Fast and Reliable Reconstruction of Phylogenetic Trees with Indistinguishable Edges*

Ilan Gronau† Shlomo Moran‡ Sagi Snir§

February 4, 2011

Abstract

Phylogenetic reconstruction methods attempt to reconstruct a tree describing the evolution of a given set of species using sequences of characters (e.g. DNA) extracted from these species as input. A central goal in this area is to design algorithms which guarantee reliable reconstruction of the tree from short input sequences, assuming common stochastic models of evolution. The *fast converging* reconstruction algorithms introduced in the last decade dramatically reduced the sequence length required to guarantee accurate reconstruction of the entire tree. However, if the tree in question contains even few edges which cannot be reliably reconstructed from the input sequences, then known fast converging algorithms may fail to reliably reconstruct all or most of the other edges. This calls for an adaptive approach suggested in this paper, called *adaptive* fast convergence, in which the set of edges which can be reliably reconstructed gradually increases with the amount of information (length of input sequences) available to the algorithm.

This paper presents an adaptive fast converging algorithm which returns a partially resolved topology containing no false edges: edges that cannot be reliably reconstructed are contracted into high degree vertices. We also present an upper bound on the weights of those contracted edges, which is determined by the length of input sequences and the depth of the tree. As such, the reconstruction guarantee provided by our algorithm for individual edges is significantly stronger than any previously published edge reconstruction guarantee. This fact, together with the optimal complexity of our algorithm (linear space and quadratic-time), makes it appealing for practical use.

1 Introduction.

Phylogenetic reconstruction is the task of figuring out the evolutionary history of a given set of extant species (terminal taxa). This history is usually described by an undirected tree whose internal vertices represent past speciation events (extinct species) and whose leaves correspond to the given set of taxa; edges of the tree are typically identified with the *splits* they define on the terminal taxa. Each such edge is associated with a weight (or length) which describes the amount of evolutionary change along the edge (often measured by the expected number of mutations). Reconstruction methods typically receive as input an alignment of sequences, each corresponding to a different taxon, and they are expected to yield a tree which closely depicts the true phylogenetic tree.

The connection between the length of input sequences and the accuracy of the reconstructed tree was first studied in the seminal work of Erdös et al. in [11]. They present an

*A preliminary version of this paper appears in [15].

†Department of Computer Science, Technion - Israel Institute of Technology, Haifa, 32000Israel

‡Department of Computer Science, Technion - Israel Institute of Technology, Haifa, 32000 Israel

§Department of Evolutionary and Environmental Biology and The Institute of Evolution, University of Haifa Mount Carmel, Haifa 31905 Israel

efficient algorithm that, with high probability, reconstructs any phylogenetic tree accurately from input sequences of length polynomial in the number of terminal taxa, provided that the minimal and maximal edge weights are bounded by constants. The main result of [11] is based on a detailed study of the sequence length required for accurate reconstruction (w.h.p.) of a *quartet* (a tree spanning four terminal taxa). The paper develops what turns out to be a tight connection (up to constants) between this sequence length, the diameter of the quartet, and the weight of its single internal edge. This basic analysis for quartets is extended to larger trees by referring to the *depth* of an edge e, which is the *minimal* diameter of a quartet of terminal taxa which are separated by e. The algorithm presented in [11] is the first algorithm which depends on the depth of the tree (i.e., the maximal depth of an edge), rather than its diameter. [11] led to a long line of algorithms using a similar approach, which are now referred to as *fast converging* phylogenetic reconstruction algorithms, because they guarantee fast convergence to the true tree (as a function of the sequences length).

The problem with nearly all fast converging algorithms is that they share an "all or nothing" nature: if the input sequences allow reliable reconstruction of all edges in the tree, then these algorithms guarantee accurate reconstruction (w.h.p.) of the tree; but if the tree contains some *indistinguishable edges*, then they do not provide any reconstruction guarantee. An indistinguishable edge is an edge for which accurate reconstruction cannot be guaranteed statistically (w.h.p.) from input sequences of some given length. This can be either because the edge is "too short" (few mutations are likely to have occurred along it) and/or because it is "too deep" (the mutations that occurred along it are likely to have been overwritten by later mutations). The stated objective of most fast converging algorithms is to reconstruct the entire tree from input sequences of minimal length, and not to reconstruct as much of the tree as possible from sequences of any length. Hence these algorithms are designed to ignore cases where the reconstructed tree has even a few indistinguishable edges.

To illustrate the issue of indistinguishable edges, consider the phylogenetic tree T in Fig. 1, which contains a terminal taxon x whose point of speciation v is very close to another speciation point u, implying that the edge (u, v) is very short. Assume an input sequence length k, for which only the edge (u, v) is indistinguishable. It is reasonable to expect that a phylogenetic reconstruction algorithm will correctly reconstruct all other edges of T, even if it fails to reconstruct (u, v). However, this is not the case with the fast converging algorithms which preceded this work: when failing to reconstruct an edge, some of these algorithms do not return any tree, while others return a tree that may contain many faulty edges[1]. This is despite the fact that when executed on all the input sequences but that of x, they do return the correct subphylogeny (w.h.p.). Note that a similar effect occurs when the edge (u, v) is not very short, but the terminal taxon x lies on a very long terminal edge (v, x), implying that both (u, v) and (u', v) are indistinguishable because they are deep. When reconstructing very large trees we are likely to encounter such cases of close-by speciation events and/or long terminal edges, and so it is important to use a reconstruction algorithm which is capable of dealing with indistinguishable edges.

The inability of fast converging algorithms to deal with even a few indistinguishable edges has been pointed out in the past [17, 25]. In this paper we present the first fast converging algorithm which returns a tree which is guaranteed (w.h.p.) to contain all edges above a certain weight. This weight threshold depends on the input sequence length and the depth of the tree. This is done by contracting indistinguishable edges and returning a tree which might contain vertices of degree greater than 3. The approach of edge contraction dates back to Buneman's algorithm [2], however, using fast converging techniques enables our algorithm to reconstruct edges which are much shorter than the ones reconstructed by Buneman's algorithm. Another practical feature of this algorithm is its optimal time complexity of $O(n^2)$ (where n is the number of terminal taxa).

[1] Previous papers on fast converging algorithms, e.g., [5, 6, 11, 12, 17, 20, 23, 24], do not explicitly describe their behavior in the presence of indistinguishable edges, and this observation is based on our analysis of these algorithms.

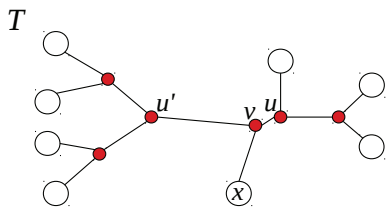

Figure 1: **A tree with a single indistinguishable edge.**

1.1 Background and Related Work.

All fast converging reconstruction algorithms (and in fact most efficient tree reconstruction algorithms) are distance-based, implying that they reconstruct the tree according to estimates of distances between terminal taxa (i.e., estimates of edge-weight sums along paths connecting leaves in the tree). However, unlike other distance-based algorithms, fast converging algorithms typically use only a subset of the $\binom{n}{2}$ pairwise distances (n denotes the number of terminal taxa). Erdös et al. initiated the study of fast converging algorithms in [11] by making the following two observations:

- In order to get any non-trivial (i.e., finite) bound on the error of distance estimation (under common stochastic models of sequence evolution), the length of input sequences needs to be exponential in the estimated distance.
- In order to reconstruct a tree from (exact) distances, it is sufficient to query distances which are typically much smaller than the diameter of the tree. The maximum distance one has to query in order to reconstruct a given tree is proportional to the *depth* of the tree (see Definition 5.1)[2]. This is significant since the depth of a tree is typically much shorter than its diameter: assuming all edges are shorter than some global constant g, the depth of a phylogenetic tree spanning n terminal taxa is bounded by $O(g\log(n))$, whereas its diameter is bounded by $O(gn)$.

Combining these two observations together, the authors of [11] designed the first algorithm which guarantees accurate reconstruction (w.h.p.) of a tree whose edge weights are bounded within the interval $[f, g]$ from sequences of length

$$k \;=\; \frac{\log(n)e^{O(depth(T))}}{f^2} \;=\; \frac{n^{O(g)}}{f^2}\;. \tag{1}$$

Treating f, g as constants, the algorithm of [11] guarantees w.h.p. a correct reconstruction of the tree from input sequences of length *polynomial* in the number of terminal taxa n. This property was later termed as *fast convergence*. It is important to note that all previously known sequence length bounds were exponential – see, e.g., [1]. In fact, Neighbor Joining [30], which is one of the most popular distance-based algorithms in use today, was shown recently to actually require (in some cases) input sequences which are exponentially long in n [21]. In attempts to minimize sequence length requirements for reconstruction of the entire tree, a related line of ongoing research is dedicated to avoiding the exponential dependence in the depth of the tree. This is typically achieved by reconstructing ancestral sequences in the tree [24, 9, 22] or by some distance-averaging which imitates reconstruction of ancestral sequences [29]. However, all these results strictly require that all edge weights do

[2]Our definition of depth is similar but slightly different from the one used in [11] in similar context.

not exceed a *certain specific threshold*. It was shown in [24], that when weights are allowed to be above this threshold, the exponential dependence in the depth of the tree (as expressed in (1)) cannot be avoided. Other lower bounds [31, 26] show that the sequence length required for correct reconstruction of an internal edge of weight w has to be proportional to $\frac{1}{w^2}$ – indicating that the dependence of k in f in (1) is asymptotically optimal.

A long series of follow up research triggered by [11] (e.g., [17, 5, 6, 20]) propose different fast converging algorithms which share the same idea of querying the smaller distances, but differ in the actual technique used to infer tree topology from the queried distances. Consequently, they vary in time/space complexity and also in the prior knowledge they require on properties of the tree, such as upper or lower bounds on edge weights. Fast converging algorithms which require no such prior knowledge are commonly referred to as being *absolute fast converging* [23].

As mentioned earlier, the main drawback of all fast converging algorithms is that they perform poorly on trees with indistinguishable edges. One possible way to deal with such trees is to relax the original requirement of returning a tree which spans *all* input taxa, and return a (fully resolved) tree which spans only a subset of these taxa. If the subphylonegy spanning this subset contains no indistinguishable edges, then it can be reconstructed reliably (for instance, in the tree depicted in Fig. 1, the subphylogeny spanning all terminal taxa other than x). This approach increases the number of reconstructed edges at the price of each reconstructed edge defining a split only on a subset of the input taxa. A nice extension of this idea was made in the forest reconstruction approach of [25, 8]. This approach suggests to reconstruct (distinguishable) edges of the tree from its leaves inwards, stopping when encountering an indistinguishable edge. If the input sequences are too short to reliably reconstruct the entire phylogeny, these algorithms return a forest of fully resolved subphylogenies portraying the periphery of the tree. Still, the presence of shallow indistinguishable edges in the tree may prevent these algorithms from reconstructing deeper edges, for which there is clear statistical support.

Another way to deal with indistinguishable edges is to return a partially-resolved version of the tree, with the indistinguishable edges contracted into high-degree vertices. There are several algorithms which contract indistinguishable edges, such as Buneman's algorithm [2], however, these algorithms also contract many distinguishable edges. As a result, they typically return very poorly resolved trees. The problem is that these algorithms guarantee reliable reconstruction of an edge e of weight $w(e)$, only from sequences of length

$$k \;\geq\; \frac{\log(n)e^{\theta(diam(T))}}{w(e)^2}\;, \tag{2}$$

where $diam(T)$ denotes the diameter of T [1]. The fast converging approach suggests that an algorithm which avoids querying distances which are much larger than the tree depth could potentially bring the required sequence length to be exponential in the *depth* of the tree (as in (1)) rather than its *diameter* (as in (2)). This hypothesis has been raised few times in the past (the first time was probably in [17]), and in this paper we present an algorithm which realizes it.

1.2 Our Results.

An *adaptive fast converging* algorithm is a fast converging algorithm which guarantees reconstruction of individual edges in the input tree, as defined below:

Definition 1.1. *A phylogenetic reconstruction algorithm is said to be* adaptive fast converging *if, for each phylogenetic tree T over n terminal taxa, it guarantees to correctly reconstruct (w.h.p.) all edges of weight greater than ε in T from sequences of length*

$$k \;=\; \frac{\log(n)e^{O(depth(T))}}{\varepsilon^2} \;=\; \frac{n^{O(g)}}{\varepsilon^2}\;, \tag{3}$$

where g is the maximal edge weight in T.

Classic (non-adaptive) fast converging algorithms guarantee reconstruction of all edges in a tree under strict requirements on the input sequence length, which depend on the tree depth and on the weight of the shortest edges in the tree. An adaptive fast converging algorithm adapts the amount of correctly reconstructed edges to the length of input sequences. In this sense, adaptive fast convergence is a natural extension of fast convergence which avoids the assumption of a global lower bound on edge weights. The main contribution of this paper is a detailed presentation and analysis of the first adaptive fast converging algorithm. Furthermore, this algorithm is realized within optimal time (and space) complexity.

Our adaptive fast converging algorithm is an incremental algorithm. In each stage it holds a subphylogeny spanning a subset of the input terminal taxa, and it then adds another taxon by attaching it (as a leaf) to an edge or vertex of this subphylogeny. The incremental approach was originally introduced in [33] and adapted to fast convergence in [6]. Its clear advantage is its efficiency: in order to produce a time optimal $O(n^2)$ algorithm (as the algorithms of [33, 6]), all that is needed is a linear-time method for selecting the next terminal taxon for insertion and finding its appropriate insertion point in the current subphylogeny. The task of finding the insertion point is typically done by querying the vertices of the subphylogeny as to the direction of the chosen taxon (relative to that specific vertex).

A typical weakness of the incremental approach is its sensitivity to reconstruction errors: a single faulty edge introduced in the early stages of the algorithm is likely to cause errors in future stages, even if the data used from that stage on is completely consistent with the true tree. Hence, it is necessary to avoid the presence of any faulty edge in the intermediate subphylogenies. In our algorithm, this is achieved by contracting edges which cannot be reliably reconstructed. This contraction introduces high-degree (multifurcating) vertices which represent unresolved portions of the tree topology. A useful side effect of this approach is that the output of our algorithm is guaranteed w.h.p. to have a zero false positive rate (it contains no false edges).

A basic algorithmic tool introduced in this paper is the *partial directional oracle* (PDO), which allows our algorithm to locate the insertion point of a terminal taxon in an intermediate subphylogeny. The inherent difference between this oracle and ones used in previous works (e.g., [33, 6]) is that it is allowed to return no answer when it does not have a strong enough statistical support for any of the possible answers. Another implied difference is that it is designed to deal with internal vertices of high degree. Designing an efficient (asymptotically optimal) implementation of the PDO is one of the main algorithmic challenges dealt with in this work.

In order to make our algorithm adaptive fast converging, we need to guarantee that it uses only distances which are at most proportional to the depth of the tree. We achieve this in two steps. In Sections 3-5 we present a basic version of the algorithm, which achieves this goal under the (rather natural) restriction that the diameter of the contracted subtrees is at most proportional to the depth of the tree. Then, in Section 7, we present a variant of the basic algorithm which gets rid of this restriction, at the price of slightly complicating the algorithm and increasing constants in the resulting bounds. Both versions of the algorithm are *absolute* fast converging (require no prior knowledge on parameters of the phylogeny being reconstructed) and have optimal time complexity of $O(n^2)$.

The rest of this paper is organized as follows. In the next section we present the notations used in the paper. In Section 3 we describe our incremental algorithm, based on the *partial directional oracle*, and prove that it returns a contracted version of the true phylogeny. Section 4 describes the properties of the partial directional oracle which are needed to bound the weights of contracted edges. Section 5 provides an upper bound on the weight of the contracted edges as a function of the tree depth, noise function and properties of the partial directional oracle. In Section 6 we analyze the performance of our basic algorithm on the CFN model of evolution (using detailed analysis provided in the appendix), and in Section 7 we present the modification needed to make our algorithm adaptive fast converging for all

possible input. In Section 8 we mention some issues for further research. In the appendix we review (and slightly tighten) the analysis of distance estimation error in the CFN model.

2 Definitions and Notations.

Trees: A tree T is an undirected connected and acyclic graph. $V(T)$ and $E(T)$ denote the sets of vertices and edges of T, respectively. $leaves(T)$ denotes the leaf-set of T, and $internal(T) = V(T) \setminus leaves(T)$ denotes the set of its internal vertices. For a vertex $v \in V(T)$, the *neighborhood* of v, $N_T(v)$, is the set of vertices adjacent to v in T. The neighborhood of a subset $U \subseteq V$ is defined by $N_T(U) = \cup_{u \in U} N(u) \setminus U$. The degree of a vertex, $deg_T(v)$, is the size of its neighborhood in T. The *parent* of a leaf x in T, $parent_T(x)$, is the unique vertex in $N_T(x)$. T is said to be a *phylogenetic tree* over a set S of terminal taxa if $leaves(T) = S$, the degree of every internal vertex is at least three, and every edge $e \in E(T)$ is associated with a strictly positive weight $w(e)$. The weight function w induces a metric $D_T = \{d_T(u,v)\}_{u,v \in V(T)}$ over $V(T)$, s.t. $d_T(u,v)$ is the length (sum of edge weights) of $path_T(u,v)$ – the unique simple path connecting u and v in T. The diameter of T ($diam(T)$) is the maximum weight of a simple path in T.

A *subtree* of a tree T is a connected subgraph of T. The notion of distances is generalized for subtrees as follows: for two vertex disjoint subtrees t_1, t_2 of T, $d_T(t_1,t_2)$ denotes the length of $path_T(t_1,t_2)$, the unique shortest path in T connecting a vertex in t_1 and a vertex in t_2. Let t_1, t_2, t_3 be mutually disjoint subtrees of T. We say that t_2 *separates* t_1 from t_3 if $path_T(t_1,t_3)$ intersects t_2. If t_2 does not separate t_1 from t_3 we say that t_1 *and* t_3 *are on the same side of* t_2 (see Fig. 2). In general, we use lower case t's to denote subtrees of a tree T. Also, the subscript T may be removed from the corresponding notation when the tree T is clear from context.

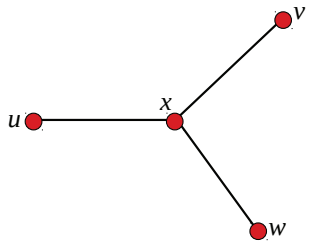

Figure 2: In the above tree, x separates u from v and from w. v, x and w are all on the same side of u.

Induced subphylogenies: Let T be a phylogenetic tree over a set of terminal taxa S, and let $S' \subseteq S$. $T(S')$, the phylogenetic tree induced by T on S', is obtained by taking the minimal subtree of T which spans S', and removing all vertices of degree two by iteratively replacing the two edges which touch such a vertex with a single edge. Note that every vertex in $V(T(S'))$ corresponds to a vertex of T and every edge in $E(T(S'))$ corresponds to a simple path in T (see Fig. 3). Edge weights in $T(S')$ are the weights of corresponding paths in T.

Internal Edge-Contraction: The contraction of an edge $e = (u,u') \in E(T)$ replaces e with a single vertex v s.t. $N(v) = N(\{u,u'\})$. In such a case we say that edge e was contracted into vertex v. $\widehat{T}$ is said to be an *internal (edge) contraction* of T if it is obtained from T by a series of contractions of internal edges (see Fig. 4). Note that if $\widehat{T}$ is an internal contraction of T then $leaves(T) = leaves(\widehat{T})$. All edge-contractions considered in this

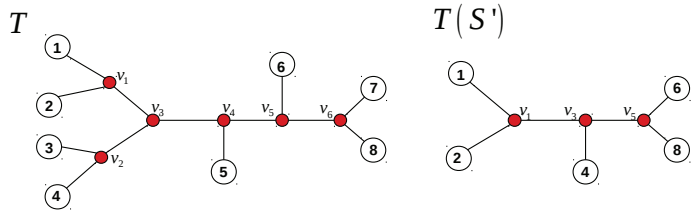

Figure 3: **Induced Sub-phylogeny.** Left: A tree over the the set of terminal taxa $S = \{1, \ldots, 8\}$. Right: The subphylogeny induced by $S' = \{1, 2, 4, 6, 8\}$. Notice that the edge (v_3, v_5) in $T(S')$ corresponds to the path $(v_3 - v_4 - v_5)$ in T.

paper are internal. $\widehat{T}$ is said to be an ε-contraction of T if each (internal) edge in T which is contracted in $\widehat{T}$ has weight at most ε. An internal contraction of T to $\widehat{T}$ induces a mapping between vertices of $internal(\widehat{T})$ and vertex disjoint subtrees of $internal(T)$: each vertex $\hat{v}$ of $internal(\widehat{T})$ corresponds either to a vertex in $internal(T)$, or to a subtree consisting of the internal edges contracted into $\hat{v}$. The subtree of T corresponding to a vertex $\hat{v}$ of $\widehat{T}$ is denoted by $t_{\hat{v}}$. Edge weights in a contraction $\widehat{T}$ of T are determined by the weight of the corresponding edges in T. This means that for neighboring vertices $\hat{u}, \hat{v}$ in $\widehat{T}$, we have $d_{\widehat{T}}(\hat{u}, \hat{v}) = d_T(t_{\hat{u}}, t_{\hat{v}})$, but otherwise, $d_T(t_{\hat{u}}, t_{\hat{v}})$ is generally larger than $d_{\widehat{T}}(\hat{u}, \hat{v})$ (see Fig. 4). We complete this section with two simple observations which are used later (often implicitly):

Lemma 2.1. *Let t be a subtree of T s.t. $V(t) \subseteq internal(T)$. Then the tree $\widehat{T}$ obtained from T by replacing t with a single vertex $\hat{v}$ such that $N_{\widehat{T}}(\hat{v}) = N_T(V(t))$ is an edge-contraction of T. If in addition the weights of all edges in t are at most ε, then $\widehat{T}$ is an ε-contraction of T.*

Lemma 2.2. *[transitivity of ε-contraction] Let T_1, T_2, T_3 be three phylogenetic trees over S. If T_1 is an ε-contraction of T_2 and T_2 is an ε-contraction of T_3, then T_1 is an ε-contraction of T_3.*

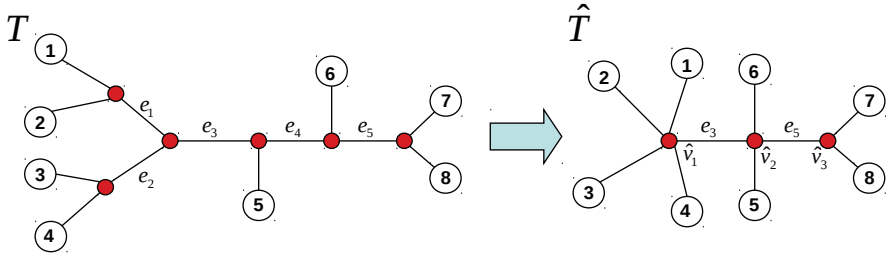

Figure 4: **Internal Edge-Contraction.** The tree $\widehat{T}$ is obtained from T by contracting e_1, e_2 into $\hat{v}_1$ and e_4 into $\hat{v}_2$. Note that $d_{\widehat{T}}(\hat{v}_1, \hat{v}_2) = w(e_3) = d_T(t_{\hat{v}_1}, t_{\hat{v}_2})$, whereas $d_{\widehat{T}}(\hat{v}_1, \hat{v}_3) = w(e_3) + w(e_5)$ and $d_T(t_{\hat{v}_1}, t_{\hat{v}_3}) = w(e_3) + w(e_4) + w(e_5)$.

3 The Incremental Algorithm.

Like other incremental algorithms (e.g., [33, 6]), our algorithm holds in each iteration a current topology $\widehat{T}$ spanning a subset $S' \subset S$ of the terminal taxa. It then selects a taxon $x \in S \setminus S'$ for insertion and attaches it to $\widehat{T}$.

Procedure *Incremental_Reconstruct(*S*)*:
- Select $x_0, x_1 \in S$, and initialize: $\widehat{T} \leftarrow (x_0, x_1)$; $S' \leftarrow \{x_0, x_1\}$.
- While $S' \neq S$ do:
  1. Select a taxon $x \in S \setminus S'$ and set $S' \leftarrow S' \cup \{x\}$.
  2. $\widehat{T} \leftarrow$*Attach_Taxon(*$\widehat{T}, x$*)*

The following lemma is central in establishing the fact that our incremental algorithm produces a tree which is an edge contraction of the true tree.

Lemma 3.1. *Let $\widehat{T}$ be a contraction of $T(S')$ and $x \in S \setminus S'$, and let $\widehat{T}_{post} =$Attach_Taxon$(\widehat{T}, x)$ be the content of $\widehat{T}$ at the end of an iteration of Incremental_Reconstruct. Then $\widehat{T}_{post}$ is a contraction of $T(S' \cup \{x\})$.*

Section 3.1 is devoted to the proof of this lemma through the use of an abstract implementation of the insertion procedure *Attach_Taxon* based on a *partial directional oracle* (PDO). Sections 3.2 and 3.3 present an efficient implementation of a PDO based on quartets. At this stage we assume an arbitrary insertion order over S, meaning that the taxon x chosen for insertion at step 1 of the loop is an arbitrary taxon in $S \setminus S'$. A natural criterion for choosing x in a way which guarantees a good upper bound on the weight of contracted edges is suggested later in Section 5. The following assumptions hold throughout the rest of the paper, unless specified otherwise: S' is a subset of the taxon set S, $\widehat{T}$ (the current topology) is an edge contraction of $T(S')$, and x (the taxon chosen for insertion) is a taxon in $S \setminus S'$.

3.1 Attaching a terminal taxon using a partial directional oracle.

The insertion process is based on queries posed to a *directional oracle*. This oracle receives x and a vertex $\hat{v}$ of $\widehat{T}$ and is expected to output a neighbor $\hat{u}$ of $\hat{v}$ which indicates the direction of x with respect to $\hat{v}$. This oracle is *partial* in the sense that it is allowed to return an answer of *'null'* if there is no such neighbor $\hat{u}$ or if the direction of x w.r.t. $\hat{v}$ cannot be reliably inferred.

Definition 3.2 (Partial Directional Oracle)**.** *A* partial directional oracle for T *is a function $PDO = PDO_T$ which receives queries of the form $(\widehat{T}, \hat{v}, x)$, where $\hat{v} \in V(\widehat{T})$, and outputs either a vertex $\hat{u} \in N_{\widehat{T}}(\hat{v})$ or 'null'. PDO is required to be* truthful*, meaning that its output must satisfy two requirements:*

- *If $\hat{v} \in leaves(\widehat{T})$, then $PDO(\widehat{T}, \hat{v}, x) = parent_{\widehat{T}}(\hat{v})$.*
- *If $PDO(\widehat{T}, \hat{v}, x) = \hat{u}$, then $t_{\hat{u}}$ and x are on the same side of $t_{\hat{v}}$ in $T(S' \cup \{x\})$. We often abuse notation and simply say that $\hat{u}$ is on the same side of $\hat{v}$ as x.*

The partial directional oracle is used to locate the *anchor* of x in $\widehat{T}$, which is an edge or vertex of $\widehat{T}$ to which x should be attached.

Definition 3.3 (Anchor)**.** *Let p_x denote the parent of x in $T(S' \cup \{x\})$. The* anchor *of x in $\widehat{T}$, denoted by $anchor_{\widehat{T}}(x)$, is defined as follows: if p_x is included in $t_{\hat{v}}$ for some $\hat{v} \in V(\widehat{T})$ (either by being a vertex in $t_{\hat{v}}$ or by lying on a path in T which corresponds to an edge of $t_{\hat{v}}$),*

then $anchor_{\widehat{T}}(x) = \hat{v}$. *Otherwise,* $anchor_{\widehat{T}}(x)$ *is the unique edge* $(\hat{u}, \hat{v}) \in E(\widehat{T})$ *for which* p_x *is an internal vertex in* $path_T(t_{\hat{u}}, t_{\hat{v}})$.

Since T might contain indistinguishable edges, it might be impossible to locate the exact location of the anchor of x in $\widehat{T}$. Therefore, the algorithm uses the partial directional oracle to compute the *insertion zone of* x *in* $\widehat{T}$ through procedure *Find_Insertion_Zone*$(\widehat{T}, x, PDO)$ described below. The truthfulness of the partial directional oracle implies that this procedure returns a subtree which contains the desired anchor (see Observation 3.4 below).

Procedure *Find_Insertion_Zone*$(\widehat{T}, x, PDO)$:

1. $\hat{t}_{inz} \leftarrow \widehat{T}$
2. For every edge $(\hat{u}, \hat{v}) \in E(\hat{t}_{inz})$ s.t. $PDO(\widehat{T}, \hat{v}, x) = \hat{u}$ do:
   - Delete from $\hat{t}_{inz}$ all the vertices which are separated from $\hat{u}$ by $\hat{v}$.

Observation 3.4. $\hat{t}_{inz}(\widehat{T}, x, PDO)$ *is the* maximal *subtree* $\hat{t}$ *of* $\widehat{T}$ *which contains* $anchor_{\widehat{T}}(x)$ *either as an edge or as an internal vertex , s.t. for each* $\hat{v} \in internal(\hat{t})$, $PDO(\widehat{T}, \hat{v}, x) =$ *'null', and for each* $\hat{v} \in leaves(\hat{t})$, $PDO(\widehat{T}, \hat{v}, x) \neq$ *'null'.*

Procedure *Attach_Taxon* specifies how to attach x to $\widehat{T}$ given $\hat{t}_{inz}(\widehat{T}, x, PDO)$ (see Fig. 5).

Procedure *Attach_Taxon*$(\widehat{T}, x)$:

1. $\hat{t}_{inz} \leftarrow \hat{t}_{inz}(\widehat{T}, x, PDO)$.
2. If $\hat{t}_{inz}$ is a single edge $(\hat{u}, \hat{v})$, then attach x to $\widehat{T}$ by introducing a new internal vertex $\hat{p}_x$ and replacing $(\hat{u}, \hat{v})$ with the three edges $(\hat{u}, \hat{p}_x), (\hat{v}, \hat{p}_x), (x, \hat{p}_x)$.
3. If $\hat{t}_{inz}$ has a single internal vertex $\hat{v}$ (i.e., $V(\hat{t}_{inz}) = \{\hat{v}\} \cup N(\hat{v})$), then add to $\widehat{T}$ the edge $(\hat{v}, x)$.
4. Else (i.e., $\hat{t}_{inz}$ has at least one internal edge), contract all internal edges of $\hat{t}_{inz}$ into a new vertex $\hat{v}$ and add to $\widehat{T}$ the edge $(\hat{v}, x)$.

In order to prove that procedure *Attach_Taxon* satisfies Lemma 3.1, we define an intermediate topology $\widehat{T}^{+x}$, which is a natural extension of $\widehat{T}$ to a contraction of $T(S' \cup \{x\})$:

- If the anchor of x in $\widehat{T}$ is a vertex $\hat{v} \in V(\widehat{T})$ then $\widehat{T}^{+x}$ is obtained by adding the edge $(\hat{v}, x)$ to $\widehat{T}$.
- Otherwise, the anchor of x in $\widehat{T}$ is an edge $(\hat{u}, \hat{v}) \in E(\widehat{T})$, and $\widehat{T}^{+x}$ is obtained by replacing the edge $(\hat{u}, \hat{v})$ with the three edges $(\hat{u}, \hat{p}_x), (\hat{p}_x, \hat{v}), (\hat{p}_x, x)$.

Next we present a proof of Lemma 3.1, which is based on the truthfulness of the partial directional oracle. An implementation of such an oracle will be presented in Section 3.2.

Proof of Lemma 3.1. $\widehat{T}^{+x}$ as defined above is clearly an edge contraction of $T(S' \cup \{x\})$. Hence, due to Lemma 2.2 (transitivity of edge contraction), all we have to show is that $\widehat{T}_{post}$ is an edge contraction of $\widehat{T}^{+x}$. By Observation 3.4, $\hat{t}_{inz} = \hat{t}_{inz}(\widehat{T}, x, PDO)$ includes $anchor_{\widehat{T}}(x)$ either as an edge or as an internal vertex. We thus distinguish between the following cases:

- $anchor_{\widehat{T}}(x)$ is the unique edge in $\hat{t}_{inz}$ (i.e., the insertion zone consists of a single edge). In this case $\widehat{T}_{post} = \widehat{T}^{+x}$.

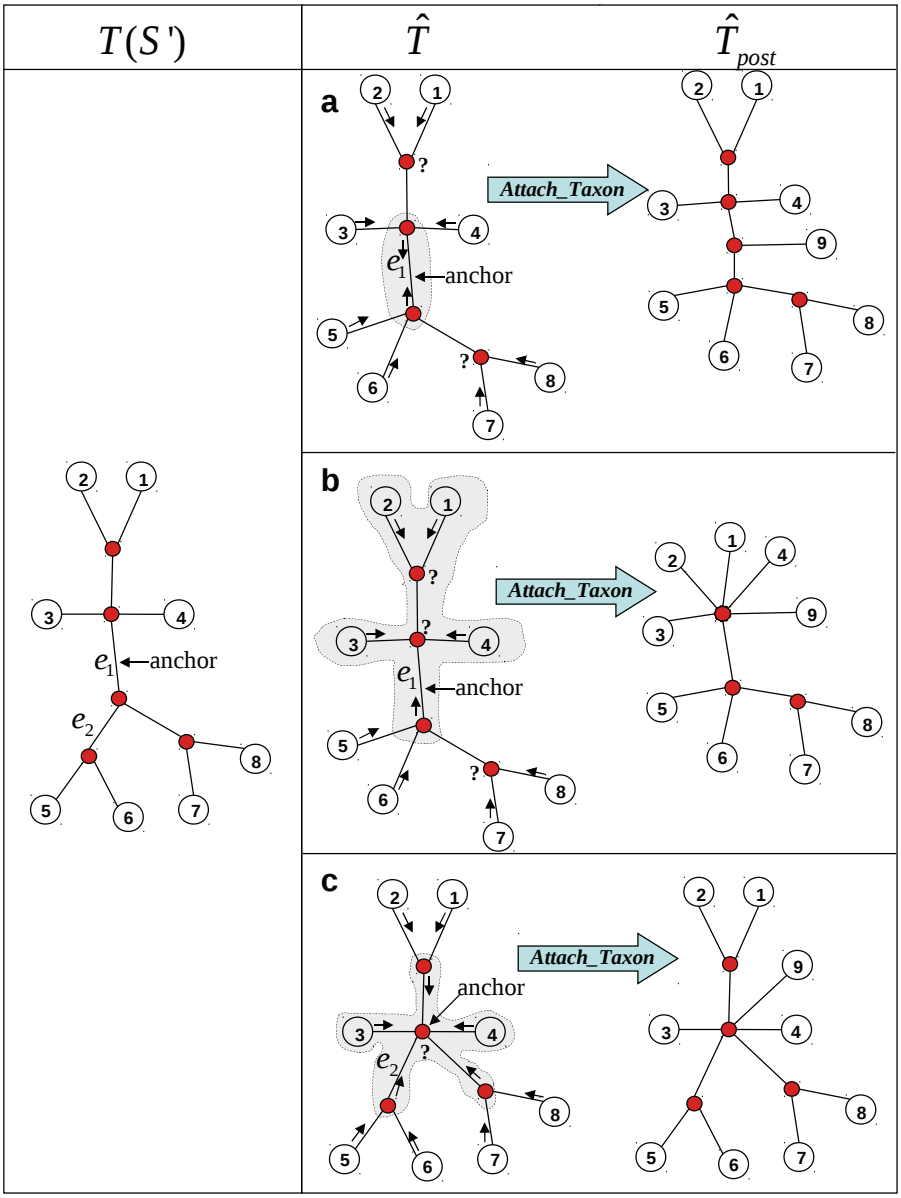

Figure 5: This figure illustrates a single insertion iteration of the algorithm. In this example we have $S' = \{1, \ldots, 8\}$ and $x = 9$. Left: the induced subphylogeny $T(S')$, in which the anchor of x (edge e_1) is indicated. Right: three scenarios for the insertion of x into $\widehat{T}$. The left tree depicts the current topology $\widehat{T}$ before insertion, with answers to directional queries and the insertion zone $\hat{t}_{inz}$ marked by a gray cloud. The right tree depicts the topology $\widehat{T}_{post}$ after the insertion. **(a,b)** The current topology $\widehat{T}$ is obtained from $T(S')$ by contracting e_2, so the anchor of x in $\widehat{T}$ is e_1. **(a)** $\hat{t}_{inz}$ consists a single edge, so $\widehat{T}_{post}$ is obtained from $\widehat{T}$ by splitting this edge to two and connecting x to the new internal vertex resulting from this. **(b)** $\hat{t}_{inz}$ contains one internal edge, which is contracted into a new internal vertex to which x is connected. **(c)** The current topology $\widehat{T}$ is obtained from $T(S')$ by contracting e_1, so the anchor of x in $\widehat{T}$ is the vertex into which e_1 is contracted. $\hat{t}_{inz}$ contains a single internal vertex to which x is connected.

- $anchor_{\widehat{T}}(x)$ is an internal vertex or edge of $\hat{t}_{inz}$. In this case $\widehat{T}_{post}$ is obtained from $\widehat{T}^{+x}$ by contracting the internal edges of $\hat{t}_{in_{\hat{z}}}$. Note that if the anchor is a vertex, then all internal edges of $\hat{t}_{inz}$ are also edges in $\widehat{T}^{+x}$. On the other hand, if $anchor_{\widehat{T}}(x)$ is an edge $(\hat{u}, \hat{v})$, then in $\widehat{T}^{+x}$ this edge is split into two edges $(\hat{u}, \hat{p}_x)$ and $(\hat{p}_x, \hat{v})$, and both these edges undergo contraction in the transition to $\widehat{T}_{post}$.
- $anchor_{\widehat{T}}(x)$ is an external edge $(\hat{u}, \hat{v})$ of $\hat{t}_{inz}$, where $\hat{v}$ is a leaf of $\hat{t}_{inz}$ and $\hat{u}$ is an internal vertex of $\hat{t}_{inz}$. In this case $\widehat{T}_{post}$ is obtained from $\widehat{T}^{+x}$ by contracting the edge $(\hat{u}, \hat{p}_x)$ as well as all the internal edges of $\hat{t}_{inz}$ (see Fig. 6). □

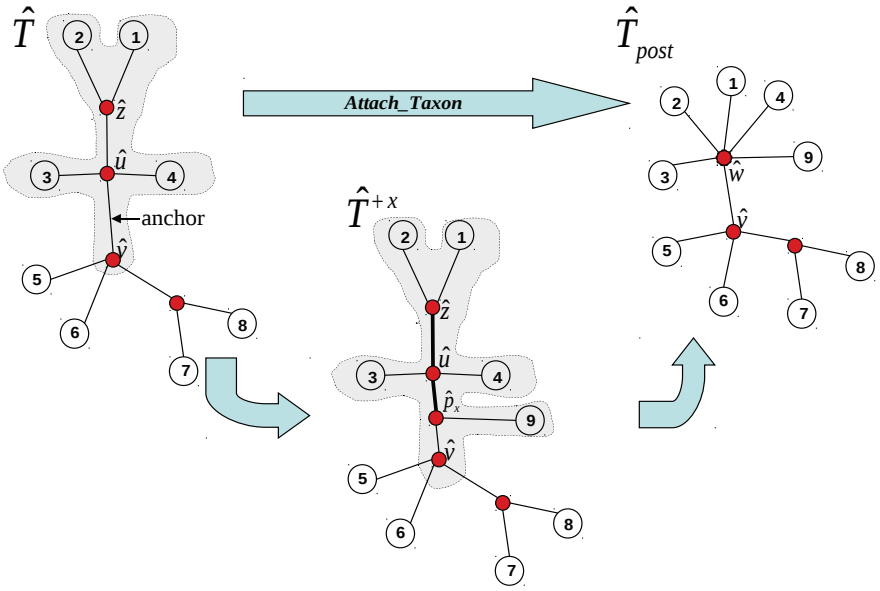

Figure 6: **Edges contracted during a single iteration.** This figure demonstrates the edges that undergo contraction in a single insertion iteration. The demonstration follows the scenario of Fig 5b. The insertion of $x = 9$ into $\widehat{T}$ results in the contraction of two edges ($(\hat{u}, \hat{z})$ and $(\hat{u}, \hat{p}_x)$) into $\hat{w}$.

3.2 A quartet-based partial directional oracle.

We implement our partial directional oracle PDO using queries to *quartet splits*. Let $T(q)$ be the subphylogeny induced by a quartet q of terminal taxa. The *split of q in T* is said to be $(a, b|c, d)$ if $T(q)$ contains an internal edge which separates $\{a, b\}$ from $\{c, d\}$. In order to determine the direction of a terminal taxon x w.r.t. vertex $\hat{v} \in V(\widehat{T})$, PDO queries a series of quartet splits. Each of these quartets consists of x and three other terminal taxa from S' which represent three different directions w.r.t. $\hat{v}$. These three taxa are *directional representatives* maintained by the incremental algorithm.

Definition 3.5 (Directional Representatives)**.** *Let $(\hat{u}, \hat{v})$ be an edge in $\widehat{T}$. A leaf s of $\widehat{T}$ is a* valid directional representative of $(\hat{v} \to \hat{u})$ *if $\hat{u} \in path_{\widehat{T}}(\hat{v}, s)$. The directional representative maintained by the algorithm for $(\hat{v} \to \hat{u})$ is denoted by $s_{\hat{v}}(\hat{u})$.*

In order to infer a quartet split, PDO queries a *partial quartet oracle*, which returns either the split of the quartet in T or *'null'* if the subphylogeny induced by the quartet is a star or if its split cannot be reliably inferred.

Definition 3.6 (Partial Quartet Oracle)**.** *Let T be a phylogenetic tree over S. A* partial quartet oracle for T *is a function $PQO = PQO_T$ which receives a quartet $q \subseteq S$ and returns either the split of q in T, or 'null'.*

Our partial directional oracle PDO consists of two main phases:

Triplets Tournament: In this phase, the set of all possible directions (represented by $N_{\widehat{T}}(\hat{v})$) is iteratively screened to end up with at most one candidate direction. In each iteration a quartet is queried and as a result at least two directions are eliminated from the set of candidates. If the tournament results in an empty candidate set, then the directional oracle returns *'null'*. Otherwise, the tournament results in a single surviving candidate. When the edge pointing to the direction of x is indistinguishable, a wrong candidate $\hat{u}$ may survive this phase. The following validation phase is needed in order to eliminate such candidates.

Validation: Validation of the direction represented by the surviving neighbor $\hat{u}$ is done by another series of quartet queries which contain both x and $s_{\hat{v}}(\hat{u})$. If all quartet queries positively validate this direction (meaning that they put x and $s_{\hat{v}}(\hat{u})$ on the same side of the split), then $\hat{u}$ is returned. Otherwise, the directional oracle returns *'null'*.

$PDO(\widehat{T}, \hat{v}, x)$**:**

1. Initialize candidate set $C \leftarrow N_{\widehat{T}}(\hat{v})$.
2. If $C = \{\hat{u}\}$ ($\hat{v}$ is a leaf), return $\hat{u}$.
3. Otherwise ($|C| \geq 3$), proceed as follows:
4. **Triplets Tournament:**
   - While $|C| > 1$ do:
     - If $|C| = 2$, then $C \leftarrow C \cup \{\hat{u}\}$, for some $\hat{u} \in N_{\widehat{T}}(\hat{v}) \setminus C$.
     - Select some triplet $\{\hat{u}_1, \hat{u}_2, \hat{u}_3\} \subseteq C$ and invoke $PQO(\{x,\ s_{\hat{v}}(\hat{u}_1),\ s_{\hat{v}}(\hat{u}_2),\ s_{\hat{v}}(\hat{u}_3)\})$.
       - If output is *'null'*, then remove $\hat{u}_1, \hat{u}_2, \hat{u}_3$ from C.
       - Otherwise, the output is $(x, s_{\hat{v}}(\hat{u}_i) \mid s_{\hat{v}}(\hat{u}_j), s_{\hat{v}}(\hat{u}_k))$ (where $\{i, j, k\} = \{1, 2, 3\}$), then remove $\hat{u}_j, \hat{u}_k$ from C.
   - If the tournament results in $C = \emptyset$, return *'null'*.
5. **Validation:** $C = \{\hat{u}\}$ for some $\hat{u} \in N_{\widehat{T}}(\hat{v})$.
   (a) Select some vertex $\hat{u}_1 \in N_{\widehat{T}}(\hat{v}) \setminus \{\hat{u}\}$.
   (b) For every $\hat{u}_2 \in N_{\widehat{T}}(\hat{v}) \setminus \{\hat{u}, \hat{u}_1\}$, invoke $PQO(\{x,\ s_{\hat{v}}(\hat{u}),\ s_{\hat{v}}(\hat{u}_1),\ s_{\hat{v}}(\hat{u}_2)\})$.
      - If output is $(x, s_{\hat{v}}(\hat{u}) \mid s_{\hat{v}}(\hat{u}_1), s_{\hat{v}}(\hat{u}_2))$, then continue.
      - Otherwise, stop and return *'null'*.
   (c) Return $\hat{u}$ (if it survived all rounds).

Lemma 3.7. *If PQO is a partial quartet oracle for T, then procedure PDO described above is a (truthful) partial directional oracle for T.*

Proof. Consider a valid input instance $(\widehat{T}, \hat{v}, x)$ for PDO. If $\hat{v}$ is a leaf, then PDO returns the unique neighbor of $\hat{v}$ in $\widehat{T}$, as required. So assume $\hat{v}$ is an internal vertex of $\widehat{T}$. It is

sufficient to show that for any vertex $\hat{u} \in N_{\widehat{T}}(\hat{v})$ which is not on the same side of $\hat{v}$ as x, there exists a vertex which fails $\hat{u}$ at step 5b of the validation phase. If there is a vertex $\hat{u}' \in N_{\widehat{T}}(\hat{v})$ which is on the same side of $\hat{v}$ as x, then $\hat{u}'$ fails the validation of $\hat{u}$ when chosen either as $\hat{u}_2$ or as $\hat{u}_1$ (see Fig. 7a). If there is no such vertex, then $\hat{v}$ is the anchor of x in $\widehat{T}$, meaning that p_x, the parent of x in $T(S' \cup \{x\}))$, is contained in $t_{\hat{v}}$. In this case, we distinguish between two subcases. First, assume that there is a neighbor $\hat{u}' \in N_{\widehat{T}}(\hat{v}) \setminus \{\hat{u}\}$ s.t. $t_{\hat{u}}$ and $t_{\hat{u}'}$ are on the same side of p_x in $T(S' \cup \{x\})$ (Fig. 7b)). Note that there is always a neighbor $\hat{u}''$ s.t. $t_{\hat{u}}$ and $t_{\hat{u}''}$ are separated by p_x and hence the quartet $\{x, s_{\hat{v}}(\hat{u}), s_{\hat{v}}(\hat{u}'), s_{\hat{v}}(\hat{u}'')\}$ fails the validation of $\hat{u}$. We are left to deal with the case in which all neighbors $\hat{u}' \in N_{\widehat{T}}(\hat{v}) \setminus \{\hat{u}\}$, are such that $t_{\hat{u}}$ and $t_{\hat{u}'}$ are separated by p_x in $T(S' \cup \{x\})$ (Fig. 7c)). In such a case, p_x is a vertex also in $T(S')$, and so the degree of p_x in $T(S' \cup \{x\})$ is at least 4. Therefore, for every choice of $\hat{u}_1$ in step 5a of the validation phase, there is $\hat{u}_2 \in N_{\widehat{T}}(\hat{v}) \setminus \{\hat{u}, \hat{u}_1\}$, s.t. $T(\{x, s_{\hat{v}}(\hat{u}), s_{\hat{v}}(\hat{u}_1), s_{\hat{v}}(\hat{u}_2)\})$ is a star, and this quartet fails the validation of $\hat{u}$. □

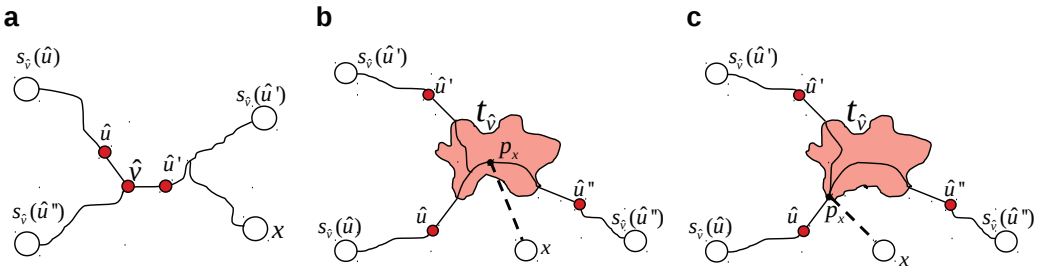

Figure 7: **Proof of Lemma 3.7.**

3.3 Updating Directional Representatives.

We now describe how directional representatives (DRs) are maintained throughout the incremental algorithm. In the initialization step, $\widehat{T}$ consists of a single edge (x_0, x_1), and this edge is assigned the obvious DRs: $s_{x_0}(x_1) = x_1$ and $s_{x_1}(x_0) = x_0$. In every insertion iteration, some DRs require updates due to changes in the edges of $\widehat{T}$ (Fig. 8). We note that when a terminal taxon x is inserted into $\widehat{T}$ it is associated with a reference taxon y which is a leaf in $\widehat{T}$ close to it (see Section 5 for more details). Hence, the new external edge $(x, \hat{p}_x)$ is assigned two DRs: $s_{\hat{p}_x}(x) = x$ and $s_x(\hat{p}_x) = y$. If an edge $(\hat{u}, \hat{v})$ is split to $(\hat{u}, \hat{p}_x), (\hat{p}_x, \hat{v})$, the DRs of the two new edges are assigned according to the DRs of $(\hat{u}, \hat{v})$ as follows: $s_{\hat{v}}(\hat{p}_x), s_{\hat{p}_x}(\hat{u}) \leftarrow s_{\hat{v}}(\hat{u})$, and $s_{\hat{u}}(\hat{p}_x), s_{\hat{p}_x}(\hat{v}) \leftarrow s_{\hat{u}}(\hat{v})$. Finally, if contractions (of the internal edges of $\hat{t}_{inz}$) take place, then edges touching the new vertex (resulting from contraction) inherit the DRs of the respective external edges of $\hat{t}_{inz}$.

3.4 Section Summary and complexity analysis.

In this section we presented a quartet-based incremental algorithm based on a *directional oracle*. A direct inductive application of Lemma 3.1 implies that this algorithm is guaranteed to return an edge contraction of the true tree, provided the directional oracle it queries is *truthful*. Such an oracle was then presented in Section 3.2. An important property of this algorithm is its optimal time and space complexity. Before turning to analyze the complexity

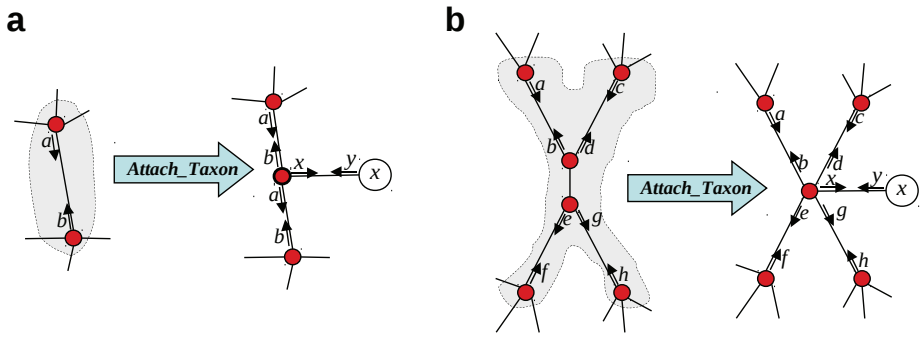

Figure 8: **Updating Directional Representatives. (a)** An edge is split during insertion of x. Its directional representatives (a, b in the figure above) are copied to the two resulting edges. **(b)** An edge is contracted during insertion of x. Directional representatives of external edges of the insertion zone (marked by a gray cloud) are copied to the edges touching the new internal vertex (into which the internal edge is contracted).

of the algorithm, we note that this analysis disregards the time required for selecting the next taxon for insertion. In Section 5 we suggest a specific order of insertion and show that it can be computed within the time constraints specified here.

The time complexity of each insertion step is dominated by the time it takes to compute the insertion zone. The insertion zone can be computed by querying the directional oracle on all vertices of $\widehat{T}$, and then pruning $\widehat{T}$ in a DFS-traversal according to the queries' results. The DFS-traversal and pruning are clearly linear in the size of $\widehat{T}$. Assuming quartet queries take constant time, an execution $PDO(\widehat{T}, \hat{v}, x)$ is linear in $deg_{\widehat{T}}(\hat{v})$. This is because the validation phase simply scans the neighborhood of $\hat{v}$, and during the triplets tournament at least two neighbors of $\hat{v}$ are eliminated from the candidate set during each iteration. Hence the total time complexity of all queries to PDO is linear in the size of $\widehat{T}$, and so each iteration is performed in linear time in the number of terminal taxa n. Therefore, the total time complexity of the incremental algorithm is $O(n^2)$, which is optimal for algorithms reconstructing phylogenetic trees with unbounded vertex-degrees [7]. The work space used by the algorithm is also optimal, because the space required for storing the current topology and directional representatives is linear in n.

4 Bounding the Weights of Contracted Edges.

The previous section describes an incremental algorithm which reconstructs an internal edge contraction of the true phylogenetic tree. This section focuses on establishing an upper bound on the weights of contracted edges. As in the previous section, our analysis focuses on a single insertion iteration. Assume that in some stage of the algorithm, $\widehat{T}$ is an ε-contraction of $T(S')$ (for some $\varepsilon \geq 0$), and let x denote the next taxon chosen for insertion. Note that the intermediate topology $\widehat{T}^{+x}$, as defined in Section 3.1, is guaranteed to be an ε-contraction of $T(S' \cup \{x\})$. In this section we study the conditions under which $\widehat{T}_{post}$, the tree obtained by applying $Attach\_Taxon(\widehat{T}, x)$, is guaranteed to be an ε-contraction of $\widehat{T}^{+x}$ (and hence, due to the transitivity of ε-contraction stated in Lemma 2.2, also an ε-contraction of $T(S' \cup \{x\})$). Lemma 4.2 below provides the main argument of our analysis through the concept of *ε-environment*.

Definition 4.1 (ε-environment)**.** *The* ε-environment *of* x *in* $\widehat{T}^{+x}$, $\hat{t}_{env}(\widehat{T}, x, \varepsilon)$, *is the max-*

imal subtree of $\widehat{T}^{+x}$ which includes x and whose internal edges have weight at most ε. (Note that an external edge of this subtree is either external in $\widehat{T}^{+x}$ or has weight greater than ε.)

Lemma 4.2. *If $V(\hat{t}_{inz}(\widehat{T}, x, PDO)) \subseteq V(\hat{t}_{env}(\widehat{T}, x, \varepsilon))$, then $\widehat{T}_{post}$ (the tree obtained by applying Attach_Taxon$(\widehat{T}, x)$) is an ε-contraction of $\widehat{T}^{+x}$.*

Proof. We use the following notations for brevity: $\hat{t}_{inz} = \hat{t}_{inz}(\widehat{T}, x, PDO)$ and $\hat{t}_{env} = \hat{t}_{env}(\widehat{T}, x, \varepsilon)$. According to the proof of Lemma 3.1, each edge $\hat{e}$ of $\widehat{T}^{+x}$ that is contracted in $\widehat{T}_{post}$ satisfies one of the following:

- One end-point of $\hat{e}$ is $\hat{p}_x$ – the parent of x in $\widehat{T}^{+x}$, and the other end-point is in $internal(\hat{t}_{inz})$.
- Both end-points of $\hat{e}$ are in $internal(\hat{t}_{inz})$.

The assumption $V(\hat{t}_{inz}) \subseteq V(\hat{t}_{env})$ implies that $internal(\hat{t}_{inz}) \subseteq internal(\hat{t}_{env})$, and the definition of ε-environment implies that $\hat{p}_x \in internal(\hat{t}_{env})$. Hence, in either of the above cases, $\hat{e}$ is an internal edge of $\hat{t}_{env}$ and its weight cannot be greater than ε. □

In order to bound the insertion zone within the ε-environment (as required by Lemma 4.2), we consider the set of quartets queried during executions of PDO on vertices in $\widehat{T}$. The quartets queried during execution of PDO on a vertex $\hat{v}$ are included in the *quartet-span* of $\hat{v}$ and x, as defined below.

Definition 4.3 (Quartet-span)**.** *The* quartet-span *of $\hat{v} \in V(\widehat{T})$ and x, is the following set:*

$$Quart(\hat{v}, x) \triangleq \left\{ q = \{x, s_1, s_2, s_3\} \mid \{s_1, s_2, s_3\} \subseteq \{s_{\hat{v}}(\hat{u}) : \hat{u} \in N_{\widehat{T}}(\hat{v})\} \right\} .$$

Note that the quartet span of $\hat{v}$ and x depends on the directional representatives maintained by the incremental algorithm for $\widehat{T}$. The ability to infer the split of a given quartet is known to depend on its diameter and the weight of its internal edge (see, e.g., [11, 8]). This is captured by the following set of definitions.

Definition 4.4 (r-short quartet)**.** *A quartet $q \subseteq S$ is said to be* r-short *if $diam(T(q)) \leq r$.*

Definition 4.5 (ε-separated quartet)**.** *A quartet $q \subseteq S$ is said to be* ε-separated *if the single internal edge in $T(q)$ has weight strictly greater than ε.*

Definition 4.6 ((r, ε)-reliability)**.** *A partial quartet oracle PQO is said to be (r, ε)-reliable if it (correctly) infers the split of every quartet which is r-short and ε-separated.*

Lemma 4.8 below translates the reliability parameters of PQO to conditions under which $PDO(\widehat{T}, \hat{v}, x)$ is guaranteed to return a non-null direction. This is done by bounding (from above) the diameter of quartets in $Quart(\hat{v}, x)$ and (from below) the *separation* of $\hat{v}$ and x, as defined next.

Definition 4.7 (Separating edge/ Separation of $\hat{v}$ and x)**.** *The* separating edge *of a vertex $\hat{v} \in V(\widehat{T})$ and x is the first edge in the path (in $\widehat{T}^{+x}$) from $\hat{v}$ to $\hat{p}_x$ – the parent of x in $\widehat{T}^{+x}$. The* separation *of $\hat{v}$ and x is the weight of this separating edge. If $\hat{v} = \hat{p}_x$, then the separation of $\hat{v}$ and x is 0.*

Lemma 4.8. *Assume that PQO is an (r, ε)-reliable partial quartet oracle for $r, \varepsilon \geq 0$. Let $\hat{v}$ be an internal vertex of $\widehat{T}$ satisfying the following:*

- *All quartets in $Quart(\hat{v}, x)$ are r-short.*
- *The separation of $\hat{v}$ and x is strictly greater than ε.*

Then $PDO(\widehat{T},\hat{v},x) \neq$ *'null'.*

Proof. We need to show that there is a neighbor $\hat{u}$ of $\hat{v}$ which survives the triplets tournament and the validation phase of $PDO(\widehat{T},\hat{v},x)$. We take $\hat{u}$ to be the unique neighbor of $\hat{v}$ which is on the same side as x (such a $\hat{u}$ exists since $\hat{v} \neq p_x$). Then for each quartet of the type $q = \{x, s_{\hat{v}}(\hat{u}), s_1, s_2\} \in Quart(\hat{v},x)$, the correct split of q is $(x, s_{\hat{v}}(\hat{u})|s_1,s_2)$. By the assumptions of the lemma, each such quartet q is r-short and ε-separated (since the internal edge of q is a path which contains the separating edge of $\hat{v}$ and x). The (r,ε)-reliability of PQO implies that it returns the correct split for each such quartet q, meaning that $\hat{u}$ survives the triplet tournament and the validation phase, as claimed □

The following lemma summarizes the results of this section by specifying the conditions under which the weights of edges contracted in a single iteration can be bounded by ε. This is done by considering $Quart(\widehat{T},x,\varepsilon)$, the *quartet-span* of $leaves(\hat{t}_{env}(\widehat{T},x,\varepsilon))$, defined by:

$$Quart(\widehat{T},x,\varepsilon) \triangleq \bigcup_{\hat{v}\in leaves(\hat{t}_{env}(\widehat{T},x,\varepsilon))} Quart(\hat{v},x) \ . \tag{4}$$

Lemma 4.9. *Let* $\widehat{T}$ *be an internal edge contraction of* $T(S')$ *for some* $S' \subset S$*, and let* x *be a taxon in* $S \setminus S'$*. Let* $\varepsilon, r \geq 0$ *be s.t. PQO is* (r,ε)*-reliable and all quartets in* $Quart(\widehat{T},x,\varepsilon)$ *are* r*-short. Then the tree* $\widehat{T}_{post}$ *obtained by applying* $Attach\_Taxon(\widehat{T},x)$ *is an* ε*-contraction of* $\widehat{T}^{+x}$*.*

Proof. Let $\hat{t}_{env}$ denote the ε-environment of x in $\widehat{T}^{+x}$. The proof is established by showing that $PDO(\widehat{T},\hat{v},x) \neq$ *'null'* for every $\hat{v} \in leaves(\hat{t}_{env}) \setminus \{x\}$. Establishing this implies (through Observation 3.4) that $\hat{t}_{inz}(\widehat{T},x,PDO)$ is contained in $\hat{t}_{env}$, and this implies (through Lemma 4.2) that $\widehat{T}_{post}$ is an ε-contraction of $\widehat{T}^{+x}$.

Consider an arbitrary vertex $\hat{v} \in leaves(\hat{t}_{env})$. If $\hat{v}$ is a leaf of $\widehat{T}$, then (by definition of PDO) $PDO(\widehat{T},\hat{v},x) \neq$ *'null'*. Otherwise, $\hat{v}$ is an internal vertex of $\widehat{T}$, and hence the separation of $\hat{v}$ and x is greater than ε. Since all quartets in $Quart(\hat{v},x)$ are r-short, Lemma 4.8 implies that $PDO(\widehat{T},\hat{v},x) \neq$ *'null'* also in this case. □

5 Applying the Algorithm on Noisy Distance Estimates.

By Lemma 4.9, in order to minimize the bound ε on the weights of the edges contracted in a given iteration, we need to minimize the bound r on the diameters of the quartets in $Quart(\widehat{T},x,\varepsilon)$, and then to devise a quartet oracle which is (r,ε)-reliable for a minimal possible ε. In this section we first provide a tight bound on the diameters of the quartets in $Quart(\widehat{T},x,\varepsilon)$ (Lemma 5.7), and then present and analyze the required quartet oracle. Both tasks depend on properties of the tree T and the accuracy of the distance estimates between the terminal taxa, $\widehat{D} = \{\hat{d}(i,j)\}_{i,j\in S}$. The main tool for minimizing the diameter of quartets in $Quart(\widehat{T},x,\varepsilon)$ is choosing an appropriate insertion order.

Taxa in S are selected for insertion according to the following greedy approach: in each iteration, the taxon selected for insertion is the one closest to the current set of terminal taxa, according to the distance estimates $\widehat{D}$. The initial taxon-pair (x_0,x_1) is chosen s.t. $\hat{d}(x_0,x_1) = \min_{i,j\in S}\{\hat{d}(i,j)\}$. Then, in each insertion iteration the algorithm identifies two taxa $x \in S \setminus S'$ and $y \in S'$ s.t. $\hat{d}(x,y) = \min_{i\notin S', j\in S'}\{\hat{d}(i,j)\}$. x is chosen as the next taxon for insertion and y is referred to as its *reference taxon* in S'. Recall that the reference taxon y is also used as the directional representative of the new external edge $(x \to \hat{p}_x)$ (see Section 3.3). Finding such a pair (x,y) can be done in linear time in each iteration by maintaining for each $i \in S \setminus S'$ the taxon $j \in S'$ closest to it under $\widehat{D}$. Notice that a similar technique is applied in the $O(n^2)$ implementation of Prim's MST algorithm [4].

Therefore, the computation of the insertion order falls within the $O(n^2)$ time complexity of the incremental algorithm.

Our greedy insertion order enables to bound the distance from the inserted taxon x to its reference taxon y in terms of the *depth* the tree, defined below (a similar notion of tree depth is used by other fast converging algorithms, e.g., [11]).

Definition 5.1 (Depth)**.** *The* depth *of a tree T (denoted by $depth(T)$) is given by:*

$$depth(T) = \max_{v \in V(T), u \in N_T(v)} \{ \ \min\{d_T(v,s) : s \in leaves(T), \ u \in path_T(v,s)\} \ \}.$$

Lemma 5.2. *Let $\emptyset \subset S' \subset S$. Then there is a pair of terminal taxa $y \in S', x \in S \setminus S'$ s.t. $d_T(x,y) \leq 2depth(T)$.*

Proof. Let T' be the subtree of T which spans S' (note that T' may have vertices of degree two). A vertex of T' is said to be full if $deg_{T'}(v) = deg_T(v)$. Observe that all leaves of T' are full, and that since $S' \neq S$, there exists a non-full vertex in T'. Now root T (and T') at some arbitrary vertex r, and let $v \in V(T')$ be a non-full (internal) vertex which maximizes $d_T(r,v)$. Let u be an arbitrary child of v in T'. Then, by the maximality of $d_T(r,v)$, u and all its descendants in T' are full. Hence there is a taxon $y \in S'$ which is a descendant of u in T' and $d_T(v,y) \leq depth(T)$. Also, since v is not full, it has a child $u' \in V(T) \setminus V(T')$ s.t. u' and all its descendants in T are not in T', implying similarly that u' has a descendant taxon $x \in S \setminus S'$ s.t. $d_T(v,x) \leq depth(T)$. Thus $d_T(x,y) = d_T(x,v) + d_T(v,y) \leq 2depth(T)$. □

The objective of the following analysis is to use Lemma 5.2 to bound the diameter of the quartets in $Quart(\widehat{T}, x, \varepsilon)$. Our analysis is similar to the one in other incremental fast converging algorithm [6, 20], and is based on the proximity of the input distance estimates $\widehat{D}$ to the true additive distances D_T. The difference between these two dissimilarity matrices is measured by a non-decreasing function α, as defined below.

Definition 5.3 (α-close dissimilarity matrices)**.** *Given a non-decreasing function $\alpha : \mathbb{R}^+ \to \mathbb{R}^+ \cup \{\infty\}$, two dissimilarity matrices D_1, D_2 over the same taxon set S are said to be* α-close *if for every $\{i,j\} \subseteq S$,*

$$|d_1(i,j) - d_2(i,j)| \ \leq \ \alpha(\min\{d_1(i,j), d_2(i,j)\}) \ . \tag{5}$$

Given a noise function α, we use the following notation for a "double portion" of noise: $\alpha^{+2}(d) = \alpha(d) \ + \ \alpha(d + \alpha(d))$. Note that $2\alpha(d) \ \leq \ \alpha^{+2}(d) \ \leq \ 2\alpha(d + \alpha(d))$. The following lemma provides two basic bounds.

Lemma 5.4. *Consider an arbitrary insertion iteration of the incremental algorithm, and let x be the terminal taxon chosen for insertion (according to the order defined above). Denote by y the reference taxon of x and by p_x the parent of x in $T(S' \cup \{x\})$. Then the following bounds are implied by the α-closeness of $\widehat{D}$ and D_T:*

$$d_T(x,y) \ \leq \ 2depth(T) + \alpha^{+2}(2depth(T)) \ . \tag{6}$$

$$d_T(x,p_x) \ \leq \ depth(T) + \alpha^{+2}(2depth(T)) \ . \tag{7}$$

Proof. Throughout the proof we use *depth* in short to denote $depth(T)$ and extensively use (5) for the α-close matrices $\widehat{D}, D_T$. We start by proving that $\hat{d}(x,y) \leq 2depth + \alpha(2depth)$. According to Lemma 5.2, there are two terminal taxa $x' \in S \setminus S'$ and $y' \in S'$ s.t. $d_T(x',y') \leq 2depth$. Due to the criterion by which x and y are chosen, we have $\hat{d}(x,y) \leq \hat{d}(x',y')$, and thus,

$$\hat{d}(x,y) \ \leq \ \hat{d}(x',y') \ \leq \ d_T(x',y') + \alpha(d_T(x',y')) \ \leq \ 2depth + \alpha(2depth) \ . \tag{8}$$

Inequality (6) is directly implied by (8) and by $d_T(x,y) \leq \hat{d}(x,y) + \alpha(\hat{d}(x,y))$.

We now turn to prove (7). Note that there must be a terminal taxon $x' \in S \setminus S'$ s.t. $d_T(p_x, x') \leq depth$ and $d_T(x, p_x) - d_T(x', p_x) = d_T(x, y) - d_T(x', y)$. Hence,

$$d_T(x, p_x) \;=\; d_T(x', p_x) + [d_T(x, y) - d_T(x', y)] \;\leq\; depth + [d_T(x, y) - d_T(x', y)] \;.$$

We are left to show that $d_T(x, y) - d_T(x', y) \leq \alpha^{+2}(2depth)$. We distinguish between two cases. If $d_T(x', y) \geq 2depth$, then (by (6)),

$$d_T(x, y) - d_T(x', y) \;\leq\; d_T(x, y) - 2depth \;\leq\; \alpha^{+2}(2depth) \;.$$

Otherwise, $d_T(x', y) \leq 2depth$, and we get the following due to $\hat{d}(x, y) \leq \hat{d}(x', y)$:

$$d_T(x, y) - d_T(x', y) \;\leq\; [d_T(x, y) - \hat{d}(x, y)] + [\hat{d}(x', y) - d_T(x', y)] \;\leq\; \alpha(\hat{d}(x, y)) + \alpha(d_T(x', y)) \;.$$

Now, since $d_T(x', y) \leq 2depth$ and $\hat{d}(x, y) \leq 2depth + \alpha(2depth)$, we get that

$$d_T(x, y) - d_T(x', y) \;\leq\; \alpha(\hat{d}(x, y)) + \alpha(d_T(x, y)) \;\leq\; \alpha^{+2}(2depth) \;. \qquad \square$$

Corollary 5.5. *At all stages of the incremental algorithm, the following is satisfied:*

1. *Every edge in $\widehat{T}$ has weight at most $depth(T) + \alpha^{+2}(2depth(T))$.*
2. *For every directional representative $s_{\hat{v}}(\hat{u})$ maintained by the algorithm, we have $d_T(t_{\hat{v}}, s_{\hat{v}}(\hat{u})) \leq 2depth(T) + \alpha^{+2}(2depth(T))$.*

Proof. The corollary clearly holds after the first iteration due to (6). Assume it holds for a certain iteration, and let x be the taxon inserted at that point. Most of the tree (including directional representatives) remains unchanged, and so the lemma holds by induction for all edges except possibly for the new external edge $(\hat{p}_x, x)$ and its two directional representatives. The weight of this edge in bounded by (7), and its directional representatives are $s_{\hat{p}_x}(x) = x$ (where $d_T(t_{\hat{p}_x}, x)$ is bounded by (7)) and $s_x(\hat{p}_x) = y$ (where $d_T(x, y)$ is bounded by (6)). $\square$

The bound on the diameters of quartets in $Quart(\widehat{T}, x, \varepsilon)$ has to take into account the diameters of subtrees contracted during the algorithm, which are shown to be bounded by the *ε-diameter* of the tree, defined next.

Definition 5.6 (ε-diameter)**.** *For $\varepsilon \geq 0$, the* ε-diameter *of T (denoted by $diam(T, \varepsilon)$) is the maximum weight of a simple path in T consisting only of edges of weight at most ε.*

The desired bound on the diameter of queried quartets is given next.

Lemma 5.7. *Let $\widehat{T}$ be an edge-contraction of $T(S')$ kept by the incremental algorithm at some stage, and let x be the taxon chosen for insertion. Further, let $\varepsilon \geq 0$ be s.t. $\widehat{T}$ is an ε-contraction of $T(S')$. Then all quartets in $Quart(\widehat{T}, x, \varepsilon)$ are r-short, where*

$$r \;=\; 4depth(T) \;+\; 3\alpha^{+2}(2depth(T)) \;+\; 2diam(T, \varepsilon) \;. \tag{9}$$

Proof. Let $\hat{v}$ be an arbitrary leaf of $\hat{t}_{env}(\widehat{T}, x, \varepsilon)$. We have to show that for every neighbor $\hat{u} \in N_{\widehat{T}}(\hat{v})$, $d_T(x, s_{\hat{v}}(\hat{u})) \leq r$, and for every neighbor-pair $\hat{u}_1, \hat{u}_2 \in N_{\widehat{T}}(\hat{v})$, $d_T(s_{\hat{v}}(\hat{u}_1), s_{\hat{v}}(\hat{u}_2)) \leq r$. First, consider $d_T(s_{\hat{v}}(\hat{u}_1), s_{\hat{v}}(\hat{u}_2))$. Corollary 5.5(2) implies that $d_T(t_{\hat{v}}, s_{\hat{v}}(\hat{u}_i)) \;\leq\; 2depth(T) + \alpha^{+2}(2depth(T))$ for $i = 1, 2$. Furthermore, since $\widehat{T}$ is an ε-contraction of $T(S')$, we get $diam(t_{\hat{v}}) \leq diam(T, \varepsilon)$. Hence,

$$d_T(s_{\hat{v}}(\hat{u}_1), s_{\hat{v}}(\hat{u}_2)) \;\leq\; 4depth(T) + 2\alpha^{+2}(2depth(T)) + diam(T, \varepsilon) \;\leq\; r.$$

Now, consider $d_T(x, s_{\hat{v}}(\hat{u}))$. This distance is bounded by dividing $path_T(x, s_{\hat{v}}(\hat{u}))$ into four sections:

- The path from x to p_x (the parent of x in $T(S' \cup \{x\})$. According to Lemma 5.4(7), the weight of this path is at most $depth(T) + \alpha^{+2}(2depth(T))$.
- The path from p_x to $t_{\hat{v}}$. The weight of the last edge touching $t_{\hat{v}}$ is at most $depth(T) + \alpha^{+2}(2depth(T))$ (according to Corollary 5.5(1)). The rest of the path consist of edges which are either contracted in $\widehat{T}$ or internal edges in $\hat{t}_{env}(\widehat{T}, x, \varepsilon)$. Either way, their weight is at most ε. Hence the weight of the path from p_x to $t_{\hat{v}}$ is at most $depth(T) + \alpha^{+2}(2depth(T)) + diam(T, \varepsilon)$.
- The path within $t_{\hat{v}}$. This path consists of contracted edges, and so its weight is at most $diam(T, \varepsilon)$.
- The path from $t_{\hat{v}}$ to $s_{\hat{v}}(\hat{u})$. According to Corollary 5.5(2), the weight of this path is at most $2depth(T) + \alpha^{+2}(2depth(T))$.

Combining these bounds together yields r as defined in (9). □

We conclude this discussion with a corollary of Lemmas 4.9 and 5.7, which establishes the reliability criterion required of the partial quartet oracle.

Corollary 5.8. *Algorithm Incremental_Reconstruct constructs an ε-contraction of T if it uses a partial quartet oracle PQO which is (r, ε)-reliable, where*

$$r \;=\; 4depth(T) \;+\; 3\alpha^{+2}(2depth(T)) \;+\; 2diam(T, \varepsilon)\ . \tag{10}$$

We now turn to describe an (r, ε)-reliable distance-based partial quartet oracle. The partial quartet oracle $FPM_{\widehat{D},\alpha}$ described below uses the input distance estimates in $\widehat{D}$ and the noise function α to reliably infer quartet splits. This oracle is a modified version of the well known *four-point method* (FPM) for inferring quartet splits, which dates back to [34]. A similar variant of FPM is used in the forest reconstruction algorithm of [8]. The reliability of this oracle is tied to the noise function α in Lemma 5.9 below.

The Partial Quartet Oracle – $FPM_{\widehat{D},\alpha}(q)$**:**
Let $q = \{a, b, c, d\}$, and assume a labeling of the four terminal taxa which satisfies:

$$\hat{d}(a,b) + \hat{d}(c,d) \;\leq\; \min\{\hat{d}(a,c) + \hat{d}(b,d)\ ,\ \hat{d}(a,d) + \hat{d}(b,c)\}\ .$$

– Compute the following three values:

$$\begin{aligned} A &= [\hat{d}(a,b) + \hat{d}(c,d)] + [\alpha(\hat{d}(a,b)) + \alpha(\hat{d}(c,d))] \\ B &= [\hat{d}(a,c) + \hat{d}(b,d)] - [\alpha(\hat{d}(a,c)) + \alpha(\hat{d}(b,d))] \\ C &= [\hat{d}(a,d) + \hat{d}(b,c)] - [\alpha(\hat{d}(a,d)) + \alpha(\hat{d}(b,c))] \end{aligned}$$

– If $A < \min\{B, C\}$, then return $(ab|cd)$. Otherwise, return *'null'*.

Lemma 5.9. *Assume that $\widehat{D}$ is α-close to D_T. Then for every $r, \varepsilon \geq 0$, s.t. $\alpha^{+2}(r) \leq \frac{\varepsilon}{2}$, $FPM_{\widehat{D},\alpha}$ is an (r, ε)-reliable partial quartet oracle for T.*

Proof. To establish the *truthfulness* of $FPM_{\widehat{D},\alpha}$ as a quartet oracle, we need to show that if $A < \min\{B, C\}$ then $(a, b|c, d)$ is the correct quartet split of $T(q)$. This is implied by the inequality $d_T(a,b) + d_T(c,d) \;<\; \min\{d_T(a,c) + d_T(b,d)\ ,\ d_T(a,d) + d_T(b,c)\}$:

$$\begin{aligned} d_T(a,b) + d_T(c,d) \;\leq\; A \;<\; B \;&\leq\; d_T(a,c) + d_T(b,d) \\ d_T(a,b) + d_T(c,d) \;\leq\; A \;<\; C \;&\leq\; d_T(a,d) + d_T(b,c) \end{aligned}$$

We turn to prove the (r, ε)-reliability of $FPM_{\widehat{D},\alpha}$. Consider an arbitrary r-short quartet $q = \{a, b, c, d\}$ (for some $r \geq 0$). Since q is r-short, every $\{i, j\} \subset q$, satisfy $|\hat{d}(i,j) -$

$d_T(i,j)| \leq \alpha(r)$, and furthermore $\alpha(\hat{d}(i,j)) \leq \alpha^{+2}(r) - \alpha(r)$. Now assume that the split of $T(q)$ is $(a,b|c,d)$ and that q is ε-separated (for some $\varepsilon \geq 0$). Then:

$$d_T(a,b) + d_T(c,d) + 2\varepsilon \quad < \quad d_T(a,c) + d_T(b,d) \quad = \quad d_T(a,d) + d_T(b,c) \ .$$

This inequality is used to obtain the following series of inequalities:

$$\begin{aligned} A &\leq [\hat{d}(a,b) + \hat{d}(c,d)] + 2\alpha^{+2}(r) - 2\alpha(r) \\ &\leq [d_T(a,b) + d_T(c,d)] + 2\alpha^{+2}(r) \\ &< [d_T(a,c) + d_T(b,d)] - 2\varepsilon + 2\alpha^{+2}(r) \\ &\leq [\hat{d}(a,c) + \hat{d}(b,d)] - 2\varepsilon + 2\alpha^{+2}(r) + 2\alpha(r) \\ &\leq B - 2\varepsilon + 4\alpha^{+2}(r) \ . \end{aligned}$$

Hence, $A < B + (4\alpha^{+2}(r) - 2\varepsilon)$, and similarly we can obtain $A < C + (4\alpha^{+2}(r) - 2\varepsilon)$. Therefore, if $\alpha^{+2}(r) \leq \frac{\varepsilon}{2}$, then $A < \min\{B, C\}$, and the split $(ab|cd)$ is recovered by $FPM_{\widehat{D},\alpha}(q)$. □

Theorem 5.10 summarizes the results in this section by specifying an upper bound ε on the weights of contracted edges as a function of the tree depth, its ε-diameter and the noise function α. In Section 7 we present a modification of the insertion process which avoids the dependence in the ε-diameter at the cost of increasing the constant in the linear dependence of ε on the depth of the tree (see Theorem 7.10).

Theorem 5.10. *Consider a phylogenetic tree T over a set S of terminal taxa. Let α be a non-decreasing function, and assume that the input dissimilarity matrix $\widehat{D}$ is α-close to the additive metric D_T. Let $\varepsilon \geq 0$ be s.t.*

$$\alpha^{+2}\left(4depth(T) + 2diam(T,\varepsilon) + 1.5\varepsilon\right) \ \leq \ \frac{\varepsilon}{2} \ . \tag{11}$$

Then, when executed on $\widehat{D}$, algorithm Incremental_Reconstruct which uses $FPM_{\widehat{D},\alpha}$, returns an ε-contraction of T.

Proof. First, denote

$$\begin{aligned} r &= 4depth(T) + 2diam(T,\varepsilon) + 1.5\varepsilon \\ r' &= 4depth(T) + 3\alpha^{+2}(2depth(T)) + 2diam(T,\varepsilon) \\ &= r + 3(\alpha^{+2}(2depth(T)) - \frac{\varepsilon}{2}). \end{aligned} \tag{12}$$

By Corollary 5.8 and Lemma 5.9, all we have to establish is that $\alpha^{+2}(r') \leq \frac{\varepsilon}{2}$. The monotonicity of α^{+2} (implied by the monotonicity of α) and (11) imply that $\alpha^{+2}(d) \leq \frac{\varepsilon}{2}$ for every $d \leq r$. Finally, $r' \leq r$ is established by the following series of inequalities (the last one implied by (12)):

$$2depth(T) \leq r \quad \Rightarrow \quad \alpha^{+2}(2depth(T)) \leq \frac{\varepsilon}{2} \quad \Rightarrow \quad r' \leq r \ .$$

□

6 Fast Convergence and Adaptive Fast Convergence.

The objective of this section is to provide a connection between the length of input sequences and reconstruction guarantees of our incremental algorithm. This is done through application of Theorem 5.10 with the appropriate noise function α_k, which depends on the length k of the input sequences. The noise function α_k is determined by assuming the input sequences evolve along the tree T through some stochastic site-substitution model. Different site substitution models have different formulas for distance estimation (see, e.g., [16]) which imply slightly different noise functions. The noise patterns for simple models like the Cavender-Farris-Neyman (CFN) model [3, 13, 27] or the Jukes-Cantor model [18] are well

studied (see, e.g., [11]). But similar noise patterns appear in more complex models [12, 16]. Since the underlying models are stochastic, the noise function α_k is 'probabilistic' in the sense that it bounds the noise with sufficiently high probability. In the appendix we develop the noise function under the CFN model. We show (Theorem A.4) that the distances computed from input sequences of length k in the CFN model are w.h.p. α_k-close to the additive distances associated with the true tree T, where α_k is the following non-decreasing function:

$$\alpha_k(d) \;=\; -\frac{1}{2}\ln\left[1 - e^{2d}\sqrt{\frac{6\ln(n)}{k}}\right] \;. \tag{13}$$

(n denotes the number of terminal taxa, and we use the convention that for $z \leq 0$, $\ln(z) = -\infty$.)

The following lemma provides a result which simplifies the use of the noise function α_k:

Lemma 6.1. *Let $k, d, \varepsilon > 0$ be s.t. $k \geq \frac{3}{2}\frac{\ln(n)}{\varepsilon^2}e^{4d+2\varepsilon}$, then $\alpha_k(d) < \varepsilon$.*

Proof. Consider $\alpha_k(d)$, for $k \geq \frac{3}{2}\frac{\ln(n)}{\varepsilon^2}e^{4d+2\varepsilon}$.

$$\begin{aligned}
\alpha_k(d) &= -\frac{1}{2}\ln\left[1 - e^{2d}\sqrt{\frac{6\ln(n)}{k}}\right] \\
&\leq -\frac{1}{2}\ln\left[1 - e^{2d}\sqrt{\frac{4\varepsilon^2}{e^{4d+2\varepsilon}}}\right] \\
&= -\frac{1}{2}\ln\left[1 - 2\frac{\varepsilon}{e^{\varepsilon}}\right] \\
\textbf{(*)} \quad &< -\frac{1}{2}\ln\left[e^{-2\varepsilon}\right] \;=\; \varepsilon \;.
\end{aligned}$$

The last inequality (*) is based on the Taylor series for the *hyperbolic sine* function:

$$\sinh(\varepsilon) \;=\; \frac{e^{\varepsilon} - e^{-\varepsilon}}{2} \;=\; \varepsilon + \frac{\varepsilon^3}{3!} + \frac{\varepsilon^5}{5!} + \cdots \;>\; \varepsilon \quad (\text{for } \varepsilon > 0),$$

which implies that (for $\varepsilon > 0$):

$$1 - 2\frac{\varepsilon}{e^{\varepsilon}} \;>\; 1 - 2\frac{\sinh(\varepsilon)}{e^{\varepsilon}} \;=\; e^{-2\varepsilon} \;.$$

□

The following lemma formulates a (rather standard) argument for establishing the fast convergence of reconstruction algorithms.

Lemma 6.2 (Fast Convergence)**.** *Let $\mathcal{A}$ be a distance-based phylogenetic reconstruction algorithm which satisfies the following condition for some positive constants c_1, c_2:*

- *For every phylogenetic tree T and every input dissimilarity matrix $\widehat{D}$ which is α-close to D_T (for some non-decreasing function α), it holds that:*
  If $c_2\alpha(c_1 depth(T)) < \min\{w(e) : e \in E(T)\}$, then $\mathcal{A}$ correctly reconstructs T.

Then $\mathcal{A}$ is fast converging.

Proof. Let T be an arbitrary tree with n terminal taxa whose edge weights are within the interval $[f, g]$. We need to prove that algorithm $\mathcal{A}$ which satisfies the condition of the lemma is guaranteed to correctly reconstruct T w.h.p. from input sequences of length $k = \frac{n^{O(g)}}{f^2}$. Let $k = c_2'\frac{\ln(n)}{f^2}e^{c_1' depth(T)}$ for $c_1' = 4c_1 + \frac{2}{c_2}$ and $c_2' = \frac{3}{2}c_2^2$ (where c_1, c_2 are the constants

from the condition of the lemma). First, note that since $depth(T) \leq cg\ln(n)$ (for some constant c), the sequence length in polynomial in n, as required. All we need to show is that this sequence length is sufficient to guarantee correct reconstruction of T (w.h.p.). Since $f \leq depth(T)$, the following holds for $\varepsilon = \frac{f}{c_2}$:

$$k \;=\; \frac{3}{2}\, c_2^2 \frac{\ln(n)}{f^2} e^{(4c_1+\frac{2}{c_2})depth(T)} \;=\; \frac{3}{2}\frac{\ln(n)}{\varepsilon^2} e^{4c_1 depth(T)+2\frac{depth(T)}{c_2}} \;\geq\; \frac{3}{2}\frac{\ln(n)}{\varepsilon^2} e^{4c_1 depth(T)+2\varepsilon}\ .$$

This implies, through Lemma 6.1, that $\alpha_k(c_1 depth(T)) < \varepsilon = \frac{f}{c_2}$, and so $c_2\alpha_k(c_1 depth(T)) < w(e)$ for every edge $e \in E(T)$. In such a case, the condition of the lemma states that if the input dissimilarity matrix computed from the input sequences is α_k-close to D_T, then $\mathcal{A}$ correctly reconstructs the tree T. According to Theorem A.4, this is guaranteed to happen with probability greater than $1 - \frac{1}{n}$. □

Theorem 6.3. *The incremental algorithm presented in Sections 3–5 is fast converging.*

Proof. We use Lemma 6.2, and show that our algorithm satisfies the condition of this lemma with $c_1 = 5.75$ and $c_2 = 4$. Let T be an arbitrary phylogenetic tree and let $\widehat{D}$ be a dissimilarity matrix which is α-close to D_T, s.t. $4\alpha(5.75depth(T)) < f \stackrel{\triangle}{=} \min\{w(e) : e \in E(T)\}$. We need to prove that the incremental algorithm correctly reconstructs T when given $\widehat{D}$ (and α) as input. Theorem 5.10 implies that this happens when $\alpha^{+2}(4depth(T) + 2diam(T,\varepsilon)+1.5\varepsilon) \leq \frac{\varepsilon}{2}$ for some $\varepsilon < f$. Denote $\varepsilon = 4\alpha(5.75depth(T))$, and $r = 4depth(T)+ 2diam(T,\varepsilon) + 1.5\varepsilon$. We need to show that if $\varepsilon < f$ then $\alpha^{+2}(r) \leq \frac{\varepsilon}{2}$. Since $\varepsilon < f$, we have $diam(T,\varepsilon) = 0$ and $\varepsilon < depth(T)$, which imply that $\alpha(r) \leq \alpha(5.5depth(T)) \leq \frac{\varepsilon}{4}$. Hence,

$$\alpha^{+2}(r) \;\leq\; 2\alpha(r+\alpha(r)) \;\leq\; 2\alpha(r+\frac{\varepsilon}{4}) \;\leq\; 2\alpha(5.75depth(T)) = \frac{\varepsilon}{2}\ ,$$

as required. □

Note that our algorithm is actually *absolute* fast converging, since it requires no prior knowledge on the parameters of the tree (e.g., its depth or minimal edge weight) in order to reconstruct it. As such, it is the first (absolute) fast converging algorithm which provides an upper bound on the weights of edges it does not reconstruct. The following lemma and the subsequent discussion imply that our algorithm is in fact *adaptive* fast converging in nearly all cases, excluding some pathological scenarios.

Lemma 6.4 (Adaptive Fast Convergence)**.** *Let $\mathcal{A}$ be a distance-based phylogenetic reconstruction algorithm which satisfies the following condition for some positive constants c_1, c_2:*

- *For every $\varepsilon \geq 0$, phylogenetic tree T, and input dissimilarity matrix $\widehat{D}$ which is α-close to D_T (for some non-decreasing function α), if $c_2\alpha(c_1 depth(T)) \leq \varepsilon$ then $\mathcal{A}$ correctly reconstructs all edges of T of weight greater than ε.*

Then $\mathcal{A}$ is adaptive fast converging.

Proof. Let T be an arbitrary tree with n terminal taxa whose edge weights do not exceed g. We need to prove that algorithm $\mathcal{A}$ which satisfies the condition of the lemma is guaranteed to reconstruct w.h.p. all edges of T of weight greater than ε from input sequences of length $k = \frac{n^{O(g)}}{\varepsilon^2}$. Note that if $\varepsilon > depth(T)$, then this claim holds vacuously (since no edge has weight greater than $depth(T)$). Assuming that $\varepsilon \leq depth(T)$, we can use a similar line of argument used in the proof of Lemma 6.2: we define $k = c_2' \frac{\ln(n)}{\varepsilon^2} e^{c_1' depth(T)}$ for $c_1' = 4c_1 + \frac{2}{c_2}$ and $c_2' = \frac{3}{2}c_2^2$, and show that $c_2\alpha_k(c_1 depth(T)) \leq \varepsilon$. By denoting $\varepsilon' = \frac{\varepsilon}{c_2}$ and using the assumption that $\varepsilon \leq depth(T)$, we obtain:

$$k \;=\; \frac{3}{2}\, c_2^2 \frac{\ln(n)}{\varepsilon^2} e^{(4c_1+\frac{2}{c_2})depth(T)} \;=\; \frac{3}{2}\frac{\ln(n)}{\varepsilon'^2} e^{4c_1 depth(T)+2\frac{depth(T)}{c_2}} \;\geq\; \frac{3}{2}\frac{\ln(n)}{\varepsilon'^2} e^{4c_1 depth(T)+2\varepsilon'}\ .$$

This implies, through Lemma 6.1, that $c_2\alpha_k(c_1 depth(T)) < \varepsilon$, and so through application of Theorem A.4 and the condition of the lemma we get that all edges of weight greater than ε are reconstructed w.h.p.. □

Discussion: Assume, for the moment, that for all inputs T and ε which are of interest, it holds that $2diam(T,\varepsilon) \leq depth(T)$ (the coefficient 2 could be replaced by any other positive constant). Then Lemma 6.4 implies (through Theorem 5.10) that our algorithm is adaptive fast converging. The above assumption ($2diam(T,\varepsilon) \leq depth(T)$) is likely to hold in practice whenever we are interested in a weight threshold ε which is not too large, and the phylogenetic tree of interest does not contain long paths consisting only of very short edges. However, in order to make our algorithm adaptive fast converging without this assumption, any dependence of our result on the diameter of contracted components needs to be eliminated. This task is addressed in the next section.

7 Eliminating Dependence on the Diameter of Contracted Subtrees.

This section presents an alternative insertion procedure (*Attach_Taxon_revised*) which eliminates the dependence of the analysis on the diameter of contracted subtrees. The intuition behind the revised insertion procedure is simple: when inserting a new terminal taxon x into a subphylogeny $\widehat{T}$ induced by the taxon set S', we ignore terminal taxa which are far from x, and insert x in a subphylogeny of $\widehat{T}$ induced by a subset $S'' \subseteq S'$ containing only terminal taxa which are close to x. To ensure that the insertion in the induced subphylogeny $\widehat{T}(S'')$ guarantees a correct insertion in $\widehat{T}$, we need that $\widehat{T}(S'')$ *preserves* the anchor of x in $\widehat{T}$, in some natural sense to be defined.

Lemma 7.1. *Assume that $\widehat{T}(S'')$ is an internal contraction of $T(S'')$ and that PQO is (r,ε)-reliable for $r = diam(T(S'' \cup \{x\}))$. Then $V(\hat{t}_{inz}(\widehat{T}(S''), x, PDO)) \subseteq V(\hat{t}_{env}(\widehat{T}(S''), x, \varepsilon))$.*

Proof. Follows the same lines as the proof of Lemma 4.9. □

Lemma 7.2. *Let $\widehat{T}$ be an internal contraction of $T(S')$, and Let $S'' \subseteq S'$. Then*

1. *$\widehat{T}(S'')$, the subtree of $\widehat{T}$ induced by S'', is an internal contraction of $T(S'')$.*
2. *An edge $(u,v) \in E(T(S''))$ is contracted into a vertex $\hat{v}$ of $\widehat{T}(S'')$ iff* all *the edges in $path_{T(S')}(u,v)$ are contracted into $\hat{v}$ in $\widehat{T}$.*

Proof. We prove the lemma for the case where $S'' = S' \setminus \{s\}$ (the general case follows by induction on $|S' \setminus S''|$). The proof (of 1 and 2 above) is established by mapping each internal vertex $\hat{v}$ of $\widehat{T}(S'')$ onto a subtree $t''_{\hat{v}}$ of $T(S'')$, as follows. Consider an arbitrary vertex $\hat{v} \in internal(\widehat{T})$, and the subtree $t_{\hat{v}}$ of $T(S')$ contracted into it in $\widehat{T}$. If $V(t_{\hat{v}}) \subseteq V(T(S''))$, then $t_{\hat{v}}$ is also a subtree of $T(S'')$, and $t''_{\hat{v}} = t_{\hat{v}}$. If this is not the case, then $t_{\hat{v}}$ must contain p_s, the parent of s in $T(S')$ (since p_s is the only possible vertex in $internal(T(S')) \setminus internal(T(S''))$). Furthermore, in this case we must have that $deg_{T(S')}(p_s) = 3$, and $N_{T(S')}(p_s) = \{s,u,v\}$ for some edge $(u,v) \in E(T(S''))$. There are three possible cases:

1. $t_{\hat{v}}$ contains neither (u,p_s) nor (p_s,v), in which case $V(t_{\hat{v}}) = \{p_s\}$, and $\hat{v} \notin V(\widehat{T}(S''))$.
2. $t_{\hat{v}}$ contains exactly one of the edges $(u,p_s),(p_s,v)$ - w.l.o.g. it is (u,p_s). Then $t''_{\hat{v}} = t_{\hat{v}} \setminus \{(u,p_s)\}$.
3. $t_{\hat{v}}$ contains both (u,p_s) and (p_s,v), in which case $t''_{\hat{v}} = t_{\hat{v}} \setminus \{(u,p_s),(p_s,v)\} \cup \{(u,v)\}$.

Note that the edge (u,v) of $T(S'')$ is contracted in $\widehat{T}(S'')$ only in the third case above. □

The correctness of our insertion procedure depends on the condition that the subset S'' *preserves* the anchor of x in $T(S')$, as specified by the following definitions and results.

Definition 7.3. *Let v be an internal vertex of a phylogenetic tree T. The* v-components *of T are the connected components of $T(V \setminus \{v\})$.*

Definition 7.4. *Let T be a phylogenetic tree, let $v \in V(T)$ and let $\tilde{S} \subseteq leaves(T)$. We say that $\tilde{S}$* preserves v in T*, if either v is a terminal taxon in $\tilde{S}$, or $\tilde{S}$ has nonempty intersections with at least three of the v-components of T. $\tilde{S}$ is said to* preserve an edge (u, v) *in T if it preserves both u and v in T.*

Observation 7.5. *A vertex v of $T(S')$ is also a vertex in $T(S'')$ iff S'' preserves v in $T(S')$. Similarly, an edge (u, v) of $T(S')$ is also an edge of $T(S'')$ iff S'' preserves the edge (u, v) in $T(S')$. (see Fig. 9)*

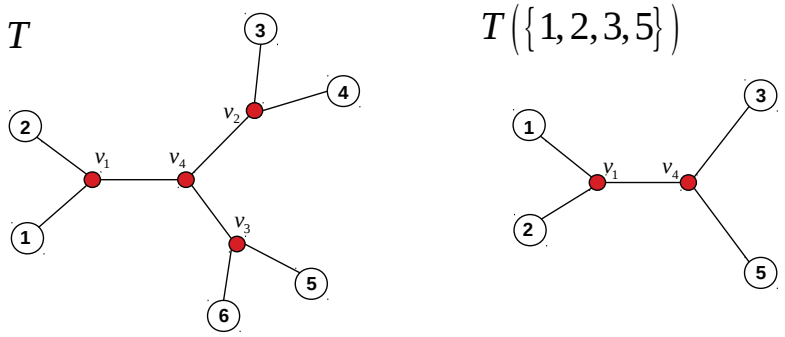

Figure 9: **Vertex and Edge Preservation.** The taxon subset $\{1, 2, 3, 5\}$ preserves the internal vertices v_1, v_4 and the edges $(1, v_1), (2, v_1), (v_1, v_4)$ in T.

Lemma 7.6. *Let $\widehat{T}$ be an internal edge-contraction of $T(S')$, and assume that S'' preserves $anchor_{T(S')}(x)$. Then $anchor_{\widehat{T}}(x) = anchor_{\widehat{T}(S'')}(x)$.*

Proof. Assume that $anchor_{T(S')}(x)$ is an edge e (the proof for the case that $anchor_{T(S')}(x)$ is an internal vertex is similar but simpler). Then by Observation 7.5, e is also an edge in $T(S'')$. Assume first that e is contracted in $\widehat{T}$ into some vertex $\hat{v}$, implying that $anchor_{\widehat{T}}(x) = \hat{v}$. Then by Lemma 7.2(2), e is also contracted into $\hat{v}$ in $\widehat{T}(S'')$, implying that $anchor_{\widehat{T}(S'')}(x) = \hat{v}$ as well. Now assume that e is not contracted in $\widehat{T}$, then (by Lemma 7.2(2)) it is also not contracted in $\widehat{T}(S'')$. Thus, both $\widehat{T}$ and $\widehat{T}(S'')$ contain an edge $\hat{e}$ which is identical to e (i.e., $\hat{e}$ and e induce the same splits in the corresponding trees). Hence $\hat{e}$ is the anchor of x in both $\widehat{T}$ and $\widehat{T}(S'')$. □

Lemmas 7.1 and 7.6 suggest the following insertion procedure:

Procedure *Attach_Taxon_revised*($\widehat{T}, x$):

1. Calculate a subset $S'' \subseteq S'$, s.t. $diam(T(S'' \cup \{x\})) = O(depth(T))$ and S'' preserves the anchor of x in $T(S')$.
2. $\hat{t}''_{inz} \leftarrow \hat{t}_{inz}(\widehat{T}(S''), x, PDO)$.
3. If $\hat{t}''_{inz}$ is a single edge $(\hat{u}, \hat{v})$, then attach x to $\widehat{T}$ by introducing a new internal vertex $\hat{p}_x$ and replacing $(\hat{u}, \hat{v})$ with the three edges $(\hat{u}, \hat{p}_x), (\hat{v}, \hat{p}_x), (x, \hat{p}_x)$.
4. If $\hat{t}''_{inz}$ has a single internal vertex $\hat{v}$ (i.e., $V(\hat{t}''_{inz}) = N(\hat{v}) \cup \{\hat{v}\}$), then add to $\widehat{T}$ the edge $(\hat{v}, x)$.
5. Else (i.e., $\hat{t}''_{inz}$ has at least one internal edge), contract into a new vertex $\hat{v}$ all edges in $\widehat{T}$ which lie on paths corresponding to internal edges of $\hat{t}''_{inz}$, and add to $\widehat{T}$ the edge $(\hat{v}, x)$.

Implementation note: Before computing the insertion zone in step 2, the algorithm computes S'' (as defined in Lemma 7.9) and constructs $\widehat{T}(S'')$. This can be done by rooting $\widehat{T}$ at some arbitrary leaf r and performing a DFS scan from that root-taxon, eliminating all vertices which do not have descendants in S'' (and contracting internal vertices of degree two). Since directional representatives have to be taxa in S'', they are not kept from one iteration to the next. However, by the way S'' is defined, a directional representative $s_{\hat{v}}(\hat{u})$ can be any taxon in S'' which is separated from $\hat{v}$ by $\hat{u}$. So if $\hat{u}$ is the parent of $\hat{v}$ in the above rooting, then $s_{\hat{v}}(\hat{u})$ is set to the root-taxon r, and if $\hat{u}$ is a child of $\hat{v}$, then $s_{\hat{v}}(\hat{u})$ is set to an arbitrary leaf in the subtree rooted by $\hat{u}$. These directional representatives can be computed as $\widehat{T}(S'')$ is being constructed during the same DFS scan. Hence, each insertion iteration of the modified algorithm is still performed in linear time.

Lemma 7.7. *Assume that $\widehat{T}$ is an ε-contraction of $T(S')$, and let $\widehat{T}_{post}$ be the topology resulting from Attach_Taxon_revised($\widehat{T}, x$). Then if* (a) *S'' preserves the anchor of x in $T(S')$ and* (b) *PQO is (r, ε)-reliable, where $r = diam(T(S'' \cup \{x\}))$, then $\widehat{T}_{post}$ is an ε-contraction of $T(S' \cup \{x\})$.*

Sketch of proof. Attach_Taxon_revised differs from *Attach_Taxon* only in that the insertion zone $\hat{t}''_{inz}$ is computed on the induced subtree $\widehat{T}(S'')$ and not on $\widehat{T}$. Thus the fact that all contracted edges are of weight at most ε is proved as for *Attach_Taxon*, noting that internal edges of $\hat{t}''_{inz}$ correspond to edges of weight at most ε in $\widehat{T}$. The fact that the taxon x is attached to its anchor in $\widehat{T}_{post}$ is based on a case analysis similar to the one in the proof of Lemma 3.1, using the fact that S'' preserves the anchor of x in $T(S')$. The lemma follows. □

We are left to specify how the procedure finds a reduced subset of terminal taxa S'' s.t. $diam(T(S'' \cup \{x\})) = O(depth(T))$ and S'' preserves the anchor of x in $T(S')$. We assume, as in Section 5, that the algorithm has access to a dissimilarity matrix $\widehat{D}$ which is α-close to the tree-induced metric D_T. We also assume the insertion order suggested in Section 5. Let S'_i be the set of attached terminal taxa in the beginning of the i^{th} iteration, and let $x_i \notin S'_i$ be the taxon chosen for insertion at that stage. Denote by y_i the reference taxon of x_i in S'_i (s.t. $\hat{d}(x_i, y_i) = \min\{\hat{d}(x', y') : x' \notin S'_i, y' \in S'_i\}$). Our definition of S'' relies on the following observation (which can be proved by induction on i):

Observation 7.8. *Let $\bar{d}_i = \max_{j \le i}\{\hat{d}(x_j, y_j)\}$. Then $depth(T(S'_i \cup \{x_i\})) \le \bar{d}_i + \alpha(\bar{d}_i)$.*

Lemma 7.9. *Consider the i^{th} iteration in which taxon $x = x_i$ is inserted into $\widehat{T}$ (where $S' = leaves(\widehat{T})$). Denote*

$$r_i \;=\; 3(\bar{d}_i + \alpha(\bar{d}_i)) + \alpha\left(3(\bar{d}_i + \alpha(\bar{d}_i))\right) \;. \tag{14}$$

Then the set $S'' = \{s \in S' : \hat{d}(x,s) \le r_i\}$ *preserves the anchor of* x *in* $T(S')$.

Proof. Assume that the anchor of x in $T(S')$ is an edge (u,v) (the other case is proved similarly). Then we need to show that S'' preserves both u and v in $T(S')$. We will show w.l.o.g. that it preserves v. This is done by proving that each v-component of $T(S')$ contains a terminal taxon s s.t. $\hat{d}(x,s) \le r_i$. Notice that each v-component contains a terminal taxon s s.t. $d_T(v,s) \le depth(T(S'))$. By decomposing the path connecting x and s in $T(S' \cup \{x\})$ into 3 parts we get:

$$d_T(x,s) \;\le\; w(x,p_x) + w(p_x,v) + d_T(v,s) \;\le\; 3depth(T(S' \cup \{x\})) \;\le\; 3(\bar{d}_i + \alpha(\bar{d}_i)) \;.$$

The lemma then follows since $\hat{d}(x,s) \le d_T(x,s) + \alpha(d_T(x,s))$. □

The final part left in the analysis is to bound the diameter of $T(S'' \cup \{x\})$. Let s_1, s_2 be two arbitrary taxa in S''. Then $\hat{d}(x,s_1), \hat{d}(x,s_2) \le r_i$, and we get:

$$d_T(s_1,s_2) \le d_T(s_1,x) + d_T(s_2,x) \;\le\; 2(r_i + \alpha(r_i)) \;. \tag{15}$$

Equation (8) in the proof of Lemma 5.4 implies that $\bar{d}_i \le 2depth(T) + \alpha(2depth(T))$ in every iteration of the algorithm. Plugging this into r_i obtains the following bound:

$$\begin{aligned} diam(T(S'' \cup \{x\})) &\le 2(r_i + \alpha(r_i)) \\ &= 6(\bar{d}_i + \alpha(\bar{d}_i)) + 2\alpha^{+2}\left(3(\bar{d}_i + \alpha(\bar{d}_i))\right)\big) \\ &\le 12depth(T) + 6\alpha^{+2}(2depth(T)) \\ &\quad + 2\alpha^{+2}\left(6depth(T) + 3\alpha^{+2}(2depth(T))\right) \;. \end{aligned} \tag{16}$$

We conclude this section by providing a bound on the weight of contracted edges, which depends only on the depth of the tree (and not on the ε-diameter). This establishes the adaptive fast convergence of the algorithm (Theorem 7.11). The major downside of this bound compared to the one of Theorem 5.10 is that the linear dependence of ε on the depth of the tree is inflated – from factor 4 in Theorem 5.10 to factor 12 in Theorem 7.10. Hence, in cases where the ε-diameter is not too large, the previous bound is actually tighter.

Theorem 7.10. *Let* T *be a phylogenetic tree over a set* S *of terminal taxa, and assume that the input dissimilarity matrix* $\widehat{D}$ *is* α*-close to the additive metric* D_T *(for some non-decreasing function* α*). Let* $\varepsilon \ge 0$ *be s.t.*

$$\alpha^{+2}\left(12depth(T) + 4\varepsilon\right) \;\le\; \frac{\varepsilon}{2} \;. \tag{17}$$

Then, when executed on $\widehat{D}$*, algorithm Incremental_Reconstruct which uses Attach_Taxon_revised and* $FPM_{\widehat{D},\alpha}$*, returns an* ε*-contraction of* T.

Proof. The proof is established by showing that the two conditions of Lemma 7.7 are satisfied in every iteration of the incremental algorithm. The first condition (preservation of the anchor) is given by Lemma 7.9. Hence, we are left to show that $FPM_{\widehat{D},\alpha}$ is (r,ε)-reliable for $r = diam(T(S'' \cup \{x\}))$. This is implied, through Lemma 5.9, by showing that $\alpha^{+2}\left(diam(T(S'' \cup \{x\}))\right) \le \frac{\varepsilon}{2}$. Hence, all we are left to show is that

$$\alpha^{+2}\left(12depth(T) + 4\varepsilon\right) \;\le\; \frac{\varepsilon}{2} \quad\Rightarrow\quad diam(T(S'' \cup \{x\}) \le 12depth(T) + 4\varepsilon \;.$$

The assumption $\alpha^{+2}\left(12depth(T) + 4\varepsilon\right) \le \frac{\varepsilon}{2}$ implies that $\alpha^{+2}(2depth(T)) \le \frac{\varepsilon}{2}$, and so

$$\begin{aligned} 6\alpha^{+2}(2depth(T)) &\le 3\varepsilon \;, \\ 2\alpha^{+2}\left(6depth(T) + 3\alpha^{+2}(2depth(T))\right) &\le \varepsilon \;, \end{aligned}$$

which imply (together with (16)) that $diam(T(S'' \cup \{x\})) \le 12depth(T) + 4\varepsilon$. □

Theorem 7.11. *The revised incremental algorithm which uses Attach_Taxon_revised is adaptive fast converging.*

Proof. We use Lemma 6.4 to establish adaptive fast convergence by showing that our algorithm satisfies the condition of this lemma with $c_1 = 16.25$ and $c_2 = 4$. Let T be an arbitrary phylogenetic tree and let $\widehat{D}$ be a dissimilarity matrix which is α-close to D_T. Denote $\varepsilon = 4\alpha(16.25depth(T))$. We will show that the revised algorithm returns an ε-contraction of T, when given $\widehat{D}$ (and α) as input. If $\varepsilon > depth(T)$, then any internal contraction of T is an ε-contraction (since the weights of all edges in a tree are no greater than its depth). So assume $\varepsilon \leq depth(T)$. We will show that in this case, T and ε satisfy the condition of Theorem 7.10 stating that $\alpha^{+2}(12depth(T) + 4\varepsilon) \leq \frac{\varepsilon}{2}$:

$$\begin{aligned}\alpha^{+2}(12depth(T) + 4\varepsilon) &\leq \alpha^{+2}(16depth(T)) \leq 2\alpha\left(16depth(T) + \alpha(16depth(T))\right) \\ &\leq 2\alpha(16.25depth(T)) = \frac{\varepsilon}{2}\,.\end{aligned}$$

□

8 Conclusion and Discussion.

In this paper we introduced the concept of adaptive fast convergence of phylogenetic reconstruction algorithms, which draws a direct connection between the amount of information in the input (i.e., length of input sequences) and the amount of topological resolution provided by the output tree (i.e., the set of edges for which correct reconstruction is guaranteed). Then we presented an adaptive fast converging algorithm which also provides a *zero false positive* rate, meaning that the edges in its output tree do not induce false splits (w.h.p.). Our algorithm is based on a *partial directional oracle*, which is a model-independent primitive for constructing phylogenetic trees in the presence of noisy information, and hence could be useful for other related tasks. We presented an efficient implementation of the partial directional oracle, which led to an optimal $O(n^2)$ time implementation of our algorithm.

One direction in which our result may be improved is by tightening the tradeoff between the length of input sequences and the upper bound on the weight of contracted edges. This can be achieved, for instance, by reducing the size of the constant preceding the term $depth(T)$ in formulas (11) and (17). One question which stems from this is what is the minimal dependence in $depth(T)$ one can obtain through a time-optimal $O(n^2)$ algorithm.

Another direction is to design a "forest reconstruction" version of our adaptive fast converging incremental algorithm. An extension of adaptive fast convergence to forest reconstruction was established in a recent work of Daskalakis et al. [10]. Specifically, they present a polynomial time algorithm which, for user controlled parameters τ and M, returns a set of subtrees whose leaves form a partition of the input-taxon set, s.t. each edge of weight greater than τ and depth smaller than M is included in one of the subtrees (note that edges here correspond to splits of proper subsets of the taxa). The tradeoff between the parameters τ, M is governed by the length of input sequences much the same way that ε depends on $depth(T)$ in (17). The user can thus specify the depth M he wishes to reach in the tree, and this dictates the weight of edges which will be contracted on the way. The user can alternatively restrict the weight of contracted edges τ, and this dictates the depth the algorithm reaches. Such an approach gives potential users flexible control on the set of edges they require to correctly reconstruct. An interesting research direction could be to improve the efficiency of the forest reconstruction algorithm in [10] by incorporating the time optimal incremental technique presented in this paper.

A more general future research direction concerns the inherent information loss in any case where the full tree is not returned, due to edge contractions and/or to returning a forest rather than a tree. When contracting an edge of a partially reconstructed tree (as done by our algorithm), the information encapsulated in the partial split defined by this edge before

it was contracted is lost and does not appear in the final output tree. When returning a forest (of at least two trees), *all* of the edges in the output represent only *partial* splits, even when some (or maybe even most) of these partial splits can be reliably extended to the entire set. Another problem related to this is that standard measures (such as Robinson-Foulds distance [28]) are not sufficient for evaluating the amount of information in an output forest, since a forest edge is contained within a tree which spans only a subset of the taxa being studied, and thus contains less information than an edge of a tree spanning the full taxon set.

One possible way to deal with both types of information loss is to design new data structures that can combine various kinds of splits (partial and full) in a way which is comprehensible to users of common tree reconstruction tools. We note that for any such data structure which represents partial splits it is also important to suggest a fair scale for comparing it with the actual phylogenetic tree of interest.

Acknowledgement:

We would like to thank Elchanan Mossel for his constructive comments on a preliminary version of this work. We would also like to thank the anonymous reviewers for their useful comments, and one of the reviewers for pointing out [34] and a tighter bound for Lemma 6.1.

References

[1] K. Atteson. The performance of neighbor-joining methods of phylogenetic reconstruction. *Algorithmica*, 25:251–278, 1999.

[2] P. Buneman. The recovery of trees from measures of dissimilarity. In F. Hodson, D. Kendall, and P. Tautu, editors, *Mathematics in the Archeological and Historical Sciences*, pages 387–395. Edinburgh University Press, 1971.

[3] J. Cavender. Taxonomy with confidence. *Math Biosci*, 40:271–280, 1978.

[4] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MITP, 2001. 2nd edition.

[5] M. Cryan, L. Goldberg, and P. Goldberg. Evolutionary trees can be learned in polynomial time in the two-state general markov model. *SIAM Journal on Computing*, 31(2):375–397, 2001.

[6] M. Csürös. Fast recovery of evolutionary trees with thousands of nodes. *Journal of Computational Biology*, 9(2):277–297, 2002.

[7] J. Culberson and P. Rudnicki. A fast algorithm for constructing trees from distance matrices. *Inform Process Lett*, 30(4):215–220, 1989.

[8] C. Daskalakis, C. Hill, A. Jaffe, R. Mihaescu, E. Mossel, and S. Rao. Maximal accurate forests from distance matrices. In *RECOMB '06: Proceedings of the tenth annual international conference on Computational molecular biology*, pages 281–295, 2006.

[9] C. Daskalakis, E. Mossel, and S. Roch. Optimal phylogenetic reconstruction. In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 159–168, 2006.

[10] C. Daskalakis, E. Mossel, and S. Roch. Phylogenies without branch bounds: Contracting the short, pruning the deep. In *RECOMB '09: Proceedings of the thirteenth annual international conference on Computational molecular biology*, pages 451–465, 2009.

[11] P. Erdos, M. Steel, L. Szekely, and T. Warnow. A few logs suffice to build (almost) all trees (I). *Random Structures Algorithms*, 14:153–184, 1999.

[12] P. Erdos, M. Steel, L. Szekely, and T. Warnow. A few logs suffice to build (almost) all trees (II). *Theoret Comput Sci*, 221:77–118, 1999.

[13] J. Farris. A probability model for inferring evolutionary trees. *Systematic Zoology*, 22:250–256, 1973.

[14] J. Felsenstein. *Inferring Phylogenies*. Sinauer Associated, Inc., Sunderland, MA, 2004.

[15] I. Gronau, S. Moran, and S. Snir. Fast and Reliable Reconstruction of Phylogenetic Trees with Very Short Edges. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*, 2008.

[16] I. Gronau, S. Moran, and I. Yavneh. Towards optimal distance functions for stochastic substitution models. *J Theor Biol*, 260(2):294–307, 2009.

[17] D. Huson, S. Nettles, and T. Warnow. Disk-Covering, a fast-converging method for phylogenetic tree reconstruction. *J Comp Biol*, 6:369–386, 1999.

[18] T. Jukes and C. Cantor. Evolution of protein molecules. In H. Munro, editor, *Mammalian Protein Metabolism*, pages 21–132. Academic Press, New York, 1969.

[19] M. Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *J Mol Evol*, 16(2):111–120, December 1980.

[20] V. King, L. Zhang, and Y. Zhou. On the complexity of distance-based evolutionary tree reconstruction. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*, pages 444–453, 2003.

[21] M. Lacey and J. Chang. A signal-to-noise analysis of phylogeny estimation by neighbor-joining: Insufficiency of polynomial length sequences. *Math Biosci*, 199(2):188–215, 2006.

[22] R. Mihaescu, C. Hill, and S. Rao. Fast phylogeny reconstruction through learning of ancestral sequences. http://arxiv.org/abs/0812.1587, 2008.

[23] B. Moret, K. St. John, and T. Warnow. Absolute convergence: true trees from short sequences. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*, pages 186–195, 2001.

[24] E. Mossel. Phase transitions in phylogeny. *Trans Amer Math Soc*, 356:2379–2404, 2004.

[25] E. Mossel. distorted metrics on trees and phylogenetic forests. *IEEE/ACM Transactions on computational biology and bioinformatics*, 4:108–116, 2007.

[26] E. Mossel and M. Steel. How much can evolved characters tell us about the tree that generated them? In Olivier Gascuel, editor, *Mathematics of Evolution and Phylogeny*, chapter 14, pages 384–412. Oxford University Press, 2005.

[27] J. Neymann. Molecular studies of evolution: a source of novel statistical problems. In S. Gupta and Y. Jackel, editors, *Statistical Decision Theory and Related Topics*, pages 1–27. Academic Press, New York, 1971.

[28] F. Robinson and R. Foulds. Comparison of phylogenetic trees. *Math Biosci*, 53:131–147, 1981.

[29] S. Roch. Sequence length requirement of distance-based phylogeny reconstruction: Breaking the polynomial barrier. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 729–738, 2008.

[30] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol*, 4:406–425, 1987.

[31] M. Steel and L. Szekely. Inverting random functions II: Explicit bounds for discrete maximum likelihood estimation, with applications. *SIAM J Discrete Math*, 15(4):562–575, 2002.

[32] L. Wasserman. *All of Statistics*. Springer, New York, 2004.

[33] M. Waterman, T. Smith, M. Singh, and W. Beyer. Additive evolutionary trees. *J Theor Biol*, 64(2):199–213, January 1977.

[34] K. Zaretskii. Constructing a tree on the basis of a set of distances between the hanging vertices. *Uspekhi Mat Nauk*, 20(6):90–92, 1965. in Russian.

Appendix

A Distance Estimation under Stochastic Models of Sequence Evolution.

We review here the Cavender-Farris-Neyman (CFN) model of site-substitution [3, 13, 27], which was assumed in our analysis. Our main objective is to figure out the noise function α implied by this model. Since the underlying model is stochastic, α is 'probabilistic' in the sense that it bounds the noise only with sufficiently high probability. The noise is expressed as a function of the distance being estimated and the length k of the evolved sequences – the longer the sequences, the smaller the noise. We note that similar results on the CFN model have been shown in other works discussing fast convergence (e.g., in [11, 8]). However, the analysis presented here is somewhat tighter and provides results with slightly better constants, and it is perhaps also simpler. We also note that more complex models (such as the Jukes-Cantor model [18] and Kimura's 2-parameter model [19]) can be shown to induce very similar noise functions to the one induced by CFN (with different constants) (see, e.g., [12, 29, 16]).

The CFN model is a model for site substitution in binary sequences. It is applied to DNA sequences by considering purines (bases `A,G`) as one character and pyrimidines (bases `C,T`) as another. The model assumes a rooted tree T, whose edges are associated with symmetric *changing probabilities* $\{p_e\}_{e\in E(T)}$. The process of evolution is modeled by uniformly randomizing a binary state (0 or 1) at the root and propagating mutations along the tree edges according to their changing probabilities. A *site* is defined by the n random states generated by the above process at the leaves of the tree. Note that under a given model-tree, the probability distribution of a specific site is well defined. Repeating this process k times (independently), yields n binary sequences of length k (corresponding to k sites), which may serve as input to a phylogenetic reconstruction method.

The *additive metric* D_T associated with the model-tree T is defined by assigning the following weight to each edge e in T:[3] $w(e) = -\frac{1}{2}\ln(1-2p_e)$. For $u, v \in V(T)$, denote by p_{uv} the *compound changing probability* between u and v, which is the probability of observing different states in u and v. The following equation describes the (well-known) relation between distances in D_T and changing probabilities:

[3] The constant $-\frac{1}{2}$ in this formula comes from modeling the number of mutation events along an edge according to a Poisson distribution (see ,e.g., [14] pp. 156-157). Analysis in other papers sometimes uses different constants. Consequently, the constants in the resulting noise bounds are changed as well.

$$d_T(u,v) \;=\; \sum_{e \in path(u,v)} w(e) \;=\; -\frac{1}{2}\ln(1-2p_{uv})\ .$$

Given a pair of sequences (of length k) corresponding to terminal taxa i,j, the *observed* compound changing probability $\hat{p}_{ij}$ is estimated by the normalized hamming distance (i.e., the number of sites in the two sequences with different states divided by k). The *observed pairwise dissimilarities* are defined accordingly – $\hat{d}(i,j) = -\frac{1}{2}\ln(1-2\hat{p}_{ij})$. The main objective of our analysis is to find a "good" noise function α_k which bounds the difference between these dissimilarities and the exact additive distances associated with the model tree.

Lemma A.1. *Let d be the additive distance between two terminal taxa, and let $\hat{d}$ be the observed dissimilarity between these two taxa (computed from k-long sequences). Then for any $\delta > 0$, $\Pr\left(|d-\hat{d}| > \alpha_{k,\delta}(\min\{d,\hat{d}\})\right) < \delta$, where $\alpha_{k,\delta}$ is defined as follows*[4]*:*

$$\alpha_{k,\delta}(d) = -\frac{1}{2}\ln\left[1 - e^{2d}\sqrt{\frac{2}{k}\ln\left(\frac{2}{\delta}\right)}\right]\ . \tag{18}$$

The main tool used in the proof of this lemma is the following basic bound, implied by Hoeffding's inequality.

Lemma A.2 ([32], Theorem 4.5)**.** *Let $X_1,\ldots,X_k$ be independent random variables which get the value 1 with probability p and 0 with probability $1-p$, and let $\hat{X}_k = \frac{\sum_{i=1}^{k} X_i}{k}$. Then for every $\lambda > 0$,*

$$\Pr\left(\hat{X}_k - p > \lambda\right) \;\leq\; \exp(-2k\lambda^2) \tag{19}$$

$$\Pr\left(\hat{X}_k - p < -\lambda\right) \;\leq\; \exp(-2k\lambda^2) \tag{20}$$

It is not hard to see that for every taxon-pair i,j, the observed compound changing probability $\hat{p}_{ij}$ is an average of k random variables satisfying the conditions of Lemma A.2. Hence, the deviation of $\hat{p}_{ij}$ from its expected value p_{ij} can be bounded using this lemma. Lemma A.3 below translates this deviation to the deviation between the additive distances and the observed dissimilarities.

Lemma A.3. *Let d be the additive distance between two terminal taxa, and let $\hat{d}$ be the observed dissimilarity between these two taxa. Then for any $\beta > 0$ we have:*

$$\Pr\left(d-\hat{d} > \beta\right) \;\leq\; \exp\left(-\frac{k}{2}\frac{(e^{2\beta}-1)^2}{e^{4d}}\right) \tag{21}$$

$$\Pr\left(d-\hat{d} < -\beta\right) \;\leq\; \exp\left(-\frac{k}{2}\frac{(1-e^{-2\beta})^2}{e^{4d}}\right)\ . \tag{22}$$

Proof. Let $p,\hat{p}$ be the real and observed compound changing probabilities between the two terminal taxa mentioned in the lemma. First we establish (21):

$$\begin{array}{rcll} d-\hat{d} \;>\; \beta & \iff & \frac{1}{2}\ln\left(\frac{1-2\hat{p}}{1-2p}\right) \;>\; \beta & \iff \\ & & \frac{1-2\hat{p}}{1-2p} \;>\; e^{2\beta} & \iff \\ & & 1+2\frac{p-\hat{p}}{1-2p} \;>\; e^{2\beta} & \iff \\ & & p-\hat{p} \;>\; \frac{1}{2}(e^{2\beta}-1)(1-2p) & \iff \\ & & p-\hat{p} \;>\; \frac{1}{2}(e^{2\beta}-1)e^{-2d}\ . & \end{array} \tag{23}$$

[4] we use the convention that for $z \leq 0$, $\ln(z) = -\infty$.

The inequality in (21) is now obtained by plugging $\lambda = \frac{1}{2}(e^{2\beta}-1)e^{-2d}$ in (20). The inequality in (22) is similarly obtained by first showing (as in (23)) that:

$$d - \hat{d} \;<\; -\beta \iff p - \hat{p} \;<\; \frac{1}{2}(e^{-2\beta}-1)e^{-2d} \;, \tag{24}$$

and then plugging $\lambda = \frac{1}{2}(1-e^{-2\beta})e^{-2d}$ in (19). □

Note that the bounds we get in (21) and in (22) are not identical: the RHS of (22) is greater than the RHS of (21), because $\beta > 0$ implies that $1-e^{-2\beta} < e^{2\beta}(1-e^{-2\beta}) = e^{2\beta}-1$. Hence we get:

$$\Pr\left(|d-\hat{d}| > \beta\right) < 2\exp\left(-\frac{k}{2}\frac{(1-e^{-2\beta})^2}{e^{4d}}\right) \;. \tag{25}$$

The bound in (26) below is obtained by noticing that d and $\hat{d}$ are interchangeable in the proof of Lemma A.3:

$$\Pr\left(|d-\hat{d}| > \beta\right) < 2\exp\left(-\frac{k}{2}\frac{(1-e^{-2\beta})^2}{e^{4\min\{d,\hat{d}\}}}\right) \;. \tag{26}$$

Proof of Lemma A.1. Note that $\alpha_{k,\delta}$ is obtained by setting δ to be equal to the RHS of (26) and solving for β. By (26), this implies that

$$\Pr\left(|d-\hat{d}| > \alpha_{k,\delta}(\min\{d,\hat{d}\})\right) \;<\; \delta \;.$$ □

Lemma A.1 provides the basic result for establishing the required noise function α_k. By applying the union-bound, we get that for every $\delta > 0$, $\widehat{D}$ and D_T are $\alpha_{k,\delta}$-close with probability at least $1 - \binom{n}{2}\delta$. The noise function α_k in (27) is obtained by $\alpha_{k,\delta}$ for $\delta = \frac{2}{n^3}$.

Theorem A.4. *Let D_T be the additive metric induced by a phylogenetic tree T in the CFN model, and let $\widehat{D}$ be an observed pairwise-dissimilarity matrix derived from k-long sequences which evolved along T. Then with probability greater than $1-\frac{1}{n}$, $\widehat{D}$ and D_T are α_k-close, where α_k is given by:*

$$\alpha_k(d) \;=\; -\frac{1}{2}\ln\left[1 - e^{2d}\sqrt{\frac{6\ln(n)}{k}}\right] \;. \tag{27}$$