TreeRank: A Similarity Measure for Nearest Neighbor Searching in Phylogenetic Databases*

Jason T. L. Wang† Huiyuan Shan‡ Dennis Shasha§ William H. Piel¶

Abstract

Phylogenetic trees are unordered labeled trees in which each leaf node has a label and the order among siblings is unimportant. In this paper we propose a new similarity measure, called TreeRank*, for phylogenetic trees and present an algorithm for computing TreeRank scores. Given a query or pattern tree P and a data tree D, the TreeRank score from P to D is a measure of the topological relationships in P that are found to be the same or similar in D. The proposed algorithm calculates the TreeRank score in $O(M^2 + N)$ time where M is the number of nodes appearing in both P and D, and N is the number of nodes in D. We then develop a search engine that, given a query or pattern tree P and a database of trees $\mathcal{D}$, finds and ranks the nearest neighbors of P in $\mathcal{D}$ where the "nearness" is measured by the proposed similarity function. This structure-based search engine is fully operational and is available on the World Wide Web.*

1 Introduction

Scientists model phylogenetic relations using unordered labeled trees and develop methods for constructing these trees [2, 7, 19, 20].[1] Different theories concerning the phylogenetic relationship of the same set of species often result in different phylogenetic trees. Even the same phylogenetic theory may yield different trees for different orthologous genes. With the unprecedented number of phylogenetic trees constructed based on these various theories, the need to analyze the trees and manage phylogenetic databases is urgent and great [27]. One important problem in this domain is to be able to compare the trees, thus possibly determining how much two theories have in common [5, 11, 16, 21]. The common portion of two trees may represent the actual phylogenetic relationship of the corresponding species.

Our motivation for studying the tree matching problem comes from the development of tools for analyzing the phylogenetic data. One particular tool we are developing is a system for searching phylogenetic trees. Given a query or pattern tree P and a set of data trees $\mathcal{D}$, this search engine is able to find and rank nearest neighbors of P in $\mathcal{D}$. The importance of nearest neighbor searching for trees, particularly in phyloinformatics, has been addressed in the literature [32, 33, 34, 37]. Central to our search engine is an algorithm for computing the similarity score from P to each data tree D in $\mathcal{D}$.

Our data consists of the phylogenetic trees stored within the widely used phylogenetic information system TreeBASE [30, 31], accessible at `http://www.treebase.org`.[2] Existing algorithms are mainly concerned with constructing the phylogenetic trees and finding the consensus of two trees, as opposed to information retrieval and ranking. By taking advantage of the properties of the phylogenetic trees and by utilizing the additive distance matrix [6] widely adopted in phylogenetic analysis, we propose a new similarity

*This work was supported in part by NSF grants IIS-9988345 and IIS-9988636.

†To whom correspondence should be addressed: College of Computing Sciences, New Jersey Institute of Technology, University Heights, Newark, NJ 07102, USA. Tel: (973) 596-3396, Fax: (973) 596-5777, Email: wangj@njit.edu.

‡Department of Computer Science, New Jersey Institute of Technology, University Heights, Newark, NJ 07102, USA.

§Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012, USA.

¶Department of Biological Sciences, 608 Cooke Hall, University at Buffalo, Buffalo, NY 14260, USA.

[1] These trees could be rooted ones, or could be unrooted ones, i.e. free trees. In this paper we focus on rooted, unordered phylogenetic trees.

[2] TreeBASE is a joint research effort developed at Harvard, UC Davis, Leiden University and the University at Buffalo.

measure, called *TreeRank*, for comparing the trees and for retrieving and ranking these trees while performing nearest neighbor searches in the phylogenetic database.

The rest of the paper is organized as follows. Section 2 discusses some properties of phylogenetic trees we observed in TreeBASE. Section 3 shows how to calculate TreeRank scores for the phylogenetic trees. Section 4 presents the algorithm for nearest neighbor searching. Section 5 reports implementation efforts. Section 6 discusses related work. Section 7 concludes the paper and points out some future work.

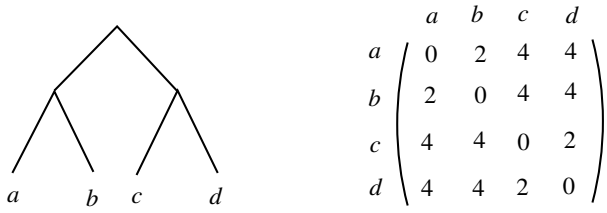

Figure 1. An additive distance tree and its distance matrix.

2 Phylogenetic Trees

In general, phylogenetic trees are structures used in biology to model the evolution of various life forms and thereby the relationship of a particular life form with other life forms. The currently existing life forms or organisms usually appear as leaf nodes in these trees.[3] Each internal node of one such tree represents an inferred ancestor organism of the organisms represented by its child nodes. There can be multiple levels of ancestors, with multiple organisms sharing the same ancestors.

For the phylogenetic trees in TreeBASE (and for those generated by any of the modern programs), the following properties hold:

- each leaf node has a label and that label appears only once in the tree, though it may appear in other trees;
- each non-leaf node either has a label that appears nowhere else in the tree or has no label;
- each unlabeled internal node has at least two children. An unlabeled internal node stands for an extinct species from which new species branched out.

We propose to adapt additive distance [2, 6, 15] to compare phylogenetic trees. Additive distance is widely used in phylogeny analysis. Given a two-dimensional additive distance matrix M in which each entry $M[u, v]$ is an integer, a free tree T is called an *additive distance tree* with respect to M if, for every pair of labeled nodes (u, v) in T, the path connecting node u and node v has exactly $M[u, v]$ edges.[4] Figure 1 shows an additive distance matrix and its corresponding additive distance tree. In the figure, for example, $M[a, d] = M[d, a] = 4$ meaning that there are four edges in the path connecting nodes a and d.[5] In general, each edge in an additive distance tree may be associated with a weight. In that case, $M[u, v]$ equals the sum of weights of the edges in the path connecting u and v. Given an additive distance matrix, many programs available on the Web can be used to reconstruct its phylogenetic tree [35].

In the following section we modify the additive distance matrix, and propose an *UpDown matrix* and a new distance measure, called *UpDown distance*, for comparing two rooted unordered phylogenetic trees.

3 UpDown Distance

3.1 Up and Down Operations

As stated above, we focus here on rooted unordered phylogenetic trees satisfying the three properties described in Section 2, and refer to these trees simply as trees when the context is clear. We consider two types of operations, up and down, between any two nodes in a tree. These operations are intended to capture the hierarchical structure in the tree (reminiscent of the "up" and "down" operations used to define the partial order on pairs of nodes in [22]). If v is a child node of u, we use $v \uparrow u$ to represent an *up* operation from v to u, and use $u \downarrow v$ to represent a *down* operation from u to v. Then, for any pair of nodes m, n in the tree T, one can count the number of up and down operations to move, say a token, from m to n.

For example, consider the tree in Figure 2 and the two nodes "fox" and "rabbit" in the tree. It takes two up operations ("fox $\uparrow$ carnivore" and "carnivore $\uparrow$ mammal")

[3] More precisely, an organism (species or taxon) name appears as a label of a leaf node in a tree.

[4] A free tree is an unrooted unordered tree, which is also known as an undirected acyclic graph [36, 40].

[5] In the paper, we often refer to a node by the label of that node and vice versa when the context is clear.

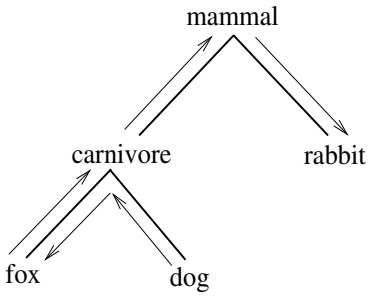

Figure 2. Illustration of up and down operations between two nodes in a tree.

and one down operation ("mammal $\downarrow$ rabbit") to go from "fox" to "rabbit" in the tree. As another example, it takes one up operation ("dog $\uparrow$ carnivore") and one down operation ("carnivore $\downarrow$ fox") to go from "dog" to "fox" in the tree.

3.2 UpDown Matrix

Given a tree T, we can now build two matrices, referred to as the *Up matrix* U and the *Down matrix* D, of integer values where $U[u, v]$ represents the number of up operations from node u to node v in T and $D[u, v]$ represents the number of down operations from u to v in the shortest path. Obviously $U[u, u] = D[u, u] = 0$ for any node u in T.

Figure 3 shows a tree and its Up and Down matrices. Notice that one of the internal nodes, namely the parent of b and c, does not have a label. The unlabeled node does not appear in the matrices. Comparing the Up and Down matrices with an additive distance matrix, cf. Figure 1, we note the main difference is that the additive distance matrix M is built for an unrooted unordered tree, i.e. a free tree. Each entry $M[u, v]$ represents the number of the undirected edges on the path connecting u and v. Thus, M is symmetric and $M[u, v] = M[v, u]$. By contrast, the Up and Down matrices are built with respect to a rooted tree and take into account the up and down operations along the directed edges between two nodes. Consider, for example, the nodes d and b in the tree T in Figure 3. If this tree were treated as an unrooted free tree, the number of edges between d and b would be 3. However, $U[d, b] = 1$ (representing 1 up operation, $d \uparrow a$) and $D[d, b] = 2$ (representing 2 down operations, $a \downarrow \lambda$ and $\lambda \downarrow b$ where λ represents the unlabeled node in T). Notice also that in this example, $U[b, d] = 2$ which is not equal to $U[d, b]$. Likewise $D[d, b] \neq D[b, d]$.

The following are some facts that can be observed directly from the above definitions.

Fact 1. *For any pair of nodes u, v in a tree T, $U[u, v] = D[v, u]$.*

Hence, from matrix U, we can obtain matrix D, and vice versa. We will therefore only use matrix U throughout the paper and refer to it as the UpDown matrix. The UpDown matrix describes the structure of T.

Fact 2. *For each node u in a tree T, $U[u, u] = D[u, u] = 0$.*

Fact 3. *Suppose u, v are two nodes in a tree T and u is the parent node of v. Then*
(i) $U[u, v] = 0$ and $U[v, u] = 1$;
(ii) For each node w in T, $w \neq u$, $w \neq v$, such that w is not a descendant of u, $U[w, v] = U[w, u]$ and $U[v, w] = U[u, w] + 1$.

3.3 TreeRank

In general, when using a search engine, if the user inputs a query tree with three nodes "fox","dog" and "tiger" plus their parent node "mammal", the user often expects to see data trees in search results containing these nodes. If the user doesn't want to see a search result containing, for example, a node "tiger", he or she can simply input a query tree having "fox", "dog" and "mammal" only.

This implies that in designing a search engine and similarity measure, the following two criteria should be considered together:

1. whether all, or at least most of, the labeled nodes of the query tree P occur in a data tree D;

2. to which extent the query tree P is similar to the data tree D in structure.

With these criteria in the mind, we seek nodes in D that match nodes in P when comparing P with D. Specifically, let V_P be the set of labeled nodes in P and let V_D be the set of labeled nodes in D. Let U_P represent the UpDown matrix of P and let U_D represent the UpDown matrix of D. Let I denote the intersection of V_P and V_D; let J denote $V_P - V_D$. We define the *UpDown*

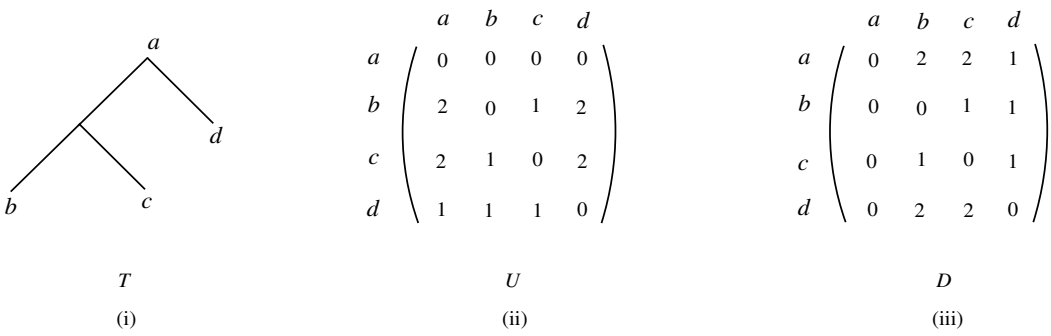

Figure 3. A tree and its Up and Down matrices.

distance from P to D, denoted $UpDown\_dist(P, D)$, as

$$UpDown\_dist(P, D) = \sum_{u \in I} \sum_{v \in I} |U_P[u, v] - U_D[u, v]| + \sum_{u \in J} \sum_{v \in J} U_P[u, v]$$

The TreeRank score from P to D, denoted $TreeRank(P, D)$, is calculated by

$$TreeRank(P, D) = (1 - \frac{UpDown\_dist(P, D)}{\sum_{u \in V_P} \sum_{v \in V_P} U_P[u, v]}) \times 100\%$$

The TreeRank score from P to D is a measure of the topological relationships in P that are found to be the same or similar in D. If P and D are the same or if one can find a substructure in D that exactly matches P, then $TreeRank(P, D)$ = 100%. On the other hand, if P and D do not have any labeled node in common, then $TreeRank(P, D)$ = 0. The time complexity of the algorithm for computing $TreeRank(P, D)$ is $O(M^2 + N)$ where M is the number of nodes appearing in both P and D, and N is the number of nodes in D.

4 Nearest Neighbor Searching

Figure 4 shows a query tree P and a data tree D that satisfy the three properties described in Section 2. In the biological sense, when comparing P with D, their distance should be 0. Motivated by this example, we incorporate a *data tree reduction* technique into our nearest neighbor searching algorithm, which works as follows.

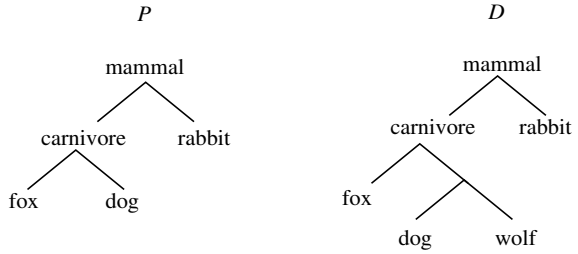

Figure 4. Example trees.

Consider a query tree P and a data tree D and their UpDown matrices. Find the column and row indexes of the nodes in the intersection of V_P and V_D. Mark those matching nodes in D with asterisks. If two distinct nodes of D are marked, then their least common ancestor is also marked. We then consider the reduced data tree D' of D that contains only the marked nodes. Equivalently, we remove unmarked nodes having only one neighbor (this must preserve connectedness). The above removal might yield additional unmarked nodes with one neighbor, which themselves will be removed. If an unmarked node n is connected to two other nodes m_1 and m_2, then remove n and link m_1 and m_2. This too preserves connectedness. Continue doing these two operations until neither can be done. The node removal operation is similar to the "degree-2 delete" operation defined in [40] where a node can be deleted when the node's degree is less than or equal to 2. Notice that after reduction, the UpDown matrices will change, and we use the new matrices to calculate the similarity score of P and D.

Figure 5 presents an example. In the figure, (i) shows a query tree, (ii) shows a data tree in which some nodes

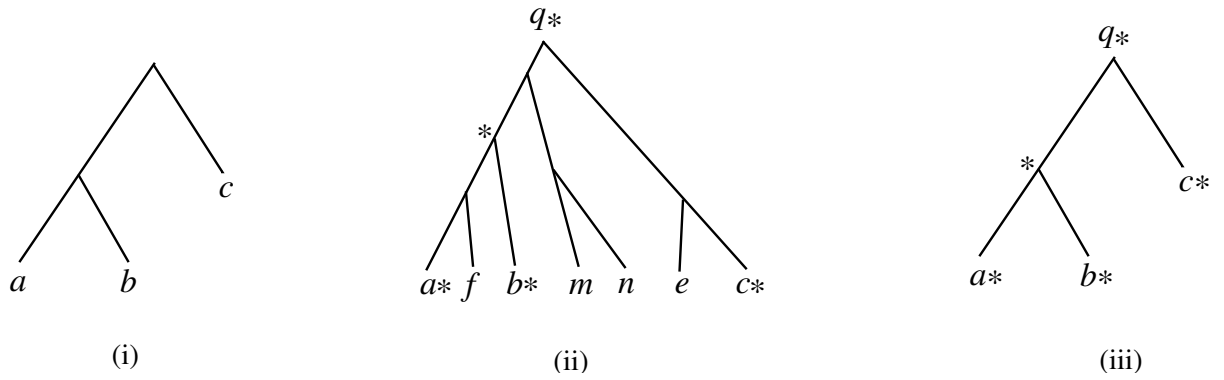

Figure 5. Example showing how the data tree reduction technique works in nearest neighbor searching.

are marked, and (iii) shows the reduced tree of the data tree in (ii). In performing nearest neighbor searches, our algorithm first applies the tree reduction technique to a data tree D, and then calculates the TreeRank score from the given query tree P to the reduced tree of D using the formula described in Section 3.3. The resulting score is then presented as the similarity score from P to D.[6] For example, in Figure 5, since the TreeRank score from the query tree in (i) to the reduced data tree in (iii) is 100%, our algorithm displays the data tree in (ii) with a 100% similarity score to the query tree. This matching technique yields a similar effect as tree matching with variable length don't cares [33, 34], though the proposed approach does not require the user to explicitly specify the don't cares in the query tree.

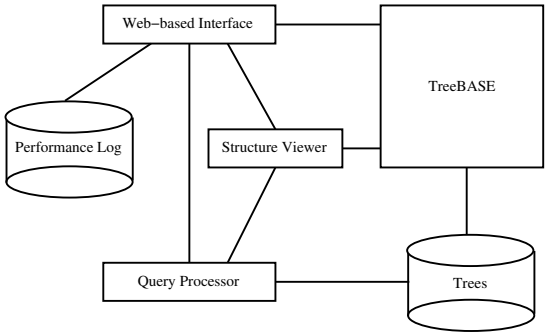

Figure 6. The software architecture of the proposed search engine.

5 Implementation

We have incorporated the nearest neighbor searching algorithm proposed here into a Web-based system. The search engine is implemented using Java, HTML, Perl CGI, and C. It is fully operational and is accessible at `http://aria.njit.edu/~biotool/treerank.html`. Figure 6 shows the software architecture of the search engine. The system is comprised of four components: Web-based Interface, Query Processor, Structure Viewer and Performance Log. From Web-based Interface, the user is able to type in his/her own query (an example tree), upload the query tree from a file, or use and modify a sample query provided by the system. Query Processor searches TreeBASE for phylogenetic trees that are nearest neighbors of the query tree using the algorithm described in Section 4. Structure Viewer displays the trees using either a parenthesized string notation or a dendrogram format, which are presented to the user via Web-based Interface. User queries and their time stamps are maintained in Performance Log, which helps to analyze user needs and better tune the system for working more effectively. The search engine is connected to TreeBASE on the Web and therefore it uses the visualization tools available in TreeBASE for displaying trees graphically.

Figure 7 shows the system's main screen and query interface (the upper left window), a query tree (the lower left window), and the query tree's nearest neighbor in

[6] We consider a data tree D to be a neighbor of P if P and D share at least one common leaf label. Otherwise D is not a neighbor of P and is not displayed as a search result.

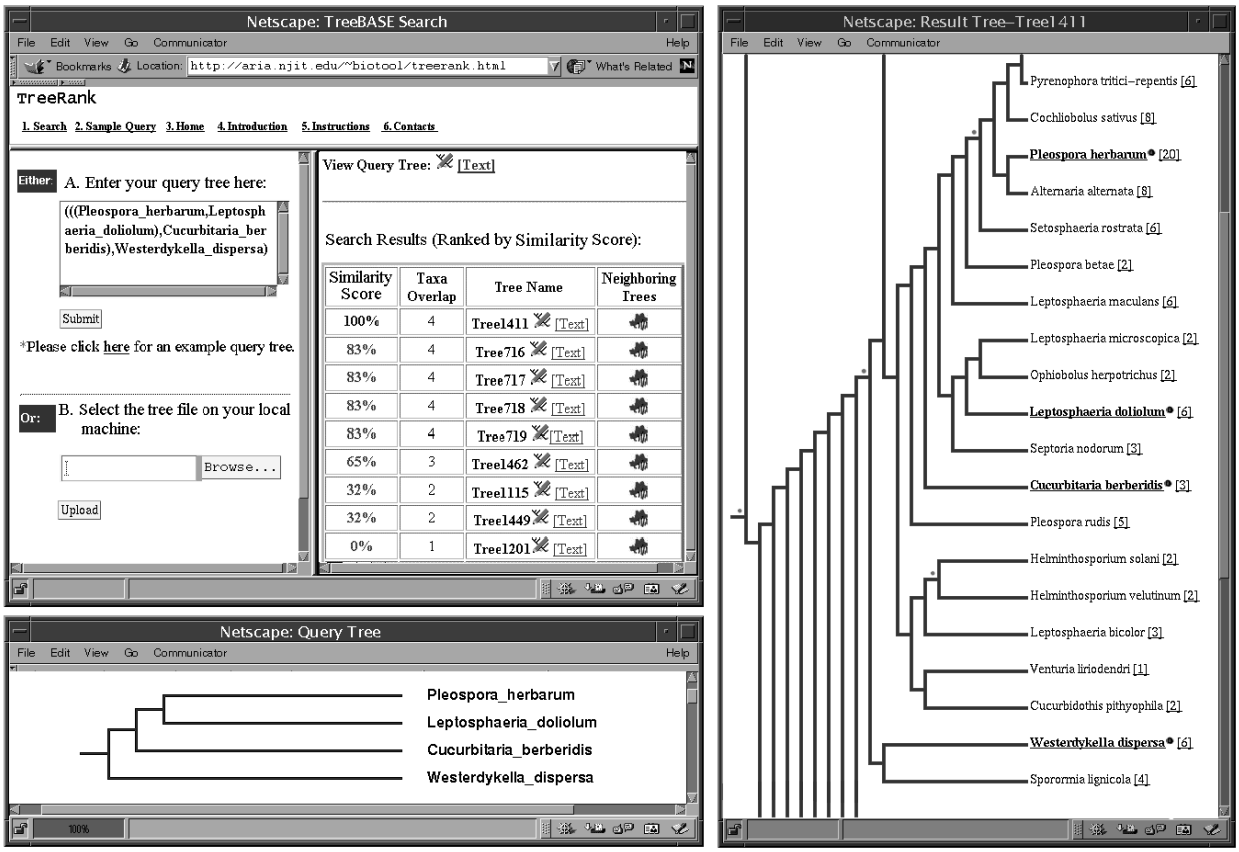

Figure 7. An example query and search results displayed via the Web-based interface of the proposed search engine.

TreeBASE (the right window). In the main screen, the query tree is expressed in the parenthesized string notation; in the other two windows this same query tree and the nearest neighboring tree are viewed in the dendrogram format. In general, to view a tree in the dendrogram format, the user would need to click the icon with the pencil overlaid upon the phylogenetic tree in the main screen. To view the parenthesized string notation, the user would need to click on the "Text" link in the main screen.

Figure 7 shows that Tree1411 is ranked highest, which is the nearest neighbor of the query tree with a 100% similarity score. Other similar trees are also displayed and ranked, from top to bottom, based on their similarity scores. For each displayed data tree D, when clicking on the icon in its "Neighboring Trees" column, the system will initiate a new search using D as the query tree, and rank and display D's nearest neighbors in TreeBASE. The "Taxa Overlap" column shows the number of leaves (taxa) D has that also appear in the query tree. The system only displays similar trees whose "Taxa Overlap" column has a number greater than 0.[7] Notice that even though the top five ranked data trees all have the same "Taxa Overlap" value, namely 4, only Tree1411 has a similarity score of 100%. The other four trees only have a similarity score of 83%.

Referring to the right window in Figure 7, the entire

[7] If a data tree does not share any common taxon with the query tree, our algorithm filters out the data tree immediately without applying tree reduction to it and calculating its similarity score. This filter works well in practice. Our benchmark result shows that for many query trees, the proposed system can perform a search on a set of approximately 1500 trees in about one second on a SUN Ultra 20 workstation.

Tree1411 is drawn and the user can use scroll bars to view portions of the tree. Named clades (labeled internal nodes) are indicated by a red dot. Each leaf node (taxon, species, or organism) has a number next to its label. This number represents the number of studies in TreeBASE the taxon is found within. If the user clicks on this number, he or she is linked to TreeBASE so that he or she can search on that taxon about those studies. The specific taxa specified within the query tree are highlighted in Tree1411 using underscored red font with a green circle next to the taxon's name. It should be pointed out that after applying the tree reduction technique to Tree1411, the reduced tree is exactly the same as the query tree. Consequently the similarity score for Tree1411 is 100%.

6 Related Work

In the past, a number of similarity and distance measures for phylogenetic trees have been proposed. Various algorithms for tree matching [1, 13, 18, 26, 29, 37, 39] and for tree reconstruction [2, 10, 22, 38] have been studied. Different theories, when applied to finding the phylogenetic relationship of the same set of species, often result in different phylogenetic trees. To determine how much two theories have in common is a fundamental problem in computational biology and in phyloinformatics.

The most comprehensive algorithmic and software tool in this field is perhaps the COMPONENT package (`http://taxonomy.zoology.gla.ac.uk/rod/cpw.html`) developed by Page at University of Glasgow. It provides several ways of finding the consensus of two phylogenetic trees. The first is to see whether they have similar "quartets" which are based on adjacency relationships among all possible subsets of four leaf species. The similarity is then computed as the proportion of quartets that are shared in the two trees [3, 5, 14].

Partition distance treats each phylogenetic tree as unrooted and analyzes the partitions of species resulting from removing one edge at a time. By removing an edge in a tree, one is able to partition that tree. The difference between two trees is defined as the number of edges for which there is no equivalent (in the sense of creating the same partitions) edge in the other tree [12].

The maximum agreement subtree between two phylogenetic trees T_1 and T_2 is a substructure of the two trees on which the two trees are the same [8, 9, 17, 22, 23, 24]. Commonly such a subtree will have fewer leaves than either T_1 or T_2. By contrast, a consensus tree has the same number of leaves as the original trees T_1 and T_2, assuming those trees have the same set of species. Consensus trees should be used gingerly, however, because a consensus tree is not a phylogeny unless the two trees are isomorphic. Instead, consensus trees are a convenient way to summarize the agreement between two or more trees. Consensus trees can be formed from cluster methods (strict, majority rule, semi-strict, or Nelson) or by intersection methods (Adams); see [12, 25, 28] for more details.

The last dissimilarity measure implemented in COMPONENT is the nearest neighbor interchange (NNI) distance. Given two unrooted unordered trees T_1 and T_2 with the same set of labeled leaves, their NNI distance is the number of NNI operations needed to transform T_1 to T_2. Finding the NNI distance between two trees is NP-hard. Brown and Day [4] developed several approximation algorithms to calculate the distance, which are implemented in COMPONENT.

In contrast to the above distance and similarity measures, which are developed for comparing two trees, possibly with some constraints (e.g. the two trees must have the same set of leaves), TreeRank is mainly designed for nearest neighbor searching in phylogenetic databases. In [34], we presented an approach called ATreeGrep, which measures the distance between two general rooted unordered trees by counting the mismatching paths in the two trees. By utilizing a suffix array index structure, ATreeGrep focuses on fast retrieval in a database of general rooted unordered trees. The tool has been applied to processing XMLs [33] and phylogenies [32]. It allows the user to add variable length don't cares to a query tree when a certain portion of a data tree is unimportant or unknown in matching with the query tree. By contrast, TreeRank is specially designed for rooted phylogenetic trees that satisfy the three properties described in Section 2. It captures the structural difference of a data tree with respect to the query tree by considering the up and down operations in the two trees. The nearest neighbor searching algorithm proposed here employs the data tree reduction technique described in Section 4, without asking the user to explicitly specify variable length don't cares in the query tree.

7 Conclusion and Future Work

In this paper we have presented a new approach to nearest neighbor searching among phylogenetic trees. Given a query or pattern tree P and a database of trees $\mathcal{D}$, the proposed approach finds and ranks data trees D where

D is similar to P, and D shares at least one common leaf label with P. If D doesn't share any common leaf label with P, our approach eliminates D without computing the similarity score from P to D.

In practice, another interesting search function is to find data trees in $\mathcal{D}$ that are within UpDown distance ϵ of the query tree P. We have implemented a filter to speed up this type of retrieval, which works as follows. For the database of trees, we create a hash table keyed by pair of node labels and each hash bin contains tree identification numbers. The pair can be in alphabetical order because $U[u, v] = D[v, u]$ for any pair of node labels (u, v) (cf. Fact 1 in Section 3). Now given the query tree P, we consider each pair of node labels in P and see which trees of the database the pair is in. (This requires time independent of the size of the database.) Sort the data trees by the number of hits.

When evaluating a data tree D, we get a lower bound on the distance from P to D by looking at $U_P[u, v]$ where U_P is the UpDown matrix of P and (u, v) is a pair in P that is missing from D. The lower bound is computed by summing up $U_P[u, v]$ for all pairs of (u, v) of P that are missing from D. If the lower bound is already greater than ϵ, we can eliminate D from consideration without calculating the distance of P and D. Furthermore, if a data tree D has a set S of k hits and it is decided D doesn't qualify to be a solution after calculating the distance of P and D, then any data tree D' that only has S' of k' hits, where $k' < k$ and S' is a subset of S, will not be a solution and hence can be eliminated from consideration. We found experimentally that this filter helps to eliminate, on average, more than 95% of data trees in performing nearest neighbor searches in TreeBASE. In separate experiments in which generated trees were used, we found that the same filtering ratio was achieved when the size of the database increased.

The proposed algorithms have been used for analyzing the structures of phylogenetic trees and for performing structure-based retrieval in TreeBASE. We have focused here on rooted unordered trees in which edges are not weighted. Our future work includes (i) adding weights to edges based on phylogenetic considerations and (ii) finding consensus trees in clusters based on this or similar measures.

8 Acknowledgement

We thank the SSDBM 2003 reviewers for their comments and suggestions.

References

[1] E. N. Adams. Consensus techniques and the comparison of taxonomic trees. *Systematic Zoology*, 21:390–397, 1972.

[2] V. Berry and D. Bryant. Faster reliable phylogenetic analysis. In *Proceedings of the 3rd Annual International Conference on Computational Molecular Biology*, pages 59–68, 1999.

[3] G. S. Brodal, R. Fagerberg, and C. N. S. Pedersen. Computing the quartet distance between evolutionary trees in time $O(nlog^2n)$. In *Proceedings of the 12th Annual International Symposium on Algorithms and Computation*, pages 731–742. Lecture Notes in Computer Science, Vol. 2223, Springer Verlag, Berlin, 2001.

[4] E. K. Brown and W. H. E. Day. A computationally efficient approximation to the nearest neighbor interchange metric. *Journal of Classification*, 1:93–124, 1984.

[5] D. Bryant, J. Tsang, P. Kearney, and M. Li. Computing the quartet distance between evolutionary trees. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, CA, 2000.

[6] P. Buneman. The recovery of trees from measures of dissimilarity. In *Mathematics in Archaeological and Historical Sciences*, pages 387–395. Edinburgh University Press, 1971.

[7] J. H. Camin and R. R. Sokal. A method for deducing branching sequences in phylogeny. *Evolution*, 19:311–326, 1965.

[8] R. Cole, M. Farach-Colton, R. Hariharan, T. M. Przytycka, and M. Thorup. An O(n log n) algorithm for the maximum agreement subtree problem for binary trees. *SIAM J. Comput.*, 30(5):1385–1404, 2000.

[9] R. Cole and R. Hariharan. An O(n log n) algorithm for the maximum agreement subtree problem for binary trees. In *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 323–332, 1996.

[10] B. DasGupta, X. He, T. Jiang, M. Li, J. Tromp, and L. Zhang. On distances between phylogenetic trees.

In *Proceedings of the 8th ACM-SIAM Symposium on Discrete Algorithms*, pages 427–436, 1997.

[11] B. DasGupta, X. He, T. Jiang, M. Li, J. Tromp, L. Wang, and L. Zhang. Computing distances between evolutionary trees. In D. Z. Du and P. M. Pardalos (eds.) *Handbook of Combinatorial Optimization*, Kluwer Academic Publishers, Volume 2, 1998, pages 35–76.

[12] W. H. E. Day. Optimal algorithms for comparing trees with labeled leaves. *Journal of Classification*, 2:7–28, 1985.

[13] P. Diaconis and S. P. Holmes. Computing with trees. In *Proceedings of the 31st Symposium on the Interface*, pages 414–419, 1999.

[14] C. R. Douchette. An efficient algorithm to compute quartet dissimilarity measures. Unpubl. BSc (Hons) Dissertation, Memorial Univ. Newfoundland, 1985.

[15] A. Dress and M. Kruger. Parsimonious phylogenetic trees in metric spaces and simulated annealing. *Advances in Applied Mathematics*, 8:8–37, 1987.

[16] M. Farach and M. Thorup. Fast comparison of evolutionary trees. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1994.

[17] M. Farach and M. Thorup. Optimal evolutionary tree comparison by sparse dynamic programming (extended abstract). In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 770–779, 1994.

[18] J. Hein, T. Jiang, L. Wang, and K. Zhang. On the complexity of comparing evolutionary trees. *Discrete Applied Mathematics*, 71:153–169, 1996.

[19] D. E. Joyce. Phylogeny and reconstructing phylogenetic trees. `http://aleph0.clarku.edu/~djoyce/java/Phyltree/cover.html`.

[20] S. Kannan, E. Lawler, and T. Warnow. Determining the evolutionary tree. In *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 475–484, 1990.

[21] S. Kannan, T. Warnow, and S. Yooseph. Computing the local consensus of trees. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1995.

[22] M. Kao, T. Lam, T. M. Przytycka, W. Sung, and H. Ting. General techniques for comparing unrooted evolutionary trees. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 54–65, 1997.

[23] E. Kubicka, G. Kubicki, and F. R. McMorris. An algorithm to find agreement subtrees. *Journal of Classification*, 12(1):91–99, 1995.

[24] T. W. Lam, W. K. Sung, and H. F. Ting. Computing the unrooted maximum agreement subtree in subquadratic time. In *Proceedings of the 5th Scandinavian Workshop on Algorithm Theory*, pages 124–135, 1996.

[25] G. Nelson. Cladistic analysis and synthesis: Principles and definitions, with a historical note on Adanson's Famille Des Plantes (1763-1764). *Syst. Zoology*, 28:1–21, 1979.

[26] A. S. Noetzel and S. M. Selkow. An analysis of the general tree-editing problem. In D. Sankoff and J. B. Kruskal (eds.) *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, pages 237–252. Addison-Wesley, Reading, MA, 1983.

[27] NSF Workshop Report at Yale University. Assembling the tree of life: Research needs in phylogenetics and phyloinformatics, July 2000.

[28] R. D. M. Page. Comments on component-compatibility in historical biogeography. *Cladistics*, 5:167–182, 1989.

[29] R. D. M. Page and M. A. Charleston. Trees within trees: Phylogeny and historical associations. *Trends in Ecology and Evolution*, 13:356–359, 1998.

[30] W. H. Piel, M. J. Donoghue, and M. J. Sanderson. TreeBASE: A database of phylogenetic information. In *Proceedings of the 2nd International Workshop of Species 2000*, 2000.

[31] M. J. Sanderson, M. J. Donoghue, W. H. Piel, and T. Eriksson. TreeBASE: A prototype database of phylogenetic analyses and an interactive tool for browsing the phylogeny of life. *American Journal of Botany*, 81(6):183, 1994.

[32] H. Shan, K. G. Herbert, W. H. Piel, D. Shasha, and J. T. L. Wang. A structure-based search engine for phylogenetic databases. In *Proceedings of the 14th*

International Conference on Scientific and Statistical Database Management, pages 7–10, 2002.

[33] D. Shasha, J. T. L. Wang, and R. Giugno. Algorithmics and applications of tree and graph searching. In *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 39–52, 2002.

[34] D. Shasha, J. T. L. Wang, H. Shan, and K. Zhang. ATreeGrep: Approximate searching in unordered trees. In *Proceedings of the 14th International Conference on Scientific and Statistical Database Management*, pages 89–98, 2002.

[35] TreeGen: Tree generation from distance data. Computational Biochemistry Research Group, ETH Zurich, `http://cbrg.inf.ethz.ch/Server/subsection3_1_6.html`.

[36] J. T. L. Wang, K. Zhang, G. Chang, and D. Shasha. Finding approximate patterns in undirected acyclic graphs. *Pattern Recognition*, 35(2):473–483, 2002.

[37] J. T. L. Wang, K. Zhang, K. Jeong, and D. Shasha. A system for approximate tree matching. *IEEE Transactions on Knowledge and Data Engineering*, 6(4):559–571, 1994.

[38] L. Wang, K. Zhang, and L. Zhang. Perfect phylogenetic networks with recombination. In *Proceedings of the ACM Symposium on Applied Computing*, pages 46–50, 2001.

[39] K. Zhang, R. Statman, and D. Shasha. On the editing distance between unordered labeled trees. *Information Processing Letters*, 42:133–139, 1992.

[40] K. Zhang, J. T. L. Wang, and D. Shasha. On the editing distance between undirected acyclic graphs. *International Journal of Foundations of Computer Science*, 7(1):43–57, 1996.