New Absolute Fast Converging Phylogeny Estimation Methods with Improved Scalability and Accuracy

Qiuyi (Richard) Zhang[1]
Department of Mathematics, University of California at Berkeley, Berkeley CA 94720
qiuyi@math.berkeley.edu

Satish Rao[2]
Division of Computer Science, University of California at Berkeley, Berkeley CA 94720
satishr@berkeley.edu

Tandy Warnow[3]
Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, 61801, USA
warnow@illinois.edu

Abstract

Absolute fast converging (AFC) phylogeny estimation methods are ones that have been proven to recover the true tree with high probability given sequences whose lengths are polynomial in the number of number of leaves in the tree (once the shortest and longest branch lengths are fixed). While there has been a large literature on AFC methods, the best in terms of empirical performance was DCM_{NJ}, published in SODA 2001. The main empirical advantage of DCM_{NJ} over other AFC methods is its use of neighbor joining (NJ) to construct trees on smaller taxon subsets, which are then combined into a tree on the full set of species using a supertree method; in contrast, the other AFC methods in essence depend on quartet trees that are computed independently of each other, which reduces accuracy compared to neighbor joining. However, DCM_{NJ} is unlikely to scale to large datasets due to its reliance on supertree methods, as no current supertree methods are able to scale to large datasets with high accuracy. In this study we present a new approach to large-scale phylogeny estimation that shares some of the features of DCM_{NJ} but bypasses the use of supertree methods. We prove that this new approach is AFC and uses polynomial time. Furthermore, we describe variations on this basic approach that can be used with leaf-disjoint constraint trees (computed using methods such as maximum likelihood) to produce other AFC methods that are likely to provide even better accuracy. Thus, we present a new generalizable technique for large-scale tree estimation that is designed to improve scalability for phylogeny estimation methods to ultra-large datasets, and that can be used in a variety of settings (including tree estimation from unaligned sequences, and species tree estimation from gene trees).

2012 ACM Subject Classification Applied computing → Molecular evolution

Keywords and phrases phylogeny estimation, short quartets, sample complexity, absolute fast converging methods, neighbor joining, maximum likelihood

Digital Object Identifier 10.4230/LIPIcs.WABI.2018.8

[1] Supported by NSF grant 1535989
[2] Supported by NSF grant 1535989
[3] Supported by NSF grant 1535977

18th International Workshop on Algorithms in Bioinformatics (WABI 2018).
Editors: Laxmi Parida and Esko Ukkonen; Article No. 8; pp. 8:1–8:12

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

The inference of phylogenies from molecular sequence data is generally approached as a statistical estimation problem, in which a model tree (equipped with a model of sequence evolution) is assumed to have generated the observed data, and the properties of the statistical model are then used to infer the tree. Various statistical approaches can be applied for this estimation, including maximum likelihood, Bayesian techniques, and methods that operate by computing a distance matrix and then computing the tree from the distance matrix.

Many stochastic sequence evolution models have been developed, starting with the Cavender-Farris-Neyman [15, 6, 15] symmetric two-state model (referred to henceforth as "CFN") and including increasingly complex molecular sequence evolution models (with four states for DNA, 20 states for amino acids, and 64 states for codon sequences). However, typically the theory that can be established under the CFN model can also be established under the more complex molecular sequence evolution models used in phylogeny estimation.

Under standard sequence evolution models, many methods are known to be statistically consistent (meaning that they will provably converge to the true tree as the sequence lengths increase), including maximum likelihood [19] and many distance-based methods [1, 23]. A key challenge is to have methods that can scale to large datasets; hence, polynomial time methods are desirable. High accuracy is also needed, so methods that recover the true tree with high probability from limited numbers of sites are best. In this respect, methods that are "absolute fast converging" [8, 9, 25] (i.e., methods that recover the true tree with high probability from polynomial length sequences) are the most promising.

There are several methods that have been established to be absolute fast converging (AFC) under the CFN model, including maximum likelihood [19] (if solved exactly) and various distance-based methods [8, 9, 25, 14, 17, 18, 12, 4]. Some of these algorithms achieve a poly-logarithmic sample complexity but require a balanced model tree and an upper bound on g, the maximum edge length (defining the expected number of changes of a random site) in the CFN model. Specifically, methods based on reconstruction of ancestral sequences provide the best sample complexity bounds but cannot handle the case where g is larger than what is known as the Kesten-Stigum threshold, which is $\ln(\sqrt{2})$ [13] for the CFN model. Other methods, which are AFC in the regime where g is unbounded, are not guaranteed to recover a tree in certain situations [8, 9, 25]. Two of the fastest AFC methods are the Harmonic Greedy Triplets (HGT+FP) method by Csűrös [7] and a method developed by King et al. [11]; these methods use $O(n^2)$ time and are based on quartet trees. Another approach is $\texttt{DCM}_{NJ}$, which uses a divide-and-conquer technique [25]. In the first phase, $O(n^2)$ trees are computed, each based on dividing the sequence dataset into overlapping subsets, constructing trees on each subset using the polynomial time distance-based method neighbor joining (NJ) [20], and then combining the subset trees using a supertree method. This approach differs from other AFC methods in its use of neighbor joining to construct trees on subsets, whereas the other AFC methods in essence construct the tree by independently constructing quartet trees, and then assembling the quartet trees together using a quartet amalgamation method.

Very few AFC methods have been implemented; however, a study [14] comparing `HGT+FP` [7] and $\texttt{DCM}_{NJ}$ [25] showed that $\texttt{DCM}_{NJ}$ had better accuracy despite being slower. Since the theory does not predict this, the results on simulated data suggest that the use of `NJ` to construct trees on subsets and then combine the trees using a supertree method may be empirically advantageous compared to methods that combine quartet trees that are estimated independently. Unfortunately, the reliance on a supertree method to combine subset trees

means that $\texttt{DCM}_{NJ}$ is unlikely to scale to ultra-large datasets, because no current supertree method has shown the ability to maintain good accuracy and reasonable running times on large datasets [24].

The purpose of this paper is to describe a new polynomial time AFC phylogeny estimation method that should improve on $\texttt{DCM}_{NJ}$: it is designed to have comparable accuracy to $\texttt{DCM}_{NJ}$ but also to be able to analyze ultra-large datasets (i.e., more than 100,000 sequences). The basic approach of this new method is similar to $\texttt{DCM}_{NJ}$ in that it uses divide-and-conquer, applies neighbor joining to subsets of the sequence dataset, and then merges the subtrees together. However, it differs from $\texttt{DCM}_{NJ}$ in a few important ways, which we describe below. Most importantly, it divides the taxon set into disjoint subsets and then merges the subset trees without relying on any supertree method; thus, it avoids the challenge of relying on existing supertree methods, none of which are likely to scale to large datasets. We present the results here for the CFN model, but the extension to even complex DNA sequence evolution models is straightforward (e.g., see [9]). Our arguments for correctness are very similar to those in [7, 11]. The rest of the manuscript is organized as follows. We provide background material in Section 2. We present the basic method, INC, in Section 3, and its extension to allow for disjoint constraint trees (computed using various methods) in Section 4. We conclude with a discussion of future work in Section 5.

2 Background Material

2.1 Absolute Fast Convergence under the CFN Model.

Under the Cavender-Farris-Neyman (CFN) Model, we have a rooted binary tree T and substitution probabilities $p(e)$ on the edges e of T. The state at the root is 0 of 1 with equal probability, and the state changes on edge e with probability $p(e)$, with $0 < p(e) < 0.5$ for all edges e. This model can be used for sequence evolution by requiring that all the sites evolve *i.i.d.* down the tree. Finally, we define $w(e) = -\frac{1}{2}\log(1-2p_e)$. We also define $CFN_{f,g}$ to be the set of all CFN model trees (T, Θ) (where Θ denotes the set of numeric parameters on the edges) with $f \leq w(e) \leq g$ for all edges in T, for arbitrarily selected positive real numbers $f \leq g$.

▶ **Definition 1.** A phylogeny estimation method Φ is said to be absolute fast converging (**AFC**) under the CFN model if, for all positive values f, g, ϵ (with $f \leq g$), there is a polynomial p such that for all CFN model trees (T, Θ) in $CFN_{f,g}$ the method Φ will recover the unrooted tree topology T given sequences of length $p(n)$ with probability at least $1 - \epsilon$. Note that the polynomial will in general depend on f, g and ϵ.

2.2 DCM_{NJ}+SQS: an AFC method with good empirical performance but low scalability

Here we describe the approach used in [25] called "DCM_{NJ}+SQS", which was shown to have high accuracy in simulation studies with up to 1600 sequences [14]. (Note: "DCM" refers to "disk-covering method" and "SQS" refers to the "short quartet support" criterion; see [25, 23]). The input is a set of sequences generated by an unknown model tree and a dissimilarity matrix d (i.e., a symmetric matrix that is zero on the diagonal) where d_{ij} is the estimated distance between taxa s_i and s_j, based on the selected sequence evolution model. For example, when the model is CFN, then $d_{ij} = -\frac{1}{2}\ln(1-2H_{ij})$ will be the "empirical CFN distance", where H_{ij} is the Hamming distance between sequences i and j divided by the sequence length (i.e., the normalized Hamming distances). Note that these empirical

CFN distances converge in probability to $D_{ij} = -\frac{1}{2}\log(1-2E_{ij})$ where E_{ij} is the expected normalized Hamming distance between leaves i and j, and that D is an additive matrix for the model tree. $\texttt{DCM}_{NJ}$+SQS uses a two-phase structure, as follows:

- Phase 1: A set $\mathcal{T}$ of $O(n^2)$ trees is computed, with at most one tree t_q for each entry q in the dissimilarity matrix d.
- Phase 2: All the trees in $\mathcal{T}$ are scored using the SQS criterion (where "SQS" refers to the short quartet support, defined in [25]) and the best-scoring tree is returned.

Before we can describe these phases, we need to provide some definitions:

▶ **Definition 2.** (From [10].) For the given dissimilarity matrix d and positive real number q, we define the threshold graph $TG(d, q)$ to be the graph with the n taxa as the vertex set and edges (i, j) if and only if $d_{ij} \leq q$. We also assign weight d_{ij} to each edge (i, j) in $TG(d, q)$.

We use a standard technique, called the Four Point Method, to compute quartet trees (i.e., unrooted binary trees on four leaves) that is based on the Four Point Condition [5].

▶ **Definition 3.** (From [8]) Given a four taxa set $\{u, v, w, z\}$ and a dissimilarity matrix d, the Four Point Method (FPM) infers tree $uv|wz$ (meaning the quartet tree with an edge separating u, v from w, z) if $d(u, v) + d(w, z) \leq \min\{d(u, w) + d(v, z), d(u, z) + d(v, w)\}$. If equality holds, then the FPM infers an arbitrary topology.

Phase 1 for $\texttt{DCM}_{NJ}$+SQS is performed as follows. Given q (the selected threshold), a threshold graph $TG(d, q)$ is computed, then edges are added to the graph to make it triangulated (if necessary), where a triangulated graph is one that has no simple cycles of size four or more; furthermore, if d is additive, then $TG(d, q)$ is triangulated. Once the triangulated graph is computed, the set of all maximal cliques can be extracted in polynomial time. See [25] for additional details and proofs.

Trees are computed for each maximal clique using neighbor joining [20], and the trees on these cliques are then combined into a tree on the full set of species using a selected supertree method. Phase 2 uses the SQS criterion, but other criteria also have good theoretical properties. The SQS score of a tree T, defined by $SQS(T)$, is the maximum l such that for all quartets $\{u, v, w, x\}$ of taxa with diameter at most l, the Four Point Method on $\{u, v, w, x\}$ produces a four-leaf tree that agrees with the T.

▶ **Theorem 4** (From [25, 14])**.** *The short quartet support (SQS) criterion score can be calculated for each tree $t_q \in \mathcal{T}$ in polynomial time. Also, there is a polynomial $p(n)$ such that if $\mathcal{T}$ contains the true tree T and the sequences are of length $p(n)$ (where n is the number of leaves), then with probability at least $1 - \epsilon$, the tree in $\mathcal{T}$ with the highest SQS score is the true tree.*

The basic approach has good theoretical guarantees (i.e., it is AFC, and more generally if X is any exponentially converging base method (meaning X recovers the tree with high probability given exponentially many sites) and Y is any true tree selection criteria that has the same theoretical properties as SQS (as given in Theorem 4), then $\texttt{DCM}_X$+Y is AFC. Empirical performance of $\texttt{DCM}_{NJ}$+SQS on simulated data was excellent, substantially outperforming NJ and fast triplet/quartet-based greedy tree growing methods, such as HTP+FP, on large datasets especially when there is a high rate of evolution [14]. However, these studies were limited to datasets with at most 1600 sequences. In other words, $\texttt{DCM}_{NJ}$+SQS was not tested on very large datasets, which is to some extent the point of absolute fast converging methods.

Furthermore, the design of $\texttt{DCM}_{NJ}$ suggests some definite problems in terms of scalability to large datasets. Most importantly, supertree methods do not have good scalability, as all

current supertree methods with good accuracy are attempts to solve NP-hard optimization problems, and so become computationally intensive on large datasets [24]. Any reasonably fast method will need to completely avoid the supertree calculation step.

3 The basic technique: Incremental Tree-Building

We begin by describing the INC method in the unconstrained condition, which uses short quartet trees to build a tree, and we prove that the method is AFC. We then show how INC can be modified to take an arbitrary set of disjoint constraint trees, and we prove that when the constraint trees are constructed using an AFC method then the result is also AFC. We then present the $O(n^2)$ INC-NJ method, which is AFC and uses neighbor joining on carefully selected subsets to achieve this optimal running time. We then discuss the use of constrained-INC with maximum likelihood methods (or other computationally intensive but highly accurate methods) to construct the subset trees.

Throughout this section, we denote d_{ij} to be the empirical CFN dissimilarity between taxa i, j and D_{ij} to be the underlying CFN model distance between taxa i, j defined by the model tree (T, Θ). Given matrices d and D and positive real q, we define the $\epsilon(q) = \max\{|d_{ij} - D_{ij}| : d_{ij} \leq q \text{ or } D_{ij} \leq q\}$.

We now describe the fast incremental tree-growing method (INC) that inserts taxa in a specific order and outputs a binary tree. We note that the structure of our method is similar to that of HGT+FP. INC achieves a faster runtime when compared to DCM_{NJ} for two primary reasons. First, our method simply requires a minimum spanning tree of our dissimilarity matrix without the need to triangulate and compute maximal cliques, as well as iterate over multiple thresholding values. Second, our method uses the spanning tree as a guide for an incremental tree-growing method that runs in $O(n^2)$, as opposed to the slower runtimes required by NJ techniques. Together, this algorithm runs in the optimal $O(n^2)$ time.

The input is the dissimilarity matrix d (which is based on computing CFN distances, and so will converge to an additive matrix as the sequence length increases). We begin by constructing the minimum spanning tree S of $TG(d, \infty)$ and use S during an incremental tree growing procedure that inserts taxa one by one in an order such that the subgraph induced by the inserted vertices is connected in S. We let q_0 be the maximum distance of an edge in S. We will show that by setting $q = 8q_0$, we can develop a tree construction method based on this threshold q that runs in polynomial time and is AFC. Specifically, we order the vertices as follows.

Insertion Ordering: Choose an arbitrary leaf in S to be the starting vertex in the ordering. Then, order the vertices according to order of traversal in a BFS (or DFS) from the starting vertex.

We initialize our tree t by choosing the first three taxa according to our insertion ordering and forming the three-leaf tree with an internal node that connects to all three taxa. Now suppose we wish to add a vertex x into t. We will select the edge of t into which we add x by performing a collection of "short quartet queries" (see [8, 9] for definition and discussion of short quartets) at the different internal nodes of t, and then insert x into the edge that has the highest support (in the best case, it "agrees" with each query). Note that this node query insertion procedure was explored earlier in [3, 22] to produce a fast $O(n \log n)$ tree-growing algorithm, but analyzed under a different model of quartet tree error than we use here and a direct implementation of their model does not produce an AFC algorithm.

Given an internal node u of t, because t is binary the removal of u splits t into three non-empty vertex components t_1, t_2, t_3 (with internal nodes included). Because the leaves of t form a connected induced subtree of S, we can find some taxon u_i in t_i such that (u_i, v_i) is an edge in S and $v_i \in V(t) \setminus t_i$ is also a taxon, where $V(t)$ is the vertex set of t.

▶ **Definition 5.** Let u be an internal node of a binary tree t and let u_1, u_2, u_3 be leaves in the three components of t upon removing u such that v_i is not in the same component as u_i and there exists v_i with (u_i, v_i) an edge in S. Then, for some choice of q, a quartet query on $\{u_1, u_2, u_3, x\}$ is **q-valid** iff the d-diameter (maximum pairwise distance with respect to the input matrix d) of $\{u_1, u_2, u_3, x\}$ is less than q. We use **valid** when q is clear from context.

By looking at all the valid queries and the corresponding quartet trees calculated using the Four Point Method, we can choose an edge e in the tree t on which to add x that maximizes the support for the placement; we then subdivide e and then make x adjacent to the node created. To choose an edge, each valid short quartet query identifies some subset of edges of t into which x should be placed. Specifically, we assign votes according to a valid query as follows:

Vote Assignment: If the valid query on internal node u shows that t_i and x should be on the same side of the quartet using the Four Point Method, then all edges in the induced subtree on $t_i \cup u$ will receive a vote.

The support for an edge e is then the total number of votes it receives from all the short quartets, and the edge that receives the highest number of votes wins. If there is a tie, then one of the edges receiving the most votes is selected (randomly). However, for favorable values of d and q, there will be a unique edge on which all queries agree, and so the algorithm will correctly add x into t (in such a way that it agrees with the model tree T), and so inductively the greedy algorithm will construct the model tree. In particular, we show that for our choice of $q = 8q_0$, if $\epsilon(q) < f/2$, then there is always a unique edge chosen by this voting procedure, and furthermore that unique edge is the correct edge. Hence, this greedy algorithm will correctly reconstruct the true tree.

$t =$ `INC`(d)
1. Calculate a spanning tree S of $TG(d, \infty)$.
2. Set $q = 8q_0$, where q_0 is the maximum edge length of S.
3. Find the insertion ordering according to a `BFS` of S.
4. Initialize t as the three-leaf tree on the first three taxa in the ordering.
5. Insert each taxon according to the ordering on the edge with the most votes from valid queries.
6. Return the resulting tree t.

▶ **Theorem 6.** *Let d be a dissimilarity matrix, D an additive matrix defining a model tree T with edge weights $w : E(T) \to \mathbb{R}^+$, and $q = 8q_0$, where q_0 is the maximum distance in the minimum spanning tree of $TG(d, \infty)$. Let f be the length of the shortest internal edge in T and suppose $\epsilon(q) < f/2$ and $q_0 \geq f/2$, then `INC`(d) returns T.*

Proof. We proceed by induction on the size of T. Our claim holds when T is size 3, since there is only one such topology. Let t be the tree maintained by `INC` before the insertion of the last taxon x according to our insertion ordering. By induction, t must have the correct topology. It suffices to show that valid queries can determine the accurate placing of x. Assume that the correct location on which to place x is $e = (u, v)$ of t. Note that when

$\epsilon(q) < f/2$, all valid node queries will return the correct quartet tree. Therefore, e will receive all possible votes.

To show that all other edges will miss at least one vote, it suffices to show that node queries are valid at u, v because all other edges will miss a vote from one of the two such queries, confirming that x is inserted at e (if one of u, v is a leaf vertex, the same conclusion is achieved).

We will show that a node query at u is valid; a similar argument is done for v. Let u_1, u_2, u_3 be selected upon the deletion of u with their corresponding v_i. First, for all i, $D(u_i, v_i) \leq q_0 + f/2$ since $d(u_i, v_i) \leq q_0$ and $\epsilon(q) \leq f/2$. Then, since $v_i \in V \setminus t_i$ and t is the correct tree topology, $D(u_i, u) \leq D(u_i, v_i) \leq q_0 + f/2$, where D is the true distance matrix and u is the corresponding internal node in T.

Next, we claim that $D(x, u) \leq 2q_0 + f$ and together we will have concluded that $\{u_1, u_2, u_3, x\}$ is a quartet of d-diameter at most $3q_0 + 2f$ since by triangle inequality, we deduce $\{u_1, u_2, u_3, x\}$ is of D-diameter $3q_0 + 3f/2$ and so the d-diameter is bounded at $3q_0 + 2f \leq 7q_0 \leq q$, proving validity.

For the last claim, since x is within q_0 of some leaf x' in t, $D(x, x') \leq d(x, x') + f/2 \leq q_0 + f/2$. Since the path x to x' must pass through u or v in T and $e = (u, v)$ is the correct edge insertion location, we conclude that $\min(D(x, u), D(x, v)) \leq q_0 + f/2$.

If $D(x, u) \leq q_0 + f/2$, then our claim follows (we automatically get $D(x, u) \leq 2q_0 + f$). Else, $D(x, v) \leq q_0 + f/2$. Since $v \in V(t_3)$, $D(x, u) \leq D(x, v) + D(u, v) \leq q_0 + f/2 + D(u, v) \leq q_0 + f/2 + D(u, u_3) \leq q_0 + f/2 + q_0 + f/2 \leq 2q_0 + f$.

Therefore, if the correct edge is u, v, then the query at u votes against all edge placements for x in $t_1 \cup t_2$, and symmetrically for v. Hence all other edges will miss at least one vote. ◀

▶ **Theorem 7.** *Given the $n \times n$ dissimilarity matrix d and a given q, `INC`(d) can be implemented in $O(n^2)$.*

Proof. It suffices to show that as we grow our tree t, each insertion step can be implemented to take $O(n)$ time. First, each internal node, when initialized due to an inserted taxon, can store the necessary vertices u_1, u_2, u_3 and that can be done in $O(n)$ time. Each node query is $O(1)$, so in total the $|V(t)|$ node queries take $O(n)$ time.

The only possible difficulty is efficiently finding the edge with the most votes. A naive implementation will lead to an $O(n^3)$ runtime. For any edge $e = (u, v)$ on the tree, let $n(e)$ denote the total number of votes that e has. Note that if e, e' are adjacent edges in t with common vertex u , then $n(e) - n(e')$ can be determined by simply looking at the short quartet query at u. Specifically, let $e = (u, v)$ and $e' = (u, v')$ be adjacent edges at vertex u. If the query at u was invalid or it showed that x should be in t_j with $v, v' \notin t_j$, then $n(e) - n(e') = 0$. Otherwise, if the query returned t_j with $v \in t_j$, then $n(e) - n(e') = 1$; a similar argument shows that if $v' \in t_j$ then $n(e) - n(e') = -1$. Therefore, by using this local property, we can calculate $n(e) + C$ for some constant C in $O(n)$ time by performing `BFS` starting from any leaf of the tree. The `BFS` then simply returns the edge with the highest score. ◀

▶ **Theorem 8.** *[2](Azuma's Inequality) Suppose $X = (X_1, X_2, ..., X_k)$ are independent random variables taking values in any set S, and let $L : S^k \to \mathbb{R}$ be any function that satisfies the condition: $|L(u) - L(v)| \leq t$ whenever u and v differ in just one coordinate. Then,*

$$P(|L(X) - E[L(X)]| \geq \lambda) \leq 2\exp(-\frac{\lambda^2}{2t^2k}).$$

▶ **Theorem 9.** *For $k = \Omega(\frac{\ln(n)e^{4q}}{f^2})$, with high probability, we have $\epsilon(q) < f/2$ for all $q = O(g \log n)$. Furthermore, if q_0 is the minimum value of q such that $TG(d, q)$ is connected, then $q_0 = O(g \log n)$. and $q_0 \geq f/2$.*

Proof. To show that $\epsilon(q) < f/2$, we must show that $|D_{ij} - d_{ij}| < f/2$ when $D_{ij} < q$ or $d_{ij} < q$. First, if $D_{ij} < q$ we show that $k = \Omega(\frac{\ln(n)e^{2q}}{f^2})$ suffices to show $|D_{ij} - d_{ij}| < f/2$ with high probability. We express our probability of failure as:

$$\begin{aligned} P(|D_{ij} - d_{ij}| \geq f/2) &= P(|\log(\frac{1-2H_{ij}}{1-2E_{ij}})| \geq f) \\ &\leq P((1-2E_{ij})e^{-f} \geq 1-2H_{ij}) + P(1-2H_{ij} \geq (1-2E_{ij})e^{f}) \end{aligned}$$

The first expression can be written as:

$$\begin{aligned} P(H_{ij} - E_{ij} \geq \frac{1}{2}(1-e^{-f})(1-2E_{ij})) &\leq P(H_{ij} - E_{ij} \geq \frac{1}{2}(1-e^{-f})e^{-2D_{ij}}) \\ &\leq 2\exp(-\Omega(k(1-e^{-f})^2 e^{-2q})) \\ &\leq 2\exp(-\Omega(\ln n)) \end{aligned}$$

The second line follows from Azuma's with $t = 1/k$. Similarly, the second expression is equivalent to:

$$\begin{aligned} P(E_{ij} - H_{ij} \geq \frac{1}{2}(e^{f}-1)(1-2E_{ij})) &\leq P(E_{ij} - H_{ij} \geq \frac{1}{2}(e^{f}-1)e^{-2D_{ij}}) \\ &\leq 2\exp(-\Omega(k(e^{f}-1)^2 e^{-2q})) \\ &\leq 2\exp(-\Omega(\ln n)) \end{aligned}$$

Next, if $d_{ij} < q$, then we show that $D_{ij} < 2q + 1$ with high probability and then apply the previous result with $q' = 2q + 1$. First, if we let $r_{ij} = D_{ij} - q$, then by simple algebra our probability that $d_{ij} < q$ when $r_{ij} > q + 1$ is bounded by

$$\begin{aligned} P(d_{ij} < q) &= P(D_{ij} - d_{ij} > D_{ij} - q) \\ &= P(-\frac{1}{2}\log(\frac{1-2E_{ij}}{1-2H_{ij}}) > D_{ij} - q) \\ &= P(1-2H_{ij} > e^{2r_{ij}}(1-2E_{ij})) \\ &= P(E_{ij} - H_{ij} > \frac{1}{2}(e^{2r_{ij}}-1)e^{-2D_{ij}}) \\ &= P(E_{ij} - H_{ij} > \frac{1}{4}e^{2r_{ij}-2D_{ij}}) \\ &\leq 2\exp(-\Omega(ke^{-2q})) \\ &\leq 2\exp(-\Omega(\ln n)) \end{aligned}$$

The fourth to fifth line follows since $e^{2r_{ij}} - 1 > \frac{1}{2}e^{2r_{ij}}$ whenever $r_{ij} > 1$. Therefore, we conclude our claim that $\epsilon(q) < f/2$ with high probability.

We now show that $q_0 = O(g \log n)$. Note that in our model tree, since g is the maximum length of an edge in a binary tree with n leaves, it follows that $TG(D, O(g \log n))$ is connected. By our previous part, we know that $|d_{ij} - D_{ij}| < f/2$ for all edges in $TG(D, O(g \log n))$. Therefore, we conclude that $TG(d, O(g \log n))$ is also connected, and so $q_0 = O(g \log n)$.

Lastly, we show $q_0 \geq f/2$. Consider all edges in the minimum spanning tree of $TG(d, \infty)$. These edges have true length (in D) at least f and by our previous part the length of the edges in the minimum spanning tree deviate from the true length (as defined by the model tree) by at most $f/2$ with high probability. Thus, we conclude that $q_0 \geq f/2$. ◀

▶ **Theorem 10.** *INC is absolute fast converging for the CFN model and takes $O(n^2)$ time (assuming distances are precomputed).*

Proof. All we need to establish is that for every triplet ϵ, f, g with $0 < f < g < \infty$ and $\epsilon > 0$, there is a polynomial $p(n)$ such that for all model trees (T, Θ) in $CFN_{f,g}$, given sequences of length at least $p(n)$ generated by (T, Θ) and empirical CFN dissimilarity matrix d computed on these sequences, the tree returned by $\texttt{INC}(d)$ is the model tree T with probability at least $1 - \epsilon$.

Let q_0 be the maximum edge length on the minimum spanning tree. By Theorem 9, with high probability, we know $q_0 = O(g \log n)$ and thus $q = 8q_0 = O(g \log n)$. Thus, when $k = \Omega(\ln(n)e^{4q}/f^2) = poly(n)$, we have $\epsilon(q) < f/2$. Furthermore, $q_0 \geq f/2$. Therefore, by Theorem 12, the insertion algorithm will return the model tree T. Hence, INC is absolute fast converging for the CFN model.

Finally, given the dissimilarity matrix, the running time to compute the constraint trees andthe minimum spanning tree and to run the insertion algorithm are all $O(n^2)$ by Theorem 7. ◀

The method we described runs in $O(n^2)$ time and is AFC. As shown by [11], any algorithm that reconstructs the true tree with high probability and uses distance calculations as its only source of information about the phylogeny on sequences of length $O(poly \log n)$ will have $\Omega(n^2)$ runtime, up to logarithmic factors. Hence, our runtime is optimal.

4 Boosting INC using constraint trees

We show how INC can be modified to take an arbitrary set of disjoint constraint trees, and we prove that when the constraint trees are constructed using an AFC method then the result is also AFC. We present several variants of this method: one that optimizes speed while still guaranteeing that the resultant algorithm is AFC, and other AFC variants that are designed for improved accuracy but with an increase in running time.

4.1 Constrained INC

The input to the constrained version of INC includes a set of leaf-disjoint trees. Therefore, the constraint set is compatible (i.e., a compatibility tree exists). We will describe a straightforward modification to INC so that it never violates the topological information in the constraint trees, by which we mean only that the final output tree is required to induce each constraint tree, when restricted to that specific set of leaves. Thus, the constraint trees are not required to be clades in the output tree.

Before proceeding, we define the **induced tree topology**, t^A, on a tree t and a subset of leaves A as follows. Consider the minimal subtree of t that contains the leaves A, and then suppress all nodes of degree two (i.e., replace all maximal paths of degree two nodes by a single edge). For an edge $e \in t^A$, note that its endpoints correspond to vertices in t, and that e corresponds to a (possibly length 1) path in t; we denote the corresponding endpoints of e as $e_t(u)$ and $e_t(v)$. For an induced tree t^A, the **component in t corresponding to an edge** $e \in t^A$ is the set of edges and vertices that can be reached by a walk starting from $e_t(u)$ and ending at $e_t(v)$ without using $e_t(u)$ or $e_t(v)$ multiple times. Note this will be a subgraph of t that includes $e_t(u)$ and $e_t(v)$ as leaves.

When working with constraint trees, we alter the insertion algorithm to account for the constraints. Recall that the constraint trees are on disjoint sets of taxa. Thus, when inserting a taxon, we identify the corresponding constraint tree t_c that includes that new taxon. Let

A be the taxa shared between t_c and t and note that $t_c^A = t^A$ since the trees are consistent. Thus, in $t_c^{A\cup\{x\}}$, the taxon x can be viewed as being attached to some edge e in t^A. The eligible edges are in the component in t corresponding to e. We then simply run `INC` on the component to find the desired edge of insertion.

4.2 INC-NJ

We continue by presenting the $O(n^2)$ `INC-NJ` method, which is AFC and uses neighbor joining on carefully selected subsets to achieve this running time. Given the selected threshold q, we take our threshold graph $TG(d, q)$ and compute a greedy disjoint clique decomposition, such that each clique is of size $O(\sqrt{n})$. This is done by a simple ball-growing procedure that is interrupted when the ball includes more than $O(\sqrt{n})$ vertices; it is easy to see this takes $O(n^2)$ time. Next, we run `NJ` on each of these clique; this produces a set of neighbor joining subset trees that will serve as our constraint trees. Note that since the subsets are disjoint, the subset trees are compatible (in that a compatibility supertree exists). Finally, if $\epsilon(q) < f/2$, then the following theorem guarantees that all constraint trees produced by our technique are correct. Furthermore, since `NJ` applied to each clique of size $O(\sqrt{n})$ takes at most $O(n^{1.5})$ time, our total runtime over all cliques is $O(n^2)$.

▶ **Theorem 11.** *[From [1]] Neighbor Joining (`NJ`) will recover the true tree T whenever the input matrix $\mathbf{d}$ satisfies $L_\infty(d, D) < f/2$, where D is an additive matrix defining an edge-weighted version of the true tree T and f is the shortest internal branch length in T. Furthermore, the runtime is $O(n^3)$.*

▶ **Theorem 12.** *`INC-NJ` is AFC and takes $O(n^2)$ (assuming distances are precomputed).*

Proof. Follows directly from combining Theorem 10 and 11. ◀

4.3 Boosting INC using other constraint trees

The version of constrained-`INC` we presented in the previous section achieves an optimal running time, but is based on obtaining constraint trees using neighbor joining on small subsets. However, we could modify the decomposition to produce larger subsets and thus take advantage of the likely improvement in accuracy obtained by using neighbor joining on these larger subsets to produce the constraint trees. This would still produce a polynomial time algorithm, but one with a higher (and hence suboptimal) running time.

Another variation would be to use other phylogeny estimation methods besides neighbor joining. In particular, maximum likelihood could be used to construct the constraint trees. As shown in [19], maximum likelihood under the CFN model is AFC. Hence, running maximum likelihood on any subsets of the taxon set will provide correct subset trees from polynomial length sequences with high probability. Hence, the same approach we describe for use with neighbor joining (where that subset trees are computed using neighbor joining) can be modified to be used with maximum likelihood, and will be an AFC method. Furthermore, because maximum likelihood is AFC, we can afford to use maximum likelihood on arbitrarily large subsets, without losing the overall AFC property (as we show below); the only negative impact of doing this would be running time, since maximum likelihood is NP-hard [16] and heuristics for maximum likelihood are computationally more intensive than neighbor joining.

▶ **Theorem 13.** *If the disjoint constraint trees are estimated by a method that is AFC under the CFN model, then constrained `INC` with the estimated constraint trees is AFC under the CFN model. Hence, using maximum likelihood to produce constraint trees and combining the constraint trees using `INC` is AFC under the CFN model.*

Proof. Let Φ be an AFC method, and fix ϵ, f, g and some CFN model tree (T, Θ). Hence, we can find a polynomial $p_1(n)$ (where the degree may depend on f, g) such that given sequences of length $p_1(n)$ guarantees that with probability at least $1 - \epsilon$, all the constraint trees computed using Φ are correct. Then, by Theorem 10, we can find $p_2(n)$ (where the degree may depend on f, g) such that `INC` returns the model tree with probability at least $1 - \epsilon$ given sequences of length $p_2(n)$. Hence, given sequences of length $\max\{p_1(n), p_2(n)\}$, constrained `INC` returns the model tree with probability at least $1 - 2\epsilon$. Thus, constrained `INC` is AFC. By [19], maximum likelihood is AFC. Hence, using `INC` with disjoint maximum likelihood constraint trees is AFC, for all ways of decomposing the dataset into disjoint subsets. ◀

5 Discussion

This paper presents a novel algorithmic technique for constructing phylogenetic trees, which allows statistically consistent and highly accurate methods to be used on subsets of the taxa in a divide-and-conquer framework. We proved that several of these variants are absolute fast converging (AFC) under the CFN sequence evolution model, and that some of these achieve $O(n^2)$ time (where n is the number of sequences) once the dissimilarity matrix relating the sequences is computed. Although this theory was presented for the CFN model, the extension to the Jukes-Cantor DNA sequence evolution models is trivial (i.e., just substitute the CFN distances by Jukes-Cantor distances), and equally simple for more complex models, provided that statistically consistent distance estimators exist (see discussion in [23]). Fortunately, all the standard sequence evolution models, including the General Markov Model [21], have such distance estimators. More generally, tree estimation methods could be scaled to large datasets using this approach, provided that it is possible to compute a matrix of pairwise distances that is guaranteed to converge to an additive matrix as the amount of data increases. The goal of this work was improved empirical performance, compared to prior AFC methods. Thus, additional work is needed to evaluate the empirical benefits of this approach in these various applications.

References

1. Kevin Atteson. The performance of neighbor-joining methods of phylogenetic reconstruction. *Algorithmica*, 25(2-3):251–278, 1999.
2. Kazuoki Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal, Second Series*, 19(3):357–367, 1967.
3. Daniel G Brown and Jakub Truszkowski. Fast error-tolerant quartet phylogeny algorithms. In *Annual Symposium on Combinatorial Pattern Matching*, pages 147–161. Springer, 2011.
4. Daniel G Brown and Jakub Truszkowski. Fast phylogenetic tree reconstruction using locality-sensitive hashing. In *Workshop on Algorithms for Bioinformatics (WABI)*, pages 14–29. Springer, 2012.
5. Peter Buneman. A note on the metric properties of trees. *Journal of Combinatorial Theory (B)*, 17:48–50, 1974.
6. James A Cavender. Taxonomy with confidence. *Mathematical biosciences*, 40(3-4):271–280, 1978.
7. Miklós Csűrös. Fast recovery of evolutionary trees with thousands of nodes. *Journal of Computational Biology*, 9(2):277–297, 2002.
8. Peter L. Erdös, Michael A. Steel, Laszlo Székely, and Tandy Warnow. A few logs suffice to build (almost) all trees (i). *Random Structures and Algorithms*, 14:153–184, 1999.

9 Peter L. Erdös, Michael A. Steel, Laszlo Székely, and Tandy Warnow. A few logs suffice to build (almost) all trees (ii). *Theoretical Computer Science*, 221:77–118, 1999.

10 Daniel Huson, Scott Nettles, Laxmi Parida, Tandy Warnow, and Shibu Yooseph. The disk-covering method for tree reconstruction. In *Algorithms and Experiments (ALEX)*, pages 62–75, 1998.

11 Valerie King, Li Zhang, and Yunhong Zhou. On the complexity of distance-based evolutionary tree reconstruction. In *14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 444–453. Society for Industrial and Applied Mathematics, 2003.

12 Radu Mihaescu, Cameron Hill, and Satish Rao. Fast phylogeny reconstruction through learning of ancestral sequences. *Algorithmica*, 66(2):419–449, 2013.

13 Elchanan Mossel. Phase transitions in phylogeny. *Transactions of the American Mathematical Society*, 356(6):2379–2404, 2004.

14 Luay Nakhleh, Usman Roshan, Katherine St. John, Jerry Sun, and Tandy Warnow. Designing fast converging phylogenetic methods. *Bioinformatics*, 17(suppl_1):S190–S198, 2001.

15 Jerzy Neyman. Molecular studies of evolution: a source of novel statistical problems. In *Statistical decision theory and related topics*, pages 1–27. Elsevier, 1971.

16 Sébastien Roch. A short proof that phylogenetic tree reconstruction by maximum likelihood is hard. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(1):92–94, 2006.

17 Sébastien Roch. Sequence-length requirement for distance-based phylogeny reconstruction: Breaking the polynomial barrier. In *Proc. of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 729—-738, 2008.

18 Sébastien Roch. Towards extracting all phylogenetic information from matrices of evolutionary distances. *Science*, 327(5971):1376–1379, 2010.

19 Sébastien Roch and Allan Sly. Phase transition in the sample complexity of likelihood-based phylogeny inference. *Probability Theory and Related Fields*, 169(1):3–62, Oct 2017.

20 Naruya Saitou and Masatoshi Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4):406–425, 1987.

21 Michael A. Steel. Recovering a tree from the leaf colourations it generates under a Markov model. *Applied Mathematics Letters*, 7(2):19 – 23, 1994.

22 Jakub Truszkowski, Yanqi Hao, and Daniel G Brown. Towards a practical O(n log n) phylogeny algorithm. *Algorithms for Molecular Biology*, 7(1):32, 2012.

23 Tandy Warnow. *Computational Phylogenetics: An introduction to designing methods for phylogeny estimation.* Cambridge University Press, 2018.

24 Tandy Warnow. Supertree construction: Opportunities and challenges. *arXiv:1805.03530 [q-bio.PE]*, 2018.

25 Tandy Warnow, Bernard M.E. Moret, and Katherine St. John. Absolute convergence: true trees from short sequences. In *Proceedings of SODA*, pages 186–195. ACM, 2001.