the Robinson-Foulds distance, this measure is based on comparing clusters. However, it considers not only the identity of clusters, but also more subtle differences between clusters, and transforms the values of dissimilarity between clusters to the final score of the dissimilarity of compared trees. The proposed measure can be regarded as a weighted extension of the Robinson-Foulds distance, and may offer better discrimination than it. We propose three algorithms to compute this new measure whose running times are $O(n^3)$, $O(n^2)$ and $O(n^2 \log n)$, respectively, where n is the number of leaves in a tree. The running time of the third algorithm becomes $O(n \log^2 n)$ if the compared trees are balanced, *i.e.*, trees of the height at most $O(\log n)$.

The remainder of this paper is organized as follows. In Section 2, we review terminology needed, and then define a new dissimilarity measure. In Section 3, we present three algorithms to compute the new measure efficiently. We conclude this paper in Section 4.

2. PRELIMINARIES

The notations and definitions used here mainly follow Semple and Steel [1]. For sets A, B, let $A \oplus B = (A \setminus B) \cup (B \setminus A)$ be their *symmetric difference*. Denote by $|A|$ the *cardinality* of set A. A *graph* $G = (V, E)$ is a structure consisting of a set V of vertices and a set $E \subseteq \{\{x, y\} : x, y \in V\}$ of edges. A sequence of contiguous edges in G is called a *path*. A *cycle* is obtained from a path by identifying its two ends without any retracing. G is *connected* if, for any pair of vertices $u, v \in V(G)$, there is a path in G from u to v.

An *unrooted tree* $T = (V, E)$ is a connected graph with no cycles. An *unrooted phylogenetic tree* (*on* L) is an unrooted tree whose leaves are labeled bijectively by a set L (species) and no vertex has degree 2. A *rooted tree* contains a distinguished vertex, called the root, from which every vertex can be reached through exactly one path. A *rooted phylogenetic tree* (*on* L) is a rooted tree whose leaves are labeled bijectively by a set L and no vertex but the root may have degree 2.

Since we consider rooted phylogenetic trees in this paper, we neglect the unrooted case. Hence, we often use the term "tree" instead of "rooted phylogenetic tree" in the following for brevity. We identify leaves with their labels. That is, for tree T, let $L(T)$ denote the set of leaves of T or the set of labels of those leaves. See Fig. (**1**) for an example.

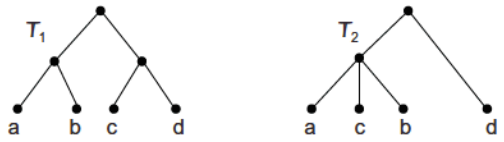

Fig. (1). An example. T_1 and T_2 are two rooted phylogenetic trees with $L(T_1) = L(T_2) = \{a, b, c, d\}$. T_1 is a binary tree, while T_2 is not.

A rooted tree defines naturally a partial order relation $\leq_T$ on its vertices. For two vertices $u, v \in V(T)$, we have $u \leq_T v$ if the path in T from u to the root of T contains v. We shall say that u is a *descendant* of v and also that v is an *ancestor* of u. In particular, $v \leq_T v$ for any $v \in V(T)$.

Given a vertex v of T, the *subtree of* T *rooted at* v is a subgraph of T, $T' = (V', E')$, such that V' is the set of descendants of v (including v itself) in T and E' consists of those edges of T with both ends in V'.

Each vertex u of T associates with a *cluster* denoted by $L(u)$, *i.e.*, the set of leaf labels in the subtree rooted at u. The set of all such clusters for T is called the *cluster representation* of T and is denoted by $\sigma(T)$, from which T can be reconstructed in linear time [1]. Among the clusters in $\sigma(T)$, those associated with the leaves or the root are called *trivial clusters* since they can be found in every tree, and the remaining clusters are called *non-trivial clusters*. Denote by $\sigma_*(T)$ the set of all non-trivial clusters of T. For the rooted trees in Fig. (**1**), $\sigma_*(T_1) = \{\{a, b\}, \{c, d\}\}$ and $\sigma_*(T_2) = \{\{a, b, c\}\}$.

Cluster representations play an important role in designing dissimilarity measures between phylogenetic trees. For example, the Robinson-Foulds distance is defined to be the number of different clusters in compared trees (divided by 2). For the rooted trees in Fig. (**1**) the Robinson-Foulds distance is 1.5.

We now introduce the new dissimilarity measure. Without loss of generality, we assume that $L = \{1, 2, \ldots, n\}$ from now on.

Each cluster of a tree T associates with a binary vector α of length n: For any leaf i, set $\alpha[i] = 1$ if i is in this cluster, otherwise set $\alpha[i] = 0$. Denote by $BV(T)$ and $BV_*(T)$ the sets of binary vectors associated with the clusters and non-trivial clusters of T, respectively. For a binary vector $\alpha \in BV(T)$, let C_α be the cluster associated with it.

Given two trees T_1 and T_2 with $L(T_1)=L(T_2)=L$. The *cluster dissimilarity* between T_1 and T_2 is defined as follows:

$$Cd(T_1,T_2)=(\sum_{\alpha\in BV_*(T_1)} \underset{\beta\in BV(T_2)}{Min} d_H(\alpha,\beta)+\sum_{\beta\in BV_*(T_2)} \underset{\alpha\in BV(T_1)}{Min} d_H(\beta,\alpha))/2 \quad (1)$$

where $d_H(\alpha,\beta)$ is the classical Hamming distance between the two vectors α and β. It is easy to see that $d_H(\alpha,\beta)=|C_\alpha \oplus C_\beta|$.

Each binary vector associated with a cluster of a tree has a most similar binary vector associated with a (not necessarily non-trivial) cluster of another tree. The most similar binary vector has the smallest Hamming distance from the binary vector. Compute the sum of the Hamming distances between each binary vector and its most similar binary vector. The cluster dissimilarity between two trees is equal to this sum divided by 2. Since each binary vector associated with a trivial cluster of a tree has the same binary vector in another tree, we need not count the Hamming distance between them in (1).

For the trees T_1 and T_2 shown in Fig. (**1**), the cluster dissimilarity between T_1 and T_2, $Cd(T_1,T_2)$, is equal to $(1+1+1)/2=1.5$.

3. THREE ALGORITHMS

In this section we will present three algorithms to compute $Cd(T_1,T_2)$ efficiently. These algorithms are all based on tree post-order traversal, but different in the methods to compute $\underset{\beta\in BV(T_2)}{Min} d_H(\alpha,\beta)$ and $\underset{\alpha\in BV(T_1)}{Min} d_H(\beta,\alpha)$ before using (1), and thus have different time complexity.

Algorithm 1:

Step 1. Traverse T_1 and T_2 in post-order respectively, and get all of the binary vectors associated with the clusters in T_1 and T_2.

Step 2. For each $\alpha\in BV_*(T_1)$, compute the Hamming distance between α and each $\beta\in BV(T_2)$. Choose the smallest one from among the obtained values as $\underset{\beta\in BV(T_2)}{Min} d_H(\alpha,\beta)$.

Step 3. For each $\beta\in BV_*(T_2)$, compute the Hamming distance between β and each $\alpha\in BV(T_1)$. Choose the smallest one from among the obtained values as $\underset{\alpha\in BV(T_1)}{Min} d_H(\beta,\alpha)$.

Step 4. Compute $Cd(T_1,T_2)$ by (1).

We then get the following theorem.

Theorem 1. Algorithm 1 computes the cluster dissimilarity between two trees T_1 and T_2 in $O(n^3)$ time, where n is the number of leaves in T_1 and T_2.

Proof. Steps 1 and 4 can be executed in $O(n)$ time. Computing the Hamming distance between two clusters requires $O(n)$ time. Since there are at most $2n-1$ clusters in a tree, each $\underset{\beta\in BV(T_2)}{Min} d_H(\alpha,\beta)$ and $\underset{\alpha\in BV(T_1)}{Min} d_H(\beta,\alpha)$ can be obtained in $O(n^2)$ time. Since there are at most $n-2$ non-trivial clusters in a tree, Steps 2 and 3 can be executed in $O(n^3)$ time. Hence the running time of Algorithm 1 is $O(n^3)$.

We now modify Algorithm 1 to reduce the time complexity to $O(n^2)$. We only need to show that in fact each $\underset{\beta\in BV(T_2)}{Min} d_H(\alpha,\beta)$ and $\underset{\alpha\in BV(T_1)}{Min} d_H(\beta,\alpha)$ can be computed in $O(n)$ time.

Fix a non-trivial cluster $C_\alpha\in\sigma_*(T_1)$. Denote by n_1 the number of labels in C_α. Traverse T_2 in post-order. Suppose that the current cluster is $C_\beta\in\sigma(T_2)$ during the traversal. Let l_1 be the number of labels that are in both C_α and C_β, and l_0 be the number of labels that are in C_β but not in C_α. Then we get $d_H(\alpha,\beta)=n_1-l_1+l_0$. Compute the Hamming distance between α and each $\beta\in BV(T_2)$ in this way. Choose the smallest one from among the obtained values as $\underset{\beta\in BV(T_2)}{Min} d_H(\alpha,\beta)$.

For the clusters associated with the leaves of T_2, the values of l_1 and l_0 can be obtained in $O(1)$ time. For any cluster associated with an interior vertex of T_2, the values of l_1 and l_0 can be computed respectively by adding the values of l_1 and l_0 of all the children. Hence for each $\alpha\in BV_*(T_1)$, $\underset{\beta\in BV(T_2)}{Min} d_H(\alpha,\beta)$ can be computed in $O(n)$ time.

Similarly, for each $\beta\in BV_*(T_2)$, $\underset{\alpha\in BV(T_1)}{Min} d_H(\beta,\alpha)$ can be computed in $O(n)$ time.

Algorithm 2:

Step 1. Traverse T_1 in post-order, and get all of the clusters in T_1. For each $\alpha \in BV_*(T_1)$, compute $\underset{\beta \in BV(T_2)}{Min} d_H(\alpha, \beta)$ in $O(n)$ time using the above method.

Step 2. Traverse T_2 in post-order, and get all of the clusters in T_2. For each $\beta \in BV_*(T_2)$, compute $\underset{\alpha \in BV(T_1)}{Min} d_H(\beta, \alpha)$ in $O(n)$ time using the above method.

Step 3. Compute $Cd(T_1, T_2)$ by (1).

We then get the following theorem.

Theorem 2. Algorithm 2 computes the cluster dissimilarity between two trees T_1 and T_2 in $O(n^2)$ time, where n is the number of leaves in T_1 and T_2.

In order to compute $\underset{\beta \in BV(T_2)}{Min} d_H(\alpha, \beta)$ for $\alpha \in BV_*(T_1)$, we have to compare α with each element in $BV(T_2)$ in the two algorithms presented above. Actually, a technique used in [35] allows us to compare α with just a proper subset of $BV(T_2)$. This is the basic idea of Algorithm 3 presented below.

Before we describe the algorithm, some preliminary definitions and initial lemmas are necessary.

The *Least Common Ancestor* (LCA) of vertices u and v in a tree T, denoted by $\mathrm{LCA}(u, v)$, is the shared ancestor of u and v that is located farthest from the root, *i.e.*, $\mathrm{LCA}(u, v) = \min\{w \mid u \leq_T w, v \leq_T w\}$.

Lemma 1. [36, 37] Given a tree T. There are algorithms that can answer LCA query in T in constant time after only linear time preprocessing of the tree.

Given $X \subseteq L(T)$. The *restriction* of T to X, denoted by $T \mid X$, is constructed by first finding the minimal subtree of T containing X, and then suppressing all vertices of degree two except the root.

Lemma 2. [35] A tree T can be preprocessed in linear time so that for any $X \subseteq L(T)$, $T \mid X$ can be computed in $O(|X| \log n)$ time, where n is the number of leaves in T.

In the proof of this lemma in [35], the authors first sort the leaves of T according to the in-order traversal, and then find the leaves with labels in X. Suppose that these leaves are $s_1, s_2, \ldots, s_m$ in the order. $T \mid X$ is constructed as follows: $\mathrm{LCA}(s_1, s_m)$ is the root, $s_2, \ldots, s_{m-1}$ is inserted one by one such that each interior vertex of $T \mid X$ is actually an interior vertex of T representing an LCA.

For a set S of vertices in T, if there is a vertex $u \in S$ such that $v \leq_T u$ for any $v \in S$, then we let $\max S$ be u, otherwise $\max S$ is undefined.

For any vertex $u \in V(T)$, let $D(u, X)$ be the set of all the vertices in $T \mid X$ that are under u in T, *i.e.*, $D(u, X) = \{v \mid v \leq_T u \text{ and } v \in T \mid X\}$.

Lemma 3. [35] If $D(u, X) \neq \phi$, then $\max D(u, X)$ is well defined.

For $\alpha \in BV_*(T_1)$, denote by $M_\alpha \in \sigma(T_2)$ the cluster associated with $\arg \underset{\beta \in BV(T_2)}{Min} d_H(\alpha, \beta)$. Similarly, For $\beta \in BV_*(T_2)$, denote by $M_\beta \in \sigma(T_1)$ the cluster associated with $\arg \underset{\alpha \in BV(T_1)}{Min} d_H(\beta, \alpha)$.

Lemma 4. Given two trees T_1 and T_2. For each $\alpha \in BV_*(T_1)$, M_α must be associated with a vertex in $T_2 \mid C_\alpha$. Similarly, for each $\beta \in BV_*(T_2)$, M_β must be associated with a vertex in $T_1 \mid C_\beta$.

Proof. Consider $\alpha \in BV_*(T_1)$. Suppose that $\alpha[i] = 1$ for some $i \in \{1, 2, \ldots, n\}$. Let e_α be the binary vector in which $e_\alpha[i] = 1$ and $e_\alpha[j] = 0$ for any $j \neq i$. We have $d_H(\alpha, e_\alpha) = |C_\alpha| - 1$. It follows that $\underset{\beta \in BV(T_2)}{Min} d_H(\alpha, \beta) \leq |C_\alpha| - 1$.

Now consider a vertex v such that $v \in V(T_2)$ but $v \notin V(T_2 \mid C_\alpha)$. Let β_v be the binary vector associated with the cluster $L(v)$. We distinguish between two different cases:

If $D(v, C_\alpha) = \phi$, then M_α cannot be associated with v because otherwise we get $\underset{\beta \in BV(T_2)}{Min} d_H(\alpha, \beta) > |C_\alpha|$.

If $D(v, C_\alpha) \neq \phi$, then set $w = \max D(v, C_\alpha)$. Lemma 3 ensures that w is well defined. Let β_w be the binary vector associated with the cluster $L(w)$. Since $L(w) \subset L(v)$ and $L(w) \cap C_\alpha = L(v) \cap C_\alpha$, we get $d_H(\alpha, \beta_v) > d_H(\alpha, \beta_w)$. Therefore, M_α cannot be associated with v.

The second half of the lemma can be proved similarly.

We are now ready to describe the third algorithm.

Algorithm 3:

Step 1. Traverse T_1 and T_2 in post-order respectively, and store at each vertex the number of leaves under it.

Step 2. For each $\alpha \in BV_*(T_1)$, compute $\underset{\beta \in BV(T_2)}{Min}\, d_H(\alpha,\beta)$ as follows. First construct $T_2 \mid C_\alpha$, and then traverse it in post-order. During the traversal compute for each vertex $v \in V(T_2 \mid C_\alpha)$ the number of leaves under v in $T_2 \mid C_\alpha$, *i.e.*, $\left|C_\alpha \cap L(v)\right|$, where $L(v)$ is the set of leaf labels in the subtree of T_2 rooted at v. (Every vertex of $T_2 \mid C_\alpha$ must be a vertex of T_2.) Let β_v be the binary vector associated with the cluster $L(v)$. We can now compute $d_H(\alpha,\beta_v) = \left|C_\alpha \oplus L(v)\right| = \left|C_\alpha\right| + \left|L(v)\right| - 2\left|C_\alpha \cap L(v)\right|$. For each $v \in V(T_2 \mid C_\alpha)$ compute $d_H(\alpha,\beta_v)$ as described above and then choose the smallest one from among the obtained values as $\underset{\beta \in BV(T_2)}{Min}\, d_H(\alpha,\beta)$.

Step 3. For each $\beta \in BV_*(T_2)$, compute $\underset{\alpha \in BV(T_1)}{Min}\, d_H(\beta,\alpha)$ as follows. First construct $T_1 \mid C_\beta$, and then traverse it in post-order. During the traversal compute for each vertex $u \in V(T_1 \mid C_\beta)$ the number of leaves under u in $T_1 \mid C_\beta$, *i.e.*, $\left|C_\beta \cap L(u)\right|$, where $L(u)$ is the set of leaf labels in the subtree of T_1 rooted at u. (Every vertex of $T_1 \mid C_\beta$ must be a vertex of T_1.) Let α_u be the binary vector associated with the cluster $L(u)$. We can now compute $d_H(\beta,\alpha_u) = \left|C_\beta \oplus L(u)\right| = \left|C_\beta\right| + \left|L(u)\right| - 2\left|C_\beta \cap L(u)\right|$. For each $u \in V(T_1 \mid C_\beta)$ compute $d_H(\beta,\alpha_u)$ as described above and then choose the smallest one from among the obtained values as $\underset{\alpha \in BV(T_1)}{Min}\, d_H(\beta,\alpha)$.

Step 4. Compute $Cd(T_1,T_2)$ by (1).

We then get the following theorem.

Theorem 3. Algorithm 3 computes the cluster dissimilarity between two trees T_1 and T_2 in $O(\sum_{u \in V(T_1)} \left|L(u)\right| \log n + \sum_{v \in V(T_2)} \left|L(v)\right| \log n)$ time, where n is the number of leaves in T_1 and T_2. The time complexity is $O(n^2 \log n)$ in the worst case and $O(n \log^2 n)$ when T_1 and T_2 are balanced.

Proof. Lemma 4 ensures the correctness of Algorithm 3. Steps 1 and 4 can be executed in $O(n)$ time. By Lemma 2, Steps 2 and 3 can be executed in $O(\sum_{u \in V(T_1)} \left|L(u)\right| \log n + \sum_{v \in V(T_2)} \left|L(v)\right| \log n)$ time.

Each $\left|L(u)\right|$ and $\left|L(v)\right|$ is at most n and there are at most $2n-1$ vertices in each tree, hence the time complexity of Algorithm 3 in the worst case is $O(n^2 \log n)$. Observe that the total cardinality of the clusters at the same depth of a tree is at most n. Hence if T_1 and T_2 have height at most $O(\log n)$, the time complexity of Algorithm 3 reduces to $O(n \log^2 n)$.

CONCLUSION

We introduced a new dissimilarity measure for comparing rooted phylogenetic trees. We showed that this measure can be computed in quadratic time in the worst case, and in roughly linear time for balanced trees. It would be interesting to investigate whether this measure can be computed in subquadratic time in the worst case.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China (Nos. 61173173, 61272016, 61272430, 61373079 and 61432010), Key project of Chinese Ministry of Education (No. 212101), Shandong Provincial Natural Science Foundation of China (Nos. ZR2013FM015 and ZR2011FL004), A Project of Shandong Province Higher Educational Science and Technology Program (No. SX12J4).

REFERENCES

[1] C. Semple, and M. A. Steel, *Phylogenetics*. Oxford University Press, 2003.

[2] M. Salemi, A.-M. Vandamme, and P. Lemey, *The phylogenetic handbook: a practical approach to phylogenetic analysis and hypothesis testing*. Cambridge University Press, 2009.

[3] W. H. Day, D. S. Johnson, and D. Sankoff, "The computational complexity of inferring rooted phylogenies by parsimony", *Mathematical Biosciences*, vol. 81, pp. 33-42, 1986.

[4] S. Guindon, and O. Gascuel, "A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood", *Systematic Biology*, vol. 52, pp. 696-704, 2003.

[5] S. Roch, "A short proof that phylogenetic tree reconstruction by maximum likelihood is hard", *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 3, pp. 92, 2006.

[6] R. R. Sokal, "A statistical method for evaluating systematic relationships", *University of Kansas Science Bulletin*, vol. 38, pp. 1409-1438, 1958.

[7] N. Saitou, and M. Nei, "The neighbor-joining method: a new method for reconstructing phylogenetic trees", *Molecular Biology and Evolution*, vol. 4, pp. 406-425, 1987.

[8] W. M. Fitch, and E. Margoliash, "Construction of phylogenetic trees", *Science*, vol. 155, pp. 279-284, 1967.

[9] J. P. Huelsenbeck, and F. Ronquist, "MRBAYES: Bayesian inference of phylogenetic trees," *Bioinformatics*, vol. 17, pp. 754-755, 2001.

[10] A. J. Drummond, and A. Rambaut, "BEAST: Bayesian evolutionary analysis by sampling trees", *BMC Evolutionary Biology*, vol. 7, pp. 214, 2007.

[11] L. R. Foulds, and R. L. Graham, "The Steiner problem in phylogeny is NP-complete", *Advances in Applied Mathematics*, vol. 3, pp. 43-49, 1982.

[12] B. Y. Wu, K. -M. Chao, and C. Y. Tang, "Approximation and exact algorithms for constructing minimum ultrametric trees from distance matrices", *Journal of Combinatorial Optimization*, vol. 3, pp. 199-211, 1999.

[13] J. T. Wang, H. Shan, D. Shasha, and W. H. Piel, "Fast structural search in phylogenetic databases", *Evolutionary Bioinformatics*, vol. 1, pp. 37-46, 2005.

[14] W. Williams, and H. Clifford, "On the comparison of two classifications of the same set of elements", *Taxon*, vol. 20, no. 4, pp. 519-522, 1971.

[15] J. Handl, J. Knowles, and D. B. Kell, "Computational cluster validation in post-genomic data analysis", *Bioinformatics*, vol. 21, pp. 3201-3212, 2005.

[16] G. Restrepo, H. Mesa, and E. J. Llanos, "Three dissimilarity measures to contrast dendrograms", *Journal of Chemical Information and Modeling*, vol. 47, pp. 761-770, 2007.

[17] D. Bryant, "A classification of consensus methods for phylogenetics", *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 61, pp. 163-184, 2003.

[18] D. L. Swofford, "When are phylogeny estimates from molecular and morphological data incongruent", In: M. M. Miamoto, J. Cracraft, eds, Phylogenetic analysis of DNA sequences, pp. 295-333, 1991.

[19] D. Bryant, "Building trees, hunting for trees, and comparing trees: theory and methods in phylogenetic analysis," 1997.

[20] W. H. Day, "Optimal algorithms for comparing trees with labeled leaves", *Journal of Classification*, vol. 2, pp. 7-28, 1985.

[21] M. S. Waterman, and T. F. Smith, "On the similarity of dendrograms", *Journal of Theoretical Biology*, vol. 73, pp. 789-800, 1978.

[22] B. L. Allen, and M. Steel, "Subtree transfer operations and their induced metrics on evolutionary trees", *Annals of Combinatorics*, vol. 5, pp. 1-15, 2001.

[23] B. D. Gupta, X. He, T. Jiang, M. Li, J. Tromp, and L. Zhang, "On distances between phylogenetic trees", In: *Proceedings of the 8th annual ACM-SIAM symposium on Discrete algorithms*, Philadelphia, PA, USA, 1997, pp. 427-436.

[24] G. Hickey, F. Dehne, A. Rau-Chaplin, and C. Blouin, "SPR distance computation for unrooted trees", *Evolutionary Bioinformatics Online*, vol. 4, pp. 17, 2008.

[25] C. Finden, and A. Gordon, "Obtaining common pruned trees", *Journal of Classification*, vol. 2, pp. 255-276, 1985.

[26] Y. Zhong, C. A. Meacham, and S. Pramanik, "A general method for tree-comparison based on subtree similarity and its use in a taxonomic database", *Biosystems*, vol. 42, pp. 1-8, 1997.

[27] D. Robinson, and L. R. Foulds, "Comparison of phylogenetic trees", *Mathematical Biosciences*, vol. 53, pp. 131-147, 1981.

[28] D. E. Critchlow, D. K. Pearl, and C. Qian, "The triples distance for rooted bifurcating phylogenetic trees", *Systematic Biology*, vol. 45, pp. 323-334, 1996.

[29] G. Cardona, M. Llabrés, F. Rosselló, and G. Valiente, "Nodal distances for rooted phylogenetic trees", *Journal of Mathematical Biology*, vol. 61, pp. 253-276, 2010.

[30] G. Cardona, A. Mir, F. Rosselló, L. Rotger, and D. Sánchez, "Cophenetic metrics for phylogenetic trees, after Sokal and Rohlf", *BMC Bioinformatics*, vol. 14:3, 2013.

[31] G. Cardona, A. Mir, and F. Rossello, "The expected value of the squared euclidean cophenetic metric under the Yule and the uniform models", arXiv preprint arXiv:1301.5131, 2013.

[32] N. D. Pattengale, E. J. Gottlieb, and B. M. Moret, "Efficiently computing the Robinson-Foulds metric", *Journal of Computational Biology*, vol. 14, pp. 724-735, 2007.

[33] M. A. Steel, and D. Penny, "Distributions of tree comparison metrics - some new results", *Systematic Biology*, vol. 42, pp. 126-141, 1993.

[34] D. Bryant, and M. Steel, "Computing the distribution of a tree metric", *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 6, pp. 420-426, 2009.

[35] L. Zhang, "On matching nodes between trees", Technical Report, 2003-67, HP Labs2003.

[36] D. Harel, and R. E. Tarjan, "Fast algorithms for finding nearest common ancestors", *SIAM Journal on Computing*, vol. 13, pp. 338-355, 1984.

[37] M. A. Bender, and M. Farach-Colton, "The LCA problem revisited", In: *LATIN 2000: Theoretical Informatics*, Springer, 2000, pp. 88-94.

Received: June 10, 2015 Revised: July 29, 2015 Accepted: August 15, 2015