Near-Linear Time Edit Distance for Indel Channels

Aaron Sy
Arun Ganesh

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2019-60
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-60.html

May 17, 2019

Acknowledgement

First, I'd like to thank Arun Ganesh, who contributed the other parts of the write-up, and whose
helpful guidance allowed me to produce the relevant proofs. I'd also like to thank my second reader
Prof. Nir Yosef, whose discussion and comments allowed us to extend the results to more commonlyused models of sequence evolution. Finally I'd to thank my advisor Prof. Satish Rao for his advice
throughout my year as a first-time graduate student.

Near-Linear Time Edit Distance for Indel Channels

by Aaron Sy

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

[illegible]

Professor Satish Rao
Research Advisor

May 17, 2019

(Date)

****

[illegible]

Professor Nir Yosef
Second Reader

May 17, 2019

(Date)

Acknowledgements and Contributions to “Near-Linear Time Edit Distance for Indel Channels”

Aaron Sy

May 2019

Abstract

I contributed to the below write-up (draft) as part of the 5th-year M.S. program offered by the EECS department at UC Berkeley.

Acknowledgements

First, I’d like to thank Arun Ganesh, who contributed the other parts of the write-up, and whose helpful guidance allowed me to produce the relevant proofs. I’d also like to thank my second reader Prof. Nir Yosef, whose discussion and comments allowed us to extend the results to more commonly-used models of sequence evolution. Finally I’d to thank my advisor Prof. Satish Rao for his advice throughout my year as a first-time graduate student.

Summary of Contributions

- I helped to (re)formulate the indel channel’s setup in Section 2, after discussion with Prof. Yosef.
- I generated most of the proofs found in Section 3, with some help from Arun for when I was stuck. In particular, this was a proof that under the substitution-only model, any alignment A with only large differences (of size $O(k \log n)$) from the canonical alignment A^* (here the main diagonal of the DP table) would always be worse.
- I produced most of Section 5, which represents a modest but important extension of the results of Section 3 to that of indels.

Near-Linear Time Edit Distance for Indel Channels

Abstract

We show that the edit distance between pairs of strings generated by a standard model called the indel channel can be computed in $O(n \log n)$ time with high probability. The algorithm is simple and only uses the textbook dynamic programming algorithm as a primitive, first computing an approximate alignment between the two strings, and then running the dynamic programming algorithm restricted to entries close to the approximate alignment. The analysis of our algorithm provides theoretical justification for alignment heuristics used in practice such as BLAST, FASTA, and MAFFT, which also start by computing approximate alignments quickly and then find the best alignment near the approximate alignment. Our main technical contribution is a partitioning of alignments such that the number of the subsets in the partition is not too large and every alignment in one subset is worse than an alignment considered by our algorithm with high probability. We hope similar techniques might be useful in the average-case analysis of other programs commonly solved via dynamic programming.

1 Introduction

Edit distance is an important string similarity measure whose computation has applications in fields such as computational biology. Its simplest variant is the Levensthein distance, which is the minimum number of insertions, deletions, or substitutions required to turn the first string into the second. A textbook dynamic programming algorithm computes the edit distance between two length n strings in $O(n^2)$ time, and the best known worst-case exact algorithm runs in $O(\frac{n^2}{\log^2 n})$ time [MP80]. Assuming the Strong Exponential Time Hypothesis, Backurs and Indyk showed that no $O(n^{2-\epsilon})$ time algorithm exists [BI14] for any $\epsilon > 0$, and Bringmann and Künnemann extended this result to bitstrings, showing that these algorithms are near-optimal [BK15]. algorithm which finds the edit distance in time $O(n + d^2)$.

In many practical applications, a quadratic runtime is prohibitively expensive. For example, using the textbook algorithm to align the full genomes of two species, each of which is several billion sites long, has been estimated to take $\tilde{1}00$ CPU years [Fri08]. When the edit distance is small, one can do better. An immediate result is that if the edit distance between two length n strings is at most d, it can be computed in time $O(nd)$ (by just considering only entries in the dynamic programming table which are distance at most d from the diagonal), and Landau et al. give a more nuanced algorithm which finds the edit distance in time $O(n + d^2)$ [LMS98]. However, in practical applications we may still expect the edit distance to be $\Omega(n)$, so these results do not offer substantial improvements.

Motivated by this and the aforementioned lower bounds, there have been many efforts to design faster algorithms. Many worst-case approximation algorithms exist for the problem (e.g. [BES06, AKO10, AO11, CDG$^+$18]). However, most results give super-constant approximation ratios, and the constant-approximation ratios are perhaps too large for practical applications. For example, a pop culture result in biology suggests that a 3-approximation algorithm[1] for edit distance is not guaranteed to determine which of humans, dogs, and bananas are most closely related.

However, there is good reason to believe that in biological applications, the subquadratic lower bound is not applicable. Roughly speaking, the lower bounds of [BI14, BK15] say that every part of one string must be compared to every part of another string in order to compute the edit distance exactly. In applications, this should rarely be true. e.g. when aligning two genomes, there is good reason to believe that the beginning of the first genome only needs to be compared to the beginning of the second genome. Observations like this motivate the need for *average-case analysis* of edit distance algorithms. There are several results on average-cases analysis of edit distance [AK08, Gaw12, Kus18], but these results often require losing an approximation factor or fairly specific conditions to hold for the input.

1.1 Our Contribution

In this paper, we consider a model for average-case analysis of edit distance called the indel channel which is motivated by biological applications. In this model, we generate a random bitstring of length n as our first string (using bitstrings simplifies the presentation, and the results generalize easily to larger alphabets), and then at each position in the string randomly apply each of the three types of mutations (insertion, deletion, substitution) independently with some probability to get the second string. We let $ID(n)$ denote the distribution of pairs of strings and sets of mutations generated by this model. This model of random string mutation is popular as an extension of the CFN model for biological mutations in computational biology, and problems based on the indel channel have been defined and studied in the areas of sequence alignment [Fri19], phylogenetic reconstruction [DR10, ABH10, ADHR12, GZ18], and trace reconstruction [HMPW08, NP17, HPP18]. We show that for pairs of strings generated by this model, we can compute their exact edit distance in near-linear time with high probability:

Theorem 1. *There exists an algorithm running in time $O(n \log n)$ that computes $ED(s_1, s_2)$ for $(s_1, s_2, \mathcal{E}) \sim ID(n)$ with probability $1 - n^{\Omega(1)}$ (over the realization of $(s_1, s_2, \mathcal{E})$).*

Our Techniques. Our algorithm is simple, using only the dynamic programming algorithm as a primitive. The high-level approach is as follows: While we cannot use the dynamic programming algorithm to compute the edit distance between the two strings and get a near-linear time algorithm, we can use it to compute the edit distance between two substrings of small length $O(\log n)$. Under the indel channel, a substring of the first string s_1 and the corresponding substring of the second string s_2 have low edit distance compared to two random substrings with high probability. So by computing the edit distance between two short substrings, we can determine if they are approximately alignment.

We can now use this as a primitive to find an alignment of the two strings that is an approximation of the "canonical" alignment as defined by the indel channel. If we know bit i of s_1 is aligned with bit j of s_2, then there are only $O(\log n)$ indices in s_2 that bit $i + k \log n$ of s_1 can be aligned with with high probability. Even if our estimate for where i is aligned is $O(\log n)$ bits off, this only increases the number of indices bit $i + k \log n$ can be aligned with by an additive $O(\log n)$. So, once we have computed an approximate alignment for the first i bits of s_1, we can extend the approximate alignment by using a small number of edit distance computations on bitstrings of length $O(\log n)$ to determine approximately where bit $i + k \log n$ of s_1 should be aligned.

[1]The approximation ratio proven by [CDG+18] is in the thousands, though they conjecture their algorithm is actually a $(3 + \epsilon)$-approximation.

Once we have an approximate alignment, our algorithm is straightforward: Use the dynamic programming algorithm, but only compute entries in the dynamic programming table which are close to the approximate alignment. We show that with high probability, the best alignment is close to the canonical alignment suggested by the indel channel, which is close to our approximate alignment, giving the correctness of this algorithm. To show this statement holds, we would like to use the fact that that any alignment which differs from the true alignment is better than the canonical alignment with probability decaying exponentially in the difference between the two alignments. However, there are too many alignments for us to conclude by combining this fact with a union bound. Instead, we construct a partition of the alignments with not too many subsets such that for any subset in this partition, all alignments in that subset are worse than the canonical alignment with high probability. We can then take a union bound over the subsets in this partition to get the desired statement.

We note that techniques similar to finding an approximate alignment and then computing the DP table restricted to entries near this alignment are used in heuristics in practice such as BLAST [AGM+90], FASTA [PL88], and MAFFT [KMKM02]. Our analysis thus can be viewed as theoretical support for these kinds of heuristics.

2 Preliminaries and Definitions

2.1 Problem Setup

We sample a uniformly random bitstring $s_1 \sim \{0,1\}^n$. We pass s_1 through an indel channel to arrive at a new bitstring s_2. When passed through the indel channel, for each bit $b_j = b_1, \ldots b_n$:

- b_j is substituted, i.e. flips, with probability p_s
- b_j is deleted, with probability
  - p_d if the previous bit b_{j-1} was not deleted
  - $q_d > p_d$ if the previous bit b_{j-1} was deleted
- An insertion event occurs with probability p_i, inserting a uniformly random bit string $t \sim \{0,1\}^I$ with length $I \sim \mathrm{Geo}(q_i)$ to the right of b_j. Inserted bits are not further acted upon by the indel channel.

We call each of these edits, and use $\mathcal{E}$ to denote the set of edits occurring in the indel channel. Each mutation happens independently for each bit and across different bits. As mentioned before, this definition of the indel channel parallels other definitions in both the theory and computational biology communities. We require that

$$p_s \leq \rho_s, \qquad p_d \leq \rho_d, \frac{1-\rho_d}{1-q_d} \leq \rho_d' \qquad p_i \leq \rho_i, 1/q_i \leq \rho_i'$$

for some constants $\rho_s, \rho_d, \rho_d', \rho_i, \rho_i'$. Our proofs will specify certain inequalities which must hold for ρ_s, ρ_d, ρ_i, thus specifying a range of values for these upper bounds for which our algorithm is proven to work. We use $ID(n)$ to denote the distribution of tuples $(s_1, s_2, \mathcal{E})$ arrived at by this process for some p, q values - we often make statements about $(s_1, s_2, \mathcal{E}) \sim ID(n)$ which apply for any realization of these values satisfying the constraints given by the ρ values, in which case we will not specify what these values are. Given $(s_1, s_2, \mathcal{E}) \sim ID(n)$ we want to compute the edit distance

$ED(s_1, s_2)$ between s_1 and s_2 as quickly as possible. Our main result is Theorem 1, which shows that we can do this in near-linear time.

For simplicity, we specifically use the Levenshtein distance in our proofs, but they can easily be generalized to other sets of penalties for edits.

We start with some basic facts and definitions that will simplify the presentation later. We do not aim to optimize constants, so we will use the following standard simplified Chernoff bound in our proofs:

Fact 1 (Chernoff bound). *Let $X_1 \dots X_n$ be independent Bernoulli random variables and $X = \sum_{i=1}^n X_i$ and $\mu = \mathbb{E}[X]$. Then for $0 < \epsilon < 1$:*

$$Pr[X \geq (1+\epsilon)\mu] \leq e^{-\frac{\epsilon^2\mu}{3}}, \qquad Pr[X \leq (1-\epsilon)\mu] \leq e^{-\frac{\epsilon^2\mu}{2}}$$

We will also use the following simplified negative binomial tail bound:

Fact 2 (Negative binomial tail bound). *Let $X \sim NBinom(n,p)$, i.e. X is a random variable equal to the number of probability p success events needed before n successes are seen. Then for $k > 1$:*

$$Pr[X \geq kn/p] \leq e^{-\frac{kn(1-1/k)^2}{2}}$$

Proof. This follows from noticing that $Pr[X \geq kn/p] = Pr[Binom(kn/p, p) < n]$ and applying a Chernoff bound with $\epsilon = 1 - 1/k$. □

We'll chain together these facts to get a tail bound for a binomial number of geometric random variables:

Lemma 3. *Consider $X \sim NBinom(m,q)$ where $m = \sum_{i=1}^t m_i$, $m_i \sim Bern(p_i) \forall i$, i.e. X is a random variable obtained by first sampling m, the sum of t independent Bernoullis, and then sampling $X \sim NBinom(m,q)$, where $Bern$ and $NBinom$ denote the standard Bernoulli and negative binomial distributions. Then for $1 < k \leq 4$, $\mu = \sum_{i=1}^t p_i$:*

$$Pr[X \geq k \cdot \frac{\mu}{q}] \leq e^{-\frac{(\sqrt{k}-1)^2\mu}{3}} + e^{-\frac{k\mu(1-1/\sqrt{k})^2}{2}}$$

Proof. We consider two cases for the realization of m and apply tail bounds to each case:

$$Pr[X \geq k \cdot \frac{\mu}{q}] = Pr[X \geq k \cdot \frac{\mu}{q} \wedge m \geq \sqrt{k}\mu] + Pr[X \geq k \cdot \frac{\mu}{q} \wedge m < \sqrt{k}\mu]$$
$$\leq Pr[m \geq \sqrt{k}\mu] + Pr[X \geq k \cdot \frac{\mu}{q} | m \leq \sqrt{k}\mu]$$

A Chernoff bound gives $Pr[m \geq \sqrt{k}\mu] \leq e^{-\frac{(\sqrt{k}-1)^2\mu}{3}}$, the negative binomial tail bound (and noticing that $Pr[X \geq k \cdot \frac{\mu}{q} | m \leq \sqrt{k}\mu]$ is maximized when $m = \sqrt{k}\mu$) gives $Pr[X \geq k \cdot \frac{\mu}{q} | m < \sqrt{k}\mu] \leq e^{-\frac{\sqrt{k}\mu(1-1/\sqrt{k})^2}{2}}$, giving the lemma. □

There are many identical definitions for solutions to the edit distance problem - we will define solutions as paths through the dependency graph as doing so simplifies the presentation of the analysis.

Definition 1. *Consider the dependency graph of the edit distance dynamic programming table. For two strings s_1, s_2 of length n_1, n_2, the dependency graph of s_1, s_2 has vertices (i, j) for $i \in [n_1], j \in [n_2]$ and directed edges from (i, j) to $(i+1, j), (i, j+1)$ and $(i+1, j+1)$ for $i \in [n_1], j \in [n_2]$ if these vertices exist.*

*An **alignment** (of two strings s_1, s_2) is any path $A = \{(i_1 = 1, j_1 = 1), (i_2, j_2) \ldots (i_{L-1}, j_{L-1}), (i_L = n_1, j_L = n_2)\}$ from $(1, 1)$ to (n_1, n_2) in the dependency graph of s_1, s_2. Denote the set of all alignments by $\mathcal{A}$.*

For convenience, we will abuse notation and sometimes use A to also denote the cost of alignment A, e.g. using $A \geq A'$ to denote that the cost of A is at least the cost of A'. (recall that in the dependency graph view, using Levenshtein distance the cost of an alignment is the length of the path, if edges of the form $\{(i, j), (i+1, j+1)\}$ with bits i of s_1 and j of s_2 being the same have length 0, and all other edges have length 1).

Definition 2. *For $(s_1, s_2, \mathcal{E}) \sim ID(n)$, the **canonical alignment** of s_1, s_2, denoted A^*, is informally the alignment corresponding to $\mathcal{E}$. More formally, A^* starts at $(1, 1)$, and for each row i of the dependency graph, if the first vertex in A^* in this row is (i, j), we extend A^* as follows according to $\mathcal{E}$:*

- *If no insertion or deletion occurs on the ith bit, we include the edge $\{(i, j), (i+1, j+1)\}$.*
- *If an insertion of I bits occurs on the ith bit and no deletion occurs, we include the path $\{(i, j), (i, j+1), \ldots (i, j+I)\}$*
- *If a deletion and no insertion occurred on the ith bit, we include the edge $\{(i, j), (i+1, j)\}$.*
- *If an insertion of I bits occurred and a deletion, we include the path $\{(i, j), (i, j+1), \ldots (i, j+I), (i+1, j+I)\}$.*

The definition of (canonical) alignments depends on the pair of strings s_1, s_2, but throughout the paper usually it will be clear that the pair of strings being referred to is sampled from $ID(n)$, so for brevity's sake we may refer to an alignment without referring to strings, letting the strings be implicit.

Note that the canonical alignment is not necessarily the optimal alignment (in fact, even in the substitution-only case, substitutions cause other alignments to be better with reasonably large probability). However, alignments which differ sufficiently from the canonical alignment should not perform better than the canonical alignment with high probability. For alignments which aren't the canonical alignment, we characterize their differences from the canonical alignment in terms of where they break from the canoncial alignment.

Definition 3. *Fix a canonical alignment A^*, and let A be any alignment. A **break** of A (from A^*) is any subpath $(\{(i_1, j_1), (i_2, j_2) \ldots (i_L, j_L)\})$ of A such that (i_1, j_1) and (i_L, j_L) are in A^* but none of (i_2, j_2) to (i_{L-1}, j_{L-1}) are in A^*. The **length** of the break is the value $i_L - i_1$.*

*For $(s_1, s_2, \mathcal{E}) \sim ID(n)$, a break of alignment A from (i_1, j_1) to (i_L, j_L) is **long** if its length is at least $k \log n$ (for a constant k to be specified later) and **short** otherwise. An alignment is **good** if it has no long breaks and **bad** if it has at least one long break.*

Intuitively, short breaks are smaller and might make an alignment better than the canonical alignment, so we can't hope to rule them out in our analysis. On the other hand, long breaks are sufficiently large such that replacing them with the corresponding part of the canonical alignment should be an improvement with high probability. Lastly, we define two functions that take alignments and make them look more like the canonical alignment A^*.

Definition 4 (Short and Long Break Replacement)**.** *We define $\mathcal{SBR} : \mathcal{A} \mapsto \mathcal{A}$ as a function from alignments to alignments, such that for any alignment A, $\mathcal{SBR}(A)$ is the alignment arrived at by applying the following modification to all short breaks in A: For a break from (i_1, j_1) to (i_L, j_L), replace it with the subpath of A^* from (i_1, j_1) to (i_L, j_L). We define $\mathcal{LBR}$ analogously, except $\mathcal{LBR}$ applies the modification to all long breaks instead of short breaks.*

Note that all alignments in the range of $\mathcal{LBR}$ are good by definition. The idea behind these functions and the definitions of good and bad alignments is to use them in the analysis as follows: It is possible to compute the best of the good alignments quickly by only considering a narrow region within the DP table. So it suffices to show any bad alignment is not the best alignment. For a single bad alignment A, it is fairly straightforward to show that A^* is better than A with high probability. However, there are many bad alignments and thus a simple union bound does not suffice to complete the analysis. We instead use $\mathcal{LBR}$ to show that it suffices if all alignments in the range of $\mathcal{SBR}$ are not better than A^* with high probability. There are considerably fewer of these alignments and they are can be partitioned in a way that is easy to analyze, and so simple counting and probability techniques let us show this holds.

3 Substitution-Only Case

As a warmup, let's consider the easier case when only substitutions are present in the indel channel. In this case, A^* is just the diagonal $\{(1,1),(2,2)\dots(n,n)\}$. We show the following theorem:

Theorem 2. *For $(s_1, s_2, \mathcal{E}) \sim ID(n)$ with $p_i, p_d = 0$, there is an $O(n \log n)$ time algorithm for calculating the $ED(s_1, s_2)$ which is correct with probability $1 - n^{-\Omega(1)}$*

The algorithm is simple - compute entries of the canonical DP table indexed by (i, j) where $|i - j| \le k \log n$, ignoring dependencies on entries for which $|i - j| > k \log n$. The value of k used in the algorithm and the definition of long breaks will be specified by the analysis, which will determine a lower bound for k needed to make the failure probability sufficiently small.

We start by showing that "off-diagonal" alignments, i.e. alignments which do not share any edges with A^*, are not better than A^* with high probability. While there are many bad alignments which are not entirely off-diagonal, this will be useful as later we can show that a bad alignment A in the range of $\mathcal{SBR}$ being better than A^* corresponds to an off-diagonal alignment being better than A^* in a subproblem.

Lemma 4. *For $(s_1, s_2, \mathcal{E}) \sim ID(n)$ with probability $1 - e^{-\Omega(n)}$, $A > A^*$ for all alignments A such that A and A^* do not share any edges.*

Proof. The cost of A^* can be upperbounded using a Chernoff bound: The expected number of edits is at most $\rho_s n$, so Fact 1 gives

$$Pr[A^* \le \frac{3}{2}\rho_i n] \le 1 - e^{-\frac{\rho_s n}{12}}$$

Now our goal is to show the existance of an alignment A that does not share edges with A^* has cost lower than cn (where $c = \frac{3}{2}\rho_s$) has low probability.

We achieve this using a union bound over alignments, and then group alignments by their number of deletions (which for the substitution-only case is also the number of insertions). We can eliminate alignments with more than $cn/2$ deletions, as they will of course have cost more than cn.

$$\begin{aligned}
Pr[\exists A, A \leq cn] &\leq \sum_{d=1}^{cn/2} \sum_{A \in \mathcal{A} \text{ with } d \text{ deletions}} Pr[A \leq cn] \\
&\leq \sum_{d=1}^{cn/2} \binom{n+d}{d, d, n-d} Pr[Bin(n-d, \frac{1}{2}) \leq cn - 2d] \\
&\leq \frac{cn}{2} \binom{(1+\frac{c}{2})n}{\frac{c}{2}n, \frac{c}{2}n, (1-\frac{c}{2})n} Pr[Bin((1-\frac{c}{2})n, \frac{1}{2}) \leq cn]
\end{aligned}$$

The second line counts the number of alignments with d deletions, and it expresses the probability of success in terms of the number substitutions, or edges in A of the form $((i,j),(i+1,j+1))$. Because the bits of s_1 are independent, the cost of off-diagonal edges is $Bern(\frac{1}{2})$. In the third line we upper bound the probability for simplicity.

A Chernoff bound gives an upper bound for the substitution term:

$$\begin{aligned}
Pr[Bin((1-\frac{c}{2})n, \frac{1}{2}) \leq cn] &= Pr[Bin((1-\frac{c}{2})n, \frac{1}{2}) \leq (1 - \frac{2-5c}{2-c})\frac{1}{2}(1-\frac{c}{2})n] \\
&\leq \exp\left(-\frac{(2-5c)^2}{8(2-c)}n\right) \qquad (1)
\end{aligned}$$

Next we upper bound the trinomial term using Stirling's approximation:

$$\begin{aligned}
\binom{(1+\frac{c}{2})n}{\frac{c}{2}n, \frac{c}{2}n, (1-\frac{c}{2})n} &\leq \frac{e}{(2\pi)^{3/2}} \frac{((1+\frac{c}{2})n)^{(1+\frac{c}{2})n+\frac{1}{2}}}{(\frac{c}{2}n)^{cn+1}((1-\frac{c}{2})n)^{(1-\frac{c}{2})n+\frac{1}{2}}} \\
&\leq \frac{e}{(2\pi)^{3/2}} \frac{2}{cn} \sqrt{\frac{2+c}{2-c}} \left[\frac{(1+\frac{c}{2})^{(1+\frac{c}{2})}}{(\frac{c}{2})^c (1-\frac{c}{2})^{(1-\frac{c}{2})}}\right]^n \qquad (2)
\end{aligned}$$

Putting everything together, we have the following upper bound

$$Pr[\exists A, A \leq cn] \leq \frac{e}{(2\pi)^{3/2}} \frac{2}{cn} \sqrt{\frac{2+c}{2-c}} \left[\frac{(1+\frac{c}{2})^{(1+\frac{c}{2})}}{(\frac{c}{2})^c (1-\frac{c}{2})^{(1-\frac{c}{2})}}\right]^n \left[\exp\left(-\frac{(2-5c)^2}{8(2-c)}n\right)\right]^n$$

To have a bound exponentially decaying in n, we must determine the appropriate values of c such that

$$\frac{(1+\frac{c}{2})^{(1+\frac{c}{2})}}{(\frac{c}{2})^c (1-\frac{c}{2})^{(1-\frac{c}{2})}} \exp\left(-\frac{(2-5c)^2}{8(2-c)}\right) < 1 \qquad (3)$$

which gives $c < 0.04206104$ as the requirement for exponential decay. This means we must restrict the substitution probability by $\rho_s < \frac{2}{3}c$.

Therefore with an appropriate value of c, with high probability $A^* < cn$ and $A > cn$ for any A disagreeing with A^* everywhere. □

We now make the following observations which will allow us to apply Lemma 4 to make more powerful statements about the set of all alignments:

Fact 5. *Fix any $s_1, s_2, \mathcal{E}$ in $ID(n)$, and let A, A' be any two alignments with the same set of long breaks. Then $\mathcal{LBR}(A) - A = \mathcal{LBR}(A') - A'$.*

This follows because applying $\mathcal{LBR}$ to A, A' results in the same pairs of subpaths being swapped (and thus the same change in cost) as A, A' have the same long breaks.

Corollary 6. *Fix any $(s_1, s_2, \mathcal{E})$ in the support of $ID(n)$. If for all alignments A in the range of $\mathcal{SBR}$, $A \geq A^*$, then any lowest-cost good alignment is also a lowest-cost alignment.*

Proof. It suffices to show that for any alignment A, $\mathcal{LBR}(A)$ (a good alignment) satisfies $\mathcal{LBR}(A) \leq A$. Letting A' be a lowest-cost good alignment, we then get $A \geq \mathcal{LBR}(A) \geq A'$ for all A, i.e. A' is the lowest cost alignment. Note that applying a composition of $\mathcal{LBR}$ and $\mathcal{SBR}$ to any alignment gives A^*, and that applying $\mathcal{SBR}$ to any alignment does not change the set of long breaks. Then:

$$\mathcal{LBR}(A) - A \overset{\text{Fact 5}}{=} \mathcal{LBR}(\mathcal{SBR}(A)) - \mathcal{SBR}(A) = A^* - \mathcal{SBR}(A) \leq 0 \implies \mathcal{LBR}(A) \leq A$$

□

We complete the argument by showing that the assumption of Corollary 6 holds with high probability.

Lemma 7. *For $(s_1, s_2, \mathcal{E}) \sim ID(n)$, with probability $1 - n^{\Omega(1)}$ for all alignments A in the range of $\mathcal{SBR}$, $A \geq A^*$.*

Proof. As in Lemma 4, we apply a union bound over the range of $\mathcal{SBR}$, grouped by the total length of breaks from A^*.

Consider the set of alignments $\mathcal{A}_i$ of total break length $b \in [ik \log n, (i+1)k \log n)$, whose elements have at most i breaks from A^*, each of length at least $k \log n$. Then the set $\{\mathcal{A}_i : 0 \leq i \leq \frac{n}{k \log n}\}$ forms a disjoint cover of the range $\mathcal{SBR}(\mathcal{A})$. Note $\mathcal{A}_0$ is a singleton set containing only A^*.

Let $\mathcal{B}_i$ be the set of all possible breakpoint configurations corresponding to the alignments in $\mathcal{A}_i$. We can view $B \in \mathcal{B}_i$ as a binary assignment of each edge in A^* to either agree or disagree with $A \in \mathcal{A}_i$. For a fixed set of break points $B \in \mathcal{B}_i$, let $\mathcal{A}_B$ be the set of all alignments having the breakpoints corresponding to B (i.e. every alignment in $\mathcal{A}_B$ has the same breaks from A^*). Note that the set $\{\mathcal{A}_B : B \in \mathcal{B}_i\}$ forms a disjoint cover of $\mathcal{A}_i$.

For any fixed set of breaks $B \in \mathcal{B}_i$, let s_1^B, s_2^B denote the restriction of s_1, s_2 to indices contained in the breaks in B, and $(A)_B$ denote the restriction of an alignment A to these indices. s_1^B, s_2^B are distributed according to $ID(ik \log n)$. Furthermore, for $A \in \mathcal{A}_B$, $A < A^*$ if and only if $(A)_B < (A^*)_B$ So by Lemma 4:

$$\Pr[\exists A \in \mathcal{A}_B, A < A^*] = \Pr[\exists A \in \mathcal{A}_B, (A)_B < (A^*)_B] \leq e^{-\Omega(ik \log n)} = n^{-\Omega(ik)}$$

.

This reduces our problem to that of counting the cardinality of $\mathcal{B}_i$:

$$Pr[\exists A \in \mathcal{SBR}(\mathcal{A}), A < A^*] = \sum_{i=1}^{\frac{n}{k \log n}} Pr[\exists A \in \mathcal{A}_i, A < A^*] = \sum_{i=1}^{\frac{n}{k \log n}} \sum_{B \in \mathcal{B}_i} Pr[\exists A \in \mathcal{A}_B, A < A^*]$$
$$\leq \sum_{i=1}^{\frac{n}{k \log n}} \sum_{B \in \mathcal{B}_i} n^{-\Omega(ik)} = \sum_{i=1}^{\frac{n}{k \log n}} |\mathcal{B}_i| \, n^{-\Omega(ik)}$$

Now we must count the cardinality of $\mathcal{B}_i$. This amounts to assigning which bits of A^* participate in a break, and which do not, taking care to have no breaks of size less than $k \log n$.

Because there are $b \in [ik \log n, (i+1)k \log n)$ breaks, we begin by removing $ik \log n$ bits from the sequence, and then inserting i bits as break points, representing the first edges of a break from A^*. Note that if breakpoints are placed adjacently here, they will together become one large break. Then each break point is extended by inserting $k \log n - 1$ bits to the right. Finally, the remaining at most $k \log n$ bits must be used to extend one of the i current breaks, or not used at all.

Altogether

$$|\mathcal{B}_i| = (n - ik \log n + 1)^i (k \log n)^{i+1} \leq n^i (k \log n)^{i+1}$$

Finally, assuming k is a sufficiently large constant:

$$\begin{aligned} Pr[\exists A \in \mathcal{SBR}(\mathcal{A}), A < A^*] &\leq \sum_{i=1}^{\frac{n}{k \log n}} n^i (k \log n)^{i+1} n^{-\Omega(ik)} \\ &\leq \sum_{i=1}^{\frac{n}{k \log n}} (k \log n)^{i+1} n^{-\Omega(ik)} \\ &\leq \frac{n}{k \log n} (k \log n)^2 n^{-\Omega(k)} \\ &\leq (k \log n) n^{-\Omega(k)} \leq n^{-\Omega(1)} \end{aligned}$$

□

Proof of Theorem 2. The algorithm is to use the standard DP algorithm restricted to entries indexed by (i, j) where $|i - j| \leq k \log n$, ignoring dependencies on entries for which $|i - j| > k \log n$.

Theorem 2 follows immediately from Corollary 6, Lemma 7, and the observation that this algorithm finds an alignment with cost at most that of the lowest-cost good alignment by the correctness of the DP algorithm. □

4 Finding an Approximate Alignment

We now consider the case where insertions and deletions are present. While in the substitution case it is obvious that the canonical alignment is the diagonal, in the presence of insertions and deletions there is the additional algorithmic challenge of finding something close to the canonical alignment. We now use our previous definition for alignments to define an alignment function, which will be useful in analyzing the approximate alignment algorithm.

Definition 5. *Given an alignment A of $(s_1, s_2, \mathcal{E}) \sim ID(n)$, let $f_A : [n] \to \mathbb{Z}$ be the function such that for all $i \in [n]$, $(i, f_A(i))$ is the first vertex in A of the form (i, j).*

Using this definition, $f_{A^*}(j)$ gives the location of the jth bit of s_1 in s_2, or if the jth bit is deleted, where the location would be had it not been deleted. To find the edit distance between s_1, s_2, our algorithm will start by computing an approximate alignment function which does not differ much from f_{A^*}. Before describing our algorithm, it will help to prove some properties about edit distances between pairs of strings sampled from $ID(n)$.

4.1 Properties of the Indel Channel

Lemma 8. *For $(s_1, s_2, \mathcal{E}) \sim ID(\ell)$, let s_1' be the substring formed by bits m to $m+\ell-1$ of s_1, and s_2' be the substring formed by bits $f_{A^*}(m)$ to $f_{A^*}(m+\ell)-1$ of s_2. Then:*

$$\Pr_{(s_1,s_2,\mathcal{E})\sim ID(\ell)}[ED(s_1', s_2') \geq \frac{3}{2}(\rho_s + \rho_i\rho_i' + (\rho_d + 1/\ell)(\rho_d'+1))\ell] \leq e^{-\rho_s\ell/12} + 2e^{-\rho_i\ell/60} + 3e^{-\rho_d\ell/60}$$

Proof. The edit distance between s_1 and s_2 is upper bounded by the number of substitutions, deletions, and insertions that occur in the channel on bits m to $m+\ell-1$ of s_1. So it suffices to show this total is at most $\frac{3}{2}(\rho_s + \rho_i + \rho_d)\ell$ with high probability. In turn, it suffices to show the number of substitutions is at most $\frac{3}{2}\rho_s\ell$, the number of insertions is at most $\frac{3}{2}\rho_i\rho_i'\ell$, and the number of deletions is at most $\frac{3}{2}(\rho_d\ell+1)(\rho_d'+1)$ with high probability. We do this using a union bound over the three types of mutations.

The number of substitutions is at most $\rho_s\ell$ in expectation. A Chernoff bound with $\epsilon = 1/2$ gives that the number of substitutions exceeds $\frac{3}{2}\rho_s\ell$ with probability at most $e^{-\rho_s\ell/12}$.

To bound the number of insertions, the probability the number of insertions exceeds $\frac{3}{2}\rho_i\rho_i'\ell$ is maximized when $p_i = \rho_i, 1/q_i = \rho_i'$. The number of insertions is then the random variable $NBinom(Binom(\ell, \rho_i), 1/\rho_i')$ with expectation $\rho_i\rho_i'\ell$, and by Lemma 3 with $k = 3/2$ to show that the probability it exceeds $\frac{3}{2}\rho_i\rho_i'\ell$ is at most $2e^{-\rho_i\ell/60}$ [2].

To bound the number of deletions, we consider the following process for deciding where deletions occur in s_1:

- For each bit of s_1 a "type 1" deletion occurs with probability p_d, except bit 1 of s_1 where the probability is q_d.
- For each bit j where a type 1 deletion occurs, we sample $m \sim Geo(\frac{1-q_d}{1-p_d})$. Let Δ be the number of bits between j and the next bit with a type 1 deletion. A type 2 deletion occurs on the $\min\{m, \Delta\}$ bits following j.

For bit 1, its probability of seeing a deletion in the indel channel is upper bounded by q_d. Otherwise, if no deletion occurs on bit $j-1$, then for bit $j > m$, the only way bit j sees a deletion is if it has a type 1 deletion, which occurs with probability p_d. If a deletion occurs on bit $j-1$ and bit j does not have a type 1 deletion, it sees a type 2 deletion with probability $(1 - \frac{1-q_d}{1-p_d}) = \frac{q_d-p_d}{1-p_d}$ by the properties of the geometric distribution (this is regardless of the type of deletion on bit $j-1$). So its overall probably of seeing a deletion is $p_d + (1-p_d)\frac{q_d-p_d}{1-p_d} = q_d$. So, the number of deletions in this process stochastically dominates the number of deletions on bits m to $m+\ell-1$ of s_1.

Then, the number of deletions is stochastically dominated by the random variable $X+m$ arrived at by sampling $m \sim Binom(\ell-1, p_d) + Bern(q_d), X \sim NBinom(m, \frac{1-q_d}{1-p_d})$, which exceeds $\frac{3}{2}(\rho_d + \rho_d\rho_d')\ell$ with maximum probability when $p_d = \rho_d$, $\frac{1-\rho_d}{1-q_d} = \rho_d'$. The probability m exceeds $\frac{3}{2}(\rho_d\ell)+1$ is at most $e^{-\rho_d\ell/12}$ by a Chernoff bound. The probability X exceeds $\frac{3}{2}(\rho_d\ell+1)\rho_d'$ is at most $2e^{-\rho_d\ell/60}$ by Lemma 3 with $k = 3/2$. So by a union bound the probability the number of deletions exceeds $\frac{3}{2}(\rho_d\ell+1)(\rho_d'+1)$ is at most $3e^{-\rho_d\ell/60}$.

□

Lemma 9. *Let s_1, s_2 be bitstrings of length ℓ_1, ℓ_2, chosen independently and uniformly at random from all bitstrings of length ℓ_1, ℓ_2. Then $Pr[ED(s_1, s_2) \leq D] \leq \frac{(4e\frac{\ell_1}{D}+5e+\frac{4e}{D})^D}{2^{\ell_2}}$*

[2]The constant 1/60 is slightly smaller/worse than the actual constant given by Lemma 3, we use 1/60 as it is cleaner and this exponent will appear frequently in the analysis.

The proof of this lemma is fairly standard, but we include it for completeness.

Proof. We first bound the number of strings within edit distance D of s_1. Fix any set of up to D edits that can be applied to a bitstring initially of length ℓ_1, that does not contain redundant edits (such as substituting the same bit more than once, deleting an inserted bit). This set can be mapped to a set of D tuples as follows:

- For a substitution (or deletion) applied to the bit in the ith position (using the indexing prior to insertions and deletions), it is encoded as the tuple (i, S) (or (i, D) for a deletion). Note that by the assumption that there are no redundant edits, all substitution and deletion edits in the set of edits map to distinct tuples.
- For insertions, we handle indexing differently to still ensure no two insertions are mapped to the same tuple. Suppose the set of D edits inserts the bitstring $b_1b_2 \dots b_k$ to the right of index of i (using the original indexing - we treat bits are being inserted to the left of the entire bitstring as being inserted to the right of bit 0). Let i' be i plus the number of insertions in the set of edits occurring before bit i. Then we map these k insertions to the tuples $(i', I_{b_1}), (i'+1, I_{b_2}) \dots (i'+k-1, I_{b_k})$. This ensures that the insertions in the set of edits also get mapped to different tuples, since the first index will be distinct for all tuples that insertions are mapped to.
- If the number of edits is $D-k$, we include $(1, N), (2, N) \dots (k, N)$ in the final set of tuples so the final set of tuples still has size D.

Every tuple that can be mapped to in this encoding scheme is of the form (i, E) for $0 \le i \le \ell_1 + D, E \in \{S, D, I_0, I_1\}$ or (i, N) for $1 \le i \le D$. So, there are at most $\binom{4\ell_1+5D+4}{D}$ sets of D tuples that any set of up to D edits can be mapped to. Furthermore, note that the mapping is injective, i.e. given a set of D tuples, using the reverse of the above process it can be uniquely mapped to set of edits. So, there are also at most $\binom{4\ell_1+5D+4}{D}$ possible ways to apply at most D edits to a bitstring which is initially length ℓ_1. Stirling's approximation gives that this is at most $(4e\frac{\ell_1}{D} + 5e + \frac{4e}{D})^D$. So there are at most $(4e\frac{\ell_1}{D} + 5e + \frac{4e}{D})^D$ strings s' such that $ED(s_1, s') \le D$. The number of bitstrings of length ℓ_2 is 2^{ℓ_2}. So the probability $ED(s_1, s_2) \le D$ is at most $\frac{(4e\frac{\ell_1}{D}+5e+\frac{4e}{D})^D}{2^{\ell_2}}$. □

Lemma 10. *For constant $k > 0$, $i \le n - k\log n$,*

$$Pr_{(s_1,s_2,\mathcal{E})\sim ID(n)}\left[|f_{A^*}(i + k\log n) - f_{A^*}(i) - k\log n| \le \frac{3}{2}(\rho_i\rho_i' + (\rho_d + \frac{1}{k\log n})(\rho_d' + 1))k\log n\right] \ge$$
$$1 - 2e^{-\rho_i k\log n/60} - 3e^{-\rho_d k\log n/60}$$

.

Proof. $f_{A^*}(i + k\log n) - f_{A^*}(i) - k\log n$ is the difference between the number of insertions and deletions happening in indices i to $i + k\log n - 1$ of s_1. A lazy upper bound for this difference is the sum of the number of insertions and deletions. The same analysis as Lemma 8 gives the Lemma. □

Corollary 11. *Consider the following random process, which we denote $\mathcal{P}$: we choose i_1 such that $i_1 < n - k\log n$, sample $(s_1, s_2, \mathcal{E}) \sim ID(n)$, and then choose an arbitrary i_2 such that $|i_2 - f_{A^*}(i_1)| \le \log n$ and i_2 is at least $k\log n$ less than the length of s_2. Let s_1' denote the string consisting of bits*

i_1 to $i_1 + k \log n - 1$ of s_1 and s'_2 the string consisting of bits i_2 to $i_2 + k \log n - 1$ of s_2. Then for any i_2 we choose satisfying the above conditions

$$\Pr_{\mathcal{P}}[ED(s'_1, s'_2) \leq (1 + \frac{3}{2}k(\rho_s + 2\rho_i\rho'_i + 2(\rho_d + \frac{1}{k \log n})(\rho'_d + 1)) \log n] \geq$$

$$1 - 2e^{-\rho_i k \log n/12} - 4e^{-\rho_i k \log n/60} - 6e^{-\rho_d k \log n/60}.$$

Proof. By Lemma 10 and the assumptions in the corollary statement, with probability at least $1-2e^{-\rho_i k \log n/60} - 3e^{-\rho_d k \log n/60}$, the edit distance between s'_2 and bits $f_{A^*}(i_1)$ to $f_{A^*}(i_1 + k \log n) - 1$ of s_2 (call this substring s^*_2) is at most $(1 + \frac{3}{2}(\rho_i\rho'_i + (\rho_d + \frac{1}{k \log n})(\rho'_d + 1))k) \log n$ (the upper bound on the difference between starting indices plus the high-probability upper bound on the difference between ending indices). s^*_2 is the result of passing s'_1 through the indel channel, so by Lemma 8 with probability at least $1 - e^{-\rho_s k \log n/12} - 2e^{-\rho_i k \log n/60} - 3e^{-\rho_d k \log n/60}$, the edit distance between s^*_2 and s'_1 is at most $\frac{3}{2}(\rho_s + \rho_i\rho'_i + (\rho_d + 1/\ell)(\rho'_d + 1))k \log n$, giving the lemma by triangle inequality. □

Corollary 12. *Consider the following random process, which we denote $\mathcal{P}$: we choose i_1 such that $i_1 < n - k \log n$, sample $(s_1, s_2, \mathcal{E}) \sim ID(n)$, and then choose an arbitrary i_2 such that*

$$|i_2 - f_{A^*}(i_1)| > \max\{1, \frac{3}{2}(\rho_i\rho'_i + (\rho_d + \frac{1}{k \log n})(\rho'_d + 1)) + 1\} \cdot k \log n$$

and i_2 is at least $k \log n$ less than the length of s_2 respectively. Let s'_1 denote the string consisting of bits i_1 to $i_1 + k \log n - 1$ of s_1 and s'_2 the string consisting of bits i_2 to $i_2 + k \log n - 1$ of s_2. Then for $0 < r < 1$,

$$\Pr_{\mathcal{P}}[ED(s'_1, s'_2) > kr \log n] \geq 1 - \left[\frac{(\frac{4e}{r} + 5e + \frac{4e}{kr \log n})^r}{2}\right]^{k \log n} - 2e^{-\rho_i k \log n/60} - 3e^{-\rho_d k \log n/60}$$

Proof. Either $i_2 < f_{A^*}(i_1) - k \log n$ or $i_2 > f_{A^*}(i_1 + k \log n)$. If $i_2 < f_{A^*}(i_1) - k \log n$, then none of the bits in s'_2 are inherited from bits in s'_1. If $i_2 > f_{A^*}(i_1) + [\frac{3}{2}(\rho_i\rho'_i + (\rho_d + 1/\ell)(\rho'_d + 1)) + 1]k \log n$, then by Lemma 10 we have with probability $1 - -2e^{-\rho_i k \log n/60} - 3e^{-\rho_d k \log n/60}$:

$$i_2 - f_{A^*}(i_1 + k \log n) = [i_2 - f_{A^*}(i_1) - k \log n] + [f_{A^*}(i_1) + k \log n - f_{A^*}(i_1 + k \log n)] \geq$$

$$\frac{3}{2}(\rho_i\rho'_i + (\rho_d + 1/\ell)(\rho'_d + 1))k \log n - \frac{3}{2}(\rho_i\rho'_i + (\rho_d + \frac{1}{k \log n})(\rho'_d + 1))k \log n = 0$$

Then since $i_2 > f_{A^*}(i_1 + k \log n)$, none of the bits are in s'_2 are inherited from bits in s'_1. In either case, s'_1, s'_2 are independent and uniformly random bitstrings, and we can apply Lemma 9 with $\ell_1, \ell_2 = k \log n$ and $D = kr \log n$ to get the lemma by a union bound. □

Note that if we choose any r which is less than a certain constant (which is approximately .1569), for all sufficiently large n, $\frac{(\frac{4e}{r} + 5e + \frac{4e}{kr \log n})^r}{2} < 1$ and thus the failure probability in Corollary 12 becomes $n^{-\Omega(k)}$. Furthermore, if $(1 + \frac{3}{2}k(\rho_s + 2\rho_i\rho'_i + 2(\rho_d + \frac{1}{k \log n})(\rho'_d + 1)) < kr$, then for sufficiently large n the lower bound on edit distance given by Corollary 12 exceeds the upper bound given by Corollary 11. So for the rest of this section, we will fix $\rho_s, \rho_i, \rho'_i, \rho_d, \rho'_d, r$ to be positive values satisfying these conditions. Once these values are fixed we can make the failure probabilities in both corollaries n^{-c} with any exponent c of our choice ($c = 2$ will suffice to achieve a final failure probability of $O(1/n)$) by choosing a sufficiently large k depending only on c. So we also fix k to be said sufficiently large value.

4.2 Algorithm for Quickly Finding an Approximate Alignment

We now describe the algorithm ApproxAlign which finds the approximate alignment f'. Informally, ApproxAlign runs as follows: It starts by initializing $f'(1) = 1$, which is of course exactly correct. By Lemma 10, we know that $f_{A^*}(k \log n + 1)$ will be within $O(\log n)$ of $1 + k \log n$. So, to decide what $f'(k \log n + 1)$ will be, we compute the edit distance between bits $k \log n + 1$ to $2k \log n$ of s_1 and bits j to $j + k \log n - 1$ of s_2 for various values of j close to $1 + k \log n$. By Corollary 11 we know that when j is near $f_{A^*}(k \log n + 1)$, the edit distance will be small, and by Corollary 12 we know that when j is far from $f_{A^*}(k \log n + 1)$ the edit distance will be large. So whichever value of j causes the edit distance to be minimized is not too far from the true value of $f_{A^*}(k \log n + 1)$. Once we've decided on the value $f'(k \log n + 1)$, we proceed analogously to choose a value for $f'(2k \log n + 1)$, using $f'(k \log n + 1)$ to decide what range of values try.

Algorithm 1 Algorithm for Approximate Alignment

```
1:  function ApproxAlign($s_1$, $s_2$)
2:      $f'(1) \leftarrow 1$
3:      $J \leftarrow 2\lceil \max\{1, \frac{3}{2}(\rho_i \rho_i' + (\rho_d + \frac{1}{k \log n})(\rho_d' + 1)) + 1\} \cdot k\rceil$
4:      for $i = 1, 2, \ldots \lfloor \frac{n}{k \log n} \rfloor - 1$ do
5:          $minED \leftarrow \infty$
6:          for $j = -J, -J+1, \ldots J$ do
7:              $s_1' \leftarrow$ bits $ik \log n + 1$ to $(i+1)k \log n$ of $s_1$
8:              $s_2' \leftarrow$ bits $f'((i-1)k \log n + 1) + (j+k) \log n$ to
                    $f'((i-1)k \log n + 1) + (j+2k) \log n - 1$ of $s_2$
9:              if $ED(s_1', s_2') \leq minED$ then
10:                 $minED \leftarrow ED(s_1', s_2')$
11:                 $f'(ik \log n + 1) \leftarrow f'((i-1)k \log n) + (j+k) \log n$
12:             end if
13:         end for
14:     end for
15:     return $f'$
16: end function
```

We now formally prove our guarantee for ApproxAlign (including the runtime guarantee).

Lemma 13. *For $(s_1, s_2, \mathcal{E}) \sim ID(n)$, ApproxAlign$(s_1, s_2)$ computes in time $O(n \log n)$ a function f' such that with probability at least $1 - n^{-\Omega(1)}$, for all i where $f'(i)$ is defined $|f'(i) - f_{A^*}(i)| \leq \lceil \max\{1, \frac{3}{2}(\rho_i \rho_i' + (\rho_d + \frac{1}{k \log n})(\rho_d' + 1)) + 1\} \cdot k \log n\rceil$.*

Proof. We proceed by induction. Clearly, $|f'(1) - f_{A^*}(1)| = |1 - 1| \leq \lceil \max\{1, \frac{3}{2}(\rho_i \rho_i' + (\rho_d + \frac{1}{k \log n})(\rho_d'+1))+1\} \cdot k \log n\rceil$. Suppose $|f'((i-1)k \log n+1) - f_{A^*}((i-1)k \log n+1)| \leq \lceil \max\{1, \frac{3}{2}(\rho_i \rho_i' + (\rho_d + \frac{1}{k \log n})(\rho_d' + 1)) + 1\} \cdot k \log n\rceil$. By Lemma 10 and our choices of constants, with probability $1 - n^{-\Omega(1)}$, $|f_{A^*}((i-1)k \log n+1) + k \log n - f_{A^*}(ik \log n+1)| \leq \frac{3}{2}(\rho_i \rho_i' + (\rho_d + \frac{1}{k \log n})(\rho_d'+1))k \log n$. This gives:

$$|[f'((i-1)k \log n + 1) + k \log n] - f_{A^*}(ik \log n + 1)| \leq$$

$$|[f'((i-1)k \log n + 1) + k \log n] - [f_{A^*}((i-1)k \log n + 1) + k \log n]|+$$

$$|[f_{A^*}((i-1)k \log n + 1) + k \log n] - f_{A^*}(ik \log n + 1)| =$$

$|f'((i-1)k\log n+1) - f_{A^*}((i-1)k\log n+1)| + |f_{A^*}((i-1)k\log n+1) + k\log n - f_{A^*}(ik\log n+1)| \leq$
$\lceil \max\{1, \frac{3}{2}(\rho_i\rho_i' + (\rho_d + \frac{1}{k\log n})(\rho_d'+1)) + 1\} \cdot k\log n\rceil + \frac{3}{2}(\rho_i\rho_i' + (\rho_d + \frac{1}{k\log n})(\rho_d'+1))k\log n \leq J$

So for some j in the range iterated over by the algorithm, $|f'((i-1)k\log n+1) + (j+k)\log n - f_{A^*}(ik\log n+1)| \leq \log n$ and thus the minimum edit distance $minED$ found by the algorithm in iterating over the j values is at most $(1 + \frac{3}{2}k(\rho_s + 2\rho_i\rho_i' + 2(\rho_d + \frac{1}{k\log n})(\rho_d'+1))\log n < kr\log n$ by Corollary 11 with probability at least $1-n^{-\Omega(1)}$. By Corollary 12, with probability at least $1-n^{-\Omega(1)}$ the final value of $f'(ik\log n+1)$ can't differ from $f_{A^*}(ik\log n+1)$ by more than $\lceil \max\{1, \frac{3}{2}(\rho_i\rho_i' + (\rho_d + \frac{1}{k\log n})(\rho_d'+1)) + 1\} \cdot k\log n\rceil$ as desired - otherwise, by the corollary with high probability $minED$ would be larger than $kr\log n$.

Thus by induction, $|f'(i) - f_{A^*}(i)| \leq \lceil \max\{1, \frac{3}{2}(\rho_i\rho_i' + (\rho_d + \frac{1}{k\log n})(\rho_d'+1)) + 1\} \cdot k\log n\rceil$ for all i if the high probability events of Lemma 10, Corollary 11, and Corollary 12 occur in all inductive steps. Across all inductive steps we require $O(n)$ such events to occur, and each occurs with probability $1-n^{-\Omega(1)}$ where the negative exponent can be made arbitrarily large, so by a union bound we can conclude that with probability $1-n^{-\Omega(1)}$, $|f'(i) - f_{A^*}(i)| \leq 2k\log n$ for all i.

For runtime, note that the for loops iterate over $O(\frac{n}{\log n})$ values of i and $O(1)$ values of j. For each i, j pair, we perform an edit distance computation between two strings of length $O(\log n)$ which can be in done in $O(\log^2 n)$ time using the canonical dynamic programming algorithm. So the overall runtime is $O(n\log n)$. □

Remark 14. There is an exact algorithm for edit distance running in time $O(n^{2-o(1)})$ [] which offers a slight asymptotic improvement in the runtime. Furthermore, if the probabilities p_s, p_d, p_i are small enough such that the resulting bounds in Lemmas 8 and 9 differ by at least a factor of α, one could instead use a subquadratic time α-approximation algorithm for edit distance to achieve a runtime speedup of ApproxAlign, while maintaining that the proof of Lemma 13 holds (e.g. the algorithm of [CDG+18] could reduce the runtime to $O(n\log^{5/7} n)$). However, if p_s, p_d, p_i are still constants, going from the approximate alignment to the edit distance value will also take $O(n\log n)$ time, so the overall asymptotic runtime of our algorithm is not sped up by either improvement.

Remark 15. Note that the proof of Lemma 13 only requires s_1 to be random so that non-overlapping strings have sufficient large edit distance with high probability. That is, for certain constants c_1, c_2, c_3, all that is needed is the following condition (which a random s_1 satisfies with high probability): For any two substrings of s_1 of length $c_1\log n$ which are disjoint and whose starting indices differ by at most $c_2\log n$, their edit distance is at least $c_3\log n$. So, in a model where instead an adversary chooses any s_1 satisfying these conditions, one can still prove ApproxAlign finds an approximate alignment with the guarantees of Lemma 13 with high probability.

5 Error Analysis with Indels

In this section, we extend the results from Section 3 to the case where indels are present.

Lemma 16. *For $(s_1, s_2, \mathcal{E}) \sim ID(n)$ with probability $1-e^{-\Omega(n)}$, $A > A^*$ for all alignments A such that A and A^* do not share any edges.*

Proof. We proceed similarly to Lemma 4, but for the case with indels.

Let s_1, s_2 be from the indel channel, and A^* the corresponding canoncial alignment. Lemma 8 has that

$$\Pr_{(s_1,s_2,\mathcal{E})\sim ID(n)}[A^* \geq \left(\frac{3}{2}\rho_s + \kappa\right)n] \leq e^{-\rho_s n/12} + 2e^{-\rho_i n/60} + 3e^{-\rho_d n/60}$$

where $\kappa = \frac{3}{2}(\rho_i\rho_i' + (\rho_d + 1/n)(\rho_d' + 1))$ is an upper bound on the proportion of indels. Our goal now is to show any alignment A that shares no edges with A^* has $A > cn$ with high probability, where $c \geq \frac{3}{2}\rho_s + \kappa$.

Let $n_1 = n$ and n_2 be the lengths of s_1 and s_2. Let $r = |n_1 - n_2|$. Lemma 10 says $r < \kappa n$ with high probability, so we need only consider that case. AWLOG that the r excess indels are insertions. The counting argument is similar if they were deletions. Then as before we sum over the number of deletions, d, which corresponds to $d + r$ insertions and $n - d$ substitutions.

$$\begin{aligned}
Pr[\exists A, A \leq cn] &\leq \sum_{d=0}^{cn/2} \sum_{A \in \mathcal{A} \text{ with } d \text{ deletions}} Pr[A \leq cn] \\
&\leq \sum_{d=0}^{cn/2} \binom{n+d+r}{d, d+r, n-d} Pr[Bin(n-d, \frac{1}{2}) \leq cn - 2d - r] \\
&\leq \frac{cn}{2} \binom{(1+\frac{c}{2})n + r}{\frac{c}{2}n, \frac{c}{2}n + r, (1-\frac{c}{2})n} Pr[Bin((1-\frac{c}{2})n, \frac{1}{2}) \leq cn]
\end{aligned}$$

Since $r < \kappa n$ with high probability, then $\binom{(1+\frac{c}{2})n+r}{\frac{c}{2}n, \frac{c}{2}n+r, (1-\frac{c}{2})n} \leq \binom{(1+\frac{c}{2}+\kappa)n}{\frac{c}{2}n, (\frac{c}{2}+\kappa)n, (1-\frac{c}{2})n}$ with high probability. Note also that $\kappa \leq \frac{3}{2}\rho_s + \kappa < c$. Hence, similar to Equation (2) from Lemma 4, Stirling's approximation gives an upperbound on the trinomial

$$\binom{(1+\frac{c}{2}+\kappa)n}{\frac{c}{2}n, (\frac{c}{2}+\kappa)n, (1-\frac{c}{2})n} \leq \frac{e}{(2\pi)^{3/2}} \frac{2}{cn} \sqrt{\frac{2+3c}{2-c}} \left[\frac{(1+\frac{3}{2}c)^{(1+\frac{3}{2}c)}}{(\frac{c}{2})^c (1-\frac{c}{2})^{(1-\frac{c}{2})}} \right]^n$$

Combine this with Equation (1) from Lemma 4 for the substitution term, and use the analogue of Equation (3) to bound the constant c. This gives the bound $c < 0.03485$ to ensure the probability decays exponentially in n. Hence requiring that $\frac{3}{2}\rho_s + \kappa < c$ ensures that $A > A^*$ with high probability. □

Lemma 17. *For $(s_1, s_2, \mathcal{E}) \sim ID(n)$, with probability $1 - n^{\Omega(1)}$ for all alignments A in the range of $\mathcal{SBR}$, $A \geq A^*$.*

Proof. The proof proceeds as in Lemma 7. Let $\mathcal{A}_i$, $\mathcal{B}_i$ be defined as in Lemma 7, with for each set $B \in \mathcal{B}_i$ of break points, the corresponding set of alignments sharing those break points $\mathcal{A}_B$. With a similar argument we have as before that

$$Pr[\exists A \in \mathcal{SBR}(\mathcal{A}), A < A^*] = \sum_{i=1}^{\frac{n}{k \log n}} |\mathcal{B}_i| \, n^{-\Omega(ik)}$$

and so we must count the cardinality of $\mathcal{B}_i$. The analysis here is similar to the substitution-only case, but with additional edges representing indels. Thus, from the n bits we remove $ik \log n$ bits for which A and A^* disagree. Then for each inserted bit, we insert $k \log n - 1$ bits to the right. To account for indels, note that the i original inserted bits may appear in sections of A^* with multiple insertions (i.e. columns in the DP table where A^* has insertions). Thus for each of the i bits, we must also choose which inserted bit of s_2 from which the break starts or ends. Because there are at most κn insertions, this gives us

$$\begin{aligned}
|\mathcal{B}_i| &\leq (n - ik \log n + 1)^i (\kappa n)^i (k \log n)^{i+1} \\
&\leq n^{2i} (k \log n)^{i+1}
\end{aligned}$$

As before, with sufficiently large k,

$$\begin{aligned} Pr[\exists A \in \mathcal{SBR}(\mathcal{A}), A < A^*] &\leq \sum_{i=1}^{\frac{n}{k \log n}} n^{2i}(k \log n)^{i+1} n^{-\Omega(ik)} \\ &\leq \frac{n}{k \log n}(k \log n)^2 n^{-\Omega(k)} \\ &\leq n^{-\Omega(1)} \end{aligned}$$

□

Proof of Theorem 1. We estimate f_{A^*} using ApproxAlign to obtain f', with an error of $\Omega(k_1 \log n)$ for some k_1. Then, we use the standard DP algorithm restricted to indices (i, j) within $k_2 \log n$ from $(i, f'(i))$ for large enough k_2 (e.g. $k_2 \geq 2k_1$). This covers A^* and the range $\mathcal{LBR}(A)$ for some constant k, and by Lemma 17, optimality of the DP algorithm gives the result.

□

References

[ABH10] Alexandr Andoni, Mark Braverman, and Avinatan Hassidim. Phylogenetic reconstruction with insertions and deletions. *Preprint*, 2010.

[ADHR12] Alexandr Andoni, Constantinos Daskalakis, Avinatan Hassidim, and Sebastien Roch. Global alignment of molecular sequences via ancestral state reconstruction. *Stochastic Processes and their Applications*, 122(12):3852–3874, 2012.

[AGM+90] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215(3):403–410, Oct 1990.

[AK08] Alexandr Andoni and Robert Krauthgamer. The smoothed complexity of edit distance. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming*, pages 357–369, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[AKO10] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Polylogarithmic approximation for edit distance and the asymmetric query complexity. *CoRR*, abs/1005.4033, 2010.

[AO11] Alexandr Andoni and Krzysztof Onak. Approximating edit distance in near-linear time. *CoRR*, abs/1109.5635, 2011.

[BES06] Tugkan Batu, Funda Ergün, and Süleyman Cenk Sahinalp. Oblivious string embeddings and edit distance approximations. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 792–801, 2006.

[BI14] Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). *CoRR*, abs/1412.0348, 2014.

[BK15] Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. *CoRR*, abs/1502.01063, 2015.

[CDG$^+$18] Diptarka Chakraborty, Debarati Das, Elazar Goldenberg, Michal Koucký, and Michael E. Saks. Approximating edit distance within constant factor in truly sub-quadratic time. *CoRR*, abs/1810.03664, 2018.

[DR10] Constantinos Daskalakis and Sebastien Roch. Alignment-free phylogenetic reconstruction. In *Annual International Conference on Research in Computational Molecular Biology*, pages 123–137. Springer, 2010.

[Fri08] Martin C. Frith. Large-scale sequence comparison: Spaced seeds and suffix arrays. 20, 2008.

[Fri19] Martin C. Frith. How sequence alignment scores correspond to probability models. *bioRxiv*, 2019.

[Gaw12] Paweł Gawrychowski. Faster algorithm for computing the edit distance between slp-compressed strings. In Liliana Calderón-Benavides, Cristina González-Caro, Edgar Chávez, and Nivio Ziviani, editors, *String Processing and Information Retrieval*, pages 229–236, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[GZ18] Arun Ganesh and Qiuyi Zhang. Optimal sequence length requirements for phylogenetic tree reconstruction with indels. *CoRR*, abs/1811.01121, 2018.

[HMPW08] Thomas Holenstein, Michael Mitzenmacher, Rina Panigrahy, and Udi Wieder. Trace reconstruction with constant deletion probability and related results. pages 389–398, 01 2008.

[HPP18] Nina Holden, Robin Pemantle, and Yuval Peres. Subpolynomial trace reconstruction for random strings and arbitrary deletion probability. In *COLT*, 2018.

[KMKM02] Kazutaka Katoh, Kazuharu Misawa, Keiichi Kuma, and Takashi Miyata. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Research*, 30(14):3059–3066, 07 2002.

[Kus18] William Kuszmaul. Efficiently approximating edit distance between pseudorandom strings. 11 2018.

[LMS98] G. Landau, E. Myers, and J. Schmidt. Incremental string comparison. *SIAM Journal on Computing*, 27(2):557–582, 1998.

[MP80] William J. Masek and Michael S. Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System Sciences*, 20:18–31, 02 1980.

[NP17] Fedor Nazarov and Yuval Peres. Trace reconstruction with exp(o(n1/3)) samples. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 1042–1046, New York, NY, USA, 2017. ACM.

[PL88] W R Pearson and D J Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences*, 85(8):2444–2448, 1988.