TreeCluster: clustering biological sequences using phylogenetic trees

Metin Balaban[1], Niema Moshiri[1], Uyen Mai[2], Siavash Mirarab[3*],

1 Bioinformatics and Systems Biology Graduate Program, UC San Diego, CA 92093, USA
2 Computer Science and Engineering, UC San Diego, CA 92093, USA
3 Department of Electrical and Computer Engineering, UC San Diego, CA 92093, USA

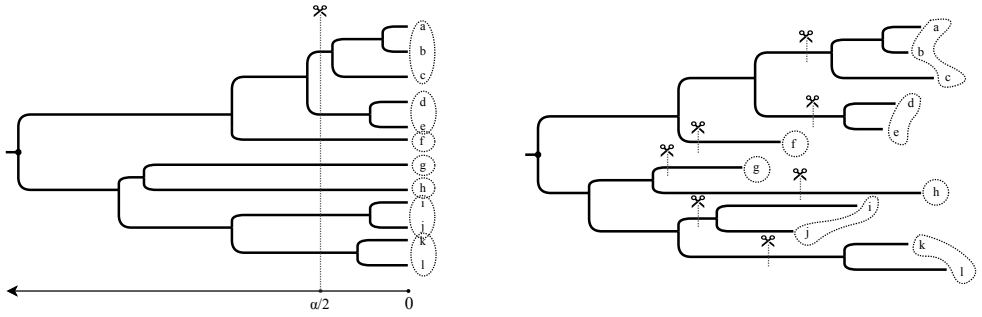
* smirarab@ucsd.edu

Abstract

Clustering homologous sequences based on their similarity is a problem that appears in many bioinformatics applications. The fact that sequences cluster is ultimately the result of their phylogenetic relationships. Despite this observation and the natural ways in which a tree can define clusters, most applications of sequence clustering do not use a phylogenetic tree and instead operate on pairwise sequence distances. Due to advances in large-scale phylogenetic inference, we argue that tree-based clustering is under-utilized. We define a family of optimization problems that, given a (not necessarily ultrametric) tree, return the minimum number of clusters such that all clusters adhere to constraints on their heterogeneity. We study three specific constraints that limit the diameter of each cluster, the sum of its branch lengths, or chains of pairwise distances. These three versions of the problem can be solved in time that increases linearly with the size of the tree, a fact that has been known by computer scientists for two of these three criteria for decades. We implement these algorithms in a tool called TreeCluster, which we test on three applications: OTU picking for microbiome data, HIV transmission clustering, and divide-and-conquer multiple sequence alignment. We show that, by using tree-based distances, TreeCluster generates more internally consistent clusters than alternatives and improves the effectiveness of downstream applications. TreeCluster is available at `https://github.com/niemasd/TreeCluster`.

Introduction

Homologous molecular sequences across different species or even within the same genome can show remarkable similarity due to their shared evolutionary history. These similarities have motivated many applications to first group the elements of a diverse set of sequences into *clusters* of set of sequences with high similarity for use in subsequent steps. The precise meaning of clusters depends on the application. For example, when analyzing 16S microbiome data, the standard pipeline is to use Operational Taxonomic Units (OTUs), which are essentially clusters of closely related sequences that do not diverge more than a certain threshold [1–3]. Another example is HIV transmission inference, a field in which a dominant approach is to cluster HIV sequences from different individuals based on their similarity (again using a threshold) and to use these clusters as proxies to clusters of disease transmission [4, 5].



(a) Ultrametric tree T_1 **(b)** Non-ultrametric tree T_2

Fig 1. When the phylogenetic tree is ultrametric, clustering is trivial: for a threshold α, cut the tree at $\frac{\alpha}{2}$ height (a) When the tree is not ultrametric, it is not obvious how to cluster leaves (b). In either cases, a set of cut edges defines a clustering.

Shared evolutionary histories, the origin of similarity among homologous sequences, can be captured by a phylogenetic tree. The true phylogeny is never known, but it can be inferred from sequence data, [6, 7] and recent advances have led to methods that can infer approximate maximum-likelihood (ML) phylogenetic trees in sub-quadratic time, which can easily scale to datasets of even millions of sequences [8]. Moreover, accurate alignment of datasets with hundreds of thousands of species (a prerequisite to most phylogenetic reconstruction methods) is now possible using divide-and-conquer methods [9, 10].

Most existing sequence clustering methods use the pairwise distances among sequences as input but do not take advantage of phylogenetic trees. For example, the widely-used UCLUST [2] searches for a clustering that minimizes the Hamming distance of sequences to the cluster centroid while maximizes the Hamming distance between centroids. Several other clustering methods have been developed for various contexts, such as gene family circumscription [11, 12] and large protein sequence databases [13].

Using phylogenies for clustering has two potential advantages. *i*) Since phylogenies explicitly seek to infer the evolutionary history, phylogeny-based clustering has the potential to not only reflect evolutionary distances (i.e., branch lengths) but also relationships (i.e., the tree topology). Recall also that branch lengths in a phylogeny are model-based "corrections" of sequence distances in a statistically-rigorous way [7], and therefore, may better reflect divergence between organisms. *ii*) When inferred using subquadratic algorithms, the tree can eliminate the need to compute all pairwise distances, which can improve speed and scalability. Moreover, a phylogeny often has to be inferred for purposes other than clustering and thus typically is readily available. However, despite these potentials, to our knowledge, no systematic method for phylogeny-guided clustering exists. Built for analyzing HIV transmissions, ClusterPicker [14] clusters sequences based on their distances while using the phylogenetic tree as a constraint; however, it still uses sequence (not tree) distances and scales cubically with respect to number of sequences in the worst case.

Given a rooted phylogenetic tree, if the tree is ultrametric (that is, distances of all the leaves to the root are identical), clustering sequences based on the tree can proceed in an obvious fashion: the tree can be cut at some distance from the root, thereby partitioning the tree into clusters (Fig. 1a). This approach extends in natural ways to unrooted ultrametric trees by first rooting the tree at the unique midpoint [15] and proceeding as before. However, inferred phylogenetic trees are rarely ultrametric. Different organisms can evolve with different rates of evolution, and even when the rates are identical (leading to an ultrametric true tree), there is no guarantee that the

inferred trees will be ultrametric. Given a non-ultrametric (and perhaps unrooted) tree, the best way to cluster sequences is not obvious (Fig. 1b).

One way to approach tree-based clustering is to treat it as an optimization problem. We can define problems of the following form: "find the minimum number of clusters such that some criteria constrain each cluster." Interestingly, at least two forms of such optimization problems have been addressed as early as the 1970s by the theoretical computer science community, in the context of proving more challenging theorems. *Tree partitioning* problems were defined to partition any arbitrary tree into minimum number of subtrees such that the maximum path length between any nodes [16] or sum of all node weights [17] in each subtree are constrained by a given threshold. Both problems can be solved exactly using straightforward linear-time algorithms; these algorithms, however, to our knowledge, are mostly ignored by bioinformaticians.[1]

Here, we argue that the tree-based clustering approach should be revitalized in the field of bioinformatics. In this paper, we introduce a family of tree partitioning problems and describe linear-time solutions for three instances of the problem (two of which correspond to the aforementioned max and sum problems with known algorithms). More importantly, we show that these tree-based clustering algorithms can result in improved downstream biological analyses in three different contexts: defining microbial OTUs, HIV transmission clustering, and divide-and-conquer multiple sequence alignment.

Materials and methods

TreeCluster

Problem definition

Let $T = (V, E)$ be an unrooted binary tree represented by an undirected acyclic graph with vertices V either degree one or three, weighted edges E, and leafset $\mathcal{L}$. We denote the path length between leaves u and v on T with $d_T(u, v)$ or simply $d(u, v)$ when clear by context. The weight of an edge (u, v) (i.e., its branch length) is denoted by $w(u, v)$. A natural way to define a clustering of the leaves in $\mathcal{L}$ is to associate a clustering to a cut set $C \subseteq E$ on the edges of T. We define a partition $\{L_1, L_2 \cdots, L_N\}$ of $\mathcal{L}$ to be an *admissible* clustering if it can be obtained by removing some edge set C from E and assigning leaves of each of the resulting connected components to a set L_i (note: $N \leq |C| + 1$).

For a given tree T, let $f_T : 2^{\mathcal{L}} \to \mathbb{R}$ be a function that maps a subset of the leafset $\mathcal{L}$ to a real number. The purpose of $f_T(.)$ is to characterize the diversity of elements at the leaves within each cluster, and it is often defined as a function of the edge weights in the cluster. For example, it can be the diameter of a subset: $f_T(L) = \max_{u,v \in L} d_T(u, v)$. We define a family of problems that seek to minimize the number of clusters while each cluster has to adhere to constraints defined using $f_T(.)$. More formally:

Definition 1 (Min-cut partitioning problem family)**.** *Given a tree T with leafset $\mathcal{L}$ and a real number α, find an admissible partition $\{L_1 \dots L_N\}$ of $\mathcal{L}$ that satisfies $\forall i, f_T(L_i) \leq \alpha$ and has the minimum possible cardinality (N) among all such clusterings.*

A natural way to limit the diversity within a cluster is to constrain all pairwise distances among members of the cluster to be less than a given threshold:

[1] In fact, in our quest to design tree-based algorithms, we reinvented these same algorithms only to later find out that they have been previously described.

Definition 2 (Max-diameter min-cut partitioning problem)**.** *The Min-cut partitioning problem (Definition 1) is called Max-diameter min-cut partitioning problem when* $f_T(L) = \max_{u,v \in L} d(u,v)$.

One potential disadvantage of max diameter min-cut partitioning is its susceptibility to outliers: the largest distance within a cluster may not be always an accurate representation of the degree of diversity in the cluster. A natural choice that may confine the effect of outliers is the following:

Definition 3 (Sum-length min-cut partitioning problem)**.** *The Min-cut partitioning problem is called Sum-length min-cut partitioning problem when* $f_T(L) = \sum_{(u,v) \in edges(T|L)} w(u,v)$ *where* $T|L$ *is the tree* T *restricted to a subset of leaves* L.

We also study a fourth problem, which we will motivate later:

Definition 4 (Single-linkage min-cut partitioning problem)**.** *The Min-cut partitioning problem is called Single-linkage min-cut partitioning problem when* $f_T(L) = \max_{S \subset L} \{ \min_{u \in S, v \in L-S} d(u,v) \}$.

Next, we will show linear-time algorithms for the Max-diameter, Sum-length, and Single-linkage min-cut partitioning problems. Note that all of these algorithms use variations of the same greedy algorithm. Two of these greedy algorithms (max and sum) are already described in the theoretical computer science literature. Nevertheless, we reiterate the solutions using consistent terminology and provide alternative proofs of their correctness.

Linear-time solution for Max-diameter min-cut partitioning

A linear-time solution for the Max-diameter min-cut partitioning problem was first published by Parley *et al.* [16]. We present Algorithm 1, which is similar to the algorithm by Parley *et al.* (adding branch lengths), and we give an alternative proof.

The algorithm operates on T', which is an arbitrary rooting of T at node r. We denote the subtree rooted at an internal node u as U. Let the two children of u be called u_l and u_r, and let the tree rooted by them be U_l and U_r. We use w_l and w_r to denote $w(u,u_l)$ and $w(u,u_r)$, respectively. We define $B(u)$ to be the length of the path from u to the farthest *connected* leaf in U under the optimal clustering.

The algorithm uses a bottom-up traversal of the tree. When we arrive at node u, one or more new paths form between the two trees U_r and U_l. Among those paths, the longest one has the length $B(u_l) + w_l + B(u_r) + w_r$. If this value exceeds the threshold, we break either (u,u_r) or (u,u_l), depending on which minimizes $B(u)$. Note that the algorithm always cuts at most one child edge of every node, and thus, $B(u)$ is always well-defined.

We now show the algorithm correct. Let $A(u)$ be the minimum number of clusters under U, each with a diameter less than α; i.e., $A(r)$ is the objective function.

Theorem 1. *Algorithm 1 computes a clustering with minimum* $A(r)$ *for rooted tree* T'. *In addition, among all possible such clusterings, the algorithm picks the solution with minimum* $B(r)$.

Proof. The proof uses induction. The base case for the induction is the simple rooted tree with root u and two leaves u_l and u_r. If $w_l + w_r > \alpha$ the algorithm cuts the longer branch whereas if $w_l + w_r \leq \alpha$ no branch is cut. In both cases, the theorem holds.

Th inductive hypothesis is that for a node u, the algorithm has computed $A(u_l)$, $A(u_r)$, $B(u_l)$, and $B(u_r)$ optimally. We need to prove that a solution other than the

Algorithm 1: Linear solution for Max diameter min-cut partitioning

Input: A tree $T' = (V, E)$ and a threshold α

$B(v) \leftarrow 0$ **for** $v \in V$
for $u \in$ *post order traversal of internal nodes of* T' **do**
  if $B(u_l) + w_l + B(u_r) + w_r > \alpha$ **then**
    if $B(u_l) + w_l < B(u_r) + w_r$ **then**
      $E \leftarrow E - \{(u, u_r)\}$
      $B(u) \leftarrow B(u_l) + w_l$
    else
      $E \leftarrow E - \{(u, u_l)\}$
      $B(u) \leftarrow B(u_r) + w_r$
  else
    $B(u) \leftarrow max(B(u_l) + w_l, B(u_r) + w_r)$
return *Leafsets of every connected component in* T'

one computed by our algorithm i) cannot have a lower number of clusters, call it $A'(u)$, and ii) when $A'(u) = A(u)$, cannot have a lower distance to the farthest connected leaf, call it $B'(u)$.

When $B(u_l) + w_l + B(u_r) + w_r \leq \alpha$, we have $A(u) = A(u_l) + A(u_r) - 1$, which is the minimum possible by inductive hypothesis and the fact that the number of clusters cannot go down by more than one on node u. Also, $B(u)$ is optimal by construction.

When $B(u_l) + w_l + B(u_r) + w_r > \alpha$, without loss of generality, assume that $B(u_l) + w_l \geq B(u_r) + w_r$ and thus, the algorithm cuts the (u, u_l) branch, getting $A(u) = A(u_l) + A(u_r)$ and $B(u) = B(u_r) + w_r$. Note that $A'(u) < A(u)$ is only possible if $A'(u_l) = A(u_l)$ and $A'(u_r) = A(u_r)$ and we do not cut any branch at u in the alternative clustering. However, this scenario is *not* possible because

$$B'(u_l) + w_l + B'(u_r) + w_r \geq B(u_l) + w_l + B(u_r) + w_r > \alpha$$

where the first inequality follows from the inductive hypothesis and the final inequality shows that we will have to cut a branch in any alternative setting. Finally, we need to show that an alternative solution with $A'(u) = A(u)$ but $B'(u) < B(u)$ is not possible. The inequality requires that either $B'(u_l) < B(u_l)$ or $B'(u_r) < B(u_r)$. First, consider the $B'(u_l) < B(u_l)$ case, which is possible only if $A'(u_l) = A(u_l) + 1$. Note that $A'(u) = A(u)$ requires $A'(u_r) = A(u_r)$ (and thus $B'(u_r) = B(u_r)$) and that $B'(u_l) + w_l + B(u_r) + w_r < \alpha$, which is possible. Under this condition, we find:

$$B'(u) = max(B'(u_l) + w_l, B(u_r) + w_r) \geq B(u_r) + w_r = B(u) \quad (1)$$

If instead $B'(u_r) < B(u_r)$, similar conditions can be written, resulting in

$$B'(u) = max(B(u_l) + w_l, B'(u_r) + w_r) \geq B(u_l) + w_l \geq B(u_r) + w_r = B(u) \quad (2)$$

Thus, $A(u)$ and $B(u)$ are optimal when $B(u_l) + w_l + B(u_r) + w_r > \alpha$.

□

Corollary 1. *Let C' be the cut set obtained by running Alg. 1 on any arbitrary rooting T' of unrooted tree T. C' optimally solves the Max-diameter min-cut partitioning problem.*

Proof. Let r_r and r_l denote the right and the left child of the root of T'. Every edge in T can be mapped to T' except the edge (r_{right}, r_{left}), from which we define a mapping

to (r, r_{right}) (w.l.o.g). Using this mapping, the optimal clustering (i.e. optimal cut-set) on T can be translated to an alternative max diameter min-cut partitioning on T'. However, by Theorem 1, $A(r)$ is optimal and cannot be improved by any alternative partitioning. Since any admissible clustering on T' is also admissible on T, Alg. 1 minimizes N.

□

Linear solution for the Sum-length min-cut partitioning problem

A linear algorithm which partitions trees into the fewest clusters having total node weights lower than or equal to α has been previously published by Kundu *et al.* [17]. In order to solve Sum-length min-cut partitioning problem, we require an altered version of the original algorithm that works on edge (instead of node) weights and focuses on binary trees. Algorithm 1 with two simple modifications solves Sum-length min-cut partitioning problem optimally (see Alg. 3 in Appendix B). The first modification is that here we define the auxiliary variable $B(u)$ denoting sum of weights of all descendent edges connected to u at the stage it is processed by the algorithm. Secondly, in the bottom-up traversal of internal nodes of T', for node u, w.l.o.g, let $B(u_l) + w_l > B(u_r) + w_r$. If the sum of branch lengths in the combined subtree exceed α, we break the edge (u, u_l). Unlike Algorithm 1, where $B(u_l) + w_l + B(u_r) + w_r \leq \alpha$, here, $B(u)$ is set to $B(u_l) + w_l + B(u_r) + w_r$. The proof for the correctness of the algorithm is analogous to that of Alg. 1 and is given in Appendix B.

Single linkage min-cut partitioning

We now address the Single-linkage problem (Definition 4), which can be considered a relaxation of the max diameter min-cut partitioning. To motivate this problem, first consider the following definition.

Definition 5 (Single-linkage clustering)**.** *We call a partition of $\mathcal{L}$ to be a Single-linkage clustering when for every $a, b \in \mathcal{L}$, a and b are in the same cluster if and only if there exist a chain $\mathcal{H} = c_0, c_1 \ldots, c_m, c_{m+1}$, where $a = c_0$ and $b = c_{m+1}$, and for every $0 \leq i \leq m$, we have $d(c_i, c_{i+1}) \leq \alpha$.*

Thus, every pair of nodes are put in the same cluster if (but not only if) their distance is below the threshold (the rest follows from transitivity). The next result (proved in Appendix B.) motivates the choice of $f_T(.)$ in Definition 4.

Proposition 1. *The optimal solution to the Single-linkage min-cut partitioning problem (Definition 4) is identical to the Single-linkage clustering of Definition 5.*

We now present Algorithm 2, a linear-time solution to the Single-linkage min-cut partitioning problem. The algorithm first computes the distances to the closest node on left, right, and outside each node u in a post-order followed by a pre-order traversal. Then, on a post-order traversal, it cuts each child edge iff the minimum distance of leaves under it to leaves under its sibling *and* to any leaf outside the node both exceed the threshold.

Theorem 2. *A min-cut partitioning computed by Algorithm 2 optimally solves Single-linkage min-cut partitioning problem (Definition 5).*

Proof. Let $a \rightsquigarrow b$ be the path between leaves a and b on T. Fixing a and b, for each node j, we use the term *support of* j denoted by $s(j)$ to refer to the unique node on all the three paths $a \rightsquigarrow b$, $a \rightsquigarrow j$, and $b \rightsquigarrow j$. We refer to a group of leaves that share a mutual support with respect to a and b as a *bubble* (e.g., triangles in Fig. 2). Among all

Algorithm 2: SINGLE-LINKAGE Single-linkage min-cut partitioning

$minBelow[u] \leftarrow minAbove[u] \leftarrow \infty$ **for** $v \leftarrow V$
for $u \in$ *post order traversal of* T' **do**
  if u ***in*** $\mathcal{L}$ **then**
    $minBelow[u] \leftarrow 0$;
  else
    $minBelow[u] \leftarrow min\big(minBelow[u_l] + w_l, minBelow[u_r] + w_r\big)$;
for $u \in$ *pre order traversal of* T' **do**
  if $u \neq r$ **then**
    $minAbove[u] \leftarrow min\big(minBelow[s] + w(v,s), minAbove[v] + w(v,v)\big)$;
for $u \in$ *post order traversal of internal nodes of* T' **do**
  if $minBelow[u_l] + w_l + minBelow[u_r] + w_r > \alpha$ ***and*** $minBelow[u_l] + w_l + minAbove[u] > \alpha$ **then**
    $E \leftarrow E \setminus (u, u_l)$
  if $minBelow[u_l] + w_l + minBelow[u_r] + w_r > \alpha$ ***and*** $minBelow[u_r] + w_r + minAbove[u] > \alpha$ **then**
    $E \leftarrow E \setminus (u, u_r)$
  if $minBelow[u_l] + w_l + minAbove[u] > \alpha$ ***and*** $minBelow[u_r] + w_r + minAbove[u] > \alpha$ **then**
    $E \leftarrow E \setminus (v, u)$
return *Leafsets of every connected component in* T'

bubbles branching out of $a \rightsquigarrow b$, let the one with the closest support to a be A'. We name the leaf closest to a on A' as a' (Fig. 2).

- If $d(a,b) \leq \alpha$ holds, the algorithm will never cut any edge on $a \rightsquigarrow b$ due to the following observation. For every internal node u on $a \rightsquigarrow b$, let v and w be the adjacent nodes on $a \rightsquigarrow u$ and $u \rightsquigarrow b$ respectively. Also let p_a be the closest leaf to u whose support $s(p_a)$ is on $a \rightsquigarrow u$, and p_b to be the closest leaf to u whose support $s(p_b)$ is on $u \rightsquigarrow b$. $d(p_a,u) + d(u,p_b) \leq d(a,u) + d(u,b) \leq \alpha$ holds, therefore regardless of the rooting, (v,u) and (u,w) is never cut by Alg. 2.
- if a chain $\mathcal{H}$ exists, due to the previous observation, there is no cuts on $c_i \rightsquigarrow c_{i+1}$ for every $0 \leq i \leq m$. Consequently, a and b are connected through a path and hence are in the same cluster.
- Assume Alg. 2 places a and b on the same cluster, i.e. does not cut any edge on $a \rightsquigarrow b$. We present a procedure to generate a chain $\mathcal{H}$ described in Definition 1. We define $p_0 = a$ and $p_{m'} = b$. For $1 \leq i \leq m'$, we let p_i be the closest leaf to p_{i-1}

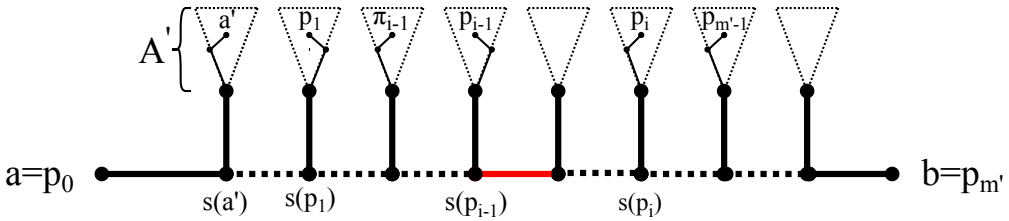


Fig 2. A sketch showing the setup for constructing the chain $\mathcal{H}$.

whose support $s(p_i)$ is on $p_{i-1} \leftrightsquigarrow b$ and $s(p_i) \neq s(p_{i-1})$ (i.e., p_i is in the bubble to the right of the bubble of p_{i-1}). Conversely, for $1 \leq i \leq m'$, we let π_i denote the closest leaf to p_i whose support is on $a \leftrightsquigarrow s(p_{i-1})$ (i.e., is in a bubble to the left of p_i); note $\pi_1 = a$. Every $\pi_i \in \{p_0 \dots p_{i-1}\}$ due to following observation: if not, $s(\pi_i)$ has to be on $s(p_{j-1}) \leftrightsquigarrow s(p_j)$ for some j; but, we would have $d(p_{j-1}, \pi_i) \leq d(p_{j-1}, p_j)$, which contradicts the definition of p_i. The fact that Alg. 2 retains $(a, s(a'))$ indicates that $min(d(a, a'), d(a, p_1)) = d(a, p_1) \leq \alpha$; therefore, we add $a \to p_1$ to an auxiliary graph $\mathcal{H}'$. Now, consider Alg. 2 when it processes the node $s(p_{i-1})$ for $1 < i$. The fact that the first edge on path $s(p_{i-1}) \leftrightsquigarrow s(p_i)$ (shown in red color in Fig. 2) is not cut indicate that either $d(\pi_{i-1}, p_i) \leq \alpha$ or $d(p_{i-1}, p_i) \leq \alpha$. Depending on which is true, we add a link from $\pi_{i-1} \to p_i$ or $p_{i-1} \to p_i$ to $\mathcal{H}'$. We repeat this process for all i until we reach $i = m'$, where we add an edge to $p_{m'} = b$. Noting that $\pi_i \in \{p_0 \dots p_{i-1}\}$, the $\mathcal{H}'$ graph becomes a directed tree, rooted at a with a directed path to the leaf b. This directed path constitute the valid chain $\mathcal{H}$.

□

Clade constraint for rooted trees

So far, we have focused on unrooted trees. This choice is partially driven by the fact that phylogenetic reconstruction tools predominantly use time reversible models of sequence evolution (e.g., GTR [18]), and therefore output an unrooted tree. Nevertheless, researches have developed various methods for rooting trees [19, 20] including scalable algorithms [15]. When a rooted tree is available, each "monophyletic clade", i.e. a group of entities that includes all descendants of their common ancestor, are biologically meaningful units. Thus, we may want to constrain each cluster to be a clade. Such "clade" constraints, in fact, make clustering easier; our algorithms can be easily altered to ascertain that each cluster is also a clade. Specifically on Algorithm 1, when we have $B(u_l) + w_l + B(u_r) + w_r > \alpha$, we simply need to cut both (u, u_l) and (u, u_r) (instead of cutting only the longer one). This small modification allows Max-Diameter, Sum-length, and Single-linkage min-cut partitioning problems to be solved in linear time while imposing the clade constraint.

Three Applications of TreeCluster

While sequence clustering has many applications, in this paper, we highlight three specific areas as prominent examples.

Application 1: OTU clustering

Biological Problem. For microbiome analyses using 16S sequences generated from whole communities, the standard pipeline uses operational taxonomic units (OTUs). Sequences with similarity at or above a certain threshold (say 97%) are grouped into OTUs, which are the most fine-grained level at which organisms are distinguished. All sequences assigned to the same OTU are treated as one organism in downstream analyses, such as taxonomic profiling, taxonomic identification, sample differentiation, or machine learning. The use of a similarity threshold instead of a biological concept of species is to avoid the notoriously difficult problem of defining species for microbial organisms. In addition, the use clusters of similar sequences as OTUs can provide a level of robustness with respect to sequencing errors.

Most applications of OTUs are closed-reference: a reference database of known organisms is selected and OTUs are defined for reference sequences using methods such as UCLUST [2] and Dotur [3]. These methods cluster sequences based on a chosen

threshold of similarity, often picking a centroid sequence to represent an OTU. Reads from a 16S sample are then compared to the OTUs, and the closest OTU is found for each read (judging distance by sequence similarity). Once all reads are processed for all samples, an OTU table can be built where the rows are samples, the columns are OTUs, and each cell gives the frequency of an OTU in a sample. This table is then used in downstream analyses. Several large reference databases exist for these OTU-based analyses [21–23]. One of these databases, popularized through pipelines such as Qiita [24], is Greengenes [23].

Regardless of the downstream application of an OTU table, one would prefer the OTUs to be maximally coherent (i.e., internally consistent) so that they represent organisms as faithfully as possible. We will focus our experiments on the closed-reference OTU picking methods and the Greengenes as the reference library. However, note that open-reference OTU picking and sub-operational-taxonomic-unit (sOTU) methods [25–27] also exist and involve a similar need for sequence clustering.

Existing methods. Several hierarchical clustering tools have been proposed for OTU clustering [3, 28]. Non-hierarchical clustering methods [2, 29] however gained popularity due to being computationally less demanding compared to hierarchical methods. Two prominent methods are UCLUST [2] and CD-HIT [29], which share the same algorithmic strategy: for a given threshold α, UCLUST determines a set of representative sequences dynamically by assigning query sequences into representative sequences (centroids) such that, ideally, distance between each query and its assigned centroid is less than α while distances between centroids is more than α. UCLUST is a heuristic algorithm, and the processing order of the queries may affect the outcome clustering. CD-HIT is different from UCLUST mostly in its strategy for computing distances.

Formulation as min-cut partitioning. We define OTUs by solving the Min-diameter, Sum-Length, or Single-linkage min-cut partitioning problems using a chosen threshold α and an inferred ML phylogeny. Each cluster in the resulting partition is designated as an OTU.

Experiments. We evaluate the quality of tree-based OTU clustering by comparing it to UCLUST as used by Greengenes [23]. We run TreeCluster on the phylogenetic tree of 203,452 sequences in Greengenes v13.5 database in three modes: max, sum, and Single-linkage. We use the following 20 thresholds: $[0.005, 0.05]$ with a step size of 0.005 and $(0.05, 0.15]$ with a step size of 0.01. For Single-linkage, we only go up to 0.1 because above this threshold, the number of clusters becomes much smaller than other methods.

From the same Greengenes database, we extract OTU clusters for all available sequence identity thresholds up to 0.15 (i.e., 0.03, 0.06, 0.09, 0.12, and 0.15). We measure the quality of a clustering $\{L_1, \ldots, L_N\}$ by its weighted average of average pairwise distance per cluster (which we call *cluster diversity* for shorthand), given by the following formula:

$$\mu(\{L_1, \ldots, L_N\}) = \frac{\sum_{k=1}^{N} |L_k| \sum_{i,j \in L_k} \frac{d(i,j)}{|L_k|^2}}{\sum_{k=1}^{N} |L_k|} = \frac{1}{n} \sum_{k=1}^{N} \sum_{i,j \in L_k} \frac{d(i,j)}{|L_k|} \tag{3}$$

where n denotes the number of sequences clustered. We compute distance $d(i, j)$ between two elements using two methods: *tree distance*, which is the path lengths on the inferred phylogenetic tree, or the sequence-based hamming distance. Hamming distances are computed pairwise from the multiple sequence alignment of all 203,452 sequences in

Greengenes database and ignore any site that includes a gap in the pairwise alignment. Clearly, cluster diversity alone is not sufficient to judge results (singletons have zero diversity). Instead, we compare methods at the same level of clustering with respect to their diversity. Thus, as we change the threshold α, we compare methods for choices of the threshold where they result in (roughly) equal numbers of clusters. Given the same number of clusters, a method with lower cluster diversity is considered preferable.

Application 2: HIV transmission cluster analyses

Biological Problem. HIV sequences evolve fast and therefore their phylogenetic relationships have a trace of the transmission chains from one person to another [30]. The ability to perform phylogenetic analyses of HIV sequences is critical for epidemiologists who design and evaluate HIV control strategies [31–35]. The results of these analyses can provide information about the genetic linkage [36] and transmission histories [37], as well as mixing across subpopulations [38]. A recent advancement in computational molecular epidemiology is the use of transmission clustering to predict at-risk individuals and epidemic growth: infer transmission clusters from pairwise sequence distances, monitor the growth of clusters over time, and prioritize clusters with the highest growth rates [39]. In this monitoring framework, two natural questions come about: What is the optimal way to infer transmission clusters from molecular data, and how can transmission cluster inference be performed more efficiently?

Existing methods. Two existing tools perform such clustering. Cluster Picker [4] is given a distance threshold, a phylogenetic tree, and sequences. It clusters individuals such that each cluster defines the leaves of a clade in the tree, the maximum pairwise sequence-based distance in each cluster is below the threshold, and the number of clusters is minimized. HIV-TRACE is a tool that, given a distance threshold and sequences, clusters individuals such that, for each pair of individuals u and v, if the Tamura-Nei 93 (TN93) distance [40] between u and v is below the threshold, u and v are placed in the same cluster [5]. Both methods scale worse than linearly with the number of sequences (quadratically and cubically, respectively, for HIV-Trace and Cluster Picker), and for large datasets, they can take hours, or even days, to run (however, HIV-Trace does enjoy trivial parallelism).

Formulation as min-cut partitioning. Transmission clustering is similar to our problem formulation in that it involves cutting edges such that the resulting clusters (as defined by the leafsets resulting from the cuts) must adhere to certain constraints. Both Cluster Picker and HIV-TRACE utilize pairwise distances computed from sequences, but when reformulated to utilize tree-based distances from an inferred phylogeny, Cluster Picker becomes analogous to our Max-diameter min-cut partitioning (with an added constraint that clusters must define clades in the phylogeny), and HIV-TRACE analogous to the Single-linkage min-cut partitioning.

Experiments. To evaluate the effectiveness of HIV transmission clustering, we first simulate HIV epidemic data using FAVITES [41]. For the simulation parameters, we use the parameters described in Moshiri *et. al.* [41] to model the San Diego HIV epidemic between 2005 and 2014. However, we deviate from the original parameter set in one key way: originally, all HIV patients were sequenced at the end time of the epidemic, yielding an ultrametric tree in the unit of time, but to better capture reality, we instead sequence each patient the first time they receive Antiretroviral Treatment (ART). In our simulations, we vary two parameters: the expected time to begin ART as well as the expected degree of the social contact network, which underlies the transmission network.

Higher ART rates and lower degrees both result in a slower epidemic and change patterns of phylogenetic branch length [41]. The complete FAVITES parameter set can be found in the supplementary materials. We infer phylogenies from simulated sequences under the GTR+Γ model using FastTree-II [8], and we use MinVar algorithm to root the trees using FastRoot [42].

We use HIV-TRACE [5] as well as multiple clustering modes of TreeCluster to infer transmission clusters. We were unable to use Cluster Picker [4] due to its excessive running time. For HIV-TRACE, we use a clustering threshold of 1.5% as suggested by its authors [39]. Because HIV-TRACE estimates pairwise sequences distances under the TN93 model, [40] which tend to be underestimates of phylogenetic distance estimated under the GTR model, we use a clustering threshold of 3% for Single-Linkage TreeCluster. The default Cluster Picker threshold for Max-diameter clustering is 4.5%, [4], so we use this as our clustering threshold for Max-Diameter TreeCluster (both with and without the Clade constraint). For Sum-length TreeCluster (with and without the Clade constraint), we simply double the Max-diameter threshold and use 9%. In addition to using these default thresholds, we also test a wide range of thresholds for each transmission clustering method for robustness.

We measure cluster growth from year 8 to year 9 of the simulation and select the 1,000 highest-priority individuals, where individuals are prioritized in descending order of respective cluster growth. To measure the risk of a given individual u, we count the number of HIV transmission events $u \to v$ between years 9 and 10. To measure the effectiveness of a given clustering, we average the risk of the selected top 1,000 individuals and use this as a metric of how effective the clustering method is. Higher numbers imply the ability to prevent more transmissions by targeting a fixed (1,000) number of individuals and thus are desirable. As a control, we also show the mean number of transmissions per population, which is what a random selection of 1,000 individuals would give in expectation (we call this "expected" risk).

Application 3: Divide-and-conquer Multiple Sequence alignment

Algorithmic idea. Tree-based clustering has also been used for divide-and-conquer, a technique that has proved particularly useful for scaling existing methods for tree inference and multiple sequence alignment (MSA) to very large datasets [10, 43–47]. Divide-and-conquer methods first use some approach to build a quick-and-dirty estimate of the phylogeny and then divide the dataset into smaller sets using the phylogeny, such that sequences inside each subset are less diverse than the full set; given the subsets, an accurate (but often computationally demanding) method is run on the subsets to infer the MSA and/or the tree; finally, the results on the subsets are merged using various techniques. The accuracy of the output depends not only on the accuracy of the base method used on the subsets and the the merging method, but also, on the effectiveness of the method used to divide the tree into subsets [48].

Here, we specifically focus on MSA using divide-and-conquer. In particular, we focus on a method called PASTA that infers both MSAs and trees for ultra-large datasets (tested for up to 1,000,000 sequences). PASTA computes an initial alignment using HMMs implemented in HMMER [49] and an initial tree using FastTree-II [8]; then, it performs several iterations (default: 3) of the divide-and-conquer strategy described before using Mafft [50] for aligning subsets and using a combination of OPAL [51] and a technique using transitivity for merging subalignments. A tree is generated using FastTree-II at the end of each iteration, which is then used as the guide tree for the next iteration. The method has shown great accuracy on simulated and real data, especially in terms of tree accuracy, where it comes very close to the accuracy obtained using the true alignment, leaving little room for improvement. However, in terms of the alignment accuracy, it has substantial room for improvement on the most challenging datasets.

The divide-and-conquer of PASTA is based on the centroid-edge decomposition. Given the *guide tree* (available from the previous iteration), the decomposition is defined recursively: divide the tree into two halves, such that the two parts have equal size (or, are as close in size as possible). Then, recurse on each subtree, until there are no more than a given number of leaves (default: 200) in each subset.

Formulation as min-cut partitioning. The centroid edge decomposition involves cutting edges and includes a constraint defined on the subsets. However, it is defined procedurally and does not optimize any natural objective function. The min-cut partitioning can produce a decomposition that is similar to the centroid decomposition in its constraints but is different in outcome. Take the guide tree and set all edge weights to 1. Then solve our Sum-length min-cut partitioning problem with the threshold set to $\alpha = 2m - 2$; the result is a partition such that no cluster has more than m leaves and the number of subsets is minimized. Thus, this "max-size min-cut partitioning" is identical to centroid decomposition in its constraints, but also guarantees to find the minimum number of clusters.

Experiments. To evaluate how our new decomposition impacts PASTA, we compare the old version to a new version that uses the decomposition based on max-size min-cut partitioning, with other parameters (including maximum subset size) all kept fixed. We run both versions on two datasets both from the original PASTA paper: 10 replicates of a simulated RNAsim dataset with 10,000 leaves and a set of 19 real HomFam datasets with 10,099 to 93,681 protein sequences. The RNASim is based on a very complex model of RNA evolution. Here, the true alignment, known in simulations, is used as the reference. For HomFam, since the true alignment is not known, following previous papers, we rely on a very small number of seed sequences with a hand-curated reliable alignment as reference [45, 52]. In both cases, we measure alignment error using two standard metrics computed using FastSP [53]: SPFN (the percentage of homologies in the reference alignment not recovered in the estimated alignment) and SPFP (the percentage of homologies in the estimated alignment not present in the reference).

Results

Results for Application 1: OTU clustering

On the Greengenes dataset, as we change the threshold between 0.005 and 0.15, we get between 181,574 and 10,112 clusters (note that singletons are also counted). The cluster diversity has a non-linear relationship with the number of clusters; it drops quicker with higher thresholds where fewer clusters are formed (Figs. 3 and S1). Comparing the three objective functions that can be used in TreeCluster, we observe that Max-diameter and Sum-length have similar trends of cluster diversity scores whereas Single-linkage min-cut partitioning has substantially lower diversity compared to the other two methods (Fig. S1). This pattern is observed whether distances are computed using tree distances or sequence distances, but differences are larger for tree distance. Finally, note that even though tree distances are, as expected, larger than sequence distances (Fig. S2), the cluster diversity is *lower* when computed using tree distances, showing that clusters are tight in the phylogenetic space.

Compared to default Greengenes OTUs, defined using UCLUST, Max-diameter min-cut partitioning defines tighter clusters for tree-based scores (Fig. 3). When distances between sequences are measured in tree distance, cluster diversity score for Greengenes OTUs is substantially lower for all thresholds, and the gap is larger for higher thresholds. For example, the cluster diversity of Greengenes OTUs is three times

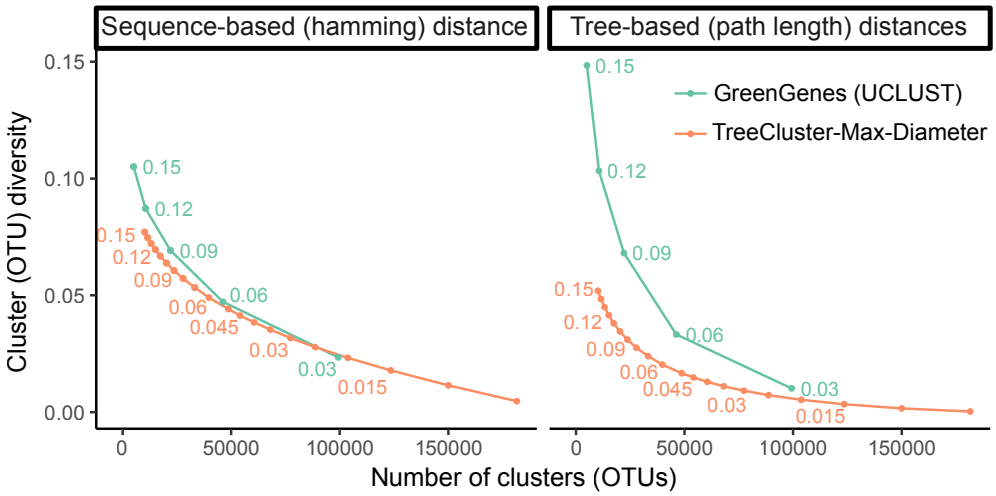


Fig 3. Clustering diversity, defined as weighted average pairwise distance within a cluster (Eq. 3) for Greengenes and TreeCluster versus the number of OTUs. For any number of OTUs (x-axis), a lower OTU diversity (y-axis) is preferrable. The threshold α is shown for all data points corresponding to GreenGeenes and for some points of TreeCluster. See Fig. S1 for comparison to other TreeCluster versions.

higher than TreeCluster OTUs for $\alpha = 0.15$. When distances between sequences are measured in hamming distance, Greengenes and TreeCluster perform similarly for low threshold values (e.g., note $\alpha = 0.03$ for Greengenes, which is similar to $\alpha = 0.02$ for TreeCluster in terms of the number of clusters). However, when the number of OTUs is reduced, remarkably, TreeCluster outperforms Greengenes OTUs by up to 1.4 folds (for $\alpha = 0.15$). This is despite the fact that UCLUST is working based on sequence distances and TreeCluster is not.

Size of the largest cluster in Greengenes is larger compared to TreeCluster (Table 1). For example, for $\alpha = 0.09$, both methods have similar number of clusters (22,090 and 23,631 for Greengenes and TreeCluster, respectively) but the size of largest cluster in Greengenes is three times that of TreeCluster (1,659 versus 540). On the other hand, for the same threshold value, the number of singleton clusters comprises 48% of all clusters for Greengenes whereas only 27% of the clusters are singletons for TreeCluster. Thus, GreenGenes has more clusters that are very small or very large, compared to TreeCluster.

	TreeCluster-Max-Diameter			**GreenGenes-UCLUST**		
α	σ	Σ	max	σ	Σ	max
0.015	86387	123456	47	(n.a)	(n.a)	(n.a)
0.03	42510	77263	96	70415	99322	527
0.045	24795	54068	171	(n.a)	(n.a)	(n.a)
0.06	15257	39809	305	26485	46256	894
0.09	6396	23631	540	10560	22090	1659
0.12	3003	15052	808	4153	10544	2131
0.15	1525	10112	1209	1735	5088	3765

Table 1. Number of singleton clusters (σ), total number of clusters (Σ), and maximum cluster size (max) for TreeCluster and GreenGenes on various threshold levels. In GreenGenes database, OTU definitions for threshold level $\alpha = 0.015$ and $\alpha = 0.045$ are not available.

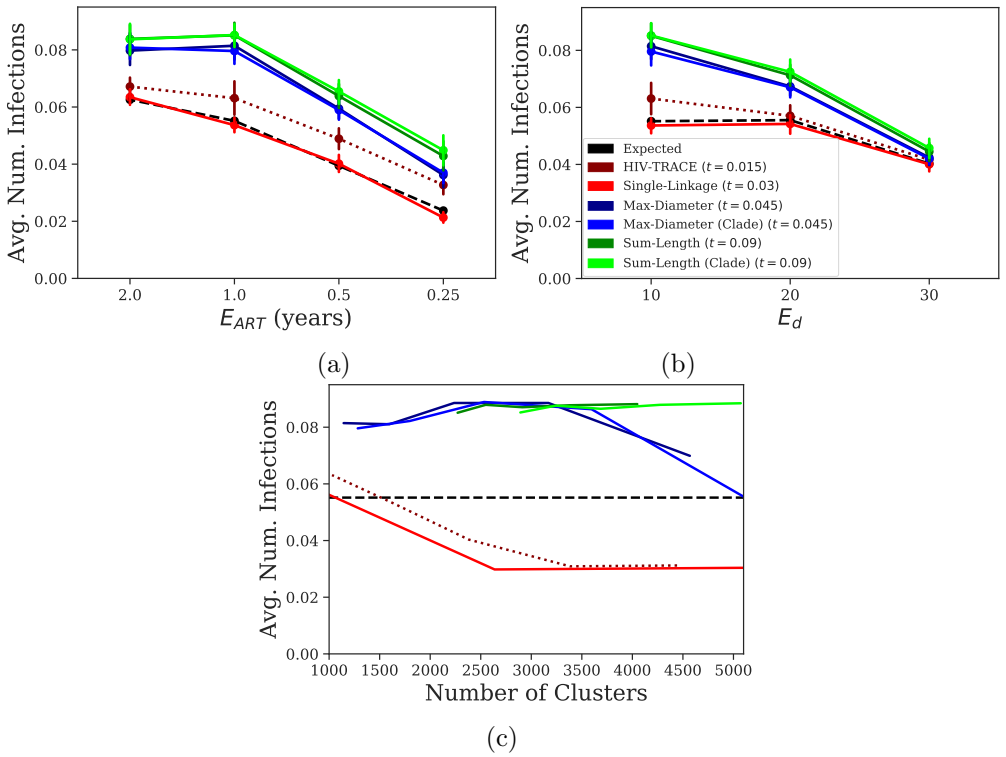


Fig 4. Effectiveness of transmission clustering, where effectiveness is measured as the average number of individuals infected by the selected 1,000 individuals. The horizontal axis depicts the expected time to begin ART (a), the expected degree (i.e., number of sexual contacts) for individuals in the contact network (b), and the number of clusters using various thresholds (c).

Results for Application 2: HIV dynamics

Comparing various versions of TreeCluster, regardless of the parameters that we vary, Sum-Branch Tree Cluster consistently outperforms the other clustering methods, and the inclusion of the Clade constraint has little impact on effectiveness (Fig. 4). Compared to a random selection of individuals, the risk of selected individuals can be substantially higher; for example, with expected ART time set to 1 year, expected risk is 0.55 transmissions while the risk of top 1,000 individuals from Sum-length clusters is 0.85. In all the conditions, a close second to TreeCluster Sum-length, is TreeCluster Max-diameter. Other methods, however, are substantially less effective than these two modes of TreeCluster.

When varying expected time to begin ART and expected degree, Single-Linkage TreeCluster and HIV-TRACE consistently perform lower than the other approaches. Single-Linkage TreeCluster typically performing around the theoretical expectation of a random selection while HIV-TRACE performing slightly better (Fig. 4a-b). Recall that these two methods are conceptually similar. Moreover, these patterns are not simply due to the chosen thresholds. Even when we change the thresholds to control the number of clusters, Single-Linkage TreeCluster and HIV-TRACE consistently perform worse than expected by random selection (Fig. 4c). The effectiveness of other methods is maximized when they create between 2,000 to 5,000 clusters for Sum-length, or between 2,000 to 3,000 clusters for Max-Diameter.

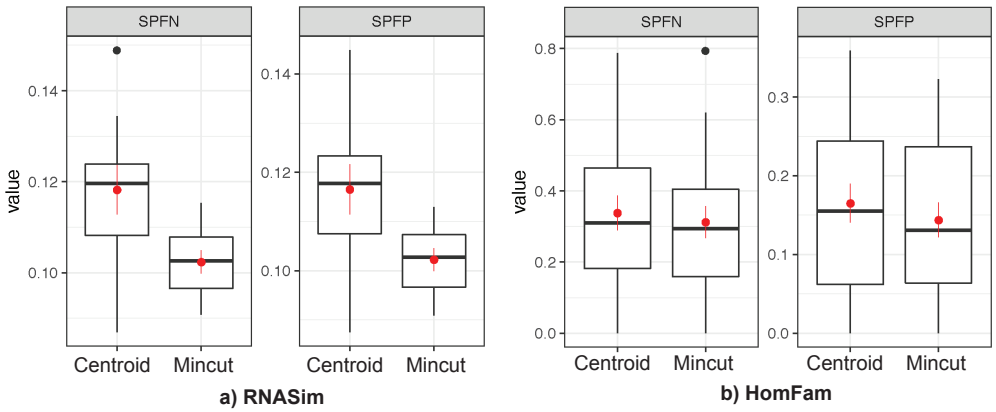


Fig 5. Alignment error for PASTA using the old (centroid) and the new (mincut) decompositions. We show Sum of Pairs False Negative (SPFN) and Sum of Pairs False Psotive (SPFP) computed using FastSP [53] over two datasetes: Simulated RNASim dataset (10 replicates) with 10 replicates and biological HomFam dataset (19 largest families; all 20 largest, except "rhv" omitted due to the warning on the Pfam website). We show boxplots in addition to mean (red dot) and standard error (red error bars).

Results for Application 3: improving PASTA

When we replace centroid decomposition with max-size min-cut partitioning in PASTA, the alignment error reduces substantially for the RNASim dataset, but less so on the Homfam dataset (Fig. 5). On the RNASim data, mean SPFN drops from 0.12 to 0.10, which corresponds to a 17% reduction in error. These drops are consistent across replicates and are substantial given the fact that the only change in PASTA was to replace its decomposition step with our new clustering algorithm, keeping the rest of the complex pipeline unchanged. In particular, the method to align subsets, to merge alignments, and to infer trees, were all kept fixed. On the HomFam dataset, too, errors decreased, but the reductions were not substantial (Fig. 5b). Based on these results, we have now changed PASTA to use max-size min-cut partitioning by default.

Discussion

Several theoretical and practical issues should be further discussed.

Mean-diameter min-cut partitioning Some of the existing methods, such as ClusterPicker [4], can define their constraints based on mean pairwise distance between nodes. Similar to those, we can define a variation of the min-cut partitioning problem where $f_T(L) = \frac{1}{\binom{|L|}{2}} \sum_{i,j \in L} d(i,j)$. Unfortunately, this "mean-diameter" min-cut partitioning problem can be solved in linear time using our greedy algorithm only if we also have clade constraints (Algorithm 4 in Appendix B). As demonstrated by the counter example given in Fig. S3, the greedy algorithm fails if we do not have clade constraints. More generally, the use of mean as function $f_T(\cdot)$ creates additional complexity and we conjecture the problem will not be solvable in linear time. Whether mean diameter is in fact a reasonable criteria is not clear. For example, it is possible that the mean diameter of a cluster is below the threshold while the mean diameter of clusters embedded in that cluster are not; such scenarios may not make sense for downstream applications.

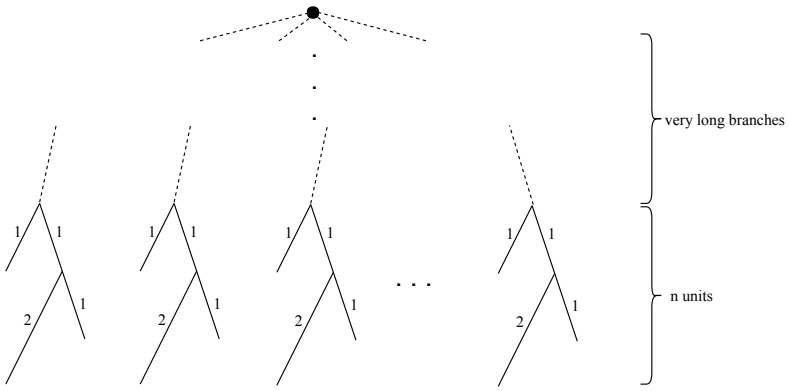


Fig 6. An example showing that number of minimal clusterings under diameter threshold can be exponential of number of leaves. When threshold is equal to 3.5, each unit has to be split into two clusters and there are three equally legitimate way of splitting. The minimum of clusters is therefore $2n$. Total number of distinct optimal solutions is 3^n whereas there are $3n$ leaves.

Set of optimal solutions. It is possible that multiple distinct partitions with equal number of clusters are all optimal solutions to any of our min-cut partitioning problems. Moreover, as the example given in Figure 6 shows, the number of optimal solutions can be exponential with respect to number of leaves in a binary phylogenetic tree. This observation renders listing all optimal solutions of diminished interest since there could be too many of them. However, finding a way to summarize all partitions may have practical utility. We do not currently have such a summarization approach. However, as shown in Lemma S1 of Appendix B, although the optimal solution space is potentially exponentially large, one can easily determine the set of all edges that could appear in any of the optimal solutions. Thus, we could find absolutely unbreakable edges that will not be cut in any optimal clustering of the data.

Choice of criterion. Among the three methods that we discussed, we observed that Max-diameter and Sum-diameter are consistently better than the Single-linkage. This observation makes intuitive sense. Single-linkage can increase the diversity within a cluster simply due to the transitive nature of its criterion. Thus, a very heterogeneous dataset may still be collapsed into one cluster, simply due to transitivity. Our desire to solve the Single-linkage problem was driven by the fact that a similar concept is used in HIV-TRACE, arguably the most widely used HIV clustering method. However, we did not detect any advantage in this type of clustering compared to Max-diameter or Sum-length; thus, our recommendation is to use these two criteria instead. Between the two, Max-diameter has the advantage that its α threshold is easier to interpret.

Centroid node. OTU picking methods, in addition to clustering, also choose a centroid node per cluster. Our clustering approach is centroid-free. However, if a representative is needed, many natural choices are available. For example, one can in linear time find the midpoint of a cluster or its balance point [15]; then, the leaf closest to the midpoint or balance point can be used as the representative. Alternatively, as some recent papers argue [54], constructing and using a consensus sequence (or perhaps even a reconstructed ancestral sequence) may be preferable to using one of the given sequences as the centroid.

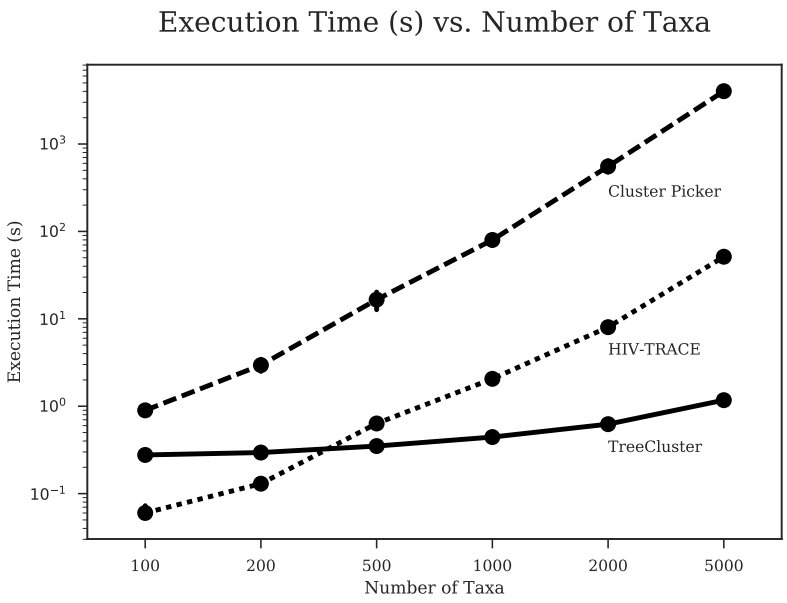


Fig 7. Execution times of Cluster Picker, HIV-TRACE, and TreeCluster in log-scale. Execution times (in seconds) are shown for each tool for various values of n sequences, with 10 replicates for each n. The full dataset was obtained by downloading all HIV-1 subtype B *pol* sequences (HXB2 coordinates 2,253 to 3,549) from the Los Alamos National Laboratory (LANL) database. All programs were run on a CentOS 5.8 machine with an Intel Xeon X7560 2.27 GHz CPU.

Running time We focused on comparing effectiveness of TreeCluster to other methods, but we note that its running time also compares favorably to other clustering methods (once the tree is inferred). For example, on a real HIV dataset, we ran HIV-TRACE, Cluster Picker, and TreeCluster for subsets of the data ranging from 100 to 5,000 sequences (Fig. 7). Even on the largest dataset, the running time of TreeCluster did not exceed 2 seconds. In contrast, the sequence-based HIV-Trace required close to a minute, (which is still quite fast) but Cluster Picker needed more than an hour. Even on the Greengenes dataset with more than 200,000 leaves, TreeCluster performed clustering in only 30 seconds. The high speed of TreeCluster makes it possible to quickly scan through a set of α thresholds to study its impact on the outcomes of downstream applications.

We note that these numbers do not include the time spent for inferring the tree, which should also be considered if the tree is not already available. For example, based on previous studies, MSA and tree inference on datasets with 10,000 sequences can take close to an hour using PASTA and 12 CPUs. Around a third of this time is spent on tree inference (e.g., see Figure 4 of [45]) and the rest is spent on the estimating alignment, which is also needed by most alternative clustering methods.

Conclusion

We introduce TreeCluster, a method that can cluster sequences at the tips of a phylogenetic tree using several optimization problems. We showed that our liner time algorithms can be used in several downstream applications, including OTU clustering, HIV transmission clustering, and divide-and-conquer alignment. Using the tree to build the cluster increases their internal consistency and improves downstream analyses.

Acknowledgments

Authors thank Xingfan Jia for useful discussions.

References

1. Goodrich JK, Di Rienzi SC, Poole AC, Koren O, Walters WA, Caporaso JG, et al. Conducting a microbiome study. Cell. 2014;158(2):250–262.

2. Edgar RC. Search and clustering orders of magnitude faster than BLAST. Bioinformatics. 2010;26(19):2460–2461.

3. Schloss PD, Handelsman J. Introducing DOTUR, a Computer Program for Defining Operational Taxonomic Units and Estimating Species Richness. Applied and Environmental Microbiology. 2005;71(3):1501–1506. doi:10.1128/AEM.71.3.1501-1506.2005.

4. Ragonnet-Cronin M, Hodcroft E, Hué S, Fearnhill E, Delpech VC, Brown AJL, et al. Automated analysis of phylogenetic clusters. BMC bioinformatics. 2013;14:317. doi:10.1186/1471-2105-14-317.

5. Kosakovsky Pond SL, Weaver S, Leigh Brown AJ, Wertheim JO. HIV-TRACE (TRAnsmission Cluster Engine): a Tool for Large Scale Molecular Epidemiology of HIV-1 and Other Rapidly Evolving Pathogens. Molecular Biology and Evolution. 2018;35(7):1812–1819. doi:10.1093/molbev/msy016.

6. Hillis DM, Allard MW, Miyamoto MM. [34] Analysis of DNA sequence data: Phylogenetic inference. In: Methods in enzymology. vol. 224. Elsevier; 1993. p. 456–487.

7. Warnow T. Computational phylogenetics: An introduction to designing methods for phylogeny estimation. Cambridge University Press; 2017.

8. Price MN, Dehal PS, Arkin AP. FastTree 2 - Approximately maximum-likelihood trees for large alignments. PLoS ONE. 2010;5(3). doi:10.1371/journal.pone.0009490.

9. Mirarab S, Nguyen N, Guo S, Wang LS, Kim J, Warnow T. PASTA: ultra-large multiple sequence alignment for nucleotide and amino-acid sequences. Journal of Computational Biology. 2015;22(5):377–386.

10. Nguyen NPDNpD, Mirarab S, Kumar K, Warnow T. Ultra-large alignments using phylogeny-aware profiles. Genome Biology. 2015;16(1):124. doi:10.1186/s13059-015-0688-z.

11. Enright AJ, Van Dongen S, Ouzounis CA. An efficient algorithm for large-scale detection of protein families. Nucleic acids research. 2002;30(7):1575–1584.

12. Li L, Stoeckert CJ, Roos DS. OrthoMCL: Identification of Ortholog Groups for Eukaryotic Genomes. Genome Research. 2003;13(9):2178–2189. doi:10.1101/gr.1224503.

13. Steinegger M, Söding J. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. Nature biotechnology. 2017;35(11):1026.

14. Ragonnet-Cronin M, Hodcroft E, Hué S, Fearnhill E, Delpech V, Brown AJL, et al. Automated analysis of phylogenetic clusters. BMC bioinformatics. 2013;14(1):317.

15. Mai U, Sayyari E, Mirarab S. Minimum variance rooting of phylogenetic trees and implications for species tree reconstruction. PLOS ONE. 2017;12(8):e0182238. doi:10.1371/journal.pone.0182238.

16. Parley A, Hedetniemi S, Proskurowski A. Partitioning trees: matching, domination, and maximum diameter. International Journal of Computer & Information Sciences. 1981;10(1):55–61.

17. Kundu S, Misra J. A linear tree partitioning algorithm. SIAM Journal on Computing. 1977;6(1):151–154.

18. Tavaré S. Some Probabilistic and Statistical Problems in the Analysis of DNA Sequences. Lectures on Mathematics in the Life Sciences. 1986;17:57–86.

19. Kluge AG, Farris JS. Quantitative phyletics and the evolution of anurans. Systematic Biology. 1969;18(1):1–32.

20. Farris JS. Estimating phylogenetic trees from distance matrices. The American Naturalist. 1972;106(951):645–668.

21. Quast C, Pruesse E, Yilmaz P, Gerken J, Schweer T, Yarza P, et al. The SILVA ribosomal RNA gene database project: improved data processing and web-based tools. Nucleic acids research. 2012; p. gks1219.

22. Maidak BL. The RDP-II (Ribosomal Database Project). Nucleic Acids Research. 2001;29(1):173–174. doi:10.1093/nar/29.1.173.

23. DeSantis TZ, Hugenholtz P, Larsen N, Rojas M, Brodie EL, Keller K, et al. Greengenes, a Chimera-Checked 16S rRNA Gene Database and Workbench Compatible with ARB. Appl Environ Microbiol. 2006;72(7):5069–5072. doi:10.1128/AEM.03006-05.

24. Gonzalez A, Navas-Molina JA, Kosciolek T, McDonald D, Vázquez-Baeza Y, Ackermann G, et al. Qiita: rapid, web-enabled microbiome meta-analysis. Nature Methods. 2018;15(10):796–798. doi:10.1038/s41592-018-0141-9.

25. Amir A, McDonald D, Navas-Molina JA, Kopylova E, Morton JT, Zech Xu Z, et al. Deblur Rapidly Resolves Single-Nucleotide Community Sequence Patterns. mSystems. 2017;2(2). doi:10.1128/mSystems.00191-16.

26. Callahan BJ, McMurdie PJ, Rosen MJ, Han AW, Johnson AJA, Holmes SP. DADA2: High-resolution sample inference from Illumina amplicon data. Nature Methods. 2016;13:581.

27. Edgar RC. UNOISE2: improved error-correction for Illumina 16S and ITS amplicon sequencing. bioRxiv. 2016;doi:10.1101/081257.

28. Cai Y, Sun Y. ESPRIT-Tree: hierarchical clustering analysis of millions of 16S rRNA pyrosequences in quasilinear computational time. Nucleic acids research. 2011;39(14):e95–e95.

29. Li W, Godzik A. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. Bioinformatics. 2006;22(13):1658–1659.

30. Leitner T, Escanilla D, Franzen C, Uhlen M, Albert J. Accurate reconstruction of a known HIV-1 transmission history by phylogenetic tree analysis. Proceedings of the National Academy of Sciences. 1996;93(20):10864–10869. doi:10.1073/pnas.93.20.10864.

31. Aldous JL, Pond SK, Poon A, Jain S, Qin H, Kahn JS, et al. Characterizing HIV transmission networks across the United States. Clinical infectious diseases : an official publication of the Infectious Diseases Society of America. 2012;55(8):1135–43. doi:10.1093/cid/cis612.

32. Hué S, Clewley JP, Cane PA, Pillay D. HIV-1 pol gene variation is sufficient for reconstruction of transmissions in the era of antiretroviral therapy. AIDS (London, England). 2004;18(5):719–28. doi:10.1097/00002030-200403260-00002.

33. Hughes GJ, Fearnhill E, Dunn D, Lycett SJ, Rambaut A, Leigh Brown AJ, et al. Molecular phylodynamics of the heterosexual HIV epidemic in the United Kingdom. PLoS pathogens. 2009;5(9):e1000590. doi:10.1371/journal.ppat.1000590.

34. Leigh Brown AJ, Lycett S, Weinert L, Hughes G, Fearnhill E, Dunn DT. Transmission network parameters estimated from HIV sequences for a nationwide epidemic. Journal of Infectious Diseases. 2011;204(9):1463–1469. doi:10.1093/infdis/jir550.

35. Mehta SR, Kosakovsky Pond SL, Young JA, Richman D, Little S, Smith DM. Associations between phylogenetic clustering and HLA profile among HIV-infected individuals in San Diego, California. Journal of Infectious Diseases. 2012;205(10):1529–1533. doi:10.1093/infdis/jis231.

36. Eshleman SH, Hudelson SE, Redd AD, Wang L, Debes R, Chen YQ, et al. Analysis of genetic linkage of HIV from couples enrolled in the HIV prevention trials network 052 trial. Journal of Infectious Diseases. 2011;204(12):1918–1926. doi:10.1093/infdis/jir651.

37. Hué S, Brown AE, Ragonnet-Cronin M, Lycett S, Dunn DT, Fearnhill E, et al. Phylogenetic analyses reveal HIV-1 infections between men misclassified as heterosexual transmissions. Aids. 2014;28(13):1967–1975. doi:10.1097/QAD.0000000000000383.

38. Bezemer D, Cori A, Ratmann O, van Sighem A, Hermanides HS, Dutilh BE, et al. Dispersion of the HIV-1 Epidemic in Men Who Have Sex with Men in the Netherlands: A Combined Mathematical Model and Phylogenetic Analysis. PLoS Medicine. 2015;12(11):e1001898. doi:10.1371/journal.pmed.1001898.

39. Wertheim JO, Murrell B, Mehta SR, Forgione LA, Kosakovsky Pond SL, Smith DM, et al. Growth of HIV-1 Molecular Transmission Clusters in New York City. The Journal of Infectious Diseases. 2018;doi:10.1093/infdis/jiy431.

40. Tamura K, Nei M. Estimation of the Number of Nucleotide Substitutions in the Control Region of Mitochondrial-DNA in Humans and Chimpanzees. Molecular biology and evolution. 1993;10(3):512–526.

41. Moshiri N, Ragonnet-Cronin M, Wertheim JO, Mirarab S. FAVITES: simultaneous simulation of transmission networks, phylogenetic trees, and sequences. Bioinformatics. 2018; p. bty921. doi:10.1093/bioinformatics/bty921.

42. Mai U, Mirarab S. TreeShrink: Efficient Detection of Outlier Tree Leaves. In: Meidanis J, Nakhleh L, editors. Comparative Genomics: 15th International Workshop, RECOMB CG 2017, Barcelona, Spain, October 4-6, 2017, Proceedings. Cham: Springer International Publishing; 2017. p. 116–140. Available from: https://doi.org/10.1007/978-3-319-67979-2_7.

43. Huson DH, Nettles SM, Warnow TJ. Disk-covering, a fast-converging method for phylogenetic tree reconstruction. Journal of computational biology. 1999;6(3-4):369–86. doi:10.1089/106652799318337.

44. Liu K, Raghavan S, Nelesen SM, Linder CR, Warnow T. Rapid and Accurate Large-Scale Coestimation of Sequence Alignments and Phylogenetic Trees. Science. 2009;324(5934):1561–1564. doi:10.1126/science.1171243.

45. Mirarab S, Nguyen N, Guo S, Wang LS, Kim J, Warnow T. PASTA: Ultra-Large Multiple Sequence Alignment for Nucleotide and Amino-Acid Sequences. Journal of Computational Biology. 2015;22(05):377–386. doi:10.1089/cmb.2014.0156.

46. Swenson MS, Suri R, Linder CR, Warnow T. SuperFine: fast and accurate supertree estimation. Systematic biology. 2012;61(2):214–27. doi:10.1093/sysbio/syr092.

47. Molloy EK, Warnow T. Statistically consistent divide-and-conquer pipelines for phylogeny estimation using NJMerge. bioRxiv. 2019; p. 469130.

48. Liu K, Warnow T, Holder MT, Nelesen SM, Yu J, Stamatakis A, et al. SATe-II: Very Fast and Accurate Simultaneous Estimation of Multiple Sequence Alignments and Phylogenetic Trees. Systematic Biology. 2011;61(1):90–106. doi:10.1093/sysbio/syr095.

49. Eddy SR. A new generation of homology search tools based on probabilistic inference. Genome Inform. 2009;23(1):205–211.

50. Katoh K, Toh H. Recent developments in the MAFFT multiple sequence alignment program. Brief Bioinform. 2008;9(4):286–298. doi:10.1093/bib/bbn013.

51. Wheeler TJ, Kececioglu JD. Multiple alignment by aligning alignments. Bioinformatics. 2007;23(13):559–568. doi:10.1093/bioinformatics/btm226.

52. Sievers F, Dineen D, Wilm A, Higgins DG. Making automated multiple alignments of very large numbers of protein sequences. Bioinformatics. 2013;29(8):989–995.

53. Mirarab S, Warnow T. FastSP: linear time calculation of alignment accuracy. Bioinformatics. 2011;27(23):3250–8. doi:10.1093/bioinformatics/btr553.

54. Page RDM. Modified Mincut Supertrees. In: Guigó R, Gusfield D, editors. Algorithms in Bioinformatics. Berlin, Heidelberg: Springer Berlin Heidelberg; 2002. p. 537–551.

A Supporting information

Figures

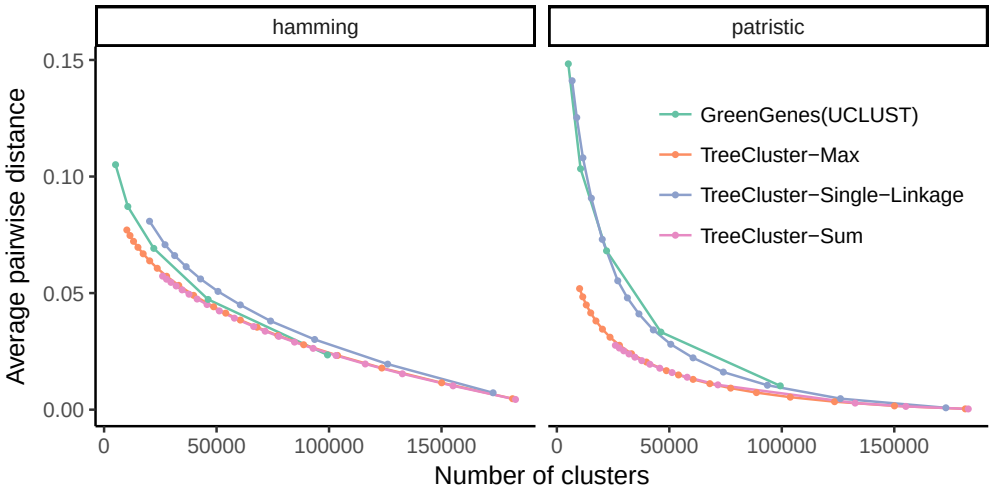


Fig S1. Comparison of various TreeCluster options and Greengenes. Clustering quality of Greengenes and various options of TreeCluster, where quality is measured as average pairwise distance within a cluster (the lower the better). The horizontal axis shows the number of clusters for a given method and a threshold value. TreeCluster OTUs based on Max-diameter and Sum-length options outperform Single-linkage option as well as Greengenes OTUs. Computation of Hamming distance based cluster diversity for $\alpha \geq 0.7$ did not complete within 24 hours and had to be terminated.

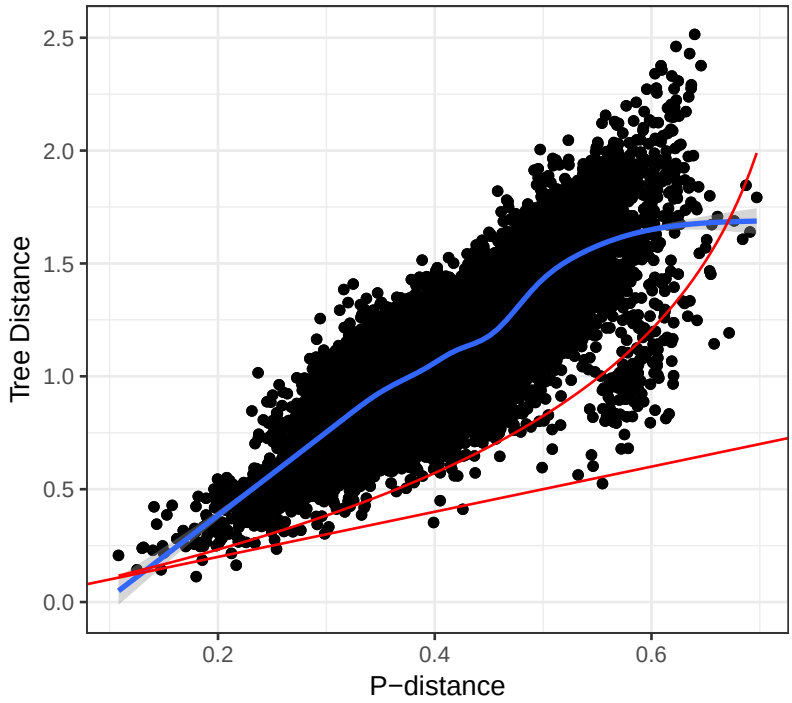
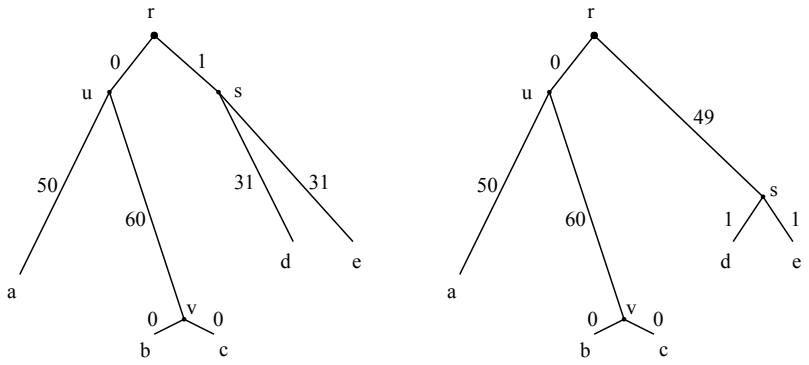


Fig S2. Tree distance versus hamming distance. On 16S data, the relationship between tree distances and hamming distances cannot be established using Jukes-Cantor formula (red curve).



(a) Example tree T_1 **(b)** Example tree T_2

Fig S3. An example showing that mean-diameter min-cut partitioning is not conforming locality when $\alpha = 72$, thus cannot be solved by a greedy algorithm analogous to Alg. 1. When a greedy algorithm is at the stage where it processes u, it makes the decision for cutting its children edges (u, v) and (u, a) based on the information available at the subtree rooted by u. When $\alpha = 72$, T_1 and T_2 require different cut-sets ($\{(u, v)\}$ and $\{(u, a)\}$ respectively) for the optimal Mean-diameter partitioning despite the fact that the subtree rooted by u remains unchanged in T_1 and T_2.

B Proofs and supplementary algorithms

B.1 Linear solution for Sum-length min-cut partitioning problem

Algorithm 3: Linear solution for Sum-length min-cut partitioning

Input: A tree $T' = (V, E)$ and a threshold α

$B(u) \leftarrow 0$ **for** $v \in V$
for $u \in$ *post order traversal of internal nodes of* T' **do**
  if $B(u_l) + w_l + B(u_r) + w_r > \alpha$ **then**
    if $B(u_l) + w_l < B(u_r) + w_r$ **then**
      $E \leftarrow E - \{(u, u_r)\}$
      $B(u) \leftarrow B(u_l) + w_l$
    else
      $E \leftarrow E - \{(u, u_l)\}$
      $B(u) \leftarrow B(u_r) + w_r$
  else
    $B(u) \leftarrow B(u_l) + w_l + B(u_r) + w_r$
return *Leafsets of every connected component in* T'

We now show the Algorithm 3 correct. Let $A(u)$ be the minimum number of clusters under U all with a diameter less than α; i.e. $A(r)$ is the objective function.

Theorem S1. *Algorithm 1 computes a clustering with minimum $A(r)$ for rooted tree T'. In addition, among all possible such clusterings, the algorithm picks the solution with minimum $B(r)$.*

Proof. The proof uses induction. The base case for the induction is the simple rooted tree with root u and two leaves u_l and u_r. If $w_l + w_r > \alpha$ the algorithm cuts the longer branch whereas if $w_l + w_r \leq \alpha$ no branch is cut. In both cases, the theorem holds.

Th inductive hypothesis is that for a node u, the algorithm has computed $A(u_l)$, $A(u_r)$, $B(u_l)$, and $B(u_r)$ optimally. We need to prove that a solution other than the one computed by our algorithm *i*) cannot have a lower number of clusters, call it $A'(u)$, and *ii*) when $A'(u) = A(u)$, cannot have a lower distance to the farthest connected leaf, call it $B'(u)$.

When $B(u_l) + w_l + B(u_r) + w_r \leq \alpha$, we have $A(u) = A(u_l) + A(u_r) - 1$, which is the minimum possible by inductive hypothesis and the fact that the number of clusters cannot go down by more than one on node u. Also, $B(u)$ is optimal by construction.

When $B(u_l) + w_l + B(u_r) + w_r > \alpha$, without loss of generality, assume that $B(u_l) + w_l \geq B(u_r) + w_r$ and thus, the algorithm cuts the (u, u_l) branch, getting $A(u) = A(u_l) + A(u_r)$ and $B(u) = B(u_r) + w_r$. Note that $A'(u) < A(u)$ is only possible if $A'(u_l) = A(u_l)$ and $A'(u_r) = A(u_r)$ and we do not cut any branch at u in the alternative clustering. However, this scenario is *not* possible because

$$B'(u_l) + w_l + B'(u_r) + w_r \geq B(u_l) + w_l + B(u_r) + w_r > \alpha$$

where the first inequality follows from the inductive hypothesis and the final inequality shows that we will have to cut a branch in any alternative setting. Finally, we need to show that an alternative solution with $A'(u) = A(u)$ but $B'(u) < B(u)$ is not possible. The inequality requires that either $B'(u_l) < B(u_l)$ or $B'(u_r) < B(u_r)$. First, consider the $B'(u_l) < B(u_l)$ case, which is possible only if $A'(u_l) = A(u_l) + 1$. Note that

$A'(u) = A(u)$ requires $A'(u_r) = A(u_r)$ (and thus $B'(u_r) = B(u_r)$) and that $B'(u_l) + w_l + B(u_r) + w_r < \alpha$, which is possible. Under this condition, we find:

$$B'(u) = B'(u_l) + w_l + B(u_r) + w_r \geq B(u_r) + w_r = B(u) \tag{4}$$

If instead $B'(u_r) < B(u_r)$, similar conditions can be written, resulting in

$$B'(u) = B(u_l) + w_l + B'(u_r) + w_r \geq B(u_l) + w_l \geq B(u_r) + w_r = B(u) \tag{5}$$

Thus, $A(u)$ and $B(u)$ are optimal when $B(u_l) + w_l + B(u_r) + w_r > \alpha$.

□

Corollary S1. *Let C' be the cut set obtained by running Alg. 1 on any arbitrary rooting T' of unrooted tree T. C' optimally solves the Max-diameter min-cut partitioning problem.*

Proof. Let r_r and r_l denote the right and the left child of the root of T'. Every edge in T can be mapped to T' except the edge (r_{right}, r_{left}), from which we define a mapping to (r, r_{right}) (w.l.o.g). Using this mapping, the optimal clustering (i.e. optimal cut-set) on T can be translated to an alternative max diameter min-cut partitioning on T'. However, by Theorem 1, $A(r)$ is optimal and cannot be improved by any alternative partitioning. Since any admissible clustering on T' is also admissible on T, Alg. 1 minimizes q.

□

B.2 Proofs for Single-linkage min-cut partitioning problem

Proof of Proposition 1. ($\Leftarrow$) If $d(a,b) \leq \alpha$ but a and b are in distinct clusters L_a, L_b respectively, N can be reduced by one by simply merging L_a and L_b. $f_T(L_a \cup L_b) \leq \alpha$ is satisfied if for any split of $L_a \cup L_b$, there exists a pair of leaves that are from distinct splits and are within α threshold. For any pair of non-empty sets S and S' that satisfy $S \subset L_a$ and $S' \subset L_b$, we have $\min_{j \in S \cup S', k \in (L_a \cup L_b) - (S \cup S')} d(j,k) \leq \min_{j \in S, k \in L_a - S} d(j,k) \leq \alpha$ and $\min_{j \in S \cup (L_b - S'), k \in S' \cup (L_a - S)} d(j,k) \leq \min_{j \in S, k \in L_a - S} d(j,k) \leq \alpha$. On the other hand, $\min_{j \in L_a, k \in L_b - S} d(j,k) \leq d(a,b) \leq \alpha$. This concludes that for $L = L_a \cup L_b$, $f_T(L) \leq \alpha$ is satisfied. L_a and L_b can still be merged if the chain $\mathcal{H}$ described above exists. It is trivial to show that there is a link $\langle c_i, c_{i+1} \rangle$ in $\mathcal{H}$ such that $c_i \in L_a$ and $c_{i+1} \notin L_a$. Using the argument above, we can iterate over $\mathcal{H}$ and keep merging clusters (and decrease N) every time we see such a link until we finally merge L_a with L_b.

($\Rightarrow$) We describe a procedure to compute the chain $\mathcal{H}$. If a and b in the same cluster L, $\min_{k \in L - \{a\}} d(a,k) \leq \max_{S \subset L} \{ \min_{j \in S, k \in L - S} d(j,k) \} \leq \alpha$ holds, implying that there is a leaf c_1 in set $L - \{a\}$ such that $d(a, c_1) \leq \alpha$. If $c_1 = b$, theorem follows. If $c_1 \neq b$, we union a and c_1, call the union set L_a, and add the link $a \to c_1$ to $\mathcal{H}'$. Iteratively, we find the pair $\langle j, k \rangle$ that yields to $\min_{j \in L_a, k \in L - L_a} d(j,k)$, add the link $j \to k$ to $\mathcal{H}'$, and add k to L_a until we finally add b to L_a. The elements forming the path between a, and b in $\mathcal{H}'$, which can be computed using depth-first-search, constitute a valid chain $\mathcal{H}$. □

B.3 Average-clade clustering

Algorithm 4: AVERAGE DIAMETER CLADE Average diameter clade min-cut partitioning

for $u \in$ *post order traversal of* T' **do**
  $totPairDist[u] \leftarrow 0$; $totLeafDist[u] \leftarrow 0$;
  if u ***in*** $\mathcal{L}$ **then**
    $numLeaves[u] \leftarrow 1$; $avgPairDist[u] \leftarrow 0$;
  else
    $numLeaves[u] \leftarrow numLeaves[u_l] + numLeaves[u_r]$;
    $totPairDist[u] \leftarrow totPairDist[u_l] + totPairDist[u_r] + totLeafDist[u_l] \times numLeaves[u_r] + totLeafDist[u_r] \times numLeaves[u_l]$;
    $totLeafDist[u] \leftarrow totLeafDist[u_l] + w_l \times numLeaves[u_l] + totLeafDist[u_r] + w_r \times numLeaves[u_r]$;
    $avgPairDist[u] \leftarrow totPairDist[u] / \binom{numLeaves[u]}{2}$;
$toExplore \leftarrow$ queue containing the root of T';
while $toExplore \neq \emptyset$ **do**
  $curr \leftarrow toExplore.dequeue()$;
  if u ***not in*** $\mathcal{L}$ ***and*** $avgPairDist[u] > \alpha$ **then**
    $E \leftarrow E \setminus (u, u_l)$; $E \leftarrow E \setminus (u, u_r)$;
    $toExplore.enqueue(u_l)$; $toExplore.enqueue(u_r)$;
return *Leafsets of every connected component in* T'

B.4 All optimal solutions

Lemma S1. *Let* $\{e_1, e_2, \cdots, e_m\}$ *be the set of edges in an unrooted tree* T*. Consider the following algorithm: root* T *at* e_j *and run* Max *algorithm , and let* $\mathcal{S}_j$ *be the set of edges cut by the algorithm in this run. Any optimal clustering for* T *has to draw its cut set from* $\Sigma = \cup_{j=1}^{m} \mathcal{S}_j$*.*

Proof. The proof is by contradiction. Assume there is an optimal cut set $\mathcal{S}'$ that does contain an edge e_i that $e_i \notin \Sigma$. Consider the T rooted at e_i. We call root of this tree as v, immediate left and right branches of v as e_l and e_r, and left and right child nodes of v as v_l and v_r. Note that e_l and e_r correspond to e_i in T. Because of our assumption, $e_l \notin \mathcal{S}_j$ and $e_r \notin \mathcal{S}_j$. When e_i is removed from T, two new trees form, called T_{left} (the one containing the node v_l) and T_{right} (the one containing the node v_r). If p number of cuts in $\mathcal{S}'$ are in T_{right}, and q number of cuts in $\mathcal{S}'$ are in T_{left}, $|\mathcal{S}'|$ equals $a + b + 1$. Number of cuts in $\mathcal{S}'$ and $\mathcal{S}_j$ are equal and e_l and e_r is not cut, which implies that either the tree rooted by v_l or v_r has an alternative clustering with one less cut. By the design of the Max algorithm, if this was the case, algorithm would choose the alternative cut. □