ELSEVIER

Theoretical Computer Science 185 (1997) 3–13

Theoretical Computer Science

Inferring a DNA sequence from erroneous copies

John Kececioglu [a,1], Ming Li [b,2], John Tromp [c,*]

[a] *Department of Computer Science, University of Georgia, Athens, GA 30602, USA*
[b] *Department of Computer Science, University of Waterloo, Waterloo, Ont., Canada N2L 3G1*
[c] *CWI, P.O. Box 94079, 1090 GB Amsterdam, Netherlands*

Abstract

We suggest a novel approach for efficiently reconstructing an original DNA sequence from erroneous copies.

Keywords: 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 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 does not 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 simplifying assumptions:

(i) the DNA sequence is a random sequence (in the sense of being algorithmically incompressible), and

* Corresponding author. E-mail: tromp@cwi.nl. Supported by an NSERC International Fellowship.
[1] Supported in part by DIMACS.
[2] Supported in part by the NSERC Operating Grant OGP0046506, ITRC, a CGAT grant and DIMACS.

PII S0304-3975(97)00013-3

(ii) all error types/distributions are equally likely.

Assumption (ii) is natural. Although assumption (i) is unrealistic, with some portions of DNA being "AT-rich", others being "GC-rich" and some regions featuring long repetitions, we expect them to behave enough like random sequences to preserve the validity of our analysis.

We propose a method of efficiently reconstructing an original DNA sequence of length n from $\log^{O(1)} n$ erroneous copies, assuming the sequence itself is random and errors are random with constant error rate $1/C$.

The obvious way to handle the problem is to compare all the erroneous copies and find the similarities that necessarily exist among them. This is commonly known as 'multiple sequence alignment'. A problem is that doing multiple sequence alignment on a collection of k sequences by any known algorithm that guarantees optimality takes time exponential in k, making it far too slow for practical use. Even if we could do multiple sequence alignment efficiently, it is not clear how many sequences are needed to converge to the real sequence. Blackwell [1] has shown that among an exponential number of erroneous copies, the original sequence is likely to appear so frequently that with high probability it can be identified as the sequence occurring most often. This approach is clearly not practical because n is usually very large, from hundreds to thousands. With our approach, we should need only a polylogarithmic number of copies. And our algorithm converges in polynomial time, rather than the worst-case superpolynomial time and space of multiple alignment algorithms like [4].

2. Preliminaries

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

Consider a Turing machine C with input alphabet $\Sigma = \{0, 1\}$. The Kolmogorov complexity of a string $x \in \Sigma^*$, given $y \in \Sigma^*$, relative to Turing machine C, is defined as

$$K_C(x|y) = \min\{|p| : C(p, y) = x\}.$$

An invariance theorem shows that for a *universal* machine U, $K_U(x|y) \leqslant K_C(x|y) + O(1)$ holds for any machine C, with the constant implicit in $O(1)$ depending on C only. It follows that the Kolmogorov complexity defined with respect to any two universal machines differs by only an additive constant. We can thus fix a universal machine U and denote $K_U(\,)$ simply as $K(\,)$. Thus, $K(x|y)$ is the *minimum* number of *bits* in a description from which x can be effectively reconstructed, given y. Let $K(x) = K(x|\lambda)$, where λ denotes the empty string.

By simple counting, for each n, constant $c < n$, and y, there are at least $2^n - (2^{n-c} - 1)$ distinct x's of length n with the property

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

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

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

where the logarithm is of base 2 (as throughout the paper). 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. This we do by prefixing x with its length (in binary), and that in turn by its length in unary, i.e. $\bar{x} = 1^{\|x\|}0|x|x$. Thus, the self-delimiting representation $\bar{x}$ of x requires $|x| + 2\log|x| + O(1)$ bits.

A DNA sequence is a word over the four-letter alphabet of *bases* $B = \{A, C, G, T\}$. We can encode a letter in B by two bits. Thus, when we say a sequence over B is Kolmogorov random, we mean the binary string encoding it is Kolmogorov random. Assume that our original DNA sequence S is of length n bases, i.e., $2n$ bits. We will assume that this string $S = s_1 \ldots s_{2n}$ contains n bases, the ith 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), replace a base (2 bits by some other 2 bits), or insert a base (also 2 bits) in between bases (i.e., after even positions in S).

Let the distance between two sequences α and β, $d(\alpha, \beta)$, be the minimum alignment score between α and β, where each insertion, deletion, and substitution is charged unit cost. The distance from one sequence α 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 α that minimizes the distance from α 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

In order to achieve polynomial-time complexity, we cannot align all the erroneous sequences at the same time. The idea is to separate them into groups. In order to be able to recognize errors one needs at least a majority of correct versions. We therefore choose to divide in groups of three. Wherever we manage to align three erroneous sequences properly and at most one sequence has an error, then it will get corrected.

Our algorithm is thus very simple. It repeatedly partitions the set of sequences into groups of three which are replaced by their Steiner sequence, thereby reducing the number of sequences by a factor of 3. In each round, it tries to reduce the error rate almost quadratically. Convergence is then easy to analyze. Removing all errors is equivalent to reducing the error rate to less than $1/n$, n being the length of the original sequence. Starting from a constant $1/C$ error rate, this takes no more than $\log\log n$ rounds, requiring $3^{\log\log n} = (\log n)^{\log 3}$ erroneous copies to start with. In this paper we analyze in detail the first round and prove that under the assumed randomness conditions, it reduces the error rate as desired. We conclude the paper with a short discussion of the difficulties in analyzing later rounds, and present some experimental data.

We end this section with two lemmas that will prove useful in the next section, where we analyze a single iteration of our algorithm.

Lemma 1. *The Steiner sequence of three sequences α, α, β is α.*

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

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

Lemma 2. *For $k < n$,*

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

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

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

the first inequality is true because $n/k \leqslant (n-i)/(k-i)$ for $k \leqslant n$. Taking logarithms, 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 of aligning three sequences

Let sequence S of length n be the original DNA sequence to be reconstructed.

At any position in S we allow at most one of the following 8 errors:

- the character is deleted,
- the character is replaced by one of three others,
- one of four possible characters is inserted just before it.

A collection of k errors can then be distributed in $\binom{n}{k}$ possible ways and be of 8^k types. For simplicity, we ignore the possibility of multiple insertions at a single position in S. This more general case could be handled with some extra complication as follows: Divide the errors into two types, insertions and non-insertions. For k_1 insertions and k_2 deletions/substitutions the number of distributions can be counted as $\binom{n+k_1}{k_1}\binom{n}{k_2}$. Summing this over all ways of writing $k = k_1 + k_2$ gives the general distribution count. Since using such expressions would severely complicate our analysis, we prefer to present our proof methods in the simpler setting.

For an error rate of $1/C$ we assume a standard encoding of an error sequence as $\log e(n)$ bits, where $e(n) = \binom{n}{n/C} \times 8^{n/C}$.

Consider any three erroneous 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 the error will persist. But since the errors are random, we expect only approximately n/C^2 such error coincidences. Then, in the Steiner sequence, we expect to have only this many errors left. Of course, since we have to do alignment of the three sequences, such a naive analysis does not suffice.

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.

Our main result is

Theorem 3. *For any $\varepsilon > 0$, there exists a constant C_0, such that for any $C \geqslant C_0$, and any sufficiently large Kolmogorov random sequence S, given three erroneous copies S_1, S_2, S_3 with Kolmogorov random errors encoded by e_1, e_2, e_3, our algorithm reduces the error rate from $1/C$ to $1/C^{2-\varepsilon}$ in one iteration.*

Proof. We start the proof with some supporting lemmas. Note that distance between errors is measured along their occurrence on S.

Lemma 4. *For any $\varepsilon > 0$, sufficiently large C and n, and sequences as above, there are less than $h = n/C^{2-\varepsilon}$ 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. The first (leftmost) error in each group is designated *group leader*. By Lemma 2, we can encode the positions of the group leaders using $\log(n/h) + O(1) = (2 - \varepsilon)\log C + O(1)$ 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)

$$\frac{h}{2}((2 - \varepsilon)\log C + 2\log\log C + O(1)) \tag{3}$$

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

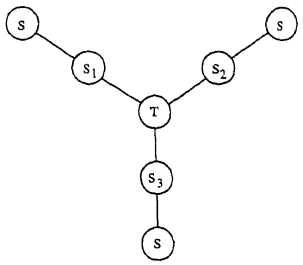

Fig. 1. Alignment tree.

logarithm of the ratio between these two:

$$\frac{\binom{n}{k}^3}{\binom{3n}{3k-h}} = \left(\frac{n\ldots(n-k+1)}{k!}\right)^3 \frac{(3k-h)!}{(3n)\ldots(3n-3k+h+1)}$$

$$\geqslant \frac{((n-k+h/3)\ldots(n-k+1))^3(3k-h)!}{k!^3 3^{3k-h}}$$

$$\geqslant \frac{((n-k+h/3)\ldots(n-k+1))^3}{(k\ldots(k-h/3+1))^3} \frac{2}{(3k-h+1)(3k-h+2)}.$$

Hence, we save at least

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

bits, which exceeds (3) for large enough C. This implies nontrivial compression, contradicting the assumed randomness of e_1, e_2, e_3. □

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 1 shows.

Fig. 1 shows the three erroneous sequences S_1, S_2, S_3 and their Steiner sequence T, which minimizes the distance to the erroneous sequences. Each S_i in turn has distance n/C to the original sequence S, as represented by the outer edges. Note that a Steiner sequence T induces a multiple alignment of the 3 erroneous sequences, under which the Steiner distance equals the following alignment 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.

The Steiner sequence can thus be chosen as the *consensus sequence* of the optimal triple alignment using this cost measure, which is defined by choosing the majority element in each column.

For example, consider an original sequence "GTCTACAGC" and three 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-alignment. 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 combination with the cost n/C alignments of each error sequence with the original sequence, in turn induces an alignment of 3 copies of the original sequence S. In fact, any tree of pairwise alignments, such as Fig. 1, induces a multiple alignment of all its leaves. 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. The 2nd column shows that the suffix match refers to the position in S rather than the base. 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 region, 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 $\log^2 C$.

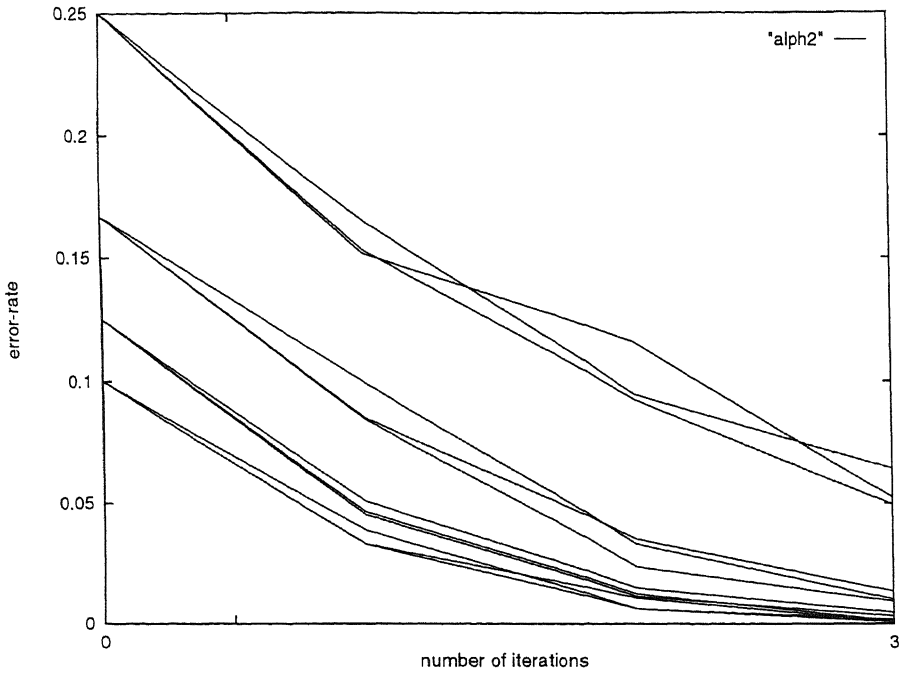

Fig. 2. Alphabet size 2.

Lemma 5. *In a* 3 *sequence alignment* (*of* S_1, S_2, S_3), *less than* $n/C^{2-\varepsilon}$ *lonely errors appear in blocks.*

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

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

$$L \geqslant \frac{l}{2} \log^2 C \geqslant \frac{n}{2C^{2-\varepsilon}} \log^2 C. \tag{4}$$

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 2, we can specify the boundaries of the blocks using no more than $2l(\log n/l + O(1)) \leqslant 2l((2-\varepsilon)\log C + O(1)) = o(L)$ bits.

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

Finally, there can be up to $n/C^{2-\varepsilon}$ non-lonely errors in the blocks, whose distribution and types takes at most $(n/C^{2-\varepsilon})((2-\varepsilon)\log C + O(1))$ bits to specify. The latter is also $o(L)$, by (4).

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

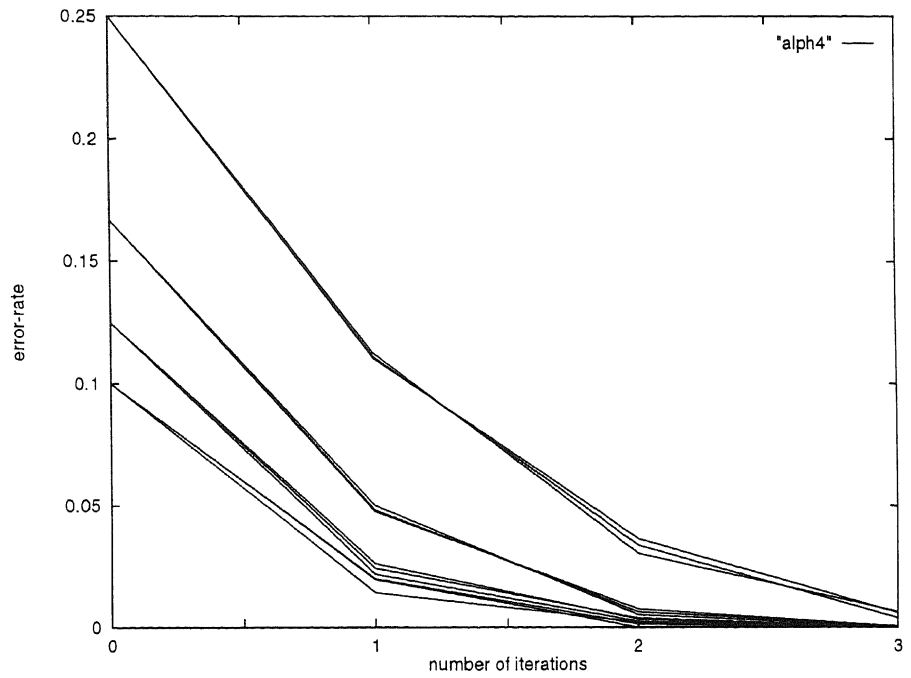

Fig. 3. Alphabet size 4.

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 $n/C^{2-\varepsilon}$ non-lonely errors, and at most $n/C^{2-\varepsilon}$ lonely errors, get corrected. Thus, by optimality, the Steiner sequence must have distance at most $2n/C^{2-\varepsilon}$ to the original sequence S.

5. Comments

Here we briefly consider the question of what the theorem on single iteration error reduction implies about the whole process. Ideally, the computed Steiner sequences would satisfy a condition of Kolmogorov randomness similar to the conditions we assumed to hold for the original error sequences. One complication arising here is that the computed sequences may not fit into our error model due to the introduction of multiple insertion errors. Another is that our analysis requires a lower bound on C in terms of ε, which suggests it will be hard to prove error reduction below a certain level.

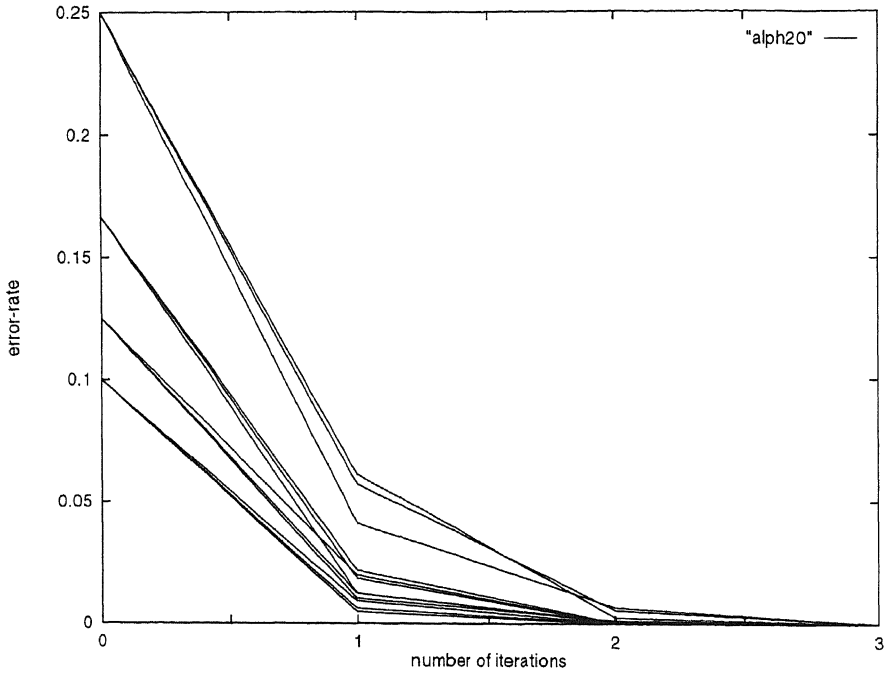

Fig. 4. Alphabet size 20.

6. Experimental results

In order to test our algorithm, we have performed the following experiments.

We generated three groups of random sequences over alphabet sizes 2, 4 (to model DNA/RNA), and 20 (to model proteins), respectively. Within each group, we generated sequences of lengths 50, 100, and 200. Much longer sequences were beyond our computational abilities, which were limited by the use of a qubic space algorithm for computing Steiner sequences.

Within each group, we tested sequences of each length with error rates of $\frac{1}{4}$, $\frac{1}{6}$, $\frac{1}{8}$, and $\frac{1}{10}$. Figs. 2, 3, and 4 show the experimental results for these three groups, respectively. Each curve in each of these pictures shows the average over a dozen runs of how the error rate goes down with successive iterations, for a particular sequence length. As the pictures show, the method converges well.

Acknowledgements

This research was initiated when the first two 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, and the anonymous referees for corrections and suggested improvements.

References

[1] T.W. Blackwell, Estimating consensus DNA sequences, Ph.D. Thesis, Harvard University, 1993.
[2] M. Li, P. Vitányi, An Introduction to Kolmogorov Complexity and its Applications (Springer, Berlin, 1997, 2nd Ed.).
[3] M. Li, P. Vitányi, Kolmogorov complexity and its applications, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, vol. A, Elsevier, Amsterdam, 1990, pp. 187–254.
[4] D.J. Lipman, S.F. Altschul and J.D. Kececioglu, A tool for multiple sequence alignment, Proc. National Acad. Sci. USA 86 (1989) 4412–4415.
[5] L. Smith, The future of DNA sequencing, Science 262 (1993) 530–532.
[6] R. Staden, Automation of the computer handling of gel reading data produced by the shotgun method of DNA sequencing, Nucleic Acids Res. 10(15) (1982) 4731–4751.