Inferring an Original Sequence from Erroneous Copies: a Bayesian Approach

Jonathan M. Keith[1] **Peter Adams**[1] **Darryn Bryant**[1] **Keith R. Mitchelson**[2,3]
Duncan A. E. Cochran[1,2] **Gita H. Lala**[1,2]

[1]Department of Mathematics
The University of Queensland
St Lucia, Qld 4072
Australia
[2]Australian Genome Research Facility
The University of Queensland
St Lucia, Qld 4072
Australia
[3]Institute for Molecular Bioscience
The University of Queensland
St Lucia, Qld 4072
Australia
Email: j.keith1@mailbox.uq.edu.au

Abstract

This paper considers the problem of inferring an original sequence from a number of erroneous copies. The problem arises in DNA sequencing, particularly in the context of emerging technologies that provide high throughput or other advantages, but at the cost of introducing many errors. We develop a Bayesian probabilistic model of the introduction of errors, and search for a sequence that has maximum posterior probability with respect to the model. We present results of extensive tests in which error-prone sequencing of real DNA was simulated. The results obtained using the new approach are compared to results obtained by deriving a consensus sequence from a multiple sequence alignment. We find that a significant improvement in accuracy is obtained using the new approach. The implication is that high error levels need not be a barrier to the adoption of sequencing technologies that are in other respects promising, because most errors can be detected and corrected using a small number of reads.

Keywords: DNA sequencing, sequencing error, Bayesian inference, consensus sequence.

1 Introduction

The need to infer an original sequence from a number of erroneous copies is common in DNA sequencing. All DNA sequencing technologies are imperfect, and it is therefore not advisable to place too much confidence in sequence obtained from a single read. By generating multiple overlapping reads, and comparing them, errors can be detected and a closer approximation to the actual sequence can be obtained. This issue is particularly important for certain emerging technologies that promise rapid sequencing at the cost of an increased number of errors. Kececioglu *et al.* (Kececioglu, Li & Tromp 1997) cites *single-molecule DNA sequencing* as an example of such a technology. In another example, the authors are currently developing a technology in which as many as 20-30% of the bases in any given read may be erroneous. If such technologies gain currency, the ability to accurately and efficiently infer a sequence from erroneous copies may become increasingly important. Indeed, that ability may be crucial in rendering such technologies feasible.

By far the most common approach to this problem, used in all major sequencing projects, is to form a multiple sequence alignment of the erroneous reads and to determine a (possibly weighted) consensus character for each column of the alignment. This approach is attributable to no single author or group, but Gusfield (Gusfield 1997) provides an interesting discussion of it. Kececioglu *et al.* (Kececioglu et al. 1997) described an elegant approach based on aligning sequences in groups of three. Li *et al.* (Li, Ma & Wang 2000) developed a number of Polynomial Time Approximation Schemes for determining consensus sequences under various assumptions. Keith *et al.* (Keith *et al.* to appear) developed a simulated annealing algorithm to search for a Steiner string under the edit distance metric. (A Steiner string is a sequence that minimises the sum of distances to each of the input sequences, and is a form of consensus sequence.)

In this paper, we advocate a Bayesian approach to the problem. The advantage of a Bayesian approach is that it enables diverse kinds of information to be taken into account when drawing an inference. For example, prior information about the length and composition of the original sequence, and more importantly, information about the kinds and frequencies of errors that can occur, can all be used to inform the inference. The result is a reconstructed sequence that is potentially more accurate, and also more meaningful in the sense that one may make probabilistic statements about what it represents. For example, one may state that the reconstructed sequence is, with some probability, the original sequence from which the erroneous copies were derived. One may also determine probabilities for other candidate sequences. In fairness, it should be noted that some of the other approaches mentioned above can also incorporate various kinds of information, mediated by the choice of value for certain parameters. However, it is not always clear how one should set the parameters in the light of the information available. A Bayesian ap-

proach is more direct, more natural, more flexible, and as we have already stated, more meaningful.

It is not our intention, in this paper, to model the actual mechanisms by which sequencing errors are introduced. Rather, we propose an idealised model of the introduction of errors. This is not to under-rate the method - we believe the resulting algorithm is a powerful, practical tool for inferring an original sequence from erroneous copies. But our main purposes here are to illustrate the Bayesian approach to this problem and to test its computational feasibility. This latter purpose is important because Bayesian models, despite their advantages, can lead to computationally inefficient or infeasible algorithms.

The paper is structured as follows. In Section 2 we describe a probabilistic model of the introduction of errors. In Sections 3 and 4 we describe computational methods used in drawing inferences from the model. In Section 5, we present and discuss results of extensive tests in which modified copies of an original DNA sequence are simulated and the original sequence is then reconstructed from these copies. For comparison, we also present results of similar tests in which consensus sequences were obtained from multiple sequence alignments produced by a well-known alignment package.

2 The probabilistic model

Most DNA sequencing techniques read molecules sequentially. Consequently, once a part of the molecule has been read, no further errors can be introduced to that part of the sequence. This observation suggests the following model, in which insertions, deletions and substitutions are introduced into the sequence in order from left to right.

Let the original sequence be X and suppose that it is formed from characters of a finite alphabet Σ. Let the left-most character of X be temporarily designated the *current character*. Now, consider inserting a character immediately to the left of the current character. Let the probability of inserting character $x \in \Sigma$, be $r(-,x)$ and the probability of not inserting any character be $r(-,-)$. Note that $r(-,-)+\sum_x r(-,x) = 1$. If a character was inserted, consider a second insertion immediately to the right of the inserted character, that is, immediately to the left of the current character. The probability of inserting a character x, or of not inserting any character, is here assumed to be the same for the second (and subsequent) insertions as for the first insertion, namely $r(-,x)$ and $r(-,-)$. Characters are inserted in this manner until a decision is made not to insert a character.

Next, choose whether to substitute the current character with a different character, delete the current character, or leave the current character unaltered. Let the value of the current character be $x \in \Sigma$ and let the probability of substituting this character with character $y \in \Sigma$ be $r(x,y)$. Let the probability of deleting character x be $r(x,-)$ and let the probability of leaving the character unaltered be $r(x,x)$. Note that $r(x,-) + \sum_y r(x,y) = 1$. Having made this choice, let the next character in the sequence be designated the current character and consider making insertions, deletions and substitutions as before. Continue this process until the last character is reached. At this point, make a final round of insertions at the end of the sequence.

In summary, the process consists of the following steps. For ease of description, we suppose that a termination character has been appended at the right end of X.

1. Set $i := 1$.
2. Consider an insertion immediately to the left of character i of X.
3. If an insertion was made at Step 2,
   (a) Set $i := i + 1$.
   (b) Go to Step 2.
4. If character i is not the termination character,
   (a) Consider deleting or substituting character i of X.
   (b) If a deletion was not made at Step 4(a), set $i := i + 1$.
   (c) Go to Step 2.

Now, we want to compute the probability that the sequence generated by applying this process to X is a given sequence Y. To do this, we must consider all possible alignments of Y to X. Each alignment may be interpreted as showing a possible way in which Y could have been obtained from X. In a given alignment, each character in Y that is aligned to a character in X is interpreted as having been substituted for that character if the characters are different, or as having been preserved from X if the characters are the same. Each character of X that is aligned to a space in Y is interpreted as having been deleted, and each character of Y that is aligned to a space in X is interpreted as having been inserted. A probability can thus be assigned to each alignment, specifically the probability that the point mutations implied by the alignment would occur under the model. The probability $p(Y|X)$ that a sequence Y is obtained by modifying a sequence X is then the sum of these probabilities over all alignments. Letting $\mathcal{M}(X,Y)$ denote the set of all such alignments, this probability may be written as shown in Equation 1.

$$p(Y|X) = r(-,-)^{m+1} \sum_{\mathcal{M} \in \mathcal{M}(X,Y)} \prod_{k=1}^{len(\mathcal{M})} r(s_k, t_k) \quad (1)$$

Here m is the length of sequence X, $len(\mathcal{M})$ is the length of the alignment $\mathcal{M}$, s_k is the kth token in the row of the alignment corresponding to X (a *token* being a character or a space), and t_k is the kth token in the row of the alignment corresponding to Y. Note that the term $r(-,-)^{m+1}$ results from the fact that the decision not to insert another character must be made for each of the $m+1$ positions between adjacent characters of X and at the ends of X.

Suppose that we are now given q independently generated modified copies $Y_1, Y_2, \ldots, Y_q$ of an unknown original sequence. Using Bayes' rule, the posterior probability that the original sequence was X is given by Equation 2

$$p(X|Y_1, Y_2, \ldots, Y_q) = \frac{p(X) \prod_{k=1}^{q} p(Y_k|X)}{\sum_Z p(Z) \prod_{k=1}^{q} p(Y_k|Z)}, \quad (2)$$

where $p(X)$ is the prior probability that the original sequence was X. The prior probability encapsulates information about X that was available prior to observing the erroneous sequence reads, such as

information about the length and composition of X. Here we assume that all sequences of equal length are equally probable prior to observing the reads. Since there are $|\Sigma|^L$ reads of length L, where $|\Sigma|$ is the number of characters in the alphabet, the prior probability can be written as $p(X) = f(L(X))/|\Sigma|^{L(X)}$, where $L(X)$ is the length of X and f is a prior probability distribution with regard to the length of the original sequence. We further assume that all sequence lengths are equally likely prior to observing the reads, and hence that f is uniform.

This model thus determines the probability $p(X|Y_1, Y_2, \ldots, Y_q)$ that any given sequence X was the original sequence from which the erroneous copies were derived. In Section 4, we mention an efficient Markov Chain Monte Carlo (MCMC) sampler that can be used to sample from probability distributions of this form. This sampler was used in the context of simulated annealing to determine the most probable original sequence X^* under the model for the test cases presented in Section 5. However, we do not regard this search technique as a defining property of the method, and there may be many alternative search techniques that could be used with equal success.

3 Computing a sum over all alignments

Equation 1 involves summing over all possible alignments of two sequences. This sounds like a formidable computational task, but in fact it can be done efficiently using dynamic programming. We require the following notation. Let the prefix of X ending at character i be denoted $X[1..i]$ and the prefix of Y ending at character j be denoted $Y[1..j]$. Moreover, let $X[1..0]$ and $Y[1..0]$ represent the null prefixes of X and Y respectively. Define

$$\sigma(i,j) = \sum_{\mathcal{M} \in \mathcal{M}(X[1..i], Y[1..j])} \prod_{k=1}^{len(\mathcal{M})} r(s_k, t_k)$$

for all $i = 0, \ldots, m$ and $j = 0, \ldots, n$ where m and n are the lengths of X and Y respectively. These values may be calculated by first setting the boundary conditions:

$$\sigma(0,0) = 1,$$

$$\sigma(i,0) = \prod_{k=1}^{i} r(x_k, -) \quad \text{for } i = 1, \ldots, m$$

and

$$\sigma(0,j) = \prod_{k=1}^{j} r(-, y_k) \quad \text{for } j = 1, \ldots, n,$$

and calculating the remaining values using the recurrence relation:

$$\begin{aligned}\sigma(i,j) = {} & r(x_i, -)\sigma(i-1, j) + \\ & r(x_i, y_j)\sigma(i-1, j-1) + \\ & r(-, y_j)\sigma(i, j-1)\end{aligned}$$

where x_i is the ith character of X and y_j is the jth character of Y. The three summands in the recurrence relation correspond to sums over all alignments of the prefixes in which the last column of the alignments is $(x_i, -)^T$, $(x_i, y_j)^T$ and $(-, y_j)^T$ respectively, where the superscript T denotes transposition. Then $p(Y|X) = r(-,-)^{m+1}\sigma(m,n)$.

Since the values calculated using this procedure are usually extremely small, it is convenient to work with logarithms as follows. Define $q(a,b) = log[r(a,b)]$ and $\rho(i,j) = log[\sigma(i,j)]$. The initial conditions become

$$\rho(0,0) = 0,$$

$$\rho(i,0) = \sum_{k=1}^{i} q(x_k, -)$$

and

$$\rho(0,j) = \sum_{k=1}^{j} q(-, y_k),$$

and the recurrence relation becomes

$$\begin{aligned}\rho(i,j) = log[& e^{q(x_i,-)+\rho(i-1,j)} + \\ & e^{q(x_i,y_j)+\rho(i-1,j-1)} + \\ & e^{q(-,y_j)+\rho(i,j-1)}].\end{aligned}$$

Then $log(p(Y|X)) = (m+1)q(-,-) + \rho(m,n)$. To ensure that at least one of the exponentials in the recurrence relation does not result in underflow, one may use the identity

$$log[e^a + e^b + e^c] = d + log[e^{a-d} + e^{b-d} + e^{c-d}]$$

where $d = \max\{a, b, c\}$.

4 Updating a sum over all alignments

The dynamic programming algorithm described above enables $P(Y|X)$ to be computed in time $O(mn)$ for sequences X and Y, where m and n are the lengths of X and Y respectively. In this section, we show how to calculate $P(Y|X)$ in time $O(n)$, assuming certain values have previously been computed and stored. We also outline a Markov Chain Monte Carlo sampler called the *string sampler*, and demonstrate that with this technique the required values can indeed be computed and stored for future use.

We require the following notation. Given a sequence X and an erroneous copy Y derived from X, define $\sigma'(i,0) = \sigma(i,0)$ for $i = 0, \ldots, m$ and $\sigma'(i,j) = \sigma(i,j) - r(-, y_j)\sigma(i, j-1)$ for $i = 0, \ldots, m$ and $j = 1, \ldots, n$, where σ is as defined above. Then $\sigma'(i,j)$ is a sum over all alignments of $X[1..i]$ and $Y[1..j]$ excluding those which end with the column $(-, y_j)^T$. In other words, $\sigma'(i,j)$ is a sum over all alignments of these prefixes in which x_i is opposite y_j or opposite a space to the right of y_j. Let $X[i..m]$ denote the suffix of X beginning at character i and $Y[j..n]$ denote the suffix of Y beginning at character j. Moreover, let $X[(m+1)..m]$ and $Y[(n+1)..n]$ represent the null suffixes of X and Y respectively. Then define:

$$\tau(i,j) = \sum_{\mathcal{M} \in \mathcal{M}(X[i..m], Y[j..n])} \prod_{k=1}^{len(\mathcal{M})} r(s_k, t_k)$$

for each $i = m+1, \ldots, 1$ and $j = n+1, \ldots, 1$. These values can be calculated via dynamic programming by setting the boundary conditions:

$$\tau(m+1, n+1) = 1,$$

$$\tau(i, n+1) = \prod_{k=i}^{m} r(x_k, -) \quad \text{for } i = m, \ldots, 1$$

and

$$\tau(m+1,j) = \prod_{k=j}^{n} r(-,y_k) \quad \text{for } j = n,\ldots,1,$$

and using the recurrence relation:

$$\begin{aligned}\tau(i,j) = {} & r(x_i,-)\tau(i+1,j) + \\ & r(x_i,y_j)\tau(i+1,j+1) + \\ & r(-,y_j)\tau(i,j+1)\end{aligned}$$

for the remaining values.

Then we have the following result.

Lemma For any $i = 0,\ldots,m$,

$$\sigma(m,n) = \sum_{j=0}^{n} \sigma'(i,j)\tau(i+1,j+1).$$

To see why this is so, observe firstly that for any i and j in the specified ranges, $\sigma'(i,j)\tau(i+1,j+1)$ expands to a sum of terms of the form $\prod_{k=1}^{len(\mathcal{M})} r(s_k,t_k)$ for some $\mathcal{M} \in \mathcal{M}(X,Y)$. Also observe that this sum is over all alignments of X and Y in which x_i is opposite y_j or a space between y_j and y_{j+1}. Consequently, for fixed i, each alignment of X and Y contributes a term to the expansion of $\sigma'(i,j)\tau(i+1,j+1)$ for one and only one j. One may therefore compute the sum of these terms over all alignments of X and Y by summing $\sigma'(i,j)\tau(i+1,j+1)$ over j for any fixed i, as stated in the lemma.

This lemma can be used to calculate $p(X'|Y)$ for a sequence X' differing by a single-character insertion, deletion or substitution from a sequence X for which certain values are known. We consider insertion, deletion and substitution separately.

Firstly, suppose that X' differs from X only by a single-character *insertion* between characters $i-1$ and i of X, for some $i \in \{1,\ldots,m+1\}$. Also suppose that the values $\sigma_{X,Y}(i-1,j)$ and $\tau_{X,Y}(i,j+1)$ are known for all $j = 0,\ldots,n$. Since $X[1..(i-1)]$ and $X'[1..(i-1)]$ are identical, we have $\sigma_{X',Y}(i-1,j) = \sigma_{X,Y}(i-1,j)$. Also, since $X[i..m]$ and $X'[(i+1)..m+1]$ are identical, we have $\tau_{X',Y}(i+1,j+1) = \tau_{X,Y}(i,j+1)$. One may therefore calculate the values $\sigma_{X',Y}(i,j)$ for $j = 0,\ldots,n$ using the recurrence relation for σ and then calculate $\sigma_{X',Y}(m,n)$ using the lemma. This requires $O(n)$ operations.

Secondly, suppose that X' differs from X only by a single-character *deletion* of character i for some $i \in \{1,\ldots,m\}$. Also suppose that the values $\sigma_{X,Y}(i-1,j)$ and $\tau_{X,Y}(i+1,j+1)$ are known for all $j = 0,\ldots,n$. Since $X[1..(i-1)]$ and $X'[1..(i-1)]$ are identical, we have $\sigma_{X',Y}(i-1,j) = \sigma_{X,Y}(i-1,j)$. Also, since $X[(i+1)..m]$ and $X'[i..(m-1)]$ are identical, we have $\tau_{X',Y}(i,j+1) = \tau_{X,Y}(i+1,j+1)$. One may therefore calculate $\sigma_{X',Y}(m,n)$ using the lemma. This requires $O(n)$ operations.

Thirdly, suppose that X' differs from X only by a single-character *substitution* at character i for some $i \in \{1,\ldots,m\}$. Also suppose that the values $\sigma_{X,Y}(i-1,j)$ and $\tau_{X,Y}(i+1,j+1)$ are known for all $j = 0,\ldots,n$. Since $X[1..(i-1)]$ and $X'[1..(i-1)]$ are identical, we have $\sigma_{X',Y}(i-1,j) = \sigma_{X,Y}(i-1,j)$. Also, since $X[(i+1)..m]$ and $X'[(i+1)..m]$ are identical, we have $\tau_{X',Y}(i+1,j+1) = \tau_{X,Y}(i+1,j+1)$. One may therefore calculate the values $\sigma_{X',Y}(i,j)$ for $j = 0,\ldots,n$ using the recurrence relation for σ and then calculate $\sigma_{X',Y}(m,n)$ using the lemma. This requires $O(n)$ operations.

We use these results to efficiently sample from the probability distribution $p(X|Y_1,Y_2,\ldots,Y_q)$ using the string sampler. The string sampler is described in detail by Keith *et al.* (Keith, Kroese & Bryant submitted) and an example of its use is given by Keith *et al.* (Keith et al. to appear). Here it is sufficient to observe that, beginning with an arbitrary initial sequence, the sampler iteratively scans from left to right through the sequence making single-character insertions, deletions and substitutions in the following manner. For ease of description we suppose that each sequence generated by the sampler has a termination character appended at its right end.

1. Set $i := 1$.
2. Generate a new sequence by either inserting a character immediately to the left of character i of the previous sequence, or else repeating the sequence unchanged.
3. If a character was inserted at Step 2,
   (a) Set i:=i+1.
   (b) Go to Step 2.
4. If character i is not the termination character,
   (a) Generate a new sequence by either deleting character i of the previous sequence, substituting a different character in place of character i, or repeating the sequence unchanged.
   (b) If a character was deleted at Step 4(a) and character i is not now the termination character, go to Step 4(a).
5. If character i is not the termination character,
   (a) Set i:=i+1.
   (b) Go to Step 2.

New sequences are selected at Steps 2 and 4(a). Whether a new sequence X' is selected at these steps depends upon the value of $p(X'|Y_1,\ldots,Y_q)$ (although we shall not go into details of the selection criteria here). Computation of this probability requires $p(Y_k|X)$ to be evaluated for each $k = 1,\ldots,q$. As we have seen, if X' differs from the previous sequence X only by an insertion immediately to the left of character i, as at Step 2, $p(Y_k|X')$ can be calculated in time $O(n_k)$, where n_k is the length of Y_k, provided that the values $\sigma_{X,Y_k}(i-1,j)$ and $\tau_{X,Y_k}(i,j+1)$ are known for all $j = 0,\ldots,n_k$. Similarly, if X' differs from the previous sequence X only by deletion or substitution of character i, as at Step 4(a), $p(Y_k|X')$ can be calculated in time $O(n_k)$, provided that the values $\sigma_{X,Y_k}(i-1,j)$ and $\tau_{X,Y_k}(i+1,j+1)$ are known for all $j = 0,\ldots,n_k$. To ensure that these values are available when they are needed, we do the following. Prior to each iteration, we calculate and store the values $\tau_{X,Y_k}(i,j)$ for all $i = m+1,\ldots,1$ and $j = n_k+1,\ldots,1$. These values do not need to be updated during the iteration, since at the time the value $\tau_{X,Y_k}(i,j)$ is needed, the characters to the right of and including character i in X will not have changed since the beginning of the iteration, and consequently this value will not have changed either. We also initially calculate and store the values $\sigma_{X,Y_k}(0,j)$ for $j = 0,\ldots,n_k$. Now, whenever an insertion is made at Step 2, and whenever a deletion is *not* made at Step 4(a), we calculate and store the values of $\sigma_{X,Y_k}(i,j)$ for all $j = 0,\ldots,n_k$, for use in future calculations. Note that once $\sigma_{X,Y_k}(i,j)$ has been calculated, $\sigma_{X,Y_k}(i-1,j)$

will not be needed again during that iteration, and hence the former value may overwrite the latter.

Using these computational techniques, it is apparent that the time required by each iteration of the string sampler is $O(L_{max} \sum_{k=1}^{q} n_k)$, where L_{max} is the length of the longest sequence generated in that iteration. The memory requirements of that iteration are $O(L_{init} \sum_{k=1}^{q} n_k)$, where L_{init} is the length of the initial sequence for that iteration. This is because we have to store the values of $\tau_{X,Y_k}(i,j)$ for all $i = L_{init}+1,\ldots,1$ and $j = n_k+1,\ldots,1$ at the beginning of the iteration. It should be possible to reduce memory requirements to $O(\sum_{k=1}^{q} n_k)$ because the recurrence relations for τ are invertible, by which we mean that the values of $\tau(i+1,j)$ for $j = n_k+1,\ldots,1$ can be back-calculated from the values $\tau(i,j)$ for $j = n_k+1,\ldots,1$. It should therefore only by necessary to store the values of $\tau_{X,Y_k}(i,j)$ for a single value of i at any one time. However, we have not tried this, nor have we analysed the consequences of finite precision arithmetic for such a procedure.

5 Results and Discussion

To test the method, the following simulations were performed. Fragments of known sequence were selected at random from a database of human DNA. A number of erroneous copies were then simulated for each original sequence. The algorithm used to generate the erroneous copies was that described in Section 2. This was done for various choices of model parameters, although all substitutions were assumed to be equally likely in all tests. That is, $r(x,y)$ was assumed to be independent of x and y for all $x,y \in \Sigma$ with $x \neq y$. We then attempted to reconstruct the original sequence by searching for a sequence with maximum posterior probability with respect to the model described above. The search was performed using the string sampler in the context of simulated annealing. That is, we used the string sampler to sample from a succession of distributions of the form $[p(X|Y_1,\ldots,Y_q)]^{1/T}$, where T is the *temperature* of the distribution. The temperature was gradually lowered until the sample converged to a single sequence. We then calculated the edit distance between each reconstructed sequence and the corresponding original sequence, as a measure of the correctness of the reconstruction. For comparison, we also attempted to reconstruct the original sequence by forming a multiple sequence alignment of the erroneous sequences using the well-known alignment program ClustalW (Thompson, Higgins & Gibson 1994) and then taking a consensus character for each column of the alignment.

Figures 1 to 4 show how the average number of errors (that is, the average edit distance between the reconstruction and the original) varied with the number of erroneous sequences used, for various probabilities of insertion, deletion and substitution. In all of these tests, the length of the original sequence was 400 bases. Each data point in the figures represents an average over approximately 1000 simulations. Results are shown for the new method (lower, black line) and for ClustalW (higher, grey line). Figure 5 shows how the number of errors varies with sequence length using five reads and with probabilities of insertion, deletion and substitution being 0.01, 0.01, and 0.2 respectively.

In Figures 1 to 4, we observe that the number of errors decreases exponentially with the number of sequences when sequences are inferred using the new algorithm. This is a desirable behaviour, as it means

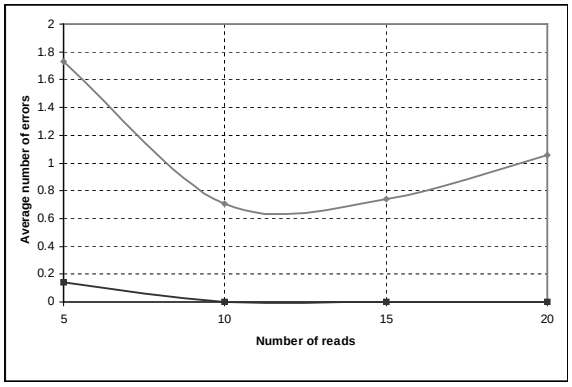

Figure 1: Probabilities of insertion, deletion and substitution are 0.01, 0.01 and 0.03 respectively.

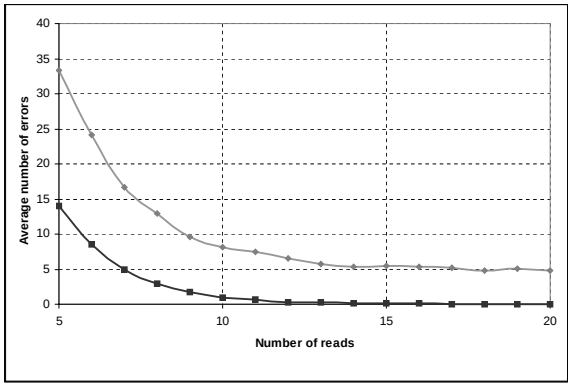

Figure 2: Probabilities of insertion, deletion and substitution are 0.01, 0.01 and 0.2 respectively.

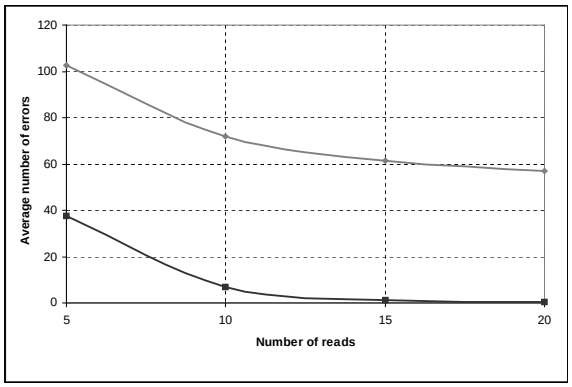

Figure 3: Probabilities of insertion, deletion and substitution are 0.05, 0.05 and 0.2 respectively.

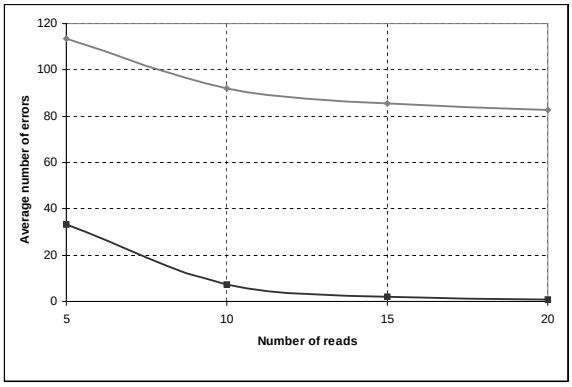

Figure 4: Probabilities of insertion, deletion and substitution are 0.1, 0.1 and 0.1 respectively.

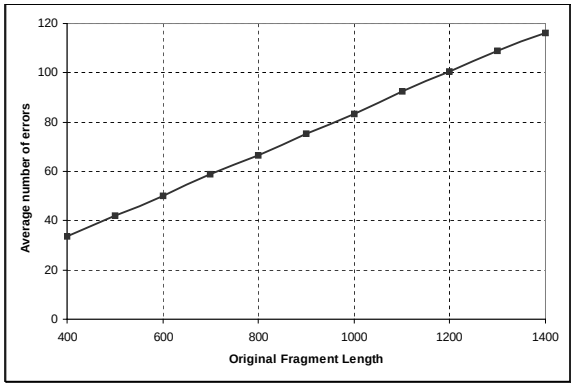

Figure 5: Graph showing the approximately linear dependence of number of errors on sequence length.

that a highly accurate sequence can be inferred from a surprisingly small number of highly inaccurate sequences. We observe that in most cases the sequences inferred using ClustalW resulted in a curve of similar shape, but that the limiting number of errors is non-zero. In Figure 1, the average number of errors actually appears to increase as the number of reads increases, when the number of reads is large. These are highly undesirable behaviours, and we do not know how to account for them. We suspect, however, that the problem is with the sequential alignment approach on which ClustalW is based.

In Figure 5, we observe that the number of errors increases approximately linearly with the length of the original sequence, using the new algorithm. Consequently, the *proportion* of errors in the reconstructed sequence is independent of length. This property should facilitate estimating the number of reads required to achieve a desired accuracy.

When the proportion of errors in the uncorrected sequences is already quite low, as it is for the data presented in Figure 1, the number of errors obtained using either approach is small. Based on these results, it would not be appropriate to claim that there is an urgent need for major sequencing projects to adopt a Bayesian approach to inferring an original sequence. We do, however, make the following points. Firstly, there seems to be a limit to the accuracy that can be achieved by obtaining a consensus sequence from a multiple sequence alignment, at any rate when the alignment is performed using ClustalW. That this should be the case for such a widely used alignment package is concerning. Whether the same is true of the consensus sequences produced by sequencing projects is something that needs to be investigated. Secondly, the Bayesian approach has the flexibility to incorporate application-specific information about the kinds and types of errors that may occur, such as specific sequence patterns that are known to cause errors, or more generally how error probabilities are affected by local sequence characteristics. Detailed models of this kind could enable automated sequence editing of a quality that is currently only possible for an informed human editor.

The advantage conferred by the Bayesian approach is much more significant when the proportion of errors is high, as it is for the data displayed in Figures 2 to 4. The ability to infer high-quality sequence from a small number of inaccurate reads could make the difference between a competitive and a non-competitive sequencing technology. An important conclusion that may be drawn from this study is that error-prone sequencing technologies may in fact be feasible if they possess compensating advantages such as high throughput or low cost.

6 References

References

Gusfield, D. (1997), *Algorithms on strings, trees and sequences*, Cambridge University Press.

Kececioglu, J., Li, M. & Tromp, J. (1997), 'Inferring a DNA sequence from erroneous copies', *Theoretical Computer Science*, **185**(1), 3–13.

Keith, J. M., Adams, P., Bryant, D., Kroese. D. P., Mitchelson, K. R., Cochran, D. A. E. &Lala, G.H. (to appear), 'A simulated annealing algorithm for finding a consensus sequence', To appear in *Bioinformatics*.

Keith, J. M., Kroese, D. P., Bryant, D. (submitted), 'A generalised Markov sampler'.

Li, M., Ma, B., Wang, L. (2000), Near Optimal Multiple Alignment Within a Band in Polynomial Time, *in* '32nd ACM Symposium on Theory of Computing (STOC2000)', pp. 425–434.

Thompson, J. D., Higgins, D. G., Gibson, T. J. (1994), 'CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice' , *Nucleic Acids Res.*, **22**, 4673–4680.