# Reconstructing a DNA Sequence from Erroneous Copies (Preliminary Report)

John Kececioglu[*]
University of Georgia

Ming Li[†]
University of Waterloo

John Tromp[‡]
University of Waterloo

**Abstract**

We show how to efficiently reconstruct an original DNA sequence of length $n$ with high probability from $\log^\delta n$ erroneous copies (for some $\delta < 2$), assuming the sequence itself is random and errors are random with constant error rate $1/C$.

Key words: DNA, sequencing, alignment, errors, Kolmogorov complexity.

## 1 Introduction

DNA sequencing is a key step and a major bottleneck in the Human Genome Project. It is relatively slow and expensive (~\$1 per base with current techniques). Since the human genome comprises no less than 3 billion bases, the development of faster and cheaper sequencing methods is crucial to the project.

Certain technologies promise the ability to obtain long DNA sequences fast but with lots of errors. In *single-molecule DNA sequencing*, the DNA strand is passed by a cutter, that cuts off a single base at a time, which then flows down a microscopic tube at high speed past an optical device. This excites the individual molecule and reads off which type of base it is. The order in

[*]Supported in part by DIMACS. Address: Department of Computer Science, University of Georgia, Athens, GA 30602. E-mail: kece@cs.uga.edu

[†]Supported in part by the NSERC Operating Grant OGP0046506, ITRC, a CGAT grant and DIMACS. Address: Department of Computer Science, University of Waterloo, Waterloo, Ont. N2L 3G1, Canada. E-mail: mli@math.uwaterloo.ca

[‡]Supported by an NSERC International Fellowship. Address: Department of Computer Science, University of Waterloo, Waterloo, Ont. N2L 3G1, Canada. E-mail: tromp@daisy.uwaterloo.ca

time of the bases as they flow down the tube past the reader is hopefully the order of bases along the DNA strand. The process doesn't always flow smoothly, sometimes sputtering (especially at the beginning and the end), hence the presence of long deletions.

In this paper, we study the problem of how to reconstruct an original DNA sequence from a small number of erroneous copies. We deal with three types of commonly considered errors: insertion, deletion, and substitution.

We will make two reasonable assumptions:

1. the DNA sequence is a random sequence (in the sense of being algorithmically incompressible), and
2. all error types/distributions are equally likely.

Assumption 2 is natural. Although assumption 1 is questionable, with non-random sequences such as $A^n$, it is unlikely that one can restore an original sequence with high accuracy. Although DNA sequences are not likely to be Kolmogorov random, we expect them behave enough like random sequences to preserve the validity of our analysis.

We show how to efficiently reconstruct an original DNA sequence of length $n$ from $\log^{1+\delta} n$ erroneous copies, assuming the sequence itself is random and errors are random with constant error rate $1/C$.

There are several difficulties. First of all, given $k$ sequences, doing multiple sequence alignment on those sequences by any known algorithm that guarantees optimality takes $\Omega(n^k)$ time. Second, even if we can do multiple alignment efficiently, is this guaranteed to yield the real sequence? And if it does, how many sequences do we need so that we can converge to the real sequence? Blackwell [1] has shown that if one takes the maximum likelihood point, then, in the limit, we can converge to the real sequence. With high probability, after sampling exponential number of erroneous copies, we converge to the original sequence. This approach is clearly not practical because $n$ is usually very large, from hundreds to thousands. With our approach, we will need only a polylogarithmic number of copies. And our algorithm converges in polynomial time, rather than in $n^{\log^{1+\delta} n}$ time if we straightforwardly use the standard multiple alignment algorithms ??.

## 2 Preliminaries

We use Kolmogorov complexity as a tool here to prove the convergence of our algorithm. To keep the paper self-contained, we briefly review the definition and some properties of Kolmogorov complexity, tailored to our needs. For a complete treatment of this subject, see [2] or the survey [3].

Fix a universal Turing machine $U$ with input alphabet $\Sigma$. The machine $U$ takes two inputs $p$ and $y$. $U$ interprets $p$ as a program and simulates $p$ on input $y$. The Kolmogorov complexity of a string $x \in \Sigma^*$, given $y \in \Sigma^*$, is defined as

$$K_U(x|y) = \min\{|p| : U(p,y) = x\}.$$

One can prove an invariance theorem which says that the Kolmogorov complexity defined with respect to any two universal machines differs by only an additive constant. We will therefore drop the subscript $U$. In fact, for the purpose of our analysis, one fixed universal Turing machine is always assumed. Thus $K(x|y)$ is the *minimum* number of *digits* (*i.e.*, letters in $\Sigma$) in a description from which $x$ can be effectively reconstructed, given $y$. Let $K(x) = K(x|\lambda)$, where $\lambda$ denotes the null string.

By simple counting, for each $n$, constant $c < n$, and $y$, there are at least $|\Sigma|^n - \frac{|\Sigma|^{n-c}-1}{|\Sigma|-1}$ distinct $x$'s of length $n$ with the property

$$K(x|y) \geq n - c. \tag{1}$$

We call a string $x$ of length $n$ *random* if

$$K(x) \geq n - \log n \tag{2}$$

where the logarithm is of base 2. Sometimes, we need to encode $x$ in a *self-delimiting* form $\bar{x}$, in order to be able to decompose $\bar{x}y$ into $x$ and $y$. One possible coding for $\bar{x}$ may be $L(x)10x$ where $L(x)$ is the binary representation of $|x|$ with each bit doubled. (Assume that $0, 1 \in \Sigma$.) For example, if $x = 0000000$, $|x| = 111$ in binary, and $L(x) = 111111$. This costs us about $2\log|x|$ extra bits. Thus, the self-delimiting representation $\bar{x}$ of $x$ requires at most $|x| + 2\log|x| + 2$ bits.

A DNA sequence is over a four letter alphabet $\Sigma = \{A, C, G, T\}$. But we can encode a letter in $\Sigma$ by two bits. Thus when we say a sequence over $\Sigma$ is Kolmogorov random, we mean the binary string (of course, of even length, but this does not matter much) encoding it is Kolmogorov random. Assume that our original DNA sequence $S$ is of length $2n$, *i.e.*, $n$ bases, $2n$ bits. We will assume that this string $S = s_1 \ldots s_{2n}$ contains $n$ bases, the $i$th base being encoded by the two bits $s_{2i-1}s_{2i}$. These two bits are treated as one unit. When we consider errors, we either delete a base (thus 2 bits), or replace a base (2 bits by some other 2 bits), or inserting a base (also 2 bits) in between bases (i.e., after even positions in $S$). These details are not essential, we could have defined

Kolmogorov complexity with alphabet size 4, but we would like to use the standard definition of Kolmogorov complexity.

Let the distance between two sequences $\alpha$ and $\beta$, $d(\alpha, \beta)$, be the minimum alignment score between $\alpha$ and $\beta$, where each insertion, deletion, and substitution is charged unit cost. The distance from one sequence $\alpha$ to a set of sequences $A = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$, is $\sum_{i=1}^{m} d(\alpha, \alpha_i)$. A *Steiner sequence* (or point) of $A$ is a sequence $\alpha$ that minimizes the distance from $\alpha$ to the sequences in $A$, *i.e.* $\sum_{i=1}^{m} d(\alpha, \alpha_i)$. Note, often such a Steiner sequence is not unique.

The expression $\log^k n$ means $\log n$, base 2, to the power of $k$.

# 3 The Reconstruction Algorithm and Some Facts

In order to to achieve polynomial time complexity, we cannot align all the erroneous sequences at the same time. The idea is to separate them into groups. Then there is a question of how many sequences should be in a group. When the group size is two, aligning two sequences does not reduce the errors. The reason is that if we take only two sequences and if one sequence has an error at some position and the other sequence is correct at that position, then even we have aligned their neighbors correctly, we still do not know which way the error has occurred at this position, hence we cannot reduce the amount of errors in the resulting sequence. On the other hand, with three sequences, we expect that most of the times the majority is correct because of the independence of errors. Thus three is the minimum group size. Our algorithm is very simple. It uses group size three and runs in $O(n^3 \log n)$ steps, where $n$ is the length of the original DNA sequence.

The Algorithm:

**Input:** $\log^{1+\delta} n$ erroneous copies. Assume $\log^{1+\delta} n$ is a power of 3.

**Repeat until one sequence is left:** Group the sequences into groups of three, and replace each group by its Steiner sequence.

**Theorem 1** *With high probability, the above algorithm produces the original sequence in $O(n^3 \log n)$ time using $\log^{1+\delta} n$ erroneous copies with uniform error rate $1/C$ for some constant $C$.*

We will prove this theorem inductively. But before we start the proof, we need a series of facts. We first state and prove these facts.

**Lemma 2** *The Steiner sequence of three sequences $\alpha, \alpha, \beta$ is $\alpha$.*

**Proof.** Denote the Steiner sequence by $s$, and consider the distance $d$ from $s$ to $\alpha, \alpha, \beta$. By choosing $s = \alpha$ we find $d \leq d(\alpha, \beta)$. By the triangle inequality, we have $d = 2d(s, \alpha) + d(s, \beta) \geq d(s, \alpha) + d(\alpha, \beta)$. Combined with the previous inequality, this shows $d(s, \alpha) = 0$. □

**Lemma 3** *Given a group $A$ of some sequences, suppose $\alpha$ has distance $d$ to $A$ and $\alpha'$ has distance $d'$ to $A$, then $d(\alpha, \alpha') \leq \frac{d+d'}{|A|}$.*

**Proof.** Choose the sequence $\beta_0$ from $A$ such that

$$d^* = d(\beta_0, \alpha) + d(\beta_0, \alpha')$$

is minimized. Obviously $d(\alpha, \alpha') \leq d^*$. We also have

$$d + d' = \sum_{\beta \in A} d(\beta, \alpha) + \sum_{\beta \in A} d(\beta, \alpha') \geq |A| d^* \geq |A| d(\alpha, \alpha').$$

Thus $d(\alpha, \alpha') \leq \frac{d+d'}{|A|}$, the lemma is proved. □

For some function $f(n) < n$ we often need to estimate $\log \binom{n}{f(n)}$.

**Lemma 4** *For $k < n$,*

$$\log \binom{n}{k} = k(\log n - \log k + O(1)).$$

**Proof.** Since $\binom{n}{k} = \prod_{i=0}^{k-1} \frac{n-i}{k-i}$, we have

$$\left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \frac{n^k}{k!},$$

the first inequality is true because $\frac{n}{k} \leq \frac{n-i}{k-i}$ for $< k \leq n$. Taking logarithm on two sides, the left part becomes $k(\log n - \log k)$. The right part becomes, using Stirling's approximation,

$$k \log n - \log k! = k \log n - k \log k - O(k).$$

The lemma follows. □

## 4 Analysis: Aligning Three Sequences

Let $S$ be the original DNA sequence to be reconstructed.

Choose a constant C whose value will be fixed later. Let there be $n/C$ errors which are uniformly distributed. Each error could be one of the following types:

- Deleting a character
- Inserting one of the four letters in $\Sigma$
- Replacing a character by an other letter in $\Sigma$

There are in total 8 types of errors, we can use three bits to specify them.[1] In total, there are $e(n) = \binom{n}{n/C} \times 8^{n/C}$ possibilities. Choose $E = \log^{1+\delta} n$ random error sequences $e_1, \ldots, e_E$ such that

$$K(e_1 \ldots e_E | S) \geq E \log e(n)$$

and

$$K(S | e_i, i = 1, \ldots, E) \geq |S| - \log n. \tag{3}$$

The first inequality on the randomness of the concatenation of all error codes, implies the following lower bound on the randomness of each individual error code:

$$K(e_i | S, e_1 \ldots e_{i-1}, e_{i+1} \ldots e_E) \geq \log e(n) - \log \log e(n). \tag{4}$$

We will assume a standard encoding. The first $\frac{n}{C} \cdot 3$ bits encode the $n/C$ error types for each $e_i$, and the rest of $e_i$ specifies the error positions. Consider any three copies of $S$, denoted $S_1, S_2, S_3$, with errors indicated by say $e_1, e_2, e_3$. Intuitively, at some position, if at most 1 copy errs, then it is possible to correct this error by majority. If two copies err at the same position, then it often impossible to remove the error. But since the errors are random, we expect only approximately $n/C^2$ such error coincidences. Then after 1 round, we expect only this many errors left. Of course, since we have to do alignment of the three sequences, the naive analysis such as above does not work.

Our strategy is to show that if errors or the alignment do not approximately follow the above pattern then either we can compress $S$ or we can compress $e_i$'s.

---

[1] But notice that this implies that insertion is 4 times as likely as deletion. It is simple to remedy this problem by using a little longer encoding and let the number of codes representing deletion be same as the number of codes representing all four types of insertions.

We now start to prove our main theorem, Theorem 1, inductively. We first show that given three erroneous copies of $S$, $S_1, S_2, S_3$, with random errors encoded by say $e_1, e_2, e_3$, our algorithm reduces the error rate from $1/C$ to $1/C^{2-\epsilon}$ after one round.

Consider a pair of errors, either may be indicated by any of $e_1, e_2, e_3$. If the distance between them on $S$ is $d$, we say they are $d$ apart. Note, it is possible that one error appears in $S_i$ and the other on $S_j$ and $i \neq j$.

**Lemma 5** *For some $\epsilon < 1$, there are less than $h = n/C^{2-\epsilon}$ errors that are within distance $d = \log^2 C$ from some other error.*

**Proof.** Assume, to the contrary, that there are at least $h$ errors that are within distance $d$ from some other error. Then we show how to compress $e_1, e_2, e_3$.

Divide the $h$ error positions $p_1, \ldots, p_h$ into at most $h/2$ groups where consecutive errors in a group are within distance $d$. By Lemma 4, we can encode the positions of the group leaders using $\log(n/h) + O(1)) = (2-\epsilon)\log C + O(1)$ bits each. The other (non-leader) positions can be encoded by $\log d = 2\log\log C$ bits each, giving the distance to the previous error and 1 bit indicating whether this previous error is a group leader. We also use $h \log 3$ bits to specify whether each error occurs in $e_1$, $e_2$, or $e_3$.

The above description uses at most (in the worst case each group consists of just 2 errors)

$$\frac{h}{2}((2-\epsilon)\log C + \log\log C + O(1)) \tag{5}$$

bits to specify the exact distribution of these $h$ errors. We want to show that this is less than what we can save from $e_1, e_2, e_3$ by excluding distribution information about $h$ errors. Let $k = n/C$. There are $\binom{n}{k}^3$ possible error distributions to start with and $\binom{3n}{3k-h}$ remaining possibilities given the distribution of any $h$ errors. Our savings is the logarithm of the ratio between these two.

$$\begin{aligned}
\frac{\binom{n}{k}^3}{\binom{3n}{3k-h}} &= (\frac{n\ldots(n-k+1)}{k!})^3 \frac{(3k-h)!}{(3n)\ldots(3n-3k+h+1)} \\
&\geq \frac{((n-k+h/3)\ldots(n-k+1))^3(3k-h)!}{k!^3 3^{3k-h}} \\
&\geq \frac{((n-k+h/3)\ldots(n-k+1))^3}{k\ldots(k-h/3+1)} \frac{2}{(3k-h+1)(3k-h+2)}.
\end{aligned}$$

Hence we save at least

$$\log\frac{\binom{n}{k}^3}{\binom{3n}{3k-h}} \geq h\log\frac{n-k}{k} + 1 - 2\log(3k) = h\log(C-1) + 1 - 2\log(3n/C)$$

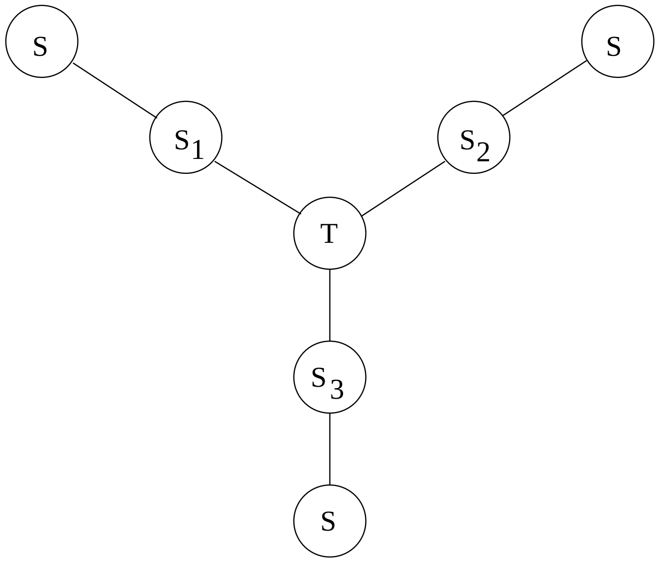


Figure 1: Alignment Tree

bits, which exceeds (5) for large enough $C$. This implies nontrivial compression, contradicting (4). $\square$

Call those errors that have distance more than $\log^2 C$ away from other errors *lonely*. A lonely error will be corrected by our algorithm if the Steiner alignment correctly aligns the surrounding characters in the 3 sequences, as Lemma 2 shows. Note that a Steiner sequence induces a multiple alignment of the 3 erroneous sequences, according to the alignment tree in Figure 1. The Steiner distance can be seen to be at least the cost of that multiple alignment under the following cost measure for columns (where a gap is considered a letter):

- 3 identical letters have 0 cost,
- 2 identical letters plus a different letter have cost 1,
- 3 different letter have cost 2.

Since this cost is the distance achieved by the consensus sequence of the alignment, it follows that the Steiner sequence coincides with the consensus sequence.

For example, consider an original sequence "GTCTACAGC" and 3 erroneous copies, $S_1$ ="TGTCTCGC", $S_2$ ="TGTCACGC", and $S_3$ ="TTATCAGAC", each having 4 errors as indicated in the following alignment:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| - | G | T | C | T | - | A | C | A | G | - | C |
| T | G | T | C | T | - | - | C | - | G | - | C |
| - | - | T | G | T | C | A | C | - | G | - | C |
| - | T | T | A | T | - | - | C | A | G | A | C |

The (unique) Steiner sequence is "TGTCACGC", and it induces the 3-alignment

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| T | - | G | T | C | T | C | G | C |
| T | - | G | T | C | A | C | G | C |
| T | T | A | T | C | A | G | A | C |

A *match* is a pair of identical characters in a column of the 3-alignemnt. In the example, the first column has a triple match, the 2nd none, the 3rd one etc. Note that the alignment of the three erroneous sequences in turn induces an alignment of 3 copies of the original sequence $S$, again according to Figure 1. We call this the $S$*-alignment.* For our example, it is

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | - | - | G | T | - | C | T | A | C | A | - | G | C |
| G | T | - | C | T | - | - | A | - | C | - | A | G | C |
| - | G | T | C | T | A | C | A | - | G | - | - | - | C |

An $S$*-match* is a pair of 2 characters in a column of the $S$-alignment. Alternatively, an $S$-match can be defined as a match of 2 characters neither of which is an insertion into the original sequence $S$. The example has no $S$-match in the 1st column, one in the 2nd, none in the 3rd, three in the 4th, etc. A *barrier* is a position in $S$ whose character fills a whole column in the $S$-alignment. Intuitively, this is where all 3 erroneous sequences are synchronized with respect to $S$. The example has only one barrier at the final position. In addition, we consider the start and end of the whole alignment to constitute 2 (pseudo) barriers.

If we picture 3 copies of $S$ above one another, then an $S$-match pairs two positions in 2 copies of $S$. The characters in these positions can be different, if there was a substitution in either position. With each lonely error we associate a region that extends to the left and to the right up to, but not including, the nearest barrier. Note that some errors will be associated with the same block, which contains all of them. Since regions are barrier-free, most $S$-matches in them must pair up different positions. This is the basis for showing that most regions must be small, i.e. consisting of only the error position. Note that the only way for a region to be larger is to contain more errors. Call the regions containing a lonely error and at least one other error *blocks.* Blocks have size at least $\log^2 C$.

**Lemma 6** *In a 3 sequence alignment (of $S_1, S_2, S_3$), less than $n/C^{2-\epsilon}$ lonely errors appear in blocks.*

**Proof.** Suppose that, on the contrary, $l \geq n/C^{2-\epsilon}$ lonely errors appear in blocks.

Let $L$ be the total length of all these blocks. We have

$$L \geq \frac{l}{2}\log^2 C \geq \frac{n}{2C^{2-\epsilon}}\log^2 C. \tag{6}$$

The idea is to show that we can specify the locations of these blocks and errors to specify $L$ characters of $S$ ($2L$ bits) using a smaller number of bits; thus compressing $S$.

We use $3\log(\log^2 n) = 6\log\log n$ bits for the id's of the 3 error codes.

By Lemma 4, we can specify the boundaries of the blocks using no more than $2l(\log\frac{n}{l}+O(1)) \leq 2l((2-\epsilon)\log C + O(1)) = o(L)$ bits.

Also, we can specify the distribution and type of the $l$ lonely errors appearing in the blocks in $l(\log\frac{n}{l} + O(1)) = o(L)$ bits.

Finally, there can be up to $\frac{n}{C^{2-\epsilon}}$ non-lonely errors in the blocks, whose distribution and types takes at most $\frac{n}{C^{2-\epsilon}}((2-\epsilon)\log C + O(1))$ bits to specify. The latter is also $o(L)$, by (6).

Using all this information we can easily reconstruct the missing blocks. Columns in the $S$-alignments are reproduced one by one. Whenever an error occurs, we just use a few bits to fill in the column. When no error occurs, we use the fact that the column is no barrier and fill in the single letter deduced from one of the $S$matches.

The last two lemmas show that all errors, except for at most $\frac{n}{C^{2-\epsilon}}$ non-lonely errors, and at most $\frac{n}{C^{2-\epsilon}}$ lonely errors, get corrected. Thus, by optimality, the Steiner sequence must have distance at most $\frac{2n}{C^{2-\epsilon}}$ to the original sequence $S$.

## 5 Analysis: the Induction Step

Given three sequences, we have shown that aligning them reduces the random errors from $n/C$ to $n/C^{2-\epsilon}$ for $\epsilon < 1$. In this section, we further inductively analyze the alignment of the aligned (Steiner) sequences as in our algorithm to show that we can keep on reducing the errors.

**Lemma 7** *Let $A$ be a group of sequences. Assume that $\alpha$ has distance $d$ to $A$. If $\beta$ is a Steiner sequence of $A$, then,*

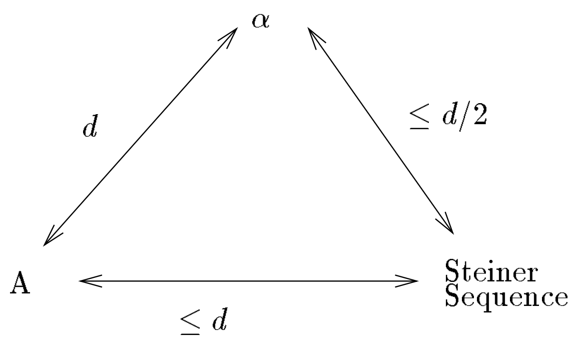


Figure 2: Lemma 7, for the case $|A| = 4$.

- *when* $|A| = 4$, $d(\alpha, \beta) \leq \frac{d}{2}$.
- *when* $|A| = 3$, $d(\alpha, \beta) \leq \frac{d}{1.5}$.

**Proof.** Because $\alpha$ has distance $d$ to $A$ and $\beta$ is a Steiner sequence of $A$, we know the distance from $\beta$ to $A$ is at most $d$. Thus, Lemma 3 implies that

$$d(\alpha, \beta) \leq \frac{d+d}{|A|} = \frac{2d}{|A|}.$$

Thus the two claims follow. The case of $|A| = 4$ is shown in Figure 2. □

Initially, we hoped to also prove $d(\alpha, \beta) \leq \frac{d}{2}$, instead of $\frac{d}{1.5}$ in Lemma 7, when $|A| = 3$. All simple examples supported this conjecture. However, the following counterexample shows that this is not ture.

$$\begin{array}{lcccccccc}
S_1 & = & \mathrm{A} & \mathrm{A} & \mathrm{C} & \mathrm{T} & \mathrm{T} & \mathrm{T} & \mathrm{T} \\
S_2 & = & - & - & - & \mathrm{G} & \mathrm{G} & \mathrm{T} & \mathrm{T} \\
S_3 & = & \mathrm{A} & \mathrm{A} & \mathrm{C} & \mathrm{G} & \mathrm{G} & \mathrm{A} & \mathrm{A} \\
\alpha & = & \mathrm{G} & \mathrm{G} & \mathrm{C} & \mathrm{T} & \mathrm{T} & \mathrm{A} & \mathrm{A}
\end{array}$$

The distance from $\alpha$ to $S_1, S_2, S_3$ is 11; but the Steiner sequence to $S_1, S_2, S_3$ is AACGGTT which has distance 6 to them.

From Lemma 7, we know that if there are $d$ errors in the sequences in three sequences $S_1, S_2, S_3$, then there are at most $d/2$ errors in the Steiner sequence $\alpha$ obtained by aligning these sequences. In fact, if a block $B$ of $\alpha$ is not mismatched, then it contains at most half the total number of errors in the corresponding blocks in $S_1, S_2, S_3$. Thus for each error $e$ in $\alpha$, we can find the smallest not-mismatched block $B$ of $\alpha$ surrounding $e$ such that the first character and the last character of $B$ are not mismatched, as shown in Figure ??.

Such a block might include $k$ errors including $e$. We know from Lemma 7 that there must be at least $2k$ errors in the corresponding blocks in $S_1, S_2, S_3$. We will quite arbitrarily say that the $i$-th error in $B$ was *caused* by the $2i$-th and $2i+1$-st errors in the matching blocks of $S_1, S_2, S_3$.

After the first round, we have at most $n/C^{2-\epsilon}$ errors in the resulting copy. We would like to show that after the $k$-th round, the following are true:

- each remaining copy is merged from $3^k$ copies;
- in each remaining copy, there are at most $n/C^{(2-\epsilon)^k}$ errors;
- each error in the remaining copies is caused by at least $1.5^k$ errors in the original copies (coded by some of $e_1, \ldots, e_{\log n}$).

Assume that the above are true after the $i$-th round. We will inductively prove that this holds after the $i+1$st round.

At the $i$-th round, let $C' = C^{(2-\epsilon)^i}$. Let $D$ be the set of errors that are *caused* by $2^i$ original errors spreading a distance more than $2^i \log C$. Distance is measured along the original sequence $S$, for a specific optimal alignment of each sequence to $S$. In fact, this needs to be defined as distance from barrier to barrier (define this carefully). This needs Lemma 7 – to count.

**Lemma 8** *For some $\epsilon < 1$, there are less than $h = n/C'^{2-\epsilon}$ errors not in $D$ that are within distance $d = \log C' \log C$ from some other error not in $D$.*

(to prove this, show the errors can be compressed, since many are close)

**Lemma 9** *In the alignment (of the three Steiner sequences at the $i$-th round), less than $n/C'^{2-\epsilon}$ lonely errors appear in blocks.*

(to prove this, show S can be compressed)

**Lemma 10** $|D| \leq n/C'^{2-\epsilon}$.

For convenience, let $C' = C^{(2-\epsilon)^i}$.

**Lemma 11** *For some $\epsilon < 1$, there are at most $n/C'^{2-\epsilon}$ pairs of errors that are less than $\log C' \log C$ apart and each is caused by $2^i$ original errors spreading over a region less than $\log C' \log C$ long.*

**Proof.** Suppose that this is not the case. Then there are $n/C'^{2-\epsilon}$ pairs of errors that are within $\log C' \log C$ to each other, and each of them is caused by at least $2^i$ original errors spreading over a distance of at most $\log C' \log C$ long.

We will specify them separately and thus compress $e_1, \ldots, e_{\log n}$. We can specify these errors in

$$\frac{n}{C'^{2-\epsilon}}(\log C'^{2-\epsilon} + 2^i \log\log C)$$

bits. But, according to Lemma 4, this saves

$$\frac{n}{C'^{2-\epsilon}} 2^i \log C$$

bits from the $e_i$'s. This compresses the $e_i$'s. □

Now, as in the basic case, we mark a region of size $\log C' \log C$ surrounding each error. Then by above lemma we know that at most $n/C'^{2-\epsilon}$ regions contain more than 2 errors each and the total number of errors such regions contain is at most $2n/C'^{2-\epsilon}$.

**Lemma 12** *Neither of the following cases is true:*

1. *there are more than $n/C'^{2-\epsilon}$ errors each caused by $2^i$ original errors spreading over a region more than $\log C' \log C$ long;*
2. *in the alignment, more than $n/C'^{2-\epsilon}$ pairs of errors of distance greater than $\log C' \log C$ are aligned together.*

**Proof.** If either case is true, then there are more than $n/C'^{2-\epsilon}$ mismatched blocks each of size $\log C' \log C$.

We would like to use this information to compress portions of $S$. As before we need to use

- at most $\frac{n}{C'^{2-\epsilon}} \log C'^{2-\epsilon}$ bits to specify the positions of these blocks, according to Lemma **??**
- within each block, at most $2^i \log\log C$ bits to specify the positions of $2^i$ errors, according to Lemma **??**
- within each block, at most $2^i \log\log C$ bits to specify the positions of the $2^i$ segments merged from the past stages.

So in total, this requires at most

$$\frac{n}{C'^{2-\epsilon}}(\log C'^{2-\epsilon} + O(2^i \log\log C))$$

bits. But we save

$$\frac{n}{C'^{2-\epsilon}} \log C' \log C$$

bits from $S$, achieving nontrivial compression. $\Box$

Now when a block with just one error is correctly aligned, then by Lemma 2, we know it will result the correct subsequence for that block.

Thus after the $i+1$st round, the following are true:

- each remaining copy is merged from $3^{i+1}$ copies;
- in each remaining copy, there are at most $n/C^{(2-\epsilon)^{i+1}}$ errors;
- each error in the remaining copies is caused by at least $2^{i+1}$ errors in the original copies.

After roughly $2\log\log n$ rounds, the errors will be all gone. So we only need less than $\log^2 n$ sequences. In fact, we need only $\log^{1+\delta} n$ for some $0 < \delta < 1$.

Finally, since almost all sequences are Kolmogorov random, i.e. satisfy Equation 2, with high probability, greater than $1 - \frac{1}{n}$, the errors are Kolmogorov random and thus the above analysis holds. This completes the proof of Theorem 1.

## 6 Acknowledgements

This research was initiated when the authors were visiting DIMACS in October 1994 during its computational biology year. We thank DIMACS for providing the stimulating environment and the financial supports. Thanks also to Tao Jiang for discussions on this research.

## References

[1] T. Blackwell, Ph.D. Thesis, Harvard University, 1994.

[2] M. Li and P. Vitányi, *An introduction to Kolmogorov complexity and its applications.* Springer-Verlag, 1993.

[3] M. Li and P. Vitányi, Kolmogorov complexity and its applications. *Handbook of Theoretical Computer Science, Vol. A*, 187–254, J. van Leeuwen, Ed., Elsevier/MIT Press, 1990.

[4] L. Smith, The future of DNA sequencing. *Science* 262, 530-532, 1993.

[5] R. Staden, Automation of the computer handling of gel reading data produced by the shotgun method of DNA sequencing. *Nucleic Acids Research 10(15), 4731-4751, 1982.*