Building Large Phylogenetic Trees in Log-Linear Time

Slava Brover*[1]

[1]National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Bethesda, MD 20894, USA.

November 21, 2022

Abstract

A practical scalable algorithm to build large phylogenetic trees is described. The trees are based on dissimilarities between objects and use the least squares optimization criterion. Tree leaves and arcs have fitting criteria as the measures of correctness. The main idea of fast tree optimization is to extract subgraphs out of the whole tree, optimize them as separate small trees and then embed them back into the whole tree. A large tree is built by inserting new objects into an existing tree and optimizing the whole tree incrementally. Dissimilarities are not computed for all pairs of objects, but for the pairs chosen by the algorithm preferring pairs closer in the tree. Different types of object outliers are identified and removed from the tree. The running time of the incremental tree building is $O(n \log^4 n)$ and space is $O(n \log^3 n)$, where n is the number of objects. The software is provided at `github`. The global phylogenetic trees of GenBank bacteria, fungi, protist genomes, and prokaryotic 16S rRNA and fungal ITS rRNA are provided.

Keywords— Phylogeny, Distance tree, Least squares, Phylogenetic placement, Bacteria, Fungi, Protists, 16S rRNA, ITS rRNA.

Contents

*E-mail: brovervv@ncbi.nlm.nih.gov

1 Introduction

When a phylogenetic tree of genomes or rRNA sequences of a given taxonomic group is built, usually the tree building is preceded by the selection of representative genomes or sequences. But it may be important to build a phylogenetic tree for all available genomes or rRNA sequences. Such a tree will be referred to as a *global* phylogenetic tree.

A global phylogenetic tree can be used:

- to verify the correctness of taxonomic assignment of genomes or rRNA sequences, especially of type strains;
- to select representative genomes at a given level of granularity;
- to identify outbreaks in epidemiological data;
- to quickly produce trees for any desired subsets of objects by removing all extra objects for pilot studies;
- to quickly find genomes close to a specific genome.

Since there is no publicly available tool to build a global phylogenetic tree for, say, all GenBank bacterial genomes, it was undertaken to create such a tool.

The optimization criterion of a global tree was chosen to be the least squares distance criterion because

- the current MLE phylogenetic methods are computationally intensive and cannot compare genomes above and below species level in the same data set;
- the minimum evolution method is not a maximum likelihood method in the statistical sense.

The currently available least squares distance tools PHYLIP [8] and PAUP [23] are too slow even for small trees. Therefore, a new tool, `makeDistTree`, was created for fast building of small phylogenetic trees, and a system of tools with the main script `distTree_inc.sh` was created to build a phylogenetic tree for a large set of genomes or rRNA sequences using the dissimilarities between them.

By the least squares optimization criterion, the tree topology and arc lengths of the tree are optimized so that the tree distances should fit the dissimilarities. A weighted least squares criterion is used as the fitness measure.

Genomes may not fit the tree well due to being hybrid, contaminated or due to sequencing problems. They are identified as outliers and removed from the tree.

The main problem with building a distance tree for a large set of genomes or sequences is that computing all pairwise dissimilarities is not feasible. This problem is solved by building an initial tree only for a subset of genomes or sequences with a complete dissimilarity matrix, and then iteratively incrementing and optimizing the tree, where the dissimilarities are not computed for all pairs of genomes or sequences, but for the pairs chosen by the algorithm. The achieved running time of this algorithm is log-linear.

The incremental algorithm allows fast adding of new genomes or sequences to an existing tree.

The provided tools allow building and maintaining a phylogenetic tree, for example, for all *Escherichia coli* genomes in a hospital database where the number of genomes is of the order of hundreds thousands.

This paper describes the algorithms used for the incremental distance tree building.

Examples of large distance trees which have been built for major microbial applications at the kingdom taxonomic level and above are provided.

2 Basic definitions

The constants defined in this Section will used in the whole paper. The constants introduced in the next Sections will be used only in those Sections.

Let S be a set of *objects*. Each object has a unique name.

$$n = |S|.$$

2.1 Dissimilarity

Let $d : S, S \to \mathbb{R}$ be a *dissimilarity* function, which may not be defined for all pairs of objects. It is required that $d_{xy} = d_{yx}$ and $d_{xx} = 0$.

$$P = \{\{x, y\} : x \in S \ \& \ y \ \in S \ \& \ x \neq y \ \& \ d_{xy} \text{ is defined}\}.$$
$$p = |P|.$$

Objects x and y having $d_{xy} \leq 0$ are *indiscernible*, otherwise they are *discernible*. Negative dissimilarities can be between interior tree nodes, see Section 5.1. The *indiscernibility relation* is

$$\{\{x, y\} \subseteq P : d_{xy} \leq 0\}.$$

Its transitive closure is an equivalence relation. Let $\mathfrak{I}$ be the set of equivalence classes of this equivalence relation.

The *neighbors* of an object $x \in S$ are the objects of S having a dissimilarity with x:

$$N(x) = \{y \in S : \{x, y\} \in P\}.$$

$$\sum_{x \in S} |N(x)| = 2p.$$
$$\text{mean } \{|N(x)| : x \in S\} = 2 \ p/n.$$

A *hybrid triangle* is a triple of objects c, p_1 and p_2 for which the triangle inequality is violated:

$$h = \frac{d_{p_1 p_2}}{d_{p_1 c} + d_{p_2 c}} > 1.$$

The object c is the hybrid triangle *child*, and the objects p_1 and p_2 are the hybrid triangle *parents*. The value h is the *hybridness* of the hybrid triangle. The child and parents of a specific hybrid triangle tr are indicated by $tr.c$, $tr.p_1$ and $tr.p_2$. A non-empty subset of the objects of a hybrid triangle tr is the set of *hybrid triangle outliers* denoted by $H(tr)$.

Let $\mathfrak{T}$ be a set of hybrid triangles. For $tr \in \mathfrak{T}$

$$Tr_c(tr) = \{x \in \mathfrak{T} : x.p_1 = tr.p_1, x.p_2 = tr.p_2\},$$
$$Tr_{p_1}(tr) = \{x \in \mathfrak{T} : x.c = tr.c, x.p_2 = tr.p_2\},$$
$$Tr_{p_2}(tr) = \{x \in \mathfrak{T} : x.c = tr.c, x.p_1 = tr.p_1\}.$$

2.2 Tree

An *undirected tree* T is a connected undirected graph with no cycles.

A *directed tree* is an undirected tree where one of the nodes is a *root*, and all arcs are directed to the root. In the sequel a directed tree will be referred to as a *tree*.

For a directed arc (x, y), node y is the *parent* of node x, and node x is a *child* of node y, and the arc is directed from x to y:

$$y = parent(x).$$
$$x \in Children(y).$$

The node $parent(root)$ does not exist.

$$Descendants(x) = \{x\} \cup \bigcup_{z \in Children(x)} Descendants(z).$$

$$Ancestors(x) = \{x\} \cup \begin{cases} \emptyset, & \text{if } x = root, \\ Ancestors(parent(x)), & \text{else.} \end{cases}$$

The nodes with degree 1 are *leaves*, denoted by $Leaves(T)$, and the other nodes of T are *interior* nodes.

$Path(x, y)$ is the set of arcs on the unique undirected path from node x to node y.

Arcs j are assigned the *length* $l_j > 0$.

The *tree distance* between nodes x and y

$$l_{xy} = \sum_{j \in Path(x,y)} l_j.$$

The tree distance satisfies the distance axioms.

The *topology* of a tree is the interior nodes and the arcs of the tree, disregarding arc lengths.

A connected subgraph of a tree is a tree.

The *boundary* of a connected tree subgraph A, denoted by $Boundary(A)$, is a subset $B \subseteq A$ such that all nodes in B have degree 1 in A.

The nodes of the tree centered at node x with radius k

$$Area(x, k) = \{y : |Path(x, y)| \leq k\}.$$

The nodes of $Area(x, k)$ make up a connected subgraph.

$$Boundary(x, k) = Boundary(Area(x, k)).$$

$$depth(x) = |Path(x, root)|.$$
$$height(x) = \max_{y \in Descendants(x)} |Path(x, y)|.$$

At each arc j the list of object pairs $\{x, y\}$ is stored:

$$Pairs(j) = \{\{x, y\} \in P : j \in Path(x, y)\}.$$

If x is a leaf then

$$Pairs(x, parent(x)) = \{(x, y) : y \in N(x)\}.$$
$$\sum_{j \text{ is a leaf arc}} |Pairs(j)| = 2\ p.$$

A *genogroup* of *barrier* β is a minimal subset G of S such that

$$\forall x \in G\ \forall y \in S : (l_{xy} \leq \beta \Rightarrow y \in G).$$

All genogroups of the same barrier make up a partition of S.

3 Assumptions

1. All nodes of a tree have degree 1 or at least 3, except the root. The root can have degree 2.
2. The degree of an interior node is at most 500. Practically it is close to 3, but can be higher if interior arcs are deleted in the middle of optimization.
3. $height(root)$ is $O(\log n)$. In other words, the tree should be balanced. This holds for evolutionary data with molecular clock.
4. The computation of dissimilarities is an external process relative to the provided tools. It is required that the computation of d_{xy} should have time $O(1)$. The dissimilarities used in this paper satisfy this requirement, see Appendix C.
5. The number of objects n is bounded by a very large number: $\log\log n = O(1)$.
6. Allocating memory of p bits and setting all bits to 0 takes time $O(1)$.

Consequences from the assumptions:

- The number of the nodes of the tree is $O(n)$ and the number of the arcs is $O(n)$.
- For a constant k, $|Area(x,k)| = O(1)$.
- The time to compute $Path(x,y)$ is $O(\log n)$ and $|Path(x,y)| = O(\log n)$.
- The cumulative time to compute $Pairs(j)$ is $O(p \log n)$.

$$\sum_{j \text{ is not a leaf arc}} |Pairs(j)| = O(p \log n). \tag{1}$$

Average $|Pairs(j)|$ is $O(p/n \log n)$.

4 Least squares model

4.1 Model and criterion

A *tree model* of a dissimilarity d is defined as a tree T, such that $Leaves(T) = S$.

The *absolute criterion* of a tree model is the weighted least squares fitness function defined as

$$|\epsilon|^2 = \sum_{\{x,y\} \in P} w_{xy} \ (d_{xy} - l_{xy})^2,$$

where the *weights* are defined as

$$w_{xy} = \frac{1}{v(l_{xy})}, \tag{2}$$

which is a classic approach, see [7]. The value $v(l)$ is the *dissimilarity variance*, see Section 4.3.

The *relative criterion* is defined as

$$\sqrt{\frac{|\epsilon|^2}{\sum_{\{x,y\} \in P} w_{xy} \ d_{xy}^2}}.$$

The absolute criterion depends on the dissimilarity measurement scale, whereas the relative criterion does not. The relative criterion is between 0 and 1:

If the dissimilarities d_{xy} are the realizations of independent random variables

$$D_{xy} \sim Normal \ (l_{xy}, \alpha \ v(l_{xy})) \tag{3}$$

for $\alpha > 0$, then the likelihood maximization is equivalent to the minimization of

$$\frac{|\epsilon|^2}{\alpha} + \sum_{\{x,y\} \in P} \ln v(l_{xy}) + p \ln \alpha,$$

which is minimized at

$$\alpha = \frac{|\epsilon|^2}{p},$$

and, therefore, is equivalent to the minimization of

$$\ln \frac{|\epsilon|^2}{p} + \frac{1}{p} \sum_{\{x,y\} \in P} \ln v(l_{xy}). \tag{4}$$

Criterion (4) is minimized approximately by iteratively computing weights and minimizing the absolute criterion.

If the variances of D_{xy} are too large, which is often the case with RNAs, then $v(d_{xy})$ can be used instead of $v(l_{xy})$ in (2) as better estimates of the dissimilarity variances. In this case the minimization of the absolute criterion is equivalent to the likelihood maximization.

Given (3),

$$\forall \{x,y\} \in P : w_{xy} \ (d_{xy} - l_{xy})^2 \sim \alpha \ \chi^2(1).$$

4.2 Object and arc criteria

The absolute criterion can be split per object or arc, and normalized object or arc criteria can be defined based on their distributions. This allows computing the outlier p-values and the degrees of certainty of leaves and arcs in the tree.

The *absolute object criterion* of object x are the components of $|\epsilon|^2$ involving only the object x:

$$|\epsilon_x|^2 = \sum_{y \in N(x)} w_{xy} \ (d_{xy} - l_{xy})^2.$$

$$\sum_{x \in S} |\epsilon_x|^2 = 2|\epsilon|^2.$$

Since var $\chi^2(1) = 2$, the *normalized object criterion* of object x

$$\frac{|\epsilon_x|^2/\alpha - |N(x)|}{\sqrt{2 \ |N(x)|}} \sim Normal(0,1)$$

for $|N(x)| \ll p$ and large p, where the value α is a parameter of the probabilistic model (3).

The *deformation* of dissimilarity d_{xy} is defined as

$$def(x,y) = \frac{(d_{xy} - l_{xy})^2}{\min(d_{xy}, l_{xy})},$$

which has approximately the same distribution as $w_{xy} \ (d_{xy} - l_{xy})^2$.

The *object deformation* of object x is defined as

$$def(x) = \max_{y \in N(x)} def(x,y).$$

The *relative object deformation* of object x is defined as

$$\frac{def(x)}{\text{mean } \{def(y,z) : \{y,z\} \in P\}} \sim \max\{Y_i : 1 \le i \le \text{mean } \{|N(y)| : y \in S\}\}, \tag{5}$$

where Y_i are i.i.d. random variables distributed as $\chi^2(1)$.

The *absolute arc criterion* of arc j are the components of $|\epsilon|^2$ involving only the arc j multiplied by the proportion of l_j in the distances:

$$|\epsilon_j|^2 = \sum_{\{x,y\} \in Pairs(j)} w_{xy}(d_{xy} - l_{xy})^2 \ \frac{l_j}{l_{xy}}.$$

$$\sum_{j \in T} |\epsilon_j|^2 = |\epsilon|^2.$$

The *absolute arc error density* of arc j is absolute arc criterion divided by the arc length: $|\epsilon_j|^2/l_j$.

The *normalized arc error density* of arc j is defined as

$$\frac{|\epsilon_j|^2/(\alpha \ l_j) - \sum_{\{x,y\} \in Pairs(j)} 1/l_{xy}}{\sqrt{2 \sum_{\{x,y\} \in Pairs(j)} 1/l_{xy}^2}} = \frac{\sum_{\{x,y\} \in Pairs(j)} (w_{xy}(d_{xy} - l_{xy})^2/\alpha - 1)/l_{xy}}{\sqrt{2 \sum_{\{x,y\} \in Pairs(j)} 1/l_{xy}^2}} \sim Normal(0,1)$$

for $|Pairs(j)| \ll p$ and large p.

4.3 Dissimilarity variance

An increasing function $v(l) > 0$, where $l > 0$, is a *dissimilarity variance function*.

The dissimilarity variance functions used in the provided tools are in Table 1:

Function name	$v(l)$
Exponential	e^l
Linear-exponential	$e^l - 1$
Power (with parameter x)	l^x

Table 1: Dissimilarity variance functions

The linear-exponential function is approximately linear if $l \approx 0$. Its model is in Appendix A.

For power dissimilarity variance function multiplying l by a positive constant multiplies $|\epsilon|^2$ by a positive constant, which preserves the tree topology. However, for the exponential and linear-exponential dissimilarity variance functions multiplying l by a positive constant can change the tree topology.

4.4 Avoiding zero dissimilarity variance

For some pairs of objects their dissimilarity variances can be 0, which makes their weights infinite, and, therefore, $|\epsilon|^2$ infinite.

For indiscernible objects this occurs always if the dissimilarity variance is $v(d_{xy})$, and may happen if the dissimilarity variance is $v(l_{xy})$. Therefore, in this work the pairs of indiscernible objects are removed from P, and therefore, from $|\epsilon|^2$, and for indiscernible objects a fixed topology is created: for each non-singleton indiscernibility class $I \in \mathfrak{I}$ a subtree of depth 1 is created whose set of leaves is I and whose arcs have length 0. Then the tree model is optimized only for discernible objects and

$$|\epsilon|^2 \geq \sum_{\{x,y\} \in P \,:\, x,y \in I, I \in \mathfrak{I}} w_{xy}\, d_{xy}^2.$$

If the dissimilarity variance is $v(l_{xy})$, then each time before the weights w_{xy} are computed, the arc of each leaf x is extended by λ_x, where

$$\lambda_x = \begin{cases} 0, & \text{if } Z_x = \emptyset, \\ \frac{1}{2} \min Z_x, & \text{else,} \end{cases}$$

$$Z_x = \{d_{xy} : \{x, y\} \in P \;\&\; l_{xy} = 0 \;\&\; d_{xy} > 0\}.$$

which changes $l_{xy} = 0$ to $0 < l_{xy} \leq d_{xy}$.

5 Tree optimization algorithms

Finding a globally optimal distance tree model is NP-hard [6].

The algorithms in this Section find a locally optimal tree model.

These algorithms allow arcs with zero length. Such arcs and their child nodes are deleted.

5.1 Optimization of a connected subgraph

Let A be a connected subgraph of a tree model T.

Let $q, r \in Boundary(A)$, such that $q \neq r$. Let T_q be the maximal subgraph of T intersecting with A in $\{q\}$, and let T_r be the maximal subgraph of T intersecting with A in $\{r\}$. The graphs T_q and T_r are trees. Let $Q = Leaves(T_q) \setminus \{q\}$, and $R = Leaves(T_r) \setminus \{r\}$. $Q \subseteq S$ and $R \subseteq S$.

Consider the components of the absolute criterion $|\epsilon|^2$, where $x \in Q$ and $y \in R$, then

$$d_{xy} - l_{xy} = d_{xy} - (l_{xq} + l_{qr} + l_{ry}) = (d_{xy} - l_{xq} - l_{ry}) - l_{qr}.$$

Let vector $\mathbf{d}_{qr}$ indexed by x and y be defined as

$$\mathbf{d}_{qr}[x, y] = d_{xy} - l_{xq} - l_{ry},$$

then

$$\sum_{x\in Q,y\in R} w_{xy}\ (d_{xy}-l_{xy})^2 = \sum_{x\in Q,y\in R} w_{xy}\ (\mathbf{d}_{qr}[x,y]-l_{qr})^2 = w_{qr}\left(\bar{\mathbf{d}}_{qr}-l_{qr}\right)^2 + \sum_{x\in Q,y\in R} w_{xy}\ (\mathbf{d}_{qr}[x,y]-\bar{\mathbf{d}}_{qr})^2,$$

where

$$\bar{\mathbf{d}}_{qr} = \frac{\sum_{x\in Q,y\in R} w_{xy}\ \mathbf{d}_{qr}[x,y]}{\sum_{x\in Q,y\in R} w_{xy}}$$

and

$$w_{qr} = \sum_{x\in Q,y\in R} w_{xy}.$$

It means that optimizing the subgraph A with dissimilarities $\bar{\mathbf{d}}_{qr}$ and weights w_{qr} as a separate tree will optimize the whole tree: the absolute criterion of the whole tree will decrease by the same amount as the absolute criterion of A.

Thus the optimization of subgraph A consists of the following steps:

1. Compute $\bar{\mathbf{d}}_{qr}$ and w_{qr} for all $q,r \in Boundary(A)$;
2. Create a new tree T_A isomorphic to A, mapping $Leaves(T_A)$ to $Boundary(A)$;
3. Optimize T_A with dissimilarities $\bar{\mathbf{d}}_{qr}$ and weights w_{qr};
4. Make A to be isomorphic to T_A;
5. Update $Pairs(j)$ for $j \in A$.

At step 3, if the number of interior nodes of T_A is more than 3 then local topology rearrangements are used to optimize the topology of T_A. Besides that, subgraph optimization can be applied recursively to T_A.

Step 1 requires finding the set of all pairs $\{x,y\} \in P$ whose $Path(x,y)$ intersects with A. This can done by iterating over $Pairs(j)$, where $j \in Boundary(A)$, and computing $Path(q,r)$ for each pair $\{x,y\}$ of this set. Therefore, the average time to do this is $O(|A|\ p/n\ \log n\ \log|A|)$.

Step 5 requires the allocation of p bits and setting them to 0. Using Assumption 6, the average time of this step is $O(|A|\ p/n\ \log n)$.

If $|A| = O(1)$, then steps 2-4 have time $O(1)$ and optimizing A has time $O(p/n\ \log n)$.

5.2 Optimization by covering subgraphs

Cover a tree by connected subgraphs $Area(x,k)$, where

$$k = \max\ \{k' : 1 \le k' \le 5,\ |Area(x,k')| \le 500\},$$

and optimize all subgraphs by the algorithm of Section 5.1.

Since each subgraph has size $O(1)$, the algorithm has time $O(p \log n)$.

5.3 Best placement of an object in a tree

For an object $a \in S$ the point x on a tree must be found so that if a is attached to the tree at point x the absolute criterion has a local minimum.

The search will use the algorithm to find the best position of on object on an arc.

Let the vector

$$\mathbf{d} = (d_{ai} : i \in N(a)).$$

It will be assumed that the weights w_{ai}, where $i \in N(a)$, are fixed.

For an arc j, let

$$S_j = \{i \in N(a) : j \in Path(a,i)\}.$$

Consider an arc (b,c), where $c = parent(b)$, as a line segment $[b,c]$. The node b of this arc will be referred to as the *base*. Let point $x \in [b,c]$, meaning that arc (b,c) is replaced by two consecutive arcs (b,x) and (x,c). The object a is attached by the arc (a,x).

$$S_{(b,x)} \cup S_{(x,c)} = N(a),$$
$$S_{(b,x)} \cap S_{(x,c)} = \emptyset.$$

Assume $S_{(b,x)} \neq \emptyset$, which is equivalent to $N(a) \cap Descendants(b) \neq \emptyset$.

Let $\mathbf{u} \in \{-1,1\}^{|N(a)|}$ be defined as

$$u_i = \begin{cases} 1, & \text{if } i \in S_{(b,x)}, \\ -1, & \text{if } i \in S_{(x,c)}. \end{cases}$$

The computation of $\mathbf{u}$ has time $O(|N(a)|\ \log n)$. This is the slowest step of this procedure.

For $y \in S$ let

$$\mathbf{l}(y) = (l_{yi} : i \in N(a))$$

and let

$$\delta(y) = \mathbf{d} - \mathbf{l}(y),$$

then attaching leaf a at the point $x \in [b,c]$ increases the absolute criterion by

$$|\epsilon_a|^2(b) = \sum_{i \in N(a)} w_{ai} \left(d_{ai} - [l_{bi} + l_{bx}u_i + l_{ax}]\right)^2 = \sum_{i \in N(a)} w_{ai} \left(\delta_i(b) - l_{bx}u_i - l_{ax}\right)^2.$$

This is a convex function of l_{bx} and l_{ax}.

Let

$$\bar{\delta}(y) = \frac{\sum_{i \in N(a)} w_{ai}\ \delta_i(y)}{\sum_{i \in N(a)} w_{ai}},$$

$$\bar{\mathbf{u}} = \frac{\sum_{i \in N(a)} w_{ai}\ u_i}{\sum_{i \in N(a)} w_{ai}}.$$

If $S_{(x,c)} \neq \emptyset$ then var $\mathbf{u} > 0$ and $|\epsilon_a|^2(b)$ attains minimum for unconstrained l_{bx} and l_{ax} at

$$l_{bx} = \frac{\operatorname{cov}(\delta(b), \mathbf{u})}{\operatorname{var}\ \mathbf{u}} = \frac{\sum_{i \in N(a)} w_{ai}\ (\delta_i(b) - \bar{\delta}(b))\ (u_i - \bar{\mathbf{u}})}{\sum_{i \in N(a)} w_{ai}\ (u_i - \bar{\mathbf{u}})^2},$$

$$l_{ax} = \bar{\delta}(b) - l_{bx}\ \bar{\mathbf{u}}.$$

If the constraints $0 \leq l_{bx} \leq l_{bc}$ and $l_{ax} \geq 0$ are not satisfied, then $|\epsilon_a|^2(b)$ attains a global minimum for $x \in [b,c]$ in one of the cases:

- $x = a$: $l_{ax} = 0$, $l_{bx} = \min\left\{\max\left\{\frac{\sum_{i \in N(a)} w_{ai}\ \delta_i(b)\ u_i}{\sum_{i \in N(a)} w_{ai}}, 0\right\}, l_{bc}\right\}$;
- $x = b$: $l_{ax} = \max\left\{\bar{\delta}(b), 0\right\}$, $l_{bx} = 0$;
- $x = c$: $l_{ax} = \max\left\{\bar{\delta}(c), 0\right\}$, $l_{bx} = l_{bc}$.

If $S_{(x,c)} = \emptyset$ then a global minimum for $x \in [b,c]$ is attained at $x = b$, $l_{ax} = \max\left\{\bar{\delta}(b), 0\right\}$ and $l_{bx} = 0$.

Now the algorithm of a best placement of an object in the tree consists in finding

$$\arg\min_{b \in B} |\epsilon_a|^2(b)$$

by examining all arcs of the tree restricted to B, where

$$B = \{b : N(a) \cap Descendants(b) \neq \emptyset\}$$

is the set of possible base nodes.

$$|B| = O(|N(a)|).$$

The running time of the algorithm is $O(|N(a)|^2\ \log n)$.

When descending the tree these equations help fast computation:

$$\mathbf{l}(c) = \mathbf{l}(b) + l_{bc}\ \mathbf{u},$$

$$\delta(c) = \delta(b) - l_{bc}\ \mathbf{u}.$$

5.4 Optimization by reinsertion

Let a be an interior node of T. Consider a subgraph A of T such that

$$Leaves(A) = \{a\} \cup Leaves(T) \setminus Descendants(a).$$

Delete the leaf a from A and find a best placement of a in A. That will find a best position of the subtree rooted at a in T.

The algorithm:

1. $B = \emptyset$;
2. For each node a of T:
   (a) Make subgraph A, remove a from A, compute $\mathbf{d} = (d_{ai} : i \in N(a))$;
   (b) Limit $|N(a)|$ by $k = O(\log n)$ objects with minimum d_{ai}, where $i \in N(a)$; in this work $k = 700\lceil \ln n \rceil$;
   (c) Find a best placement of a in A, see Section 5.3;
   (d) If the absolute criterion of a improves, add the best position of a to B;
3. For each best position in B: apply it to T, and if the absolute criterion of T does not improve revert the change.

Step 2a finds $Leaves(A)$ and traverses $Pairs(a)$ which has time $O(n + p\log(n)/n)$ and cumulative time $O(n \log n + (p \log(n)/n) \times n) = O(p \log n)$.

Step 2c has time $O(\log^3 n)$.

Step 2 has time $O((p + n \log^2 n) \log n)$.

Step 3 updates $Pairs(j)$ for $j \in A'$, where A' is the path from the old position of node a to its new best position, which according to Section 5.1 has time $O(|A|\ p/n \log(n) \log |A|) = O(p/n \log^2 n)$ because $|A| = O(\log n)$. Therefore, this step has time $O(|B|\ p/n \log^2 n)$.

The whole algorithm has time $O((p + n \log^2 n + |B|\ p/n \log n) \log n)$.

6 Outlier objects

The objects which do not fit the tree well and thus may damage the tree in their neighborhood are referred to as *outliers* and are not used in the tree.

Different types of outliers are described below.

The fitness and genogroup outliers may be undetected hybrid outliers.

6.1 Quality outliers

The *quality outliers* are the objects which have abnormal attributes or for which dissimilarities cannot be computed. Such objects are identified before the tree building and they are not included into S, see Appendix C.1.

6.2 Dissimilarity outliers

The *dissimilarity outliers*, consisting of *alien outliers* and *incomparable outliers*, are the objects which cannot be added to the tree because their dissimilarities cannot be computed, see Section 8.3.

6.3 Fitness outliers

The fitness outliers are the objects with too large fitness-related criteria.

The *criterion outliers* are the objects with normalized object criteria exceeding the right-tail normal distribution outlier threshold for e-value 10^{-6}, see Appendix B. (An e-value is a p-value multiplied by the size of a sample.)

The *deformation outliers* are the objects with relative object deformations exceeding e-value 10^{-10} according to distribution (5).

The fitness outliers may be caused by the tree deformation due to other outliers located close in the tree. Therefore, it is difficult to find all fitness outliers without false positive and false negative errors, but for a small k, the k outliers with the largest criteria are likely to be true.

6.4 Genogroup outliers

Genogroups are found for genomes using a dissimilarity barrier specific for a tree, see Appendix C. Finding genogroups has time $O(n \log n)$. For each genogroup with the number of genomes at least 10, the means and standard deviations for the genogroup attributes are computed. The genomes where one of the attributes has e-value below 0.01 are *genogroup outliers*.

6.5 Hybrid outliers

The *hybrid outliers* are the objects which cause the violation of the triangle inequality. The hybrid triangles tr with hybridness above a *minimum hybridness* threshold are found, and their members $H(tr)$ are identified as hybrid outliers.

Hybrid outliers may be the product of biological hybridization, phylogenetically distant genomes containing the same mobilome, or sequencing contamination artifacts.

It is impossible to determine exactly which objects of a hybrid triangle tr are hybrid outliers using only dissimilarity data. Therefore, the set $H(tr)$ is a heuristic, and hybrid outliers require a review outside of the tree building process and a reclassification into quality outliers and non-outliers.

Some examples of the structure of the members of hybrid triangles are in Table 2. Hybrid triangle tr_1 is a classic example where the child c is a biological hybrid of the parents p_1 and p_2. In hybrid triangle tr_2 the parts A_1 and A_2 are random halves of two random genomes of the same species sharing thus a quarter of species loci. The cases tr_1 and tr_2 can be approximately differentiated by the values: $|Tr_c(tr)|$, $|Tr_{p_1}(tr)|$, $|Tr_{p_2}(tr)|$, $|e_c|^2$, $|e_{p_1}|^2$, $|e_{p_2}|^2$.

Hybrid triangle tr	p_1	p_2	c	$H(tr)$
tr_1	AA	BB	AB	$\{c\}$
tr_2	A_1B	A_2C	AA	$\{p_1, p_2\}$

Table 2: Examples of the structure of the members of hybrid triangles

Since examining all triangles takes time $O(n^3)$, only a subset of hybrid triangles is found with running time artificially limited to $O(p^2/n)$.

Though the existence of the hybrid triangles is independent of the tree, the search of them uses heuristics based on fitness outlier objects and outlier dissimilarities, and therefore, requires a tree to be performed.

7 Building a small tree

Given a dissimilarity matrix for a set of objects S with possibly missing dissimilarities, a tree is built by these steps:

1. Build a tree model for S by the nearest neighbor-joining algorithm, [22].
2. Set $w_{xy} = 1/v(d_{xy})$ as an approximation.
3. Arc length optimization:
   (a) Multiply all l_{xy} and arc lengths by
   $$\frac{\sum_{x,y} w_{xy}\ l_{xy}\ d_{xy}}{\sum_{x,y} w_{xy}\ l_{xy}^2}.$$
   (b) Do 10 times: optimize each arc (x, y) by the subgraph optimization using subgraphs $A = \{x, y\}$, see Section 5.1.
   (c) Optimize each node x by the subgraph optimization using subgraphs $A = Area(x, 1)$, see Section 5.1.
4. Iterate `iter_max` times or until the criterion stops improving:
   (a) Set w_{xy} by (2);
   (b) Optimize by covering subgraphs, see Section 5.2.

The test files on `github` include the file `Saccharomyces.dm` containing 500 genomes of the genus *Saccharomyces*. As a small tree is built for this data with `iter_max = 20`, the decrease of the absolute criterion is shown in Table 3.

Step	Optimization method	Absolute criterion
1	Nearest neighbor joining	491.29
3a	All arc lengths	432.01
3b	Individual arc lengths	261.44
3c	Arcs of nodes	257.56
4, iteration 1	Covering subgraphs	205.60
4, iteration 2	Covering subgraphs	202.31
4, iteration 3	Covering subgraphs	202.20
4, iteration 4	Covering subgraphs	202.13
4, iteration 5	Covering subgraphs	202.13

Table 3: Typical improvement of the absolute criterion after different optimization steps

Since step 1 is slowest, the running time is $O\left(n^3\right)$. Space is $O\left(n^2\right)$.

8 Building a large tree

Let $\mathfrak{S}$ be a *reservoir* of objects, and S be a set of objects for which an initial tree has been built. After that a large tree is built incrementally by moving objects from $\mathfrak{S}$ to S, computing dissimilarities for the added objects, optimizing the tree for S and removing outlier objects from S.

Dissimilarities are not computed for all pairs of objects, but for the pairs chosen by the algorithm preferring pairs closer in the tree.

8.1 Initial tree

The initial tree is computed by these steps:

1. Select a random subset $S \subset \mathfrak{S}$, such that $|S| = 3000$; remove S from $\mathfrak{S}$;
2. Compute a complete dissimilarity matrix for S;
3. Build a small tree by the algorithm of Section 7 with `iter_max = 20`.

8.2 Necessary neighbors

Let a *representative object* for a tree node x and seed $s \geq 0$ be a function

$$repr(x, s) \in Descendants(x) \cap Leaves(T).$$

If $s > 0$ then this function is deterministic, if $s = 0$ then this function is random. Computing this function has time $O(\log n)$.

Let the set of *necessary neighbors* of an object a be defined as

$$N_n(a) = \{repr(x, s) : y \in Ancestors(a), x \in Boundary(y, k_1)\} \cup \{repr(x, s) : x \in Boundary(a, k_2)\},$$

where n is the number of objects in the tree, s is a function of the name of a (e.g., a hash code) and $k_2 \geq k_1 \geq 1$. In this work $k_1 = 3$ and $k_2 = 10$.

$$|N_n(a)| = O(\log n).$$

Computing $N_n(a)$ has time $O(\log^2 n)$.

The invariant is maintained: $N_n(a) \subseteq N(a)$.

8.3 Adding a new object to the tree

Let $N_0(a)$ be the *initial set of neighbors* of a new object a. If $N_0(a) = \emptyset$ then the object a is an *alien* outlier. It is required that $|N_0(a)| = O(1)$ and the time to compute $N_0(a)$ should be $O(\log^4 n)$. In this work $|N_0(a)| \leq 100$. The computation of $N_0(a)$ for genomes and rRNA sequences is described in Appendix D.

The algorithm of adding a new object a to the tree:

1. $N = N_0(a)$;
2. Iterate $\lfloor \ln n + 3 \rfloor$ times:
   (a) Compute dissimilarities d_{ax}, where $x \in N$;
   (b) Set $w_{ax} = 1/v(d_{ax})$ as approximation.
   (c) For $B = \{b : N \cap Descendants(b) \neq \emptyset\}$ find a best placement of a in the tree, see Section 5.3:
   $$b^* = \arg\min_{b \in B} |\epsilon_a|^2(b).$$
   (d) Compute $N_n(a)$, which equals $N_n(b^*)$;
   (e) If $N_n(a) \subseteq N$ then stop;
   (f) $N = N \cup N_n(a)$;

If there is $x \in N$ such that d_{ax} cannot be computed then a is a *incomparable* outlier.

When the algorithm stops and a is not an outlier, $N = N(a)$ and $|N(a)| = O(\log^2 n)$. The running time is $O(\log n \times |N(a)|^2 \ \log n) = O(\log^4 n)$.

In practice, if $N_0(a)$ and the dissimilarities are accurate enough and a is not a hybrid the algorithm makes 3 iterations, $|N(a)| = O(\log n)$ and the running time is $O(\log^2 n)$.

8.4 Iterative tree extension

After an initial tree has been built, see Section 8.1, the tree incrementation is done.

Let $r > 1$ be the *incrementation rate*. In this work $r = 1.015$.

1. $G = \emptyset$; / * genogroup outliers * /
2. while $\mathfrak{S} \neq \emptyset$ do:
   (a) Select a random subset of objects $New \subseteq \mathfrak{S}$ such that $|New| = \min\{(r-1)|S|, |\mathfrak{S}|\}$;
   (b) For each $x \in New$ remove x from $\mathfrak{S}$ and add x to S and T, see Section 8.3;
   (c) Compute $Pairs(j)$ for each arc j;
   (d) Delete the objects in G from S and T, add them to genogroup outliers; $G = \emptyset$;
   (e) At each $10th$ iteration of step 2 do the optimization by reinsertion, see Section 5.4.
   (f) Do 2 times:
      i. Set w_{xy}.
      ii. Optimize by covering subgraphs, see Section 5.2.
      iii. Find hybrid outliers, see Section 6.5.
      For each outlier x:
      A. delete x from S and T;
      B. optimize the subgraph $Area(parent(x), 10)$, see Section 5.1;
   (g) Find criterion outliers, see Section 6.3; delete one worst criterion outlier from S and T;
   (h) Find deformation outliers, see Section 6.3; delete one worst deformation outlier from S and T;
   (i) Find root;
   (j) Find genogroup outliers, see Section 6.4, and store them in G;
   (k) For each $x \in Leaves(T)$:
      i. compute missing dissimilarities d_{xy} for $y \in N_n(x)$;
      ii. $N(x) = N(x) \cup N_n(x)$;
3. Delete the objects in G from S and T, add them to genogroup outliers;

4. Optimize 5 times by covering subgraphs optimization, see Section 5.2.

Steps 2c, 2e, 2(f)ii and 4 use 15 threads.

Let n_i and p_i be the tree size and the number of dissimilarities respectfully at iteration i of step 2, and let $n = |\mathfrak{S}|$ be reservoir size.

Step 2b has time $O(n_i \log^4 n_i)$.

Steps 2c has time $O(p_i \log n_i)$, see Section 3.

Step 2e has time $O((p_i + n_i \log^2 n_i + |B|\ p_i/n_i \log n_i) \log n_i) = O((n_i \log n_i + p_i) \log^2 n_i)$ because $|B| = O(n_i)$.

Steps 2(f)ii has time $O(p_i \log n_i)$.

Step 2(f)iii has time $O(p_i\ (p_i/n_i + \log n_i))$.

At step 2i the root is optimized so that the average path length from the root to the leaves weighted by the subtree length is minimized. This makes the root of the tree more stable. This step has time $O(n_i)$.

Step 2j has time $O(n_i \log n_i)$.

Since $p_i = O(n_i \log^2 n_i)$, see Section 8.2, step 2 has time $O(n_i \log^4 n_i)$, where the slowest steps are 2b, 2e and 2(f)iii.

Step 4 has time $O(p \log n)$.

When the algorithm stops, the number of iterations made by step 2 will have been $a = O(\log n)$. Since $n_i = O(n_1 r^i)$, $n_a = O(n)$, and $\sum_{i=1}^{a} r^i = O(r^a)$, the total time for step 2 and for the whole algorithm is

$$\sum_{i=1}^{a} O(n_i \log^4 n_i) = \sum_{i=1}^{a} O(n_1 r^i \log^4 (n_1 r^i)) = O(n_1 r^a a^4) = O(n a^4) = O(n \log^4 n).$$

For a tree with 200,000 objects the empirical running time of step 2 using 30 threads on a 2300 MHz computer is in Table 4:

Slowest substeps	Operation	Average time, hours
2b	Adding new objects	2.0
2c	Tree structure initialization	0.5
2e	Optimization by reinsertion	10.0 / 10
2(f)ii	Optimization by covering subgraphs	2×3.5
2(f)iii	Hybrid processing	2×2.0
2k	Adding necessary neighbors	0.5
Whole step 2	One incrementation	13.0

Table 4: Empirical running time of step 2 where 3,000 objects are added to a tree of 200,000 objects with 7 % of hybrids.

By (1) the space size of the algorithm is $O(p \log n) = O(n \log^3 n)$.

9 Results: GenBank global phylogenetic trees

The global phylogenetic trees have been built for GenBank bacterial, fungal and protist genomes, and universal rRNA sequences, see Table 5. The data used are described in Appendix C.

Taxon Object type	bacteria genome	fungi genome	protists genome	prokaryotes 16S rRNA	fungi ITS rRNA
Min. hybridness	1.20	1.25	1.15	-	-
Dissimilarity variance	$l^{4.5}$	$e^l - 1$	$e^l - 1$	$l^{5.5}$	$e^d - 1$
Species barrier	1.076	1.0125	1.46	0.334	1.213
Genogroup barrier	0.225	1.0125	-	-	-
Reservoir objects	473,130	5,080	635	21,137	11,025
Quality outliers	4,493	181	12	161	595
Dissimilarity outliers	534	0	0	1	2
Fitness outliers	292	54	11	-	-
Genogroup outliers	12,187	60	-	-	-
Hybrid outliers	39,863	27	31	-	-
All outliers, %	12.1%	6.3%	8.5%	0.8%	5.4%
Objects in the tree	415,761	4,758	581	20,975	10,428
Dissimilarities	4.4×10^8	3.1×10^6	1.7×10^5	9.6×10^6	7.0×10^6
Max. node degree	8	3	4	5	4
Release	11	14	3	4	8
Data collection date	10 Jan 2020	10 Jan 2020	22 Apr 2019	29 Jun 2019	21 Oct 2019
Base file name	`bacteria-R11`	`fungi-R15`	`protists-R3`	`prok-16S-R4`	`fungi-ITS-R8`

Table 5: Examples of global phylogenetic trees of GenBank assemblies and rRNA

In Table 5 the maximum node degree is computed only over nodes with discernible children.

In the fungal genome tree all *Saccharomyces pastorianus* genomes are identified as hybrid outliers [13].

The tree of bacterial genomes restricted to 11,970 type strains is `bacteria-R11t`,

Table 6 shows the distribution of bacterial genomes per outlier type for different sources of genomes. The surveillance, metagenome and single cell genomes are enriched in outliers which make up 76% of all outliers of the bacterial tree.

Source Reservoir objects	Surveillance 261,947	Metagenome 30,503	Single cell 1,342	Type strains 12,358
Quality outliers	0%	12%	42%	0%
Dissimilarity outliers	0%	1%	1%	0%
Fitness outliers	0%	0%	0%	0%
Genogroup outliers	2%	0%	0%	0%
Hybrid outliers	11%	11%	10%	2%
Objects in the tree	87%	76%	47%	97%

Table 6: Bacterial genomes of different sources: distribution per outlier type

The comparison of the obtained trees with other publicly available trees by quality is done by taxonomy miscongruence [4] using the NCBI taxonomy as of January 13, 2020, see Table 7.

Tree T_1 (better)	Objects in T_1	Tree T_2	Objects in T_2	Common objects	τ_1	τ_2	Quality difference
`bac120_r86.1.tree` [15]	27,372	`bacteria-R11`	415,761	22,479	37,420	41,158	9.1%
`bac120_r86.1.tree` [15]	27,372	`bacteria-R11t`	11,970	6,885	11,239	11,584	3.0%
ASTRAL [27]	10,575	`bacteria-R11`	415,761	8,422	17,686	20,264	12.7%
ASTRAL [27]	10.575	`bacteria-R11t`	11,970	3,411	7,416	7,726	4.0%
`bacteria-R11`	415,761	`microreact-project -10667ecoli-tree.nwk` [1]	10667	9734	3791	4637	18.2%
`fungi-R15`	4,413	`332_24080Gs_ time-calibrated_ phylogeny...` [21]	332	319	520	528	1.5%
`prok-16S-R4`	20,975	`LTPs132_SSU` [18]	13,867	12,501	16,494	19,110	13.7%

Table 7: Comparison of the quality of phylogenetic trees T_1 and T_2 by taxonomy miscongruences τ_1 and τ_2 on common objects. Quality difference is $1 - \tau_1/\tau_2$.

Each tree is represented:

- in the Newick format: file extension `.nw`;
- as an interactive HTML file: file extension `.html`, see Appendix E. The problem of studying large trees is that they are difficult to visualize. This problem is solved by assigning taxonomic names to some interior nodes. Then setting any interior node as a root by removing all ancestor nodes and expanding the tree until the named nodes are reached will define a small subtree which is easy to visualize.

10 Discussion

The method described in this paper is currently, to our best knowledge, the only practical method to build global phylogenetic trees of acceptable quality, and the program `makeDistTree`, which is a part of the software, is the fastest tool to build least squares distance trees.

A global phylogenetic tree will usually be less accurate than a tree of a small set of objects made by the maximum likelihood method on the basis of carefully constructed multiple alignment of sequences. However, it is required that the set of objects should be selected before the tree building.

The currently best bacterial genome tree is ASTRAL [27] made by summarizing RAxML gene trees by ASTRAL-MP [26]. The next best bacterial genome tree is GTDB [15] made by the maximum likelihood program FastTree v2.1.7 [17] which requires memory of size $O(n\sqrt{n})$. In these trees the contaminated genomes have been removed by the CheckM analysis [16] which itself uses a phylogenetic tree. This removal of contaminated genomes can artificially reduce the taxonomy miscongruence, therefore, the comparison by type strains is more reliable because type strains are less likely to be affected by the contamination removal. By type strain comparison the ASTRAL tree is only 4% better than the `bacteria-R11t` tree.

The QTree program [25] is very fast, but it does not implement subgraph optimization and implements only step 2b of the algorithm of Section 8.4.

The probabilistic model (3) is chosen because the absolute criterion $|\epsilon|^2$ is easy to optimize by the linear regression technique. However, there may be a better probabilistic model, for example, based on exponential distribution. Besides that, even a local optimum (4) is not reached.

Acknowledgements

I thank Barbara Robbertse, Terence Murphy, Richa Agarwala, Josh Cherry, Conrad Schoch, Bill Klimke, Arjun Prasad and David Lipman for the discussions of this project.

Funding

This research was supported by the National Center for Biotechnology Information of the National Library of Medicine (NLM), National Institutes of Health.

Availability of data and materials

The source code for the algorithms described in this paper is written in C++ and `bash` scripts for UNIX and is available at `https://github.com/ncbi/tree-tool`.

License: Freely available to the public for use.

The documentation for this code is available at `https://github.com/ncbi/tree-tool/wiki`.

The produced trees are available at `https://github.com/vbrover/Tree`.

Authors contributions

Not applicable.

Competing interest

Authors declare that they have no competing interests.

Consent for publication

Not applicable.

Ethics approval and consent to participate

Not applicable.

Appendices

A Linear-exponential function of dissimilarity variance

Suppose the dissimilarity d of a pair of objects is a function of probability p defined as:

$$d = f(p) = -\log p.$$

Then

$$p = e^{-d}.$$

$$\frac{\partial}{\partial p} f(p) = -\frac{1}{p} = -e^{d}.$$

Let P be a random variable, such that E $P = p$, and the dissimilarity be the random variable $D = f(P)$, then

$$D = f(p + P - p) \approx f(p) + \frac{\partial}{\partial p} f(p)\ (P - p) = f(p) - e^{d}(P - p),$$

$$\text{var } D \approx e^{2d} \text{ var } P.$$

If $P \sim 1/n \times Binomial(n, p)$ then

$$\text{var } P = 1/n \times p\ (1 - p) = 1/n \times \left(p - p^2\right) = 1/n \times \left(e^{-d} - e^{-2d}\right)$$

and

$$\text{var } D \approx 1/n \times \left(e^{d} - 1\right).$$

B Right-tail normal distribution outliers

Let C be a multiset of real numbers, representing a sample of i.i.d. random variables X, and let the multiset

$$C_x = \{\{y \in C : y \leq x\}\}.$$

If $X \sim Normal$ then the right-tail normal distribution outlier threshold for e-value v is

$$\arg\max \{x : |C_x| \geq 0.5|C|, \mathrm{P}(Normal(\text{mean } C_x, \text{var } C_x) > x) \times |C_x| > v\}.$$

The outliers are the numbers larger than the outlier threshold.
The operation of removing outliers is idempotent.
Computing the outlier threshold has time $O(|C| \log |C|)$.

C Data preparation and computation of dissimilarities

The tree objects in this paper are genomes and rRNA sequences, [2], [14].

The formulae and parameters of dissimilarities as well as the variance functions have been chosen by the principle of maximum congruence estimation, see [4].

C.1 Genomes

The genomes used for tree building are GenBank bacterial, fungal and protist assemblies with N50 $\geq$ 4000 which have not been excluded from RefSeq with the reason: 'chimeric', 'contaminated', 'hybrid', 'misassembled', 'mixed', or 'low quality'.

The names of the genome objects are the NCBI GenBank assembly identifiers.

In the bacterial genomes the CDSs are found by Prodigal v2.6.3 with default parameters [10]. In the eukaryotic genomes the CDSs are found by GeneMark-ES version 4.39 with default parameters [12], [24]. For fungal genomes GeneMark-ES has the additional parameter `--fungus`.

The protein translations are submitted to `hmmsearch` [9] ver. 3.1b2 vs. the BUSCO universal protein HMMs [5], see Table 8. The BUSCO HMMs have score cutoffs. For each HMM the annotated protein with maximum score is selected, and the protein segment hit by the HMM is stored as the universal HMM marker of the genome.

The protein translations are also submitted to `hmmsearch` with parameter `-Z 10000` vs. Pfam ver. 31.0, see [20]. For each protein the Pfam accessions are sorted, deredundified and converted to 64-bit hash codes by the C++ class `hash<string>`.

The CDSs and their protein translations longer than 150 aa are converted to 64-bit hash codes by the C++ class `hash<string>`.

The CDS, protein and Pfam HMM hash codes are loaded into an SQL table `GenomeHash`, see Appendix D.1.

The quality outliers are identified by the thresholds indicated in Table 8. In addition, quality outliers are the genomes assigned to hybrid taxa in GenBank, and the genomes with the ratio of distinct CDS hash codes to distinct protein hash codes above 1.1.

Taxon	bacteria	fungi	protists
BUSCO HMM library	`bacteria_odb9`	`fungi_odb9`	`protists_ensembl`
Number of HMMs	148	290	215
HMM cutoffs are used	Yes	No	No
Min. number of HMMs	70	200	100
Hash type	CDS	PRT	PRT
Min. number of hashes	250	1000	1000
Max. number of hashes	10000	30000	200000
hash_common_{min}	50	10	10
hash_ratio_{min}	0.5	0.1	0.1
p_{univ}	0.57	0.57	0.65
k_{SD}	1.0175	1	1.5
r_{CDS}	0.25	N/A	N/A
r_{PRT}	0.34	0.3375	0.584
r_{univ}	25.0	19.89	20.0
β_{CDS}	0.9	0	0
β_{PRT}	2.5	3.0	2.5

Table 8: Parameters of genome preparation and dissimilarities

The genogroup barrier for bacteria was selected at approximately subspecies level, and the genogroup barrier for fungi was selected at approximately species level. The barriers were selected in the genetic discontinuity regions of the dissimilarities [11].

The assembly attributes used to identify genogroup outliers are:

- average CDS length.
- octamer mononucleotide frequency;
- total DNA length;
- GC%;

The deviant octamer mononucleotide frequencies may be due to sequencing errors which may result in frame shifts. The attributes "total DNA length" and "GC%" were used only for bacterial genomes.

The dissimilarity d_{xy} between genomes x and y is a combination of 3 dissimilarities: $d_{\text{CDS}}(x,y)$, $d_{\text{PROT}}(x,y)$ and $d_{\text{univ}}(x,y)$. The dissimilarity $d_{\text{CDS}}(x,y)$ is used only for bacteria.

Let $H_{\text{T}}(x)$ be the set of type T hash codes of genome x, where T is CDS or PRT. Then if

$$|H_{\text{T}}(x) \cap H_{\text{T}}(y)| < \text{hash\_common}_{min}$$

or

$$\frac{\min\{|H_{\text{T}}(x)|, |H_{\text{T}}(y)|\}}{\max\{|H_{\text{T}}(x)|, |H_{\text{T}}(y)|\}} < \text{hash\_ratio}_{min}$$

then $d_{\text{T}}(x,y) = \infty$, otherwise

$$d_{\text{T}}(x,y) = \frac{1}{2}\left(\ln\frac{|H_{\text{T}}(x)|}{|H_{\text{T}}(x) \cap H_{\text{T}}(y)|} + \ln\frac{|H_{\text{T}}(y)|}{|H_{\text{T}}(x) \cap H_{\text{T}}(y)|}\right),$$

where hash_common_{min} and hash_ratio_{min} are defined in Table 8.

For two genomes x and y the protein segments of the same universal HMM markers are aligned globally with the NCBI BLASTP parameters `gap_open = -8`, `gap_extent = -2` and matrix PAM30 for bacteria, and `gap_open = -11`, `gap_extent = -2` and matrix BLOSUM62 for eukaryotes.

Let A be the protein segment of universal HMM marker i in genome x, and B be the protein segment of universal HMM marker i in genome y. Let $s(S,T)$ be the score of the global alignment of sequences S and T.

Define the dissimilarity $d(HMM_i)$ similarly to d_{CDS} and d_{PRT}: if

$$\frac{\min\{s(A,A),s(B,B)\}}{\max\{s(A,A),s(B,B)\}} < 0.5$$

then $d(HMM_i) = \infty$, otherwise

$$d(HMM_i) = \left[\frac{1}{2}\left(\ln\frac{s(A,A)}{s(A,B)} + \ln\frac{s(B,B)}{s(A,B)}\right)\right]^{p_{univ}},$$

where p_{univ} is defined in Table 8.

Let m_i be the median value of $d(HMM_i)$ measured on the half million dissimilarities of 1000 random genomes, and let $r_i = 1/m_i$. Then $r_i\ d(HMM_i)$ converts the dissimilarity measured by each HMM_i to the same scale.

Let $var_i = \text{var}\ (r_i\ d(HMM_i))$ measured on the same half million dissimilarities of 1000 random genomes.

Then the universal HMM dissimilarity between genomes x and y is defined as

$$d_{\text{univ}}(x,y) = \frac{\sum_{i\in I}(1/var_i)\ r_i\ d(HMM_i)}{\sum_{i\in I} 1/var_i},$$

where

$$I = \{i : r_i\ d(HMM_i) - d_{\text{univ}}(x,y) \le k_{SD}\ \sqrt{var_i}\}\,.$$

The set I excludes erroneously called or horizontally transferred universal markers. The formula of $d_{\text{univ}}(x,y)$ allows for missing marker HMMs.

The values r_i and var_i are in the files `hmm-univ.stat` in the subdirectories of the directory `phylogeny/inc/`. The values k_{SD} are in the top lines of `hmm-univ.stat` and also defined in Table 8.

Since the computation of $d_{\text{univ}}(x,y)$ and the set I depends on each other, the computation is done iteratively until convergence, initially setting I to all markers found in the genomes x and y.

The combined dissimilarity is defined as

$$d_{xy} = \begin{cases} r_{\text{CDS}}\ d_{\text{CDS}}(x,y), & \text{if } d_{\text{CDS}}(x,y) < \beta_{\text{CDS}}, \\ r_{\text{PRT}}\ d_{\text{PRT}}(x,y), & \text{if } d_{\text{PRT}}(x,y) < \beta_{\text{PRT}}, \\ r_{\text{univ}}\ d_{\text{univ}}(x,y), & \text{else.} \end{cases}$$

The scale factors r_{CDS}, r_{PRT}, r_{univ} and the barriers β_{CDS}, β_{PRT} are in Table 8.

The scale factors r_{CDS}, r_{PRT} and r_{univ} are chosen to convert the dissimilarity to the same scale, and this scale is optimized for the linear-exponential variance function. The barriers β_{CDS} and β_{PRT} are chosen to be the subspecies and species barriers respectfully.

Two bacterial genomes having 3,000 long CDSs each and differing in one SNP in one of these CDSs will have $d_{\text{CDS}} = 3.3 \times 10^{-4}$, and since d_{CDS} is multiplied by $r_{\text{CDS}} = 0.25$, the combined dissimilarity will be approximately 10^{-4}.

C.2 rRNA sequences

NCBI collections of prokaryotic 16S rRNA and fungal ITS rRNA sequences have been used for tree building.

The names of the sequence objects are the NCBI RefSeq nucleotide accessions.

The 16S rRNA sequences have been downloaded from `https://www.ncbi.nlm.nih.gov/nuccore?term=33175%5BBioProject%5D+OR+33317%5BBioProject%5D` and the ITS rRNA sequwences have been downloaded from `https://www.ncbi.nlm.nih.gov/nuccore?term=177353%5BBioProject%5D`.

The ITS rRNA sequences are 5.8S ribosomal RNAs of length between 140 and 165 bp with upstream flanking sequences of length 100 bp and downstream flanking sequences of length 150 bp.

The 16S rRNA sequences have length at least 1200 bp.

The dissimilarity between two sequences is computed by making a semiglobal alignment and computing the minimum edit distance [19], which is equal to the number of elementary operations (substitutions of one symbol, deletions of one symbol, and insertions of one symbol) needed to change one sequence into the other, weighted by the penalties of specific elementary operations: match score 5, mismatch score -4, gap open 0, and gap extension -10. The minimum allowed alignment length is 140 bps for ITS rRNA sequences, and 600 bps for 16S rRNA sequences. The minimum edit distance was multiplied by 0.0082 for ITS rRNA sequences, and by 0.00155 for 16S rRNA sequences because this multiplication affects non-linearly the linear-exponential dissimilarity variance.

D Initial set of neighbors

This Section describes how to compute the sets $N_0(a)$ used in Section 8.3.

D.1 Genomes

An SQL table `GenomeHash` with genome annotations encoded as 64-bit hash codes is maintained:

```
create table GenomeHash
(
  genome int  not null
, type char(3)  not null  -- 'CDS', 'PRT' or 'HMM'
, hash numeric(20)  not null
);
create unique index GenomeHash_hash_type_uq on GenomeHash(hash, type, genome);
```

Hashes with frequencies more than 5000 in `GenomeHash` for specific hash types are stored in an SQL table `FreqHash`.

For a new genome a and a given hash type the set of hashes is loaded from a file into a temporary SQL table H. The hashes in `FreqHash` are deleted from H. Then H is joined with `GenomeHash` and a multiset M of genomes is produced. The 100 genomes with maximum frequencies in M and which are in the current tree are returned as $N_0(a)$.

Since genomes with too many proteins are quality outliers, $|H| = O(1)$. The size of `GenomeHash` is $O(n)$, and finding $N_0(a)$ has time $O(\log n)$.

The hash type 'CDS' is used first. If $|N_0(a)| < 100$ then the hash type 'PRT' is used, and if $|N_0(a)| < 100$ then the hash type 'HMM' is used.

This data structure allows a fast identification of similar GenBank genomes for new genomes which have not been submitted to GenBank.

D.2 rRNA sequences

A BLASTN search database is maintained for the sequences which are in the current tree. For a new sequence a the set $N_0(a)$ are the top 100 sequences found by BLASTN with parameters `-strand plus -evalue 1e-20` ordered by the number of identities descending, [3].

Updating a BLASTN search database takes time $O(n)$.

A BLASTN search also takes time $O(n)$, but is practically fast. For large n, BLASTN should be replaced by a search with nucleotide k-mer index using the approach of Appendix D.1. The program `kmerIndex_find` finds 100 close DNA sequences using a k-mer index within time $O(\log n_i)$, and the program `kmerIndex_add` creates a k-mer index within time $O(n_i \log n_i)$ for $k = O(\log n_i)$ for iteration i of Section 8.4.

E Interactive tree HTML file

Trees are represented as HTML files with JavaScript.

Example of the Fungal genome tree is Fig. 1.

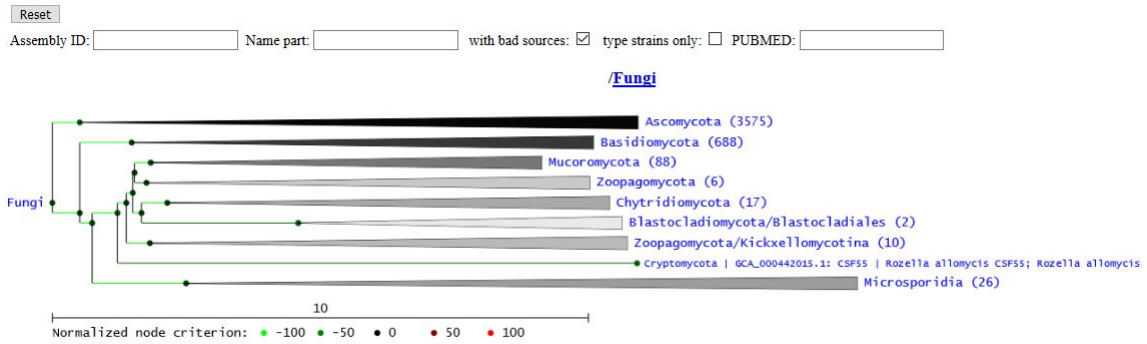

Figure 1: Fungal genomes interactive tree

The distances are standardized so that 1 is the species barrier, see Table 5.

A part of the tree can be displayed. Therefore, there are *displayed leaves* and *final leaves*, as well as *displayed root* and *global root*. The final leaves are the tree objects — genomes or rRNA sequences.

Some nodes have automatically assigned names by the novel method: the result of the solution of the maximum parsimony problem for the taxons of the objects.

For genomes the names are composed of these items:

- automatically assigned taxonomic name (optional) followed by "|";
- "bad sources": [SURVEILLANCE], [METAGENOME] or [SINGLE CELL] (optional);
- [TYPE] (optional);
- GenBank assembly accession followed by ":";
- strain (if it is in GenBank) followed by "|";
- MLST number (only for *Escherichia coli* followed by "|";
- serovar (if it is in GenBank) followed by "|";
- semicolon-delimited taxonomic lineage ordered from most specific to most general taxa.

For rRNA sequences the names are composed of these items:

- automatically assigned taxonomic name (optional) followed by "|";
- RefSeq nucleotide accession, followed by ":";
- semicolon-delimited taxonomic lineage ordered from most specific to most general taxa.

The root is on the left-hand side, and the displayed leaves are on the right-hand side. Each displayed leaf should have a name.

If a displayed leaf is an interior node, then its subtree is hidden (collapsed), and this subtree is shown as a gray triangle where the amount of black color reflects the number of leaves, followed by the node name and the number of hidden final leaves in parentheses. The length of the triangle is the average distance to the final leaves weighted by the length of the subtrees.

For each interior node the child subtrees are sorted by the number of final leaves descending, so that largest subtrees are displayed at the top. This makes the displayed topology of the trees to be stable among incrementations.

Leaf color reflects the normalized object criterion. The colors of an interior nodes are derived from the average normalized object criteria of the children. Arc color reflects the normalized arc error density.

Each interior node is identified by the pair `<top final leaf name>:<bottom final leaf name>`, so that the interior node is the least common ancestor (LCA) of these two final leaves.

Left-clicking on a node opens a Node Information window containing information on the node and buttons allowing to hide or show ancestors or descendants. To close the information window, click on any point on the tree. A single node cannot be displayed, therefore, hiding ancestors of a leaf node also expands the descendants of the leaf, and hiding descendants of a root node also expands the ancestors of the root. Double left-clicking on a leaf or any point to the right of a leaf is the same as hiding ancestors of the leaf, and double left-clicking on a root or any point on the left margin of the tree is the same as hiding descendants of the root. Descendants are expanded until named nodes are reached.

The names of final leaves are truncated to 100 characters on the canvas, but are displayed in full in the Node Information Window.

Objects can be searched by entering the object names (assembly ID or accession for genomes, or RefSeq accessions for rRNA) or any part of the names of the final leaves. A colon-delimited list of object identifiers can be entered in the "Assembly ID" or "Nucl. accession" field. On pressing the Enter key the found objects are highlighted. The button "Zoom in on highlighted" makes the displayed root to be the LCA of the highlighted objects. In the pair of numbers (M/N) to the right of a displayed leaf, M is the number of highlighted objects, and N is the total number of objects on the subtree.

If the displayed tree is too large, the "Make Clusters" button runs a simple clustering algorithm which assigns names like "Cluster N" to the root nodes of clusters and collapses the clusters, thus making the displayed tree smaller.

References

[1] Kaleb Abram, Zulema Udaondo, Carissa Bleker, Visanu Wanchai, Trudy M. Wassenaar, Dave W. Ussery, What can we learn from over 100,000 Escherichia coli genomes? Jul 19, 2019, bioRxiv 708131; doi: https://doi.org/10.1101/708131

[2] Agarwala R. et al. Database resources of the National Center for Biotechnology Information. Nucleic Acids Res. 2018 Jan 4;46(D1):D8-D13. doi: 10.1093/nar/gkx1095.

[3] Stephen F. Altschul, Thomas L. Madden, Alejandro A. Schaffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman (1997), Gapped BLAST and PSI-BLAST: a new generation of protein database search programs, Nucleic Acids Res. 25:3389-3402.

[4] Brover S., Principle of a Congruence Maximization of a Phylogenetic Tree to Reference Classifications, *in print*

[5] Robert M. Waterhouse, Mathieu Seppey, Felipe A. Simão, Mose Manni, Panagiotis Ioannidis, Guennadi Klioutchnikov, Evgenia V. Kriventseva, and Evgeny M. Zdobnov, BUSCO applications from quality assessments to gene prediction and phylogenomics. Mol Biol Evol, published online Dec 6, 2017, doi: 10.1093/molbev/msx319

[6] W.H.E. Day. Computational complexity of inferring phylogenies from dissimilarity matrices. Bulletin of Mathematical Biology, 49(4):461-467, 1987.

[7] Felsenstein J. Inferring Phylogenies. Sunderland, MA: Sinauer Associates, 2003.

[8] Felsenstein J. 1989, PHYLIP-Phylogeny inference package (version 3.2), *Cladistics* 5:164-166.

[9] HMMer. http://hmmer.org/

[10] Hyatt D, Chen GL, Locascio PF, Land ML, Larimer FW, Hauser LJ. Prodigal: prokaryotic gene recognition and translation initiation site identification. BMC Bioinformatics. 2010 Mar 8;11:119. doi: 10.1186/1471-2105-11-119. PubMed PMID: 20211023; PubMed Central PMCID: PMC2848648.

[11] Jain, C., Rodriguez-R, L.M., Phillippy, A.M. et al. High throughput ANI analysis of 90K prokaryotic genomes reveals clear species boundaries. Nat Commun 9, 5114 (2018) doi:10.1038/s41467-018-07641-9

[12] Lomsadze A., Ter-Hovhannisyan V., Chernoff Y. and Borodovsky M. "Gene identification in novel eukaryotic genomes by self-training algorithm." Nucleic Acids Research, 2005, 33: 6494-6506

[13] Monerawela C, Bond U. The hybrid genomes of *Saccharomyces pastorianus*: A current perspective. Yeast. 2018 Jan;35(1):39-50. doi: 10.1002/yea.3250. Epub 2017 Sep 26. PMID:28787090

[14] Sayers EW, Cavanaugh M, Clark K, Ostell J, Pruitt KD, Karsch-Mizrachi I. GenBank. Nucleic Acids Res. 2019 Jan 8;47(D1):D94-D99. doi: 10.1093/nar/gky989. PMID: 30365038; PMCID: PMC6323954.

[15] Parks, D., Chuvochina, M., Waite, D. et al. A standardized bacterial taxonomy based on genome phylogeny substantially revises the tree of life. Nat Biotechnol 36, 996-1004 (2018) doi:10.1038/nbt.4229

[16] Parks, D. H., Imelfort, M., Skennerton, C. T., Hugenholtz, P. & Tyson, G. W. CheckM: assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes. Genome Res. 25, 1043?1055 (2015).

[17] Price MN, Dehal PS, Arkin AP (2010) FastTree 2 – Approximately Maximum-Likelihood Trees for Large Alignments. PLoS ONE 5(3): e9490. https://doi.org/10.1371/journal.pone.0009490

[18] Quast C, Pruesse E, Yilmaz P, Gerken J, Schweer T, Yarza P, Peplies J, Glöckner FO (2013) The SILVA ribosomal RNA gene database project: improved data processing and web-based tools. Nucl. Acids Res. 41 (D1): D590-D596.

[19] Setubal J., Meidanis J., Introduction to Computational Molecular Biology. 1997. PWS Publishing Company.

[20] Schaeffer RD, Liao Y, Cheng H, Grishin NV. ECOD: new developments in the evolutionary classification of domains. Nucleic Acid Research. 2017; 45(D1):D296-D302.

[21] Shen XX et al., Tempo and Mode of Genome Evolution in the Budding Yeast Subphylum. Cell. 2018 Nov 29;175(6):1533-1545.e20. doi: 10.1016/j.cell.2018.10.023. Epub 2018 Nov 8.

[22] James A. Studier, Karl J. Keppler, A Note on the Neighbor-Joining Algorithm of Saitou and Nei, Mo. Bio. Evol. 5(6):729-731. 1988

[23] DL Swofford, Phylogenetic Analysis Using Parsimony, PAUP* 4.0, beta version 4.0 b2, Sinauer Assoc, Boston, Mass

[24] Ter-Hovhannisyan V., Lomsadze A., Chernoff Y. and Borodovsky M. "Gene prediction in novel fungal genomes using an ab initio algorithm with unsupervised training." Genome Research, 2008, 18:1979-90

[25] Truszkowski, J., Hao, Y. & Brown, D.G. Towards a practical O(n logn) phylogeny algorithm. Algorithms Mol Biol 7, 32 (2012) doi:10.1186/1748-7188-7-32

[26] Yin, J., Zhang, C. & Mirarab, S. ASTRAL-MP: scaling ASTRAL to very large datasets using randomization and parallelization. Bioinformatics. https://doi.org/10.1093/bioinformatics/btz211 (2019).

[27] Zhu, Q., Mai, U., Pfeiffer, W. et al. Phylogenomics of 10,575 genomes reveals evolutionary proximity between domains Bacteria and Archaea. Nat Commun 10, 5477 (2019). https://doi.org/10.1038/s41467-019-13443-4