# TreeLign: Simultaneous Stepwise Alignment and Phylogenetic Positioning, with its Application to Automatic Phylogenetic Assignment of 16S rRNAs

Yuan Li
Department of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL, USA, 32816-2362
liy@eecs.ucf.edu

Aaron L. Halpern
J. Craig Venter Institute
Rockville, MD, USA, 20850
Complete Genomics
2071 Stierlin Court
Mountain View, CA 94043
ahalpern@completegenomics.com

Shaojie Zhang
Department of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL, USA, 32816-2362
shzhang@eecs.ucf.edu

## ABSTRACT

Phylogenetic assignment of 16s rRNA has been frequently used for taxonomic classification. Recently, high-throughput sequencing, especially in the context of environmental or metagenomic sequencing projects, has made fast and accurate taxonomic classification an important goal. Existing classification methods are either fast, but too coarse-grained and inaccurate or fine-grained and accurate but too slow for use in practice. In this paper, we propose a new computational method, TreeLign, to rapidly and accurately conduct alignment and phylogenetic assignments for novel sequences, given a reference phylogenetic tree and an alignment. TreeLign first constructs profiles of every branch on the reference tree, then, for each query sequence, tries assigning it to every possible branch, and finally obtains a new tree and a new alignment which are jointly optimal in terms of Maximum Parsimony (MP). We tested the accuracy and robustness of TreeLign on both a large and a small 16S rRNA dataset extracted from the core set of GreenGenes. The results on the large dataset show that the assignments of TreeLign are in general consistent with the phylogenetic tree of the core set of GreenGenes. And, the results on the small dataset show that TreeLign achieves comparable accuracy compared with existing maximum likelihood based methods, but requires much less computational time.

## Categories and Subject Descriptors

I.2.8 [**ARTIFICIAL INTELLIGENCE**]: Problem Solving, Control Methods, and Search—*Dynamic Programming*

## General Terms

Algorithm, Theory


*ACM BCB* '11 Chicago Illinois


## Keywords

16S rRNA, taxonomic classification, phylogenetic assignment

## 1. INTRODUCTION

Recently, a variety of metagenomics projects [2, 14, 22–24] have produced a large amount of 16S ribosomal RNA (16S rRNA) sequences. In addition, publicly available metagenomics databases are emerging [4, 6, 19]. Given this large amount of environmental sequences, we want to know the taxonomic groups they belong to. Inferring the phylogenetic locations of these environmental sequences enables us to sort these uncultivated sequences into already known taxonomic clades. Therefore, fast and accurate taxonomic classification and phylogenetic assessment of these 16S rRNA sequences have become a challenge for metagenomics. Similar challenges apply to analysis of other marker genes.

In fact, to taxonomically classify a novel sequence would be to perform a *de novo* alignment and phylogenetic inference (comprising the novel sequence and the reference sequences) simultaneously. This will lead to improvements in both accuracy of the alignment and accuracy of the phylogeny [11, 25, 32]. But, in general, such simultaneous determination is computationally intractable. Many heuristic solutions have been employed in rendering practical approximations for this problem [25, 32]. Typically, multiple alignment and phylogeny inference are computed separately (construct an optimal phylogeny based on a fixed alignment or vice versa). Moreover, even when computed separately, both problems are still hard [5, 9, 27]. In this case, adding a novel sequence into a carefully constructed existing alignment or phylogeny is a more practical approach.

There are several current state-of-art methods of phylogenetic assignment for 16S rRNA sequences. ARB parsimony [16] can add new sequences to a phylogenetic tree without perturbing its initial topology. However, the overall sequence alignment has to be constructed before conducting any phylogenetic analysis, and the quality of the alignment has great impact on the quality of the phylogenetic analyses that follow. In addition, the construction of the overall alignment often requires human interaction.

The Ribosomal Database Project (RDP) [28] server employs the RDP Classifier, a Bayesian classifier, to classify bacterial 16S rRNA sequences into the higher-order taxon-

omy [12], which is only capable of making hierarchical classification to the genus level. GreenGenes, an ARB compatible 16S rRNA gene database, also provides taxonomic classification tools using multiple published reference taxonomies [6]. It employs a consensus tree built from taxonomies proposed by several groups [15, 16, 18, 28] and maintains a consistent multiple sequence alignment of both archaeal and bacterial 16S rRNA genes to facilitate taxonomic placements. For user-generated 16S rRNA gene sequences, the NAST aligner [7] is used to align the query sequences to the consistent alignment. In addition, GreenGenes searches the near neighbors for each query sequence based on sequence similarity. As expected, these sequence similarity based (similar to BLAST-based) approaches are very fast, but are not that accurate.

TreePuzzle [20] is a Maximum Likelihood (ML) based software for phylogenetic analysis. MLTreeMap [26] is one of the TreePuzzle based approaches for making phylogenetic inference. MLTreeMap evaluates the ensemble of topologies using TreePuzzle and chooses the phylogenetic topology with the highest ML score. Due to the computational cost of TreePuzzle, it can only be applied on small datasets. RAxML [21] infers phylogenetic trees based on sequential and parallel ML, by conducting subtree rearrangement. AMPHORA [29] is one of the RAxML based approaches, it employs a methodology similar to that of MLTreeMap, and uses RAxML to construct the phylogenetic tree. RAxML is much faster than TreePuzzle because it employs parallel computing and a rapid bootstrapping algorithm. However, the speed of RAxML is still not satisfying for large trees regarding phylogenetic inference. In addition, RAxML employs subtree rearragnement technique for phylogenetic analysis, and thus does not respect the topology of the reference tree. On the other hand, we would prefer respecting the reference trees, because most of the reference trees should have been developed through costly processes (such as manual analysis) and been extensively studied [30, 31].

Here, we propose a new computational tool, called TreeLign, to assign novel 16S rRNA sequences into a reference phylogenetic tree while updating the multiple alignment simultaneously to achieve maximum parsimony. In the remainder of this paper, we describe the algorithm of TreeLign in section 2. In section 3, we evaluate and compare the accuracy and robustness of TreeLign with existing methods, and finally give discussions and remarks in section 4.

## 2. METHOD

Given a reference phylogenetic tree, an alignment of taxa in the tree and a new novel sequence, we propose here a method to quickly and accurately conduct phylogenetic assignment, putting the new sequence into the given tree and the given alignment. Let $W = \{w_1, \ldots, w_n\}$ be a set of $n$ species. Let $T_r = (V, E)$ be the (arbitrarily) phylogenetic tree of $W$, rooted at node $r$. For each node $v \in V$, we denote the subtree of $T_r$ rooted at $v$ by $T_v$. For simplicity and without loss of generality, we require $T$ to be a proper binary tree. Let $\Sigma = \{A, T, C, G\}$ be the alphabet representing the four nucleotide bases and let '$-$' represent a gap. Then, $\Gamma = \Sigma \bigcup \{-\}$ is the complete alphabet. Let $\mathbb{A} = \{A_1, A_2, \ldots, A_n\}$ be the multiple sequence alignment of length $L$ for species in $W$, where $A_i \in \Gamma^L$ represents the aligned sequence of $w_i$. Let $s_{new}$ denote the molecular sequence of a novel species labeled $w_{new}$. We aim to construct a new phylogenetic tree $T'$ that best explains the phylogenetic relationships of $w_{new}$ to all the species in $W$ by adding $w_{new}$ to $T$, and simultaneously generate a new multiple alignment $\mathbb{A}'$ by aligning $s_{new}$ with $\mathbb{A}$. The construction is based on the MP principle, so the parsimony score of $\mathbb{A}'$ on $T'$ should be as small as possible (an indel or a mutation costs 1, with the objective being to minimize the total cost) while $\mathbb{A}$ and $T$ are respected in $\mathbb{A}'$ and $T'$.

The joint cost (the increase in parsimony score for $T'$ and $\mathbb{A}'$ relative to $T$ and $\mathbb{A}$) of adding $w_{new}$ to a given branch of $T$, assuming a fixed alignment, is computed in the following way. Assigning $w_{new}$ to $T$ along a given branch $e(u, v)$ is conducted by subdividing the branch $e(u, v)$, adding a new internal node $x$ between nodes $u$ and $v$, and attaching $w_{new}$ as a descendent node of $x$. This operation introduces additional mutations (costs) to the reference tree and the alignment. For a fixed alignment $\mathbb{A}'$, the amount of additional cost can be determined by checking the number of mutations between $s_{new}$ to the profile of the branch. The branch profile represents a consensus of all aligned ancestral sequences that may possibly occur on that branch: for each base of $s_{new}$ (or a gap introduced into $s_{new}$), the cost is 0 if it matches a possible ancestral sequence at the same position, and 1 otherwise.

To achieve jointly optimal topology and alignment for stepwise addition, it remains to find the alignment of $s_{new}$ to $\mathbb{A}$ that results in the minimum score when the sequence is added to a given branch. If this problem (referred to as the Profile sequence alignment problem and will be stated in next subsection) is solved, the decision among the $2n - 2$ possible topologies can be made by exhaustively evaluating every branch, as in typical stepwise-addition approaches to phylogeny reconstructions. Therefore, we first simply evaluate the cost of assigning the novel sequence to every branch in the reference tree, then determine the "optimal" branch that leads to the minimum cost, and finally generate the topology of $T'$. Once $s_{new}$ is aligned with a branch profile, the pattern of the alignment can be propagated back to all other nodes without disturbing the reference alignment, except for inserting necessary gaps. Consequently, we are able to construct a new alignment $\mathbb{A}'$ that achieves the minimal increase in score in conjunction with the new phylogenetic tree $T'$.

In summary, the procedure of phylogenetic assignment consists of three phases:

1. **Branch profile construction**, to construct branch profiles for all the branches in the reference phylogeny.
2. **Profile sequence alignment**, to compute the cost of aligning the novel sequence to every branch.
3. **Optimal phylogeny and alignment determination**, to determine the optimal branch that leads to a minimal increase in MP score, and thereafter generate a new phylogenetic tree and a new alignment accordingly.

### 2.1 Branch profile construction

In this section, we describe how branch profiles are constructed. By using Fitch's algorithm [10], we can compute node profiles of all the interior and exterior nodes based on the reference phylogeny $T$ and the alignment $\mathbb{A}$. For any branch $e(u, v) \in E(T)$, we denote the branch profile of $e$ by $p_{uv}$. $p_{uv}$ is the union of bases present at either endpoints of the branch, and it can be inferred from the profiles of

nodes $u$ and $v$ (denoted by $p_u$ and $p_v$). For each column $k$, let $p_u^k$, $p_v^k$ and $p_{uv}^k$ denote the $k^{th}$ column of $p_u$, $p_v$ and $p_{uv}$ respectively. Then, $p_u^k$, $p_v^k$ and $p_{uv}^k$ each represents a set of characters that may occur in the $k^{th}$ column of $p_u$, $p_v$, and $p_{uv}$ respectively. Assume each column of a profile is independent, then the value of $p_{uv}^k$ should only depend on $p_u^k$ and $p_v^k$, and $p_{uv}^k = p_u^k \bigcup p_v^k$. Clearly, $p_{uv}^k \subseteq \Gamma$ as $p_u^k \subseteq \Gamma$ and $p_v^k \subseteq \Gamma$. For every branch in $T$, as long as the node profiles of both endpoints are known, we can construct the profile of the branch.

## 2.2 Profile sequence alignment

As stated above, the problem of aligning a novel sequence to a reference tree along a branch can be turned into the problem of aligning the sequence to the branch profile of that branch, where each branch profile represents a set of all possible sequences on that branch which are consistent with the MP criterion. From another point of view, each branch profile can also be considered as a network of $L$ layers. Each layer has at most five disconnected internal nodes that are fully connected to the nodes of the adjacent layers. In this case, the problem of profile sequence alignment can be seen as a special case of the network matching problem [13]. Next, we will propose an algorithm that efficiently determines the optimal alignment of a branch profile and a sequence and computes the parsimony score.

At the first step, we need to define the score for aligning a single column of a branch profile with a single base of a sequence. Given a branch profile $p_{uv}$ and a novel sequence $s_{new}$, the cost of aligning the $k^{th}$ column of $p_{uv}$ to the $h^{th}$ base of $s_{new}$ is either 0 or 1, depending on whether the base $s_{new}^h$ is present at the set of $p_{uv}^k$ (a match) or not (a mismatch). The cost of aligning $s_{new}^h$ with $p_{uv}^k$ is 0, if $s_{new}^h \in p_{uv}^k$; and 1, otherwise.

$$score(p_{uv}^k, s_{new}^h) = \begin{cases} 0 & s_{new}^h \in p_{uv}^k \\ 1 & otherwise \end{cases} \quad (1)$$

Next, we apply dynamic programming to determine an optimal alignment of $p_{uv}$ and $s_{new}$. Let $cost[0..L][0..l]$ be an $(L+1)\times(l+1)$ array, where $L$ is the length of profile $p$ and $l$ is the length of sequence $s$. For each $i$ and $j$, $cost(i,j)$ is the cost of the optimal alignment between the sub-profile $p^{1..i}$ and the sub-sequence $s^{1..j}$. Let both the gap penalty and the cost of a mismatch be 1, we have the following recursive formulation.

For boundary conditions,

$$\begin{aligned} score(0,0) &= 0 \\ score(0,j) &= score(0,j-1)+1 \\ score(i,0) &= \begin{cases} score(i-1,0) & \text{if } - \in p^i \\ score(i-1,0)+1 & \text{if } - \notin p^i \end{cases} \end{aligned} \quad (2)$$

For $i \geq 1$ and $j \geq 1$,

$$score(i,j) = \min \begin{cases} score(i-1,j-1) & \text{if } s^j \in p^i \\ score(i-1,j-1)+1 & \text{if } s^j \notin p^i \\ score(i,j-1)+1 & \\ score(i-1,j) & \text{if } - \in p^i \\ score(i-1,j)+1 & \text{if } - \notin p^i \end{cases} \quad (3)$$

The MP cost of aligning a novel sequence to a branch profile explains the number of additional mutations introduced to the phylogeny. If the novel sequence conforms to the profile of the branch (when it happens to be in the set of all possible aligned ancestral sequences on that branch), there should be no cost associated with adding the novel sequence along that branch; otherwise, the cost is equivalent to the cost of aligning the novel sequence to the profile of that branch.

## 2.3 Optimal phylogeny and alignment determination

After the costs of aligning the novel sequence along all the branches are computed, we can select the branch that gives the minimal cost. Then, we construct the topology of the new phylogenetic tree by subdividing the selected branch and adding the novel sequence as a descendant node of the internal node that splits the branch. Further, we generate the new alignment by propagating the pairwise alignment between the novel sequence and the branch profile back to each interior and exterior node. For example, if column $i$ of the branch profile is aligned with column $j$ of the novel sequence (or a gap), then column $i$ of the reference alignment should also be aligned with column $j$ of the novel sequence. Similarly, an insertion of a gap to column $i$ of the branch profile should lead to an insertion of a gap to every sequence of the reference alignment. The above procedure describes how the topology of the resulting phylogenetic tree and the resulting alignment are constructed in TreeLign.

If branch lengths of the new phylogeny are also in need, one may resort to dedicated programs such as TreePuzzle for precise branch lengths assignments. However, for the sake of convenience, we also provide an approximation to estimate lengths of branches that are incident to the new internal node. Let $e(u,v)$ be the branch along which $w_{new}$ is inserted and $x$ be the new internal node which splits $e(u,v)$. Let $|e(u,v)|$ be the branch length of $e(u,v)$. Since, the pairwise distances between any pair of existing leaf nodes in the reference tree are kept intact, then $|e(u,x)| + |e(x,v)| = |e(u,v)|$. And, the ratio of $|e(u,x)|$ over $|e(x,v)|$ is equivalent to the ratio of the profile divergence between nodes $u$ and $x$ over the profile divergence between nodes $x$ and $v$.

## 2.4 Speed up of TreeLign

The overall time complexity of TreeLign is $O(nlL)$. We can speed up TreeLign by eliminating the number of branches to assign by using BLAST as a filter and we term this heuristic as TreeLignB (TreeLign with BLAST filter). Although BLAST is not that accurate for assigning a query sequence to a reference phylogenetic tree, it gives an indication of where the best branch to assign may reside in general. The assumption is that the ideal branch to assign should be close to, if not exactly incident to, the leaf node that represents the best hit of BLAST. If there is a high sequence similarity between the two, it is unlikely the query sequence will be inserted into the tree along a branch which is far away from the best hit node. Usually, sequences of nodes that are distantly away from the best hit node should be greatly different from the sequence of the best hit node, and also be dissimilar to the query sequence. Thus, we do not need to try every possible branch on the reference tree, but only a subset of branches that reside within a certain distance to the best hit. We can narrow down the search space by only searching branches that are within the search radius to the best hit node, and discard all the remaining branches. The radius is defined in terms of the sum of branch lengths, and it may range from 0 (in which only the best hit node is examined), to the diameter of the tree (in which all branches are tried). The time complexity of TreeLignB is $O(KlL)$, where $K$ is the number of branches within the search space.

# 3. RESULTS AND DISCUSSION

## 3.1 datasets

The reference phylogenetic tree and multiple alignment for our tests were drawn from GreenGenes [6] in order to test if TreeLign could be used as a de novo approach on a real dataset. We downloaded the GreenGenes ARB database from GreenGenes' website [1], and extracted a core set phylogenetic tree (which is named tree_core_set) as well as its corresponding multiple alignment from the database for our tests. The resulting reference alignment and tree (which is referred to as "the large dataset") contains 3730 archeal and bacterial species and has a diameter of 0.6968. The length of the reference alignment is 3408. We also randomly picked 200 Proteobacteria from the large dataset, including 2 Ralstonia, 12 Burkholderiales and 32 $\beta$-Proteobacteria, and extracted

both the corresponding phylogenetic tree and the alignment for the selected species (referred to as "the small dataset"). The diameter of the resulting tree is 0.3775.

## 3.2 Measures of observed deviated distance

Following the method used by von Mering et al. [26], we evaluated the accuracy of the phylogenetic assignment of TreeLign based on the deviated distance between the ideal phylogenetic position where the query species should belong to and the actual phylogenetic assignment observed. Figure 1 shows the procedures of removing a clade from a reference phylogenetic tree and assigning a query species to the remaining tree. In tree $T_0$, the subtree rooted at node $D$ is removed from the overall phylogeny. This causes $B$, the parent of $D$, to be also removed, and the sibling of $D$ becomes the child of $A$ in place of $B$ to keep bifurcated, as shown in $T_1$. As we respect the reference phylogeny, for any removed species, the ideal phylogenetic assignment should be on the basal position of the new link. Tree $T_2$ represents an observed phylogenetic assignment, in which a query species $X$ and an new internal node $Y$ are inserted. We can obtain the actual phylogenetic assignment by mapping the position of $Y$ onto the branch between $C$ and $G$ in $T_3$. The red star on edge $e(v_A, v_E)$ and the green star on edge $e(v_C, v_G)$ in $T_3$ represent the ideal and actual phylogenetic assigning positions respectively.

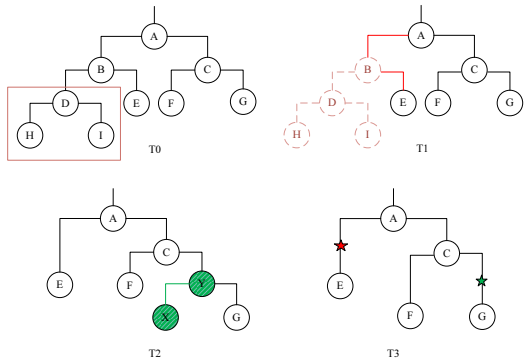


**Figure 1: The removal of a clade from phylogeny and the assignment of a new node.**

The minimum distance between the ideal and actual observed branch to assign, denoted by $M_d$, is a good measure for deviated distances. However, there are cases when branch lengths of the reference tree are absent, or one may only care about the changes of topology rather than distances. Then, the number of edges along the path from the original attachment point to the new attachment point, denoted by $M_b$, provides an alternative measure. $M_b$ is of more topological meaning than quantitative meaning, it cares about the number of branches between the ideal and observed assignments, regardless of the actual deviated distance. Finally, we define $M_p$, which represents the exact distance between the ideal (the red star) and the observed (the green star) points to assign, as the third measure. Let $e_i$ be the ideal branch to assign, $e_o$ be the observed branch to assign, and $e_n$ be any neighboring branch that shares exactly one endpoint with $e_o$. $M_d$ and $M_b$ are computed as following:

$$M_d(e_i, e_o) = \begin{cases} 0, & \text{if } e_i \text{ and } e_o \text{ share endpoints} \\ \min_{e_n}\{|e_n| + M_d(e_i, e_n)\} & \text{otherwise} \end{cases}$$

$$M_b(e_i, e_o) = \begin{cases} 0, & \text{if } e_i \text{ and } e_o \text{ are identical} \\ \min_{e_n}\{1 + M_b(e_i, e_n)\} & \text{otherwise} \end{cases} \quad (4)$$

## 3.3 Benchmarking procedure

We compared the performance of TreeLign with the aforementioned approaches on taxonomic assignments. For TreeLign, we first eliminated the poorly aligned region of the alignment by removing all the columns that were composed of more than 80% gaps. Due to the limitations of assembly and sequencing techniques, many query sequences are only partial 16S rRNAs sequences. In this case, these partial sequences should only be aligned to part of the reference alignment. Therefore, we used TreeLign with local alignment for the tests. For BLAST based approaches, we just selected the best hit of the query sequence out of all the sequences in the reference dataset. For TreePuzzle based approaches, the procedure exactly followed MLTreeMap [26] except that 16S rRNA sequences were used instead of marker genes. First, we built a Hidden Markov Model template for the alignment using HMMBuild [8] and then aligned the query sequence to the template using Hmmalign and obtained the new alignment. Next, we used GBLOCKS [3] to eliminate the poorly aligned regions. Finally, we applied TreePuzzle to compute the ML score of every topology in the ensemble, and then selected the one that had the best ML score as the resulting phylogenetic tree. Because TreePuzzle is able to assign branch lengths for the user-defined trees, we were able to compute all of $M_d$, $M_b$ and $M_p$. For RAxML based approaches, the procedures were exactly the same as TreePuzzle except that the ensemble of topologies were evaluated and scored by RAxML. (Note that RAxML is able to generate an overall phylogenetic topology based on sequence alignment under certain topological constraints.) Here, we restricted RAxML to consider user-specified trees that were consistent with the reference topology. First, we constructed an ensemble of all possible phylogenetic topologies, which integrated the query speices while respecting the reference topology. Next, we used RAxML to evaluate the likelihood of each user-specific phylogenetic tree to be a correct representation of the alignment. Since there was no branch lengths information available, we only benchmarked $M_d$ and $M_b$ for RAxML.

## 3.4 Performance of TreeLign

In order to evaluate the performance of TreeLign, we want to know, given an existing well-established reference phylogeny and its corresponding reference alignment, whether TreeLign can make fast, accurate and robust phylogenetic assignments for novel sequences. The first test is to take in a high-quality large alignment and phylogeny and see how accurately we can reconstruct the alignment and assign the phylogenetic position of each sequence if we remove it from the reference and apply the algorithm. The second test is to remove different clades of species from the existing alignment and phylogeny, and then reassign these removed species back into the remaining tree. The second test is able to estimate the robustness of TreeLign in the scenario when closely related species are missing.

### 3.4.1 *Validation of accuracy on the large dataset*

In this test, for each species in the large dataset, we simply removed it from the reference tree and the alignment. After that, we reassigned each species back to the reference tree and aligned its sequence with the reference alignment. ML-based methods were not compared in this larget dataset test, because most of them are too slow for thousands of species. For example, TreePuzzle requires more than one month to evaluate the ensemble of all possible topologies generated for assigning ONLY one query species to the reference phylogeny in the large dataset, not to mention thousands of queries. In this case, we only compared the accuracy of TreeLign with the BLAST based method.

Results show that TreeLign performs much better than BLAST, according to all three measures. TreeLign makes perfect phylogenetic assignments for 45.2% species, with average deviated distances $M_d = 0.0083, M_b = 2.40, M_p = 0.030$; while BLAST makes perfect assignments for 34.91% species, with average deviated distances $M_d = 0.015, M_b = 3.65, M_p = 0.056$. This proves that TreeLign is able to make consistent phylogenetic assignments with the reference phylogeny and thus can be successfully used as a taxonomic classification tool for de novo datasets.

**Table 1: Robustness test on the large dataset among BLAST-based mapping, TreeLign and TreeLignB**

| | Methods | Removed Clades | | | |
|---|---|---|---|---|---|
| | | Ralstonia | Burkholderiales | $\beta$-proteobacteria | Proteobacteria |
| $M_d$ | BLAST | 0.02223 | 0.03234 | 0.07381 | 0.05187 |
| | TreeLign | 0.01965 | 0.01192 | 0.03585 | 0.01951 |
| | TreeLignB | 0.02786 | 0.02297 | 0.02891 | 0.01233 |
| $M_b$ | BLAST | 3.0 | 8.522 | 12.21 | 11.79 |
| | TreeLign | 6.5 | 4.13 | 7.654 | 6.584 |
| | TreeLignB | 8.5 | 6.391 | 6.75 | 5.637 |
| $M_p$ | BLAST | 0.04050 | 0.06899 | 0.1115 | 0.10580 |
| | TreeLign | 0.04708 | 0.02571 | 0.06796 | 0.04778 |
| | TreeLignB | 0.03716 | 0.02958 | 0.04057 | 0.03143 |

### 3.4.2 *Estimation of robustness on both the large and small datasets*

The errors of classifications and assignments for species with low abundance on the reference dataset are much higher than those with high abundance. We define robustness of phylogenetic assignment as the ability to assign a novel sequence, which has little or no close relatives in the reference phylogeny, accurately. The property of robustness is important for phylogenetic assignment because although the current phylogenetic tree covers species of a lot of taxonomic groups, there are still some lineages, especially those with environmental low abundance or restricted to poorly-studied environments, remain uncovered. Without robustness, taxonomic classifications of query species from unrepresented clades are usually error prone.

We estimated the robustness of TreeLign by intentionally removing clades that are closely related to the query species from the reference tree and the alignment, and then evaluating quality of the phylogenetic assignment of each species in the clade on the remaining dataset. We selected different levels of clades (Genus, Class, Order and Phylum) to remove for the test, and compared the robustness of TreeLign against all the aforementioned methods. TreeLign and BLAST were tested on both large and small datasets, while TreePuzzle and RAxML based methods were only tested on the small dataset.

For tests on the large dataset, the removed clades were Ralstonia (Genus), Burkholderiales (Class), $\beta$-proteobacteria (Order) and Proteobacteria (Phylum), as in previous tests of ML-TreeMap [26]. The removed clades contained 2, 23, 52 and 903 species respectively. The average deviated distance between the ideal and actual assignment was determined using $M_d$, $M_b$ and $M_p$. Comparisons among TreeLign and TreeLignB (with radius 0.20) and the BLAST based method are shown in Table 1. TreeLign outperforms the BLAST based method except when Ralstonia, which contains only two species, is removed.

For tests on the small dataset, the removed clades were Ralstonia, Burkholderiales and $\beta$-proteobacteria, each containing 2, 12 and 32 species respectively. Comparisons among TreeLign, TreeLignB (with radius 0.16) and aforementioned methods are shown in Table 2. These results indicate that TreeLign can make robust phylogenetic assignments even when different levels of closely related clades are removed. In general, the quality of results generated by TreeLign is comparable to that of ML-based methods, but requires much less time. For example, when the clade of Burkholderiales is removed, on average, the deviated distance $M_p$ of reassigning the removed species back is equivalent to 0.068 using BLAST, while it equals to 0.021, 0.019 and 0.014 using TreeLign, TreePuzzle, and RAxML respectively. The deviated distance of BLAST is almost three folds of those of the other methods, but the differences in deviated distances among TreeLign, TreePuzzle and RAxML are slight. Results are similar for $M_d$ and $M_b$. This is probably because although ML is a better estimation for phylogenetic reconstruction, the result of a poor ML search may be no better than that of a more thorough search under some faster criterion [17].

We also tested the contribution of simultaneous optimization, as compared to simply using MP to score phylogenetic placement given a fixed alignment. To evaluate the latter, we conducted a test similar to that of Table 2 and we used a fixed alignment built by HMMBuild and HMMAlign rather than allowing optimization of the alignment for each branch. Run times are predictably much faster, but the results are considerably worse. Regardless of the clades removed, the deviated distance is greater than that using TreeLign by at least 22% in terms of $M_d$ and at least 35% in terms of $M_b$.

**Table 2: Robustness test on the small dataset among BLAST-based mapping, ML-based mappings, TreeLign and TreeLignB**

| | Methods | Removed Clades | | |
|---|---|---|---|---|
| | | Ralstonia | Burkholderiales | $\beta$-proteobacteria |
| $M_d$ | BLAST | 0.06479 | 0.04105 | 0.08386 |
| | TreeLign | 0 | 0.00809 | 0.02773 |
| | TreeLignB | 0 | 0.01342 | 0.04744 |
| | TreePuzzle | 0 | 0.00330 | 0.03723 |
| | RAxML | 0 | 0.00271 | 0.039 |
| $M_b$ | BLAST | 9.5 | 6.33 | 8.562 |
| | TreeLign | 0 | 3.33 | 3.344 |
| | TreeLignB | 1 | 3.33 | 5.625 |
| | TreePuzzle | 0 | 2.25 | 4.594 |
| | RAxML | 0 | 2 | 4.812 |
| $M_p$ | BLAST | 0.11830 | 0.06807 | 0.1159 |
| | TreeLign | 0.00704 | 0.02179 | 0.05565 |
| | TreeLignB | 0.02361 | 0.01907 | 0.06861 |
| | TreePuzzle | 0.00642 | 0.01403 | 0.06509 |
| | RAxML | - | - | - |

## 3.5 Time performance

The time performance of BLAST, TreeLign, TreeLignB, TreePuzzle and RAxML on both the large dataset and the small dataset is shown in Table 3. The machine configurations are 32 bit, 2.4G Hz Quad-CPU, 3.0 GB memory, Fedora 11 OS. In general, TreeLign needs more time than BLAST but is much faster than the others; while TreeLignB is even faster than TreeLign, but at the expense of accuracy. BLAST is the fastest of all but not as accurate as the others because it is designed for finding regions with local similarity between sequences, while sequence similarity is not the gold-standard for phylogenetic assignment, especially when sequences of the species are divergent. The computational time required by TreeLign is on the order of a thousand times faster than TreePuzzle for both datasets, and hundreds times faster than RAxML. ML-based methods such as TreePuzzle and RAxML require a lot of computational resources for calculating likelihoods and probabilities, while TreeLign is based on MP criterion. In addition, ML-based methods may not make extensive use of available information contained in the reference phylogeny, thus consume a lot of computational time for reference species, among which, the phylogenetic relationships are well known. In contrast, TreeLign takes advantage of the established reference

phylogeny and make incremental assignments for novel sequences without unnecessary re-computation. With the advancement of sequencing techniques, more genetic sequences of hundreds of thousands of environmental species will be available. TreeLign can be used to provide accurate phylogenetic assignments in a reasonable time.

**Table 3: Comparison of time performance**

| Methods | The large dataset | The small dataset |
|---|---|---|
| BLAST | 2 seconds | less than 1 seconds |
| TreeLign | 7 minutes 12s | 11 seconds |
| TreeLignB | 1 minute 40s | 6 seconds |
| TreePuzzle | >1 month | 1.1 hours |
| RAxML | 21 hours | 21 minutes |

In addition, the run time of TreeLign can also largely be improved using Multiple Threading Technique. TreeLign chooses from all possible phylogenetic assignments the one with minimal cost, while the evaluation of each assignment is independent. Therefore, the algorithm of TreeLign can be divided into a set of profile sequence alignment problems, and each problem can be solved independently using Multiple Threading Technique.

# 4. CONCLUSION

In this paper, we have developed TreeLign, an automated method that makes fast and accurate phylogenetic assignments for novel sequences while improving both the phylogeny and alignment. TreeLign first constructs profiles of every branch on the reference tree, then, for each query sequence, tries assigning it to every possible branch, and finally obtains a new tree and a new alignment which are jointly optimal in terms of Maximum Parsimony. By testing on 16S rRNA sequences, TreeLign is shown to make accurate assignments that are comparable to ML-based methods, while requires a lot less computational resources. The program is freely available at http://www.genome.ucf.edu/TreeLign.

# 5. ACKNOWLEDGMENTS

The project was supported in part by the J. Craig Venter Institute.

# 6. REFERENCES

[1] GreenGenes, `http://greengenes.lbl.gov/Download/Sequence_Data/Arb_databases`.

[2] M. Breitbart, P. Salamon, et al. Genomic analysis of uncultured marine viral communities. *Proc. Natl. Acad. Sci. U.S.A.*, 99:14250–14255, Oct 2002.

[3] J. Castresana. Selection of conserved blocks from multiple alignments for their use in phylogenetic analysis. *Mol. Biol. Evol.*, 17:540–552, Apr 2000.

[4] J. R. Cole, B. Chai, et al. The ribosomal database project (RDP-II): introducing myRDP space and quality controlled public data. *Nucleic Acids Res.*, 35:D169–172, Jan 2007.

[5] W. H. Day. Computational complexity of inferring phylogenies from dissimilarity matrices. *Bull. Math. Biol.*, 49:461–467, 1987.

[6] T. DeSantis, P. Hugenholtz, et al. Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB. *Appl. Environ. Microbiol.*, 72:5069–5072, Jul 2006.

[7] T. DeSantis, P. Hugenholtz, et al. NAST: a multiple sequence alignment server for comparative analysis of 16S rRNA genes. *Nucleic Acids Res.*, 34:W394–399, Jul 2006.

[8] E. S. Durbin, R., A. Krogh, and G. Mitchison. *Biological Sequences Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge, UK, 1998.

[9] I. Elias. Settling the intractability of multiple alignment. *J. Comput. Biol.*, 13:1323–1339, Sep 2006.

[10] W. Fitch and E. Margoliash. Construction of Phylogenetic Trees. *Science*, 155(3760):279–284, 1967.

[11] R. Fleissner, D. Metzler, and A. von Haeseler. Simultaneous statistical multiple alignment and phylogeny reconstruction. *Syst. Biol.*, 54:548–561, Aug 2005.

[12] G. M. Garrity, J. A. Bell, and T. G. Lilburn. *Bergey's Manual of Systematic Bacteriology, Second Edition*. Springer, New York, 2004.

[13] D. Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, New York, NY, USA, 1997.

[14] S. J. Hallam, N. Putnam, et al. Reverse methanogenesis: testing the hypothesis with environmental genomics. *Science*, 305:1457–1462, Sep 2004.

[15] P. Hugenholtz. Exploring prokaryotic diversity in the genomic era. *Genome Biol.*, 3:REVIEWS0003, 2002.

[16] W. Ludwig, O. Strunk, et al. ARB: a software environment for sequence data. *Nucleic Acids Res.*, 32:1353–1371, 2004.

[17] D. A. Morrison. Increasing the efficiency of searches for the maximum likelihood tree in a phylogenetic analysis of up to 150 nucleotide sequences. *Syst. Biol.*, 56:988–1010, Dec 2007.

[18] N. R. Pace. A molecular view of microbial diversity and the biosphere. *Science*, 276:734–740, May 1997.

[19] E. Pruesse, C. Quast, et al. SILVA: a comprehensive online resource for quality checked and aligned ribosomal RNA sequence data compatible with ARB. *Nucleic Acids Res.*, 35:7188–7196, 2007.

[20] H. A. Schmidt, K. Strimmer, et al. TREE-PUZZLE: maximum likelihood phylogenetic analysis using quartets and parallel computing. *Bioinformatics*, 18:502–504, Mar 2002.

[21] A. Stamatakis. RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics*, 22:2688–2690, Nov 2006.

[22] S. G. Tringe, C. von Mering, et al. Comparative metagenomics of microbial communities. *Science*, 308:554–557, Apr 2005.

[23] G. W. Tyson, J. Chapman, et al. Community structure and metabolism through reconstruction of microbial genomes from the environment. *Nature*, 428:37–43, Mar 2004.

[24] J. C. Venter, K. Remington, et al. Environmental genome shotgun sequencing of the Sargasso Sea. *Science*, 304:66–74, Apr 2004.

[25] M. Vingron and A. von Haeseler. Towards integration of multiple alignment and phylogenetic tree construction. *J. Comput. Biol.*, 4:23–34, 1997.

[26] C. von Mering, P. Hugenholtz, et al. Quantitative Phylogenetic Assessment of Microbial Communities in Diverse Environments. *Science*, 315(5815):1126–1130, 2007.

[27] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *J. Comput. Biol.*, 1:337–348, 1994.

[28] Q. Wang, G. M. Garrity, et al. Naive Bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy. *Appl. Environ. Microbiol.*, 73:5261–5267, Aug 2007.

[29] M. Wu and J. A. Eisen. A simple, fast, and accurate method of phylogenomic inference. *Genome Biol.*, 9:R151, Oct 2008.

[30] P. Yarza, W. Ludwig, et al. Update of the All-Species Living Tree Project based on 16S and 23S rRNA sequence analyses. *Syst. Appl. Microbiol.*, 33:291–299, Oct 2010.

[31] P. Yarza, M. Richter, et al. The All-Species Living Tree project: a 16S rRNA-based phylogenetic tree of all sequenced type strains. *Syst. Appl. Microbiol.*, 31:241–250, Sep 2008.

[32] F. Yue, J. Shi, and J. Tang. Simultaneous phylogeny reconstruction and multiple sequence alignment. *BMC Bioinformatics*, 10(Suppl 1):S11, 2009.