Phylogenetic Reconstruction with Insertions and Deletions

Alexandr Andoni[*]
Mark Braverman[†]
Avinatan Hassidim[‡]

November 5, 2009

Abstract

We study phylogenetic reconstruction of complete d-ary trees, with three possible mutations: substitutions, insertions and deletions. We give the first efficient algorithm for this problem which uses sequences of poly logarithmic length. The paper introduces two new tools:

1. A new distance measure, and a new reconstruction guarantee which are tailored to deal with insertions and deletions.
2. A robust local alignment and reconstruction algorithm, which can be used recursively.

The error analysis of the algorithm is based on defining a new process on trees, and applying percolation theory to analyze its behavior.

[*]CCI
[†]Microsoft Research
[‡]MIT, Part of the work was performed while visiting Microsoft Research.

1 Introduction

The evolutionary history of a given set of species is usually modeled as a *phylogenetic tree*. The leaves of the tree correspond to the species which exist today. The root of the tree is their closest common ancestor, and each branching indicates a specification event, in which one species is extinct, and several new species are formed. The goal of *phylogenetic reconstruction* is to infer the tree structure from the information we have on the leaves, which is usually in the form of some genetic information. The problem of phylogenetic reconstruction received a lot of attention in the recent literature; see, e.g., [Mos03, Mos04, DMR06, Roc08, BRZ95, EKPS00, Iof96, BKMP05, MSW04, BCMR06] and the excellent surveys [Roc07, SS03].

In the present work, we work with the following common formalization of the phylogenetic tree reconstruction problem. The tree is the complete d-ary tree, where d is a constant. The genetic information is modeled as a bit string (the CFN model of [Cav78, Far73, Ney71]), and we call each location in this bit string a *site*. The genetic information of the root is assumed to be sampled uniformly at random from $\{0,1\}^k$ for some k representing the amounth of the available genetic information. In every branching event, the bit string of the father node v is copied to each one of the daughter nodes $u_1, \ldots, u_d$, with the exception that the copying process undergos a random mutation process. To define the mutation process, we characterize each edge (v, u_j) by three parameters: the substitution probability of the edge $p_s((v,u_j))$, the insertion probability $p_i((v,u_j))$ and the deletion probability $p_d((v,u_j))$. Given these probabilities, when copying the genetic information x_v to the daughter u_j, every site undergoes any of the following mutations, independently of any other site:

1. Substitution: The bit is flipped with probability $p_s((v,u_j))$.
2. Deletion: The bit is deleted with probability $p_d((v,u_j))$
3. Insertion: A new site is added to the right of the current site with probability $p_i((v,u_j))$. The value of the bit in that site is a random bit.

We assume that exist global constant p_{id}, p_s, p_{min}, such that for every edge (v,u_j) we have $p_i((v,u_j)), p_d((v,u_j)) < p_{id}$, and $p_{min} < p_s((v,u_j)) < p_s$. We call p_{id} the *indel* probability, and say that a site has gone an indel if it has gone through an insertion or a deletion.

Denote the depth of the tree by $\log n$, so there are $n^{\log d}$ leaves. We want to design an algorithm for the following reconstruction problem. Consider an instance of the evolutionary process on a tree. Then, given the unsorted list of the leaves and the parameters d, p_s, p_{id}, p_{min}, the algorithm has to find the tree structure, putting every leaf in the correct place. The algorithm is required to succeed for any choice of parameters $p_i((v,u_j)), p_d((v,u_j)), p_s((v,u_j))$ with high probability over the random coin flips of the evolution process (which determine the initial values of the root and govern the mutations). We limit our attention to algorithms which are polynomial in n.

Traditionally, the hardest part of the phylogenetic problem was the presence of indel mutations. Hence, the classical approach to the phylogenetic tree reconstruction problem is to factor away the

indels by performing some multiple sequence alignment, and then running a phylogenetic reconstruction algorithm, assuming that the only mutations are substitutions. Most of the work focused on the second part, assuming that there are no indels. Under this assumption, early results (see, e.g., the work of Erdös et al. [ESSW99a, ESSW99b]) showed how to use distance estimations between all pairs of nodes in order to build the tree. However, as the correlation between the same sites in two nodes which lie in different parts of the tree can be as small as $n^{-O(1)}$, these approaches require a polynomial number of sites[1].

In the last decade, several breakthrough results [Mos03, Mos04, DMR06, Roc08] showed that there exists a critical substitution probability $p_s^\star = \frac{1}{2} - \frac{1}{\sqrt{d}}$, such that if $p_s < p_s^\star$ the correct tree can be reconstructed with high probability even if $k = O(\log n)$. A key ingredient in getting these results was to use the values of the bits (and not just the distances), and to use recursive constructions. These constructions rely heavily on the independence of the values of different sites, and therefore do not work when there are indels.

There has been a lot of empirical work on performing the two processes together [TKF91, TKF92, Met03, MLH04, SR06, RE08, LG08, LRN$^+$09]. One common heuristic is to alternate between building a phylogenetic tree (assuming no indels) based on some initial alignment, and then generating an alignment based on the current tree. However, lately there has been growing concern in the biological literature with the errors introduced by these procedures [LG08, WSH08].

In recent work, Daskalakis and Roch [DR09] proposed a new algorithm, which reconstructs the tree with high probability when there are indels. However, their approach was based on the pairwise distances, and requires a polynomial number of sites.

In present work, we give an algorithm for phylogenetic tree reconstruction problem, which support non-trivial indel mutation probability, and requires only a polylogarithmic number of sites k.

1.1 Our Results

The main result of this paper is the following:

Theorem 1.1 *For every $p_{min} > 0$ there exist a (small) constant ϵ, and (large) constants C_1, C_2, C_3 such that, if $p_{id} < \epsilon/\log^2 n$, the degree $d > C_1$, the substitution probability is at most $p_s \leq \frac{1}{2} - \frac{C_2\sqrt{\log d}}{\sqrt{d}}$ and the number of sites is $k > C_3 \log^2 n$, then, given the sequences of the $n^{\log d}$ leaves of the tree, one can reconstruct the structure of the tree with high probability.*

We note that we did not attempt to optimize C_1, C_2, C_3 and ϵ.

Remark 1.2 *Our result also extends to the binary tree case, i.e., when $d = 2$. In this case, we only handle $p_{min} = p_s = \epsilon$ for some small constant. We defer the details to the full version of the paper.*

[1] A surprising example in which reconstruction is possible (under some conditions) even when k is subpolynomial has been obtained by Roch [Roc08].

We now briefly discuss the parameters, and compare to best possible parameters one would hope to achieve. If k is polylogarithmic in n, we must have $p_{id} = O(\log\log n/\log n)$ even when there are no substitutions: otherwise, some of the leaves will be empty (all sites are deleted), and the correct reconstruction is impossible. Our algorithm works for $p_{id} = O(1/\log^2 n)$. We conjecture that the right bound is $O(1/\log n)$. In the d-ary case, our bound on p_s is close to optimal, as even when there are no indel operations, one cannot have $p_s > 1/2 - 1/\sqrt{d}$. Finally, if the number of sites is larger than $C_3 \log^2 n$, our algorithm only uses the first $C_3 \log^2 n$ sites.

2 Preliminaries

We introduce some standard notation. We use $[d]$ denote the set $\{1, \ldots, d\}$. When we say that an event happens with high probability, we mean with probability at least $1 - 1/n^c$, where we can set up the parameters such that c is as big as we need.

Given a bit string x, we let $x[i:j]$ denote the bits from location i to location j in x. Sometimes we wish to distinguish between a node v and the sequence of bits it has; in this case, we usually denote the sequence by x_v.

For each node v, we define d functions $f_i^v : \{1, \ldots K_v\} \mapsto \{1, \ldots K_{v(i)}\} \cup \{\perp\}$, where K_v is the length of the sequence at node v, $v(i)$ is the i'th daughter, and $K_{v(i)}$ is the length of the i'th daughter. We let $f_i^v(j)$ denote the place in v_i which the j'th site went to, or $\perp$ if the site was deleted. Thus, each f_i^v is strictly monotone. When v is clear from context, we omit it.

During the algorithm we will reconstruct the phylogenetic tree recursively, level by level. It will be comfortable to distinguish between the *ideal tree*, which is the tree generated by the random process, and the *reconstruction tree*, which is the tree reconstructed by the algorithm. Variables which refer to sequences of bits in the reconstructed tree will have a hat. There will also be a clear correspondence between nodes in the ideal tree and the ones in the reconstructed tree. Thus, if x_v is the subsequence of bits in node v in the ideal tree, $\hat{x}_v$ is the sequence of bits in the node which corresponds to v in the reconstructed tree.

We will usually think of each x_v or $\hat{x}_v$ as composed of *blocks*, which are consecutive sequences of length B, where $B = O(\log n)$ is a large constant times $\log n$. The algorithm is only going to use the first $O(\log n)$ blocks. To simplify the notation, the paper is written such that the algorithm uses B blocks, so it is enough to remember that B is a large constant times $logn$ (the number of blocks does not have to be equal to the length of each block in order for the algorithm to succeed). We will also have *bad* blocks (which will also be called *red blocks*), and we will later prove that with high probability every vertex has at most α red blocks, where $\alpha = O(\log n)$, but this time with a small constant. We also prove that with high probability on the path from the leaf to the root there are at most α indel operations. The use of the same bound for both these variables is again done to enable the reader to remember less parameters. Finally, the reconstruction guarantee is going to be that with probability $1 - \beta$ two sites are equal, where we can use $\beta = O(d^{-2/3})$ a small constant.

3 Intuitive Explanation

The reconstruction of the tree is done level by level, beginning with the leaves. At each level, we prove that there exists a correspondence between the set of the vertices that we reconstructed for this level $\hat{v}_1, \ldots, \hat{v}_m$ to the vertices of that level in the ideal tree $v_1, \ldots v_m$. Denoting the bit string which corresponds to $\hat{v}_i$ by $\hat{x}_i$, and the bit string which corresponds to v_i by x_i, this correspondence has the property that the $\hat{x}_i$ resembles x_i in two important manners:

1. The distance between x_i and x_j is similar to the distance between $\hat{x}_i$ and $\hat{x}_j$. The distance we use here is not hamming nor edit, but some mixture of the two.

2. there exists an alignment which matches $\hat{x}_i$ to x_i. Under this alignment, $\hat{x}_i$ and x_i have "small" hamming distance (we actually need to something a bit stronger – see below).

For the leaves this correspondence exists trivially. We now assume that the correspondence exists for level $\ell - 1$, and show how to build a set of vertices which will correspond to the set of vertices of level ℓ. First, the first property of the correspondence enables us to partition the vertices of level $\ell - 1$ into sets of size d, such that each such set constitutes of d siblings. Denoting by $\hat{v}_1, \ldots, \hat{v}_d$ the nodes which correspond to the daughters of a single node v, we want to build a node $\hat{v}$ such that $\hat{x}$ will correspond to x, the bit string of node v. We do this by performing a local alignment between $\hat{x}_1, \ldots, \hat{x}_d$, and then take the majority vote for each site. Finally, we prove that in this case $\hat{x}$ resembles x (satisfying both guarantees).

This high level approach has many complications. The first is that we may have a degradation of the quality of the reconstruction, in which the guarantee we can get for level ℓ is weaker than the one we get for level $\ell - 1$. This problem can also occur in the case where there are no indels. In that case, the reconstruction algorithm is just a place-wise majority, and since the errors are completely random and independent, one can see that (due to simple error correction bounds), there is a limiting error, and the guarantee doesn't degrade.

We cannot hope for such an easy solution. Any local reconstruction procedure might miss some of the edit distance errors (for example it might not find the exact location of a deletion). This will introduce errors into the process which are not randomized, but are generated by our algorithm. Such errors can accumulate, and may also be magnified by the recursive structure of the algorithm.

A second challenge that we face is the large size of the tree, compared to the sequence length. This means that the success probability of our reconstruction procedure must be exponential in the number of sites. A reconstruction procedure (for the case where the tree is given) appeared in [ADR10]. However, that procedure had a very high failure probability (at least $1/\mathrm{poly}\log n$).

To deal with these challenges, we use a very conservative reconstruction guarantee. We allow $\hat{x}$ to correspond to x, if one can align them with a few edit distance operations – i.e. there exists a string y such that y is obtained from the reconstructed sequence $\hat{x}$ by up to $\alpha = O(\log n)$ indel operations, and y is close to x in hamming distance. The similarity between y and x is defined as follows:

1. There are up to $\alpha = O(\log n)$ blocks of length $B = O(\log n)$ in y on which we have no guarantee. We will call these areas the red blocks.

2. In the rest of y, we have that if $z = x \oplus y$, then z is stochastically dominated by a string which has 1 in every place with i.i.d probability β, where β is a small constant[2].

We explain this seemingly weird guarantee. Suppose the first site of v was deleted, when going to all of his children. In this case, even if we have the children perfectly, we cannot expect to reconstruct the first bit of x, and we will have a deletion error. Hence we need to alow some slackness in the alignment. Note that there is a probability of $O(1/\text{poly}(n))$ that the first $\log n/\log\log n$ sites will be deleted in all the children. Therefore we must allow at least $\log n/\log\log n$ indel operations going from $\hat{x}$ to y (we did not try to optimize $\log\log n$ factors).

Since we cannot identify exactly the location of each indel operation, we settle on estimating it up to B locations. But this means that if there were many indels in the same block, we will perform the place-wise majority between sites which came from different locations in the father. In this case, we cannot say much about the result of the majority vote. Hence, we allow for bad blocks, in which there is no guarantee. Again we use α to bound the number of such blocks.

Finally, the reconstruction guarantee gives us that $z = x \oplus y$ is only stochastically dominated by a random string, and not that it is equal to a random string. This happens because sometimes we take a majority vote with a site which is adversarial. In this case, the adversary can choose to tilt the majority vote in any direction she wants. This may not sound like a problem, but it complicates all distance estimations made by the algorithm, since even if two strings x_1, x_2 are completely uncorrelated (they come from different areas in the same node or from different nodes), the adversary can create correlation between $\hat{x}_1$ and $\hat{x}_2$.

Before describing the reconstruction algorithm and its analysis, we note that when β is small enough, and there are not too many bad areas, we can estimate the distance between two nodes u, v by looking at the hamming distance between $\hat{x}_u$ and $\hat{x}_v$, when up to 4α indel operations are allowed for free. This is a good approximate of the distance ??????

, as getting from $\hat{x}_v$ to y_v requires α operations, and y_v is pretty close to x_v - up to αB bad areas (remember that α is a small constant times $\log n$), and a change of the estimate of up to $2\beta B^2$ in any direction due to the stochastic dominance.

The reconstruction procedure itself is simple - we pick a random child s, and divide it into B blocks of length B. To determine the k'th block of the father, we align all the other children according to the k'th block of s and take a majority vote[3]. The challenge is to analyze the mistakes

[2]Let X,Y be two random variables in $\{0,1\}^m$. We say that Y stochastically dominates X, if there is a joint random variable $(\tilde{X}, \tilde{Y})$ such that the marginals satisfy $X \sim \tilde{X}$ and $Y \sim \tilde{Y}$ and moreover $\mathbb{P}[\tilde{X} \leq \tilde{Y}] = 1$.

[3]To be a bit more formal, for the k'th block of s we look for blocks in the vicinity of kB of the other children which are close to the k'th block of s. Since we know that at most α indel operations happened on the path between the root and any vertex 4.1, and since we have the reconstruction guarantee for level $\ell - 1$ we know that it is very improbable that the first block of the first son and the 7th block of the second son originated from the same sites in the father. This is important - we do not want to look at far away places, as we do not want far away areas which are adversarial to interfere with our local procedure on correct parts.

introduced by this procedure. Since the reconstruction errors in the father depend on the location of the reconstruction errors in the daughter nodes, we use a percolation type argument to show that with high probability, no node in the tree will have more than α mistakes, thus meeting the reconstruction guarantee.

The percolation argument is applied to a d-ary tree of depth $\log n$, in which every node has B blocks, which can be either red or green. A red block also has a number on it. The connection between the colors of the blocks and the numbers and the reconstruction guarantee is as follows:

1. Each red block corresponds to an adversarial area, see part 1 in the description of the reconstruction guarantee.

2. The indel operations which are used to get y from $\hat{x}$ can only occur in red blocks. Moreover, if a red block has number n_r, there can be at most n_r indel operations there.

To know how many red blocks each node has, and what are the number written on them, we employ a coloring procedure, which starts from the leaves. The procedure is designed to dominate the error generation process of the reconstruction algorithm, while being simple enough to allow to prove strong concentration results. An informal (and not exact) description of the procedure is

1. For each indel operation which happened in a block - color it red, and increase its number by 1.

2. elongate each consecutive sequence of red blocks, by adding one block to the left, and one block to the right[4].

3. The father picks a random child and copies its redness structure (including the numbers).

4. If at any block, two or more children are red, we color this block red in the father.

We briefly explain why this process dominates the reconstruction algorithm, in the sense that each block which is green according to this process is reconstructed correctly (up to the hamming error β), and why the alignment can be made. The proof is by induction on the levels, and within each level, for every node, by induction on the block. The random child which is picked in Step 3 corresponds to the child which we align according to. If the majority is computed with too much adversarial influence, we cannot be sure that it's correct. To compensate for this, Step 4 colors the father block red if the block is red in at least two children. Finally, suppose that an indel operation occurred in the site j of child t for $t \neq s$, where j lies in the beginning of block k. In this case, the corresponding site in the father, $f_t^{-1}(j)$ may be either in the k'th block or in the $k-1$ block. Worse still, we might have $f_s(f_t^{-1}(j))$ either in the k'th block or in the $k-1$ block. Therefore, we need to color both blocks red in son t. To be on the safe side, we elongate each sequence of red blocks.

[4] Actually for technical reasons we elongate almost every sequence.

The organization of the rest of the paper is as follows. Section 4 describes the coloring process in more detail, and proves that with high probability the number of red blocks, and the sum of numbers on red blocks is bounded. Section 5 introduces the algorithm and the reconstruction guarantee, and Section 6 proves that the algorithm is dominated by the percolation process. Finally, Section 7 shows how to use the reconstruction guarantee to find siblings.

4 Red and Green Trees

In this section we consider the ideal tree, and prove some properties on its structure. These properties will help us design the algorithm later. Let B^2 denote the length of the sequence of the root. Most of the properties we prove are defined on blocks, which are consecutive sequences of sites of length B, which begin at kB and end at $kB + B$ for some value k. The following lemma motivates this

Lemma 4.1 *With high probability, the maximum number of indels between the root any leaf is bounded by $\alpha = O(\log n)$, and $\alpha \ll B$.*

We condition the rest of the analysis on the high probability event of Lemma 4.1, without mentioning it explicitly. Given this event, we partition the sites of each node into B blocks, each of length B, except maybe the last block (which can be a bit longer or shorter). In the rest of the paper, we ignore the length of the last block, implicitly assuming that it is B exactly[5]. Informally, the condition from Lemma 4.1 means that throughout the process indels never create shifts longer than α, which is much less than the length of a single block.

We color all the blocks in the tree in two colors: red and green, and give each red block some integer. Initially red nodes signify a misalignment between a parent node and a child node cause by indels. We then apply the following recursive process, level by level, beginning with the leaves.

Initialization: Assume that there are k an indel operation going from the father to the i'th child, which happen in the j'th block. Color block j red, and give it the number k

After coloring level $\ell - 1$, we color level ℓ by the procedure 1. This procedure temporarily expands the red blocks on level $\ell - 1$. These red blocks are only added to simplify the description (one can think of procedure 1 as first copying the daughter nodes, and then coloring the father while changing the temporary copies of the daughters). Procedure 1 is carefully tailored to dominate misalignments in our reconstruction algorithm. By analyzing it we obtain the bounds that we need for the algorithm. For example, the temporary extension in the recursive coloring (line 9) corresponds to the fact that during the algorithmic reconstruction, errors that appear in two different children may "spill over" one block when we reconstruct the parent.

The following lemma is the heart of our error analysis, and it's proof takes the rest of this section

[5]Handling the length of the last block requires some tedious details, but is not fundamentally different. In fact, in the i'th level of the tree, we could throw away the last block. This would mean that in the top level the root would have $B - \log n$ blocks, which would still not affect the distances by much as B is a large constant times $\log n$.

Algorithm 1: Recursively coloring a father y given the d daughter nodes

1 **Initialization:**
2 **for** *each vertex* t*, and each block* k **do**
3 **if** *there are* $n_{k,t} > 0$ *insertions and deletions going from* t*'s parent to* t *in block* k **then**
4 Color block k in t red, and give it the number $n_{k,t} > 0$;
5 **Recursively coloring a father** y **given the** d **daughter nodes:**
6 Let s be a random child ;
7 **for** *every child* $t \neq s$ **do**
8 **for** *every maximal consecutive sequence of red blocks in the* t*'th child,* $i, i+1, \ldots, i+k$ **do**
9 temporarily color blocks $i-1, i+k+1$ red in the child t, and give them the number 1.
10 **for** $k = 1$ *to* B **do**
11 **if** *the* k*'th block in* s *is red, and has number* i **then**
12 Color the k'th block in the father red, and add number i to it.
13 **else**
14 **if** *exist* $t_1 \neq t_2$ *in which the* k*'th block is red* **then**
15 Color the k'th block in the father red, and give it number 1.

Lemma 4.2 *With high probability, in each node of the tree there are at most* α *blocks which are not green, assuming* $\alpha \geq \frac{1200 \log d}{\log 1/(B \cdot p_{id})}$.

In the remainder of the paper, we will condition the rest of the analysis on the high probability event of Lemma 4.2, without mentioning it explicitly.

Proof of Lemma 4.2
We distinguish between *initial* red blocks that were placed during the initiation phase (lines 2-4) and *acquired* red blocks that were passed from child to parent (lines 6-15).

We first note that the probability that a node contains *any* initial red blocks at all is bounded by ε_1 where $\varepsilon_1 < B \cdot p_{id}$ is small. We say that a node is red if it contains any red blocks (initial or acquired). We first claim that with high probability the largest connected component of red nodes is small.

Claim 4.3 *Except with probability* n^{-3}*, the largest red connected component in the graph has fewer than* $\varepsilon_2 \log n$ *nodes with initial red blocks, where* $\varepsilon_2 < \frac{3 \log d}{\log 1/\varepsilon_1}$ *is small.*

Proof: For a node v in the graph, denote by $P_i(v)$ the probability that the set S_v of initial red descendants of v that are connected to it through a red path contains at least i nodes. We will prove

that

$$P_i(v) < \frac{\varepsilon_3^{i+1}}{i^2},$$

where $\varepsilon_3 = 40\sqrt{\varepsilon_1}$ is a small constant. We will show this by induction.

Base case: For S_v to be non-empty, one of the three cases has to hold: (1) v has an initial red block; (2) v has at least two red children; (3) v has one red child that has been randomly selected. This implies the following inequality:

$$P_1(v) \leq \varepsilon_1 + d^2 P_1(v)^2 + \frac{1}{d} P_1(v).$$

It is not hard to see that for a sufficiently small ε_1, $P_1(v) < 1/(2d)$, and thus we get that $P_1(v) < 3\varepsilon_1 < \varepsilon_3^2$.

Step: We want to show the bound for $P_i(v)$, $v > 1$. As in the base case, there are three possibilities that cover all the cases when $|S_v| \geq i$: (1) v has an initial red block; (2) v has at least two red children; (3) v has one red child that has been randomly selected. Denote the probabilities of the three cases by Q_1, Q_2 and Q_3. Given that there is an initial red block in v, the probability that $|S_v| \geq i$ is bounded by the probability that it is $\geq i-1$ without this information. Thus $Q_1 \leq \varepsilon_1 \cdot P_{i-1}(v)$. We also have $Q_3 \leq \frac{1}{d} P_i(v)$. Thus we have

$$P_i(v) \leq \varepsilon_1 \cdot P_{i-1}(v) + Q_2 + \frac{1}{d} P_i(v) < \frac{1}{3} \frac{\varepsilon_3^{i+1}}{i^2} + Q_2 + \frac{1}{3} P_i(v).$$

To complete the proof, all we need to show is that $Q_2 < \frac{1}{3} \frac{\varepsilon_3^{i+1}}{i^2}$. To estimate Q_2 we cover it using the following events. For each $0 < j < i$ and index $1 \leq k < d$ let Q_{jk} be the event that the children $v_1, \ldots, v_{k-1}$ of v have no initial red nodes in their subtrees, node v_k has $\geq j$ initial red nodes in their subtrees, and children $v_{k+1}, \ldots, v_d$ of v have $\geq i-j$ initial red nodes in their subtrees. These events cover Q_2. Moreover, the probability of Q_{jk} is bounded by $(d+1)P_j(v) \cdot P_{i-j}(v)$: the event that exactly one of the nodes $v_{k+1}, \ldots, v_d$ has initial red descendants is covered by $d \cdot P_j(v) \cdot P_{i-j}(v)$. The event that more than one does has probability bounded by $P_{i-j}(v)$, which we multiply by the probability $P_j(v)$ that v_k has $\geq j$ descendants. Thus, in total, we get

$$Q_2 \leq 2d^2 \sum_{j=1}^{i-1} P_j(v) \cdot P_{i-j}(v) < 2d^2 \varepsilon_3 \cdot \frac{\varepsilon_3^{i+1}}{i^2} \cdot \sum_{j=1}^{i-1} \frac{i^2}{j^2(i-j)^2} <$$

$$4d^2 \varepsilon_3 \cdot \frac{\varepsilon_3^{i+1}}{i^2} \cdot \sum_{j=1}^{\infty} \frac{4}{j^2} = \frac{8\pi^2}{3} d^2 \varepsilon_3 \cdot \frac{\varepsilon_3^{i+1}}{i^2} < \frac{\varepsilon_3^{i+1}}{i^2},$$

as long as $\varepsilon_3 < 3/(d^2 \cdot 8\pi^2)$. The claim follows immediately. ■

From now on we will assume that the conclusion of Claim 4.3 holds. Next we want to prove Lemma 4.2, namely that in each node the sum of all the red blocks is at most $\alpha \log n$. We distinguish

two types of red blocks: *natural* blocks and *extension* blocks. A red block is *natural* if either it is an initial red block, or the block is natural in one of the node's children. In other words, for each natural block in a node v there is a descendent of v connected to it via a red path where this block is an original red block. All other blocks are called *extension* blocks. Extension blocks occur because in the case when a node has two or more red children the process extends the red blocks by 1 before taking intersections.

We will bound the number of each type of blocks separately. As a first step, we present the process of red block creation above in an equivalent way as follows:

1. First of all, for each node in the tree we decide with probability ε_1 whether it contains any original red blocks at all; we also select for each node the random child that it copies;

2. we then sample the original blocks in the flagged "red" nodes conditioned on there being at least one red block;

3. we deterministically propagate the choices throughout the tree according to the rules.

Note that by Claim 4.3 the first step creates red connected components of size $< \varepsilon_2 \log n$. The propagation only happens on these connected components separately. Using this new presentation of the process we first bound the numbers of the natural red blocks in each node.

Claim 4.4 *Except with probability $< n^{-3}$ the maximum number of natural red blocks in each node is bounded by $\varepsilon_4 \log n$, where $\varepsilon_4 = 2\varepsilon_2$.*

Proof: We will prove that this is true for each individual node in the graph except with probability $n^{-3-\log d}$, thus implying the claim by union bound. Let v be a node and S_v be the nodes that contain at least one original red block, are in v's connected components and that are v's descendants. By Claim 4.3 we know that $t = |S_v| < \varepsilon_2 \log n$. Denote the nodes in S_v by $v_1, \ldots, v_t$. Each node contains at least one original red block. Denote the number of red blocks in v_i, counted with multiplicities, by B_i. Then the B_i's are i.i.d. and for $j > 1$

$$P[B_i > j] < \varepsilon_1^{j-1}.$$

since $\varepsilon_4 = 2\varepsilon_2$, and $\varepsilon_2 < \frac{3 \log d}{\log 1/\varepsilon_1}$, denoting $A = \varepsilon_4 \log n$, we have

$$P\left[\sum_{j=1}^{t} B_j > A\right] = \sum_{i=A+1}^{\infty} P\left[\sum_{j=1}^{t} B_j = i\right] < \sum_{i=A+1}^{\infty} \binom{i}{t} \varepsilon_1^{i-t} < \sum_{i=A+1}^{\infty} 2^i \varepsilon_1^{i/2} < (4\varepsilon_1)^{A/2} < n^{-3-\log d}.$$

■

Next, we bound the number of extension red blocks. Note that extension blocks always have multiplicity 1. We again consider the original red blocks in each red connected component. Let S_v be a set of nodes that contain original red blocks and all belong to the same red connected component. We know that $|S_v| < \varepsilon_2 \log n$. We denote by P_k the set of blocks that are covered more than k times by original red blocks in S_v (not counting multiplicities). For example, P_1 is just the set of blocks that appear as original red blocks in at least one of the nodes of S_v. We first argue that

Claim 4.5 *For each k,*

$$\mathbf{P}[P_k > (\varepsilon_4 \log n)/k] < n^{-3-\log d}.$$

Thus, by union bound, we can assume that this even doesn't happen. The claim just follows from counting the total number of original red blocks. The proof of Claim 4.4 implies that the total number of original red blocks cannot exceed $\varepsilon_4 \log n$, and the claim follows. Next we make a simple combinatorial observation.

Claim 4.6 *For each extension block b, there is a block b' that is i positions away from b such that $b' \in P_{2^{i/2-3}}$ (P_k is extended trivially to non-integer values by setting $P_k := P_{\lceil k \rceil}$).*

Proof: An extension block b can be traced to two children that are either in the original block or in an extension block as well. We can continue tracing the extension blocks until we obtain a binary tree with b at the root and an original red block at each leaf. Moreover, if the leaf is j levels from b then the location of its original block is $\leq j$-away from b. Denote the distances of all the leaf blocks from b by $d_1, \ldots, d_t$. We have

$$\sum_{j=1}^{t} 2^{-d_j} \geq 1.$$

Denote by n_k the number of leaf blocks *exactly* k-away from b (so that $\sum n_k = t$. Then we have

$$\sum_{k} n_k 2^{-k} \geq 1.$$

Hence there must exist a k such that $n_k > 2^{k/2}/4$. Otherwise

$$\sum_{k} n_k 2^{-k} < \frac{1}{4} \sum_{k} 2^{-k/2} = \frac{1}{4(1-1/\sqrt{2})} < 1.$$

Thus there is a location b' that is k-blocks away from b and that appears in at least $n_k/2 > 2^{k/2-3}$ original blocks, thus belonging to $P_{2^{k/2-3}}$. ■

Putting Claims 4.5 and 4.6 together we see that:

Algorithm 2: Reconstruction of a single node. Inputs: $\hat{x}_1, \ldots, \hat{x}_d$

Let s denote a random child ;
for *each block* k **do**
  $G_k = \{\hat{x}_s[kB : kB + B]\}$;
  $h_s = 0$;
  for *each* $t \neq s$ **do**
    if *exists a shift* $-4\alpha < h_t < 4\alpha$ *such that*
    $\mathrm{d_{ham}}(\hat{x}_s[kB : kB + B], \hat{x}_t[kB + h_t : kB + B + h_t]) < (2p_s(1 - p_s) + 2\beta + \epsilon)\, B$
    then
      Set $G_k \leftarrow G_k \cup \{\hat{x}_t[kB + h_t : kB + B + h_t]\}$;
  Set $\hat{x}[kB : kB + B] = Majority_{B \in G_k} B$

Claim 4.7 *Except with probability* $< n^{-3}$ *the total number of extension blocks in each connected component does not exceed* $199\varepsilon_4 \log n$.

Proof: Fix a connected component S_v. By Claim 4.6, each extension block is close to a point in one of the P_k's, and thus

$$\#\{\text{extension blocks in } S_v\} \leq \sum_{i=0}^{\infty} (2i+1) \cdot |P_{2^{i/2-3}}| \leq$$

by Claim 4.5

$$\varepsilon_4 \log n \cdot \sum_{i=0}^{\infty} (2i+1) \cdot 2^{3-i/2} < 199\varepsilon_4 \log n.$$

■

Lemma 4.2 is obtained by putting Claims 4.4 and 4.7 together. The former bounds the number of natural red blocks in every node, while the latter bounds the number of extension red blocks.

5 The Algorithm

We let $\mathrm{d_{ham}}(x, y)$ denote the hamming distance of x and y.

We explain the notation in the algorithm. α is the bound from Lemmas 4.1,4.2, which satisfies $\alpha = O(\log n)$, $\alpha \ll B$. We will only care about the result of the majority, if at least $d - 1$ children participated in it, treating it as adversarial otherwise. Note that since the original process is symmetric, the algorithm can easily be derandomized, picking s arbitrarily.

We present the reconstruction guarantee. Let $\hat{x}_1, \ldots, \hat{x}_{\hat{K}}$ be the reconstructed sequence. We decompose it into consecutive blocks (subsequences) of length B, as we did in Section 4. Let $1 \leq$

$R_1, \ldots, R_r \leq B$ denote the positions of the red blocks, where $r \leq \alpha$. Let n_i denote the number given to the i'th block if it's red, ir $n_i = 0$ if i is a green block. Let $g : [\hat{K}] \mapsto [-r, \ldots, r] \cup \{\perp\}$ with the following properties:

1. $g(0) = 0$, by definition 0 is green.
2. g is not defined over red blocks: If $R_i B \leq j < R_i B + B$, then $g(j) = \perp$.
3. g is constant over consecutive green blocks: If $j, j-1$ are both green than $g(j) = g(j-1)$.
4. g can change by at most n_i over the i'th block: If j is green but $j-1$ is red, let $k < j$ be the last green site. Then
$$|g(j) - g(k)| \leq \sum_{i,k<Bi<j} n_i$$
where the sum ranges over all the red blocks between k and j.
5. g is an alignment of $\hat{x}$ and x: Consider the string Y which is aligning $\hat{x}$ according to g, that is $y[j] = \hat{x}[j + g(j)]$, or $y[j] = \perp$ if $g(j) = \perp$. Letting $z[j] = x[j] \oplus y[j]$, or 0 if $y[j] = \perp$, we have that z is stochastically dominated by a string which has 1 in each place with i.i.d probability β.

Essentially, for a site j in a green block, $g(j)$ will give the displacement between this site and the matching site in the original string. g is not defined on sites in red blocks - it gives $\perp$. Note that although g is defined as a function of the site, it is actually equal for all the sites in the same block. The algorithm will not reconstruct g, but it will be used in the analysis.

Lemma 5.1 *Suppose that in children i and $j \neq s$ the k'th block is green. Then with high probability (that is with probability $1 - 2^{-O(B)}$) there is exactly one shift $\hat{h}_j$ for which*

$$\mathrm{d_{ham}}(\hat{x}_i[kB : kB + B], \hat{x}_j[\hat{h}_j + kB : kB + \hat{h}_j + B]) < (2p_s(1 - p_s) + 2\beta + \epsilon)\, B$$

This shift satisfies $-4\alpha < \hat{h}_j < 4\alpha$. Moreover, denoting $h_j = \hat{h}_j - g_j(kB) + g_i(kB)$, for each $1 \leq z \leq B$ we have $f_i^{-1}(kB + z) = f_j^{-1}(kB + z + h_j)$, or the same site in the father is mapped to the same site in the sons, up to the shift h_j.

Proof Sketch: Sketch: The 4α bound comes from a maximal shift of 2α for each child. The 2α shift for each child is the sum of two terms: the maximal number of indel operations, and the bound on the shift $g_i(kB)$.

To show that the distance behaves correctly, note that

$$\hat{x}_i[kB + g_i(kB) : kB + g_i(kB) + B] \oplus x_i[kB : kB + B]$$

stochastically dominates a string which has 1 with i.i.d probability β. Since i, j are siblings, we know that there exists a shift h_j between them, such that

$$\Pr(x_i[kB+t] = x_j[kB+t+h_j]) = p_{s_i}p_{s_j} + (1-p_{s_i})(1-p_{s_j})$$

where $p_{s_i} < p_s$ is the substitution probability going from the father to child i, and $p_{s_j} < p_s$ is the substitution probability going from the father to child j. The shift h_j is just

$$h_j = |\{r < kB : f_i(r) = \perp\}| - |\{r < kB : f_j(r) = \perp\}| - |\{r < kB : f_i^{-1}(r) = \emptyset\}| + |\{r < kB : f_j^{-1}(r) = \emptyset\}|$$

or the difference between insertion and deletion operations between child i and child j up to block k. Note that we rely on the fact that there are no indel operations in blocks $k-1, k+1$ of the j'th child. This is the case, as otherwise block k would have been colored red, since $j \neq s$. Also note that the shift h_j may not be equal to $\hat{h}_j$, as h_j is the optimal shift in the ideal tree, and $\hat{h}_j$ is effected by the functions g_i, g_j.

Consider the reconstructed tree. The expected hamming distance of $\hat{x}_i[kB : kB+B]$ from $\hat{x}_j[kB + g_j(kB) - g_i(kB) + h_j : kB + g_j(kB) - g_i(kB) + h_j + B]$ is at most

$$\left(p_{s_i}(1-p_{s_j}) + (1-p_{s_i})p_{s_j} + 2\beta\right) B$$

, and we find $\hat{h}_j = h_j + g_j(kB) - g_i(kB)$.

Using a Chernoff bound and the definition of stochastic dominance, one can get that with high probability this distance is well concentrated. However, as for any $h \neq h_j$, we have

$$\Pr(x_i[kB+t] = x_j[kB+t+h]) = 1/2$$

And therefore the expected distance between $\hat{x}_i[kB : kB+B]$ and $\hat{x}_j[kB+g_j(kB)-g_i(kB)+h : kB+g_j(kB)-g_i(kB)+h+B]$ is at least $(1/2-2\beta)B-2\alpha$, and again this distance is concentrated with high probability. As α is small relative to B, and β is a small enough constant, we get that with high probability the correct shift $\hat{h}_j$ passes the bound, and every other shift does not pass it. In this case, for each $1 \leq z \leq B$ we have $f_i^{-1}(kB+z) = f_j^{-1}(kB+z+h_j)$, where $h_j = \hat{h}_j - g_j(kB) + g_i(kB)$. ■

We condition on the high probability event of these lemmas for any two comparisons between blocks made in the algorithm. This is a union bound over $\tilde{O}(n)$ comparisons, but the success probability of the lemma can be taken to be $1 - 1/n^2$.

6 The algorithm is dominated by the process

In this section we utilize Lemma 5.1 to argue that assuming the d children meet the reconstruction guarantee with some redness structure, the father also meets it, when the redness structure of the father is determined by the coloring procedure 1. This ties between the algorithm and the red and green tree generated by the insertion and deletions. Let $\vec{R}$ denote all the coin flips made by the algorithm; that is, R is a sequence of length $\sum_{i=0}^{\log n - 1} d^i$ of numbers in $[d]$, which choose which son was chosen in step 1 of Algorithm 1.

Lemma 6.1 *Suppose all the children meet the reconstruction guarantee for some red and green structure. Then the father x meets the guarantee as well, when the blocks of the father are colored red according to the coloring procedure 1, and the random choice of daughter node is made according to $\vec{R}$.*

Proof: Let g_s denote the alignment function between $\hat{x}_s$ and x_s, and let f_s denote the alignment function between the father node v to the daughter s. The set of sites which were deleted when going from v to s is $D_s = \{j : f_s(j) = \perp\}$, and the set of sites which were inserted is $D_s = \{j : f_s^{-1}(j) = \emptyset\}$. We now define

$$g(j) = g_s(j) + |\{i \in D_s : \ i < j\}| - |\{i \in I_s : \ i < j\}|$$

As the sum of the numbers given to red blocks in the father before site j is at least $g_s(j) + D_s + I_s$, the definition satisfies condition 4 in the definition of g. As $g(j) \neq g(j-1)$ only when there is a red block in the father (either because $g_s(j) \neq g_s(j-1)$ or because there was an indel operation), g satisfies condition 3.

We now show that the reconstruction guarantee holds, given the alignment. Let k be a block which is green in s. Let G_k be the set of daughters for which k is green. If $|G_k| < d-1$, we make no claim about the result of the place-wise majority, as the coloring procedure 1 colors the k'th block red in the father. Otherwise applying lemma 5.1 between each one of the children in G_k and s, gives a set of shifts $\hat{H}_k$, such that for every $j \in S_k$ and site z

$$f_s^{-1}(kB + z) = f_j^{-1}(kB + z + \hat{h}_j) + g_s(kB) - g_j(kB)$$

Denote $h_j = \hat{h}_j - g_j(kB)$ and $a = f_s^{-1}(kB + z)$. Assume wlog that $x[a] = 1$. Let $b_j = x_j[kB + z + h_j]$ and $\hat{b}_j = \hat{x}_j[kB + z + \hat{h}_j]$ for $j \in G_k$, and let b be adversarial.

$$\begin{aligned} \Pr\left(\hat{x}[a + g(a)] = 1\right) &= \Pr\left(\sum_{j \in G_k} \hat{b}_j + b > d/2\right) \geq \Pr\left(\sum_{j \in G_k} \hat{b}_j > d/2\right) \quad (1) \\ &\geq \Pr\left(\sum_{j \in G_k} b_j > d/2 + 2\sqrt{d}\right) \Pr\left(|j : \ \hat{b}_j \neq b_j| < \sqrt{d}\right) \quad (2) \end{aligned}$$

However, $\sum_{j \in G} b_j$ is just a sum of indicator variables, with expectancy

$$\mathbb{E} \sum_{j \in G} b_j \geq \sum_{j} \in G(1 - p_{s_j}) \geq \frac{d-1}{2} + (d-1)2\sqrt{\frac{\log d}{d}}$$

where p_{s_j} is the substitution probability going from the father to the j'th child, and p_s is the bound on the substitution probability. Thus, we have $\Pr(\sum_{j \in G} b_j < d/2 + 2\sqrt{d}) < 1/2d^{2/3}$. As for the second term,

$$\Pr(|j : \ \hat{b}_j \neq b_j| > \sqrt{d}) = 2^{-O(d^{1/6})} < 1/2d^{2/3}$$

using $\beta = 1/d^{2/3}$ and d large enough. Putting this together gives that $\Pr(\hat{x}[j] = 1) > 1 - \beta$, and a similar analysis can be made when $x[j] = 0$.

Note that the event $\sum_{j \in G} b_j > d/2 + 2\sqrt{d} \bigcap |j : \hat{b}_j \neq b_j| < \sqrt{d}$ only depends on the i.i.d random variables which correspond to the substitutions, and on the sum of the random variables $a_j = \hat{b}_j \oplus b_j$, which are dominated by i.i.d random variables. Thus, if we let $y[j] = \hat{x}[j + g(j)] \oplus x[j]$, we have that y is stochastically dominated by a string which has 1 in each position with i.i.d probability β, as required. ■

7 Finding Siblings

In this section we finish the induction on levels, by showing that if all the nodes of level $\ell - 1$ match the reconstruction guarantee, then one can partition them to $d^{\log n - \ell}$ sets of size d, such that every set will contain d siblings, or all the daughters of some node. We begin by defining a new distance, which is motivated by our reconstruction guarantee

$$\mathrm{d}_{\mathrm{ed}}(x, y, \gamma) = \min_{\mathrm{ed}_\gamma}(\mathrm{d}_{\mathrm{ham}}(\mathrm{ed}_\gamma(x), y))$$

Where $\mathrm{ed}_\gamma(x)$ is obtained from x by performing up to γ indel operations. Note that d_{ed} is not a metric, since it does not respect the triangle inequality. To use it, we require the following claim

Claim 7.1 *There is an efficient algorithm which computes* $\mathrm{d}_{\mathrm{ed}}(x, y, \gamma)$.

The algorithm is based on dynamic programming.

It is easy to see that the distance is monotone in γ, that is $\mathrm{d}_{\mathrm{ed}}(x, y, \gamma_1) \leq \mathrm{d}_{\mathrm{ed}}(x, y, \gamma_2)$ for $\gamma_1 \leq \gamma_2$. Moreover, the distance respects some form of triangle inequality

Claim 7.2

$$\mathrm{d}_{\mathrm{ed}}(x, y, \gamma_1) + \mathrm{d}_{\mathrm{ed}}(y, z, \gamma_2) \geq \mathrm{d}_{\mathrm{ed}}(x, z, \gamma_1 + \gamma_2)$$

The main tool that we want to use is cherry picking (see e.g. [DMR06]). To use it,we need the following lemma. Let i, j be two nodes which are siblings, and v, w be arbitrary. Then

Lemma 7.3 *With high probability,*

$$\mathrm{d}_{\mathrm{ed}}(\hat{x}_i, \hat{x}_j, 4\alpha) + \mathrm{d}_{\mathrm{ed}}(\hat{x}_v, \hat{x}_w, 4\alpha) < \mathrm{d}_{\mathrm{ed}}(\hat{x}_i, \hat{x}_v, 4\alpha) + \mathrm{d}_{\mathrm{ed}}(\hat{x}_j, \hat{x}_w, 4\alpha)$$

We sketch the proof of this lemma. According to the triangle inequality, for any two vertices r, t

$$\mathrm{d}_{\mathrm{ed}}(\hat{x}_r, \hat{x}_t, 4\alpha) \leq \mathrm{d}_{\mathrm{ed}}(\hat{x}_r, x_r, \alpha) + \mathrm{d}_{\mathrm{ed}}(x_r, x_t, 2\alpha) + \mathrm{d}_{\mathrm{ed}}(x_t, \hat{x}_t, \alpha)$$

and similarly

$$\mathrm{d}_{\mathrm{ed}}(\hat{x}_r, \hat{x}_t, 4\alpha) \geq \mathrm{d}_{\mathrm{ed}}(x_r, x_t, 6\alpha) - \mathrm{d}_{\mathrm{ed}}(\hat{x}_r, x_r, \alpha) - \mathrm{d}_{\mathrm{ed}}(x_t, \hat{x}_t, \alpha)$$

The following claim is based on the reconstruction guarantee of $\hat{x}_v$

Claim 7.4 *With high probability,* $\mathrm{d_{ed}}(\hat{x}_r, x_r, \alpha) < 2\beta B^2 + \alpha B$

Proof: According to the reconstruction guarantee, implementing the alignment defined by the function g_s requires less than α edit operations. Given the alignment defined by g, the hamming distance between the cells in the green blocks of $\hat{x}_r$ and their counterparts in x_r is at most $2\beta B^2$, with exponentially good probability in B^2. Since there are at most α bad blocks, this can increase the distance by at most αB. ■

Let R_{rt} denote the path on the tree from r to t, and let

$$p_{rt} = \frac{1}{2} - \prod_{e \in R_{rt}} \left(\frac{1}{2} - p_e \right)$$

where p_e is the substitution probability of edge e, and we have $p_e \leq p_s$. Then

Claim 7.5 *With probability* $2^{-O(\epsilon B)}$*, we have* $\mathrm{d_{ed}}(x_r, x_t, 2\alpha) < (1+\epsilon) p_{rt} B^2$.

Proof: Let z be the common ancestor of r, t. According to Lemma 4.1, with high probability there were at most α indel operations on the path from z to r, and on the path from z to t. Conditioning on this event, both vertices can be aligned according to z. In this case, what we get is a simple hamming distance, which has exponentially good concentration. ■

We take $\epsilon = \beta$, which adds an error of the magnitude generated by Claim 7.4. We also need a lower bound on the distance

Claim 7.6 *For any constant* $\epsilon > 0$*, with probability* $2^{-O(\epsilon B)}$*, we have* $\mathrm{d_{ed}}(x_r, x_t, 6\alpha) > (1-\epsilon) p_{rt} B^2$.

Proof: Fix an alignment of r, t. The probability that the distance is less than $(1-\epsilon) p_{rt} B^2$ is at most $2^{-O(\epsilon B^2)}$, where the probability is taken over the substitutions, insertions and deletions of the random process which generated the tree. As there are at most

$$\binom{B}{6\alpha} \leq B^{6\alpha} = 2^{6\alpha \log B} < 2^{B \log B}$$

different alignment, it is possible to take a union bound for constant ϵ. ■

Again we take $\epsilon = \beta$.

Finally, Lemma 7.3 is proven by noticing that when $\beta, \alpha/B$ are small enough compared to the minimal substitution probability, we have that with high probability

$$\begin{aligned} \mathrm{d_{ed}}(\hat{x}_i, \hat{x}_j, 4\alpha) + \mathrm{d_{ed}}(\hat{x}_v, \hat{x}_w, 4\alpha) &\leq (1+\beta) p_{ij} B^2 + 4\beta B^2 + 2\alpha B + (1+\beta) p_{vw} B^2 + 4\beta B^2 + 2\alpha B \\ &= (p_{ij} + p_{vw}) B^2 + 10\beta B^2 + 4\alpha B < (p_{iv} + p_{jw}) B^2 - 10\beta B^2 - 4\alpha B \\ &\leq \mathrm{d_{ed}}(\hat{x}_i, \hat{x}_v, 4\alpha) + \mathrm{d_{ed}}(\hat{x}_j, \hat{x}_w, 4\alpha) \end{aligned}$$

Algorithm 3: Partition L, the nodes of level ℓ, into sets of siblings

for *every set* $S \subset L$, *with* $|S| = d$ **do**
  for *every* $i, j \in S$, *and* $v, w \in L \setminus S$ **do**
    if $\mathrm{d}_{\mathrm{ed}}(\hat{x}_i, \hat{x}_j, 4\alpha) + \mathrm{d}_{\mathrm{ed}}(\hat{x}_v, \hat{x}_w, 4\alpha) > \mathrm{d}_{\mathrm{ed}}(\hat{x}_i, \hat{x}_v, 4\alpha) + \mathrm{d}_{\mathrm{ed}}(\hat{x}_j, \hat{x}_w, 4\alpha)$ **then**
      S is not a set of siblings. Continue to the next set ;
    Add S to the partition

where we substituted $\epsilon = \beta$.

Given Lemma 7.3, it is straightforward to see the correctness of Algorithm 3. If a set S contains all the daughters of a single vertex, they will pass all tests. Otherwise, if S contains i, j which are not siblings, and v is a sibling of i, then according to Lemma 7.3, for every w, the test will fail, as $\mathrm{d}_{\mathrm{ed}}(\hat{x}_i, \hat{x}_j, 4\alpha) + \mathrm{d}_{\mathrm{ed}}(\hat{x}_v, \hat{x}_w, 4\alpha) < \mathrm{d}_{\mathrm{ed}}(\hat{x}_i, \hat{x}_v, 4\alpha) + \mathrm{d}_{\mathrm{ed}}(\hat{x}_j, \hat{x}_w, 4\alpha)$.

Note that in the current description we use the lemma $n^{O(d^2)}$ times. One can show that given the high probability event of Lemma 4.1, and the reconstruction guarantee, the failure probability of Lemma 7.3 can be made $2^{-O(\log^2 n)}$, so this is not a problem. Also, it is easy to find more efficient algorithms which find siblings.

References

[ADR10] Alex Andoni, Constantinos Daskalakis, and Avinatan Hassidim Sébastien Roch. Trace reconstruction on a tree. In *ICS*, 2010.

[BCMR06] Christian Borgs, Jennifer T. Chayes, Elchanan Mossel, and Sébastien Roch. The Kesten-Stigum reconstruction bound is tight for roughly symmetric binary channels. In *FOCS*, pages 518–530, 2006.

[BKMP05] N. Berger, C. Kenyon, E. Mossel, and Y. Peres. Glauber dynamics on trees and hyperbolic graphs. *Probab. Theory Rel.*, 131(3):311–340, 2005. Extended abstract by Kenyon, Mossel and Peres appeared in proceedings of 42nd IEEE Symposium on Foundations of Computer Science (FOCS) 2001, 568–578.

[BRZ95] P. M. Bleher, J. Ruiz, and V. A. Zagrebnov. On the purity of the limiting Gibbs state for the Ising model on the Bethe lattice. *J. Statist. Phys.*, 79(1-2):473–482, 1995.

[Cav78] J. A. Cavender. Taxonomy with confidence. *Math. Biosci.*, 40(3-4), 1978.

[DMR06] Constantinos Daskalakis, Elchanan Mossel, and Sébastien Roch. Optimal phylogenetic reconstruction. In *STOC'06: Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 159–168, New York, 2006. ACM.

[DR09] Constantinos Daskalakis and Sébastien Roch. Alignment-free phylogenetic reconstruction. In *Preprint*, 2009.

[EKPS00] W. S. Evans, C. Kenyon, Y. Peres, and L. J. Schulman. Broadcasting on trees and the Ising model. *Ann. Appl. Probab.*, 10(2):410–433, 2000.

[ESSW99a] P. L. Erdös, M. A. Steel, L. A. Székely, and T. A. Warnow. A few logs suffice to build (almost) all trees (part 1). *Random Struct. Algor.*, 14(2):153–184, 1999.

[ESSW99b] P. L. Erdös, M. A. Steel, L. A. Székely, and T. A. Warnow. A few logs suffice to build (almost) all trees (part 2). *Theor. Comput. Sci.*, 221:77–118, 1999.

[Far73] J. S. Farris. A probability model for inferring evolutionary trees. *Syst. Zool.*, 22(4):250–256, 1973.

[Iof96] D. Ioffe. On the extremality of the disordered state for the Ising model on the Bethe lattice. *Lett. Math. Phys.*, 37(2):137–143, 1996.

[LG08] Ari Loytynoja and Nick Goldman. Phylogeny-Aware Gap Placement Prevents Errors in Sequence Alignment and Evolutionary Analysis. *Science*, 320(5883):1632–1635, 2008.

[LRN+09] Kevin Liu, Sindhu Raghavan, Serita Nelesen, C. Randal Linder, and Tandy Warnow. Rapid and Accurate Large-Scale Coestimation of Sequence Alignments and Phylogenetic Trees. *Science*, 324(5934):1561–1564, 2009.

[Met03] Dirk Metzler. Statistical alignment based on fragment insertion and deletion models. *Bioinformatics*, 19(4):490–499, 2003.

[MLH04] I. Miklos, G. A. Lunter, and I. Holmes. A "Long Indel" Model For Evolutionary Sequence Alignment. *Mol Biol Evol*, 21(3):529–540, 2004.

[Mos03] E. Mossel. On the impossibility of reconstructing ancestral data and phylogenies. *J. Comput. Biol.*, 10(5):669–678, 2003.

[Mos04] E. Mossel. Phase transitions in phylogeny. *Trans. Amer. Math. Soc.*, 356(6):2379–2404, 2004.

[MSW04] F. Martinelli, A. Sinclair, and D. Weitz. Glauber dynamics on trees: boundary conditions and mixing time. *Comm. Math. Phys.*, 250(2):301–334, 2004.

[Ney71] J. Neyman. Molecular studies of evolution: a source of novel statistical problems. In S. S. Gupta and J. Yackel, editors, *Statistical desicion theory and related topics*, pages 1–27. Academic Press, New York, 1971.

[RE08] Elena Rivas and Sean R. Eddy. Probabilistic phylogenetic inference with insertions and deletions. *PLoS Comput Biol*, 4(9):e1000172, 09 2008.

[Roc07] S. Roch. *Markov Models on Trees: Reconstruction and Applications*. PhD thesis, UC Berkeley, 2007.

[Roc08] Sébastien Roch. Sequence-length requirement for distance-based phylogeny reconstruction: Breaking the polynomial barrier. In *FOCS*, pages 729–738, 2008.

[SR06] Marc A. Suchard and Benjamin D. Redelings. BAli-Phy: simultaneous Bayesian inference of alignment and phylogeny. *Bioinformatics*, 22(16):2047–2048, 2006.

[SS03] C. Semple and M. Steel. *Phylogenetics*, volume 22 of *Mathematics and its Applications series*. Oxford University Press, 2003.

[TKF91] Jeffrey L. Thorne, Hirohisa Kishino, and Joseph Felsenstein. An evolutionary model for maximum likelihood alignment of dna sequences. *Journal of Molecular Evolution*, 33(2):114–124, 1991.

[TKF92] Jeffrey L. Thorne, Hirohisa Kishino, and Joseph Felsenstein. Inching toward reality: An improved likelihood model of sequence evolution. *Journal of Molecular Evolution*, 34(1):3–16, 1992.

[WSH08] Karen M. Wong, Marc A. Suchard, and John P. Huelsenbeck. Alignment Uncertainty and Genomic Analysis. *Science*, 319(5862):473–476, 2008.