© 1992 Oxford University Press
Nucleic Acids Research, Vol. 20, No. 11 2717–2724

Computer methods for locating kinetoplastid cryptogenes

Arndt von Haeseler[1], Beat Blum[2], Larry Simpson[2], Nancy Sturm[2] and Michael S.Waterman[1,3]*
[1]Department of Mathematics, University of Southern California, Los Angeles, CA 90089-1113, [2]Department of Biology, University of California, Los Angeles, CA 90024 and [3]Department of Molecular Biology, University of Southern California, Los Angeles, CA 90089-1340, USA

Received March 18, 1992; Revised and Accepted April 7, 1992

ABSTRACT

RNA editing in the mitochondria of kinetoplastid protoza involves the insertion and/or deletion of precise numbers of uridine residues at precise locations in the transcribed RNA of certain genes. These genes are known as cryptogenes. In this paper we study computational algorithms to search for unknown cryptogenes and for the associated templates for insertion of uridines, gRNA sequences. The pairwise similarity search algorithm of Smith and Waterman (1) is modified to study this problem. The algorithm searches for unknown gRNAs given the cryptogene sequence. The method is tested on 4 known cryptogenes from *L.tarentolae* which are known to have 7 associated gRNAs. The statistical distribution of the longest gRNA when comparing random sequences is derived. Finally we develop an algorithm to search for cryptogenes using amino acid sequences from related proteins.

INTRODUCTION

RNA editing is defined as the modification of mRNA sequences within coding regions (Simpson and Shaw (2)). The type of RNA editing found in the mitochondria of kinetoplastid protozoa involves the addition or more rarely the deletion of uridine residues at a few or multiple locations (Feagin *et al.* (3)); Shaw *et al.* (4); Shaw *et al.* (5); Benne (6); Benne *et al.* (7); Benne *et al.* (8); Van der Spek *et al.* (9); Van der Spek *et al.* (10); Feagin *et al.* (11); Stuart *et al.* (12); Bhat *et al.* (13); Koslowsky *et al.* (14)). The sequence information for these editing events is located within genes known as guide RNA (gRNA) genes, since these genes are transcribed to give rise to the small gRNAs which actually mediate the editing process (Blum *et al.* (15)). The gRNAs form short duplex anchor regions immediately downstream of the pre-edited region and these are thought to represent the initial event of editing. The remainder of the gRNA forms a perfect hybrid with the mature edited mRNA, provided $G \cdot U$ basepairs are allowed.

The simplest case of editing involves the addition of four U's at three sites and is guided by a single gRNA. The most complex involves the addition of hundreds of U's at sites throughout the length of the gene (Feagin *et al.* (16); Bhat *et al.* (13); Koslowsky *et al.* (14)). This latter type of editing is known as pan-editing (Simpson and Shaw (2)) and involves multiple gRNAs, which act sequentially 3′ to 5′ on the pre-edited mRNA.

In the kinetoplastid mitochondrion, there are two types of mitochondrial DNA molecules, maxicircles and minicircles (Simpson (17)). The maxicircles contain rRNA genes and structural genes for mitochondrial membrane proteins. gRNA genes are found in both maxicircle and minicircle DNA in *L. tarentolae* (Sturm and Simpson (18)). The maxicircle gRNA genes are scattered around the genome with no apparent relation to the pre-edited genes (Blum *et al.* (15)). Five cryptogenes have been so far identified in the maxicircle genome of *L. tarentolae* and an additional six G-rich regions exist which also contain cryptogenes. Seven maxicircle-encoded gRNAs and fifteen minicircle-encoded gRNAs have been identified. The cryptogene transcripts which are edited by several of the minicircle-encoded gRNAs are not yet known. Likewise, gRNAs for several of the cryptogenes have not yet been identified.

The goal of this paper is to describe computational algorithms to search for unknown cryptogenes and gRNA genes. It is of course an easy matter to find the gRNA genes when the mRNAs or edited cryptogenes are known. We are interested in cases where the mRNAs and the gRNAs are unknown. We have modified pairwise similarity search algorithms to model this problem on a computer. More precisely we present a dynamic programming algorithm which 'detects' possible gRNA gene sequences in the genome, with only the cryptogene given. The basic idea to discover such patterns is the free addition of '*U*' into the cryptogene to increase the number of basepairs with the gRNA gene. The algorithm is similar to the algorithm for best sub-sequence alignment as proposed by Smith and Waterman (1), and Waterman and Eggert (19). The algorithm can be extended to handle deletion of '*U*' as well.

We apply the algorithm to search for gRNA gene sequences for 4 known cryptogenes which have 7 associated maxicircle encoded gRNAs. Initially we have limited success which is improved by the imposition of rules suggested from experimental results. Candidate gRNA sequences are ranked by alignment score. Even with the modified algorithm, three of the true gRNA's are not near the highest score. For this reason, we pose

*To whom correspondence should be addressed at: Department of Mathematics, University of Southern California, Los Angeles, CA 90089-1113, USA

the question of the statistical distribution of the longest candidate gRNA when comparing random sequences. Some probability theory allows us to give a good approximation to this distribution. Finally we develop an algorithm to search for cryptogenes using amino acid sequences from related proteins.

METHODS

It is a common practice to use computer methods to find similarity between two DNA, RNA or protein sequences. Similarities are usually presented in the form of sequence alignments. See Waterman (20) for a review. The problem of sequence alignment is made difficult due to the necessity of introducing alignment of non-identical letters and the necessity of including insertions or deletions (indels) in alignments. Here we adapt a dynamic programming method known as the Smith–Waterman algorithm (1) to search for cryptogenes. gRNA genes can be aligned to cryptogenes by basepairing with the exception of inserted *U*'s. The flexibility of dynamic programming method is much more critical than issues of computer memory and time for the solution of our problem.

In 1970 Needleman and Wunsch (21) introduced the first dynamic programming algorithm for comparison of two sequences. Their algorithm finds the alignments that maximize a similarity score where identical or closely related aligned amino acids receive positive score and more distantly related amino acids receive negative score. Gaps also receive a negative score. More recently algorithms have been developed to locate high scoring segments between two sequences (1).

Clearly the effort to find gRNAs must incorporate basepairing, not identical letters as in sequence matching. General schemes to predict minimum energy RNA secondary structure based on dynamic programming have existed since Waterman (22). (See Zuker (23) for a survey of this area.) It is quite easy to see that dynamic programming methods can be applied to find gRNAs when the edited message is available. This is just what Blum *et al.* (15) did to locate the first gRNAs. It is less clear that gRNA genes can be found with knowledge of only the cryptogene. Below we will present some standard dynamic programming algorithms for sequence alignment. Then we will present new algorithms to search for potential gRNA genes that correspond to a cryptogene.

Dynamic Programming

We begin with two sequences $\mathbf{a} = a_1a_2 \ldots a_n$ *and* $\mathbf{b} = b_1b_2 \ldots b_m$. In this section we briefly describe two algorithms for sequence alignment. While this paper studies RNA sequences, the sequences could be DNA, RNA or protein. To specify the scoring function, let $s(a,b)$ be the score for aligning the letters a and b. In sequence alignment, $s(a,a)$ is often set to be 1. While one gap of length k is frequently weighted less than k gaps of length 1, it is sufficient here to consider only gaps of length 1 which receives a penalty weight of δ. In the above notation $s(-,b) = s(a,-) = -\delta$, for all a and b. A convenient way to view alignments is to insert of gaps or '$-$'s into each sequence and score the implied alignment. The sequence $\mathbf{a}(\mathbf{b})$ becomes $\mathbf{a}^* = a^*_1a^*_2 \ldots a^*_L(\mathbf{b}^* = b^*_1b^*_2 \ldots b^*_L)$ where the subsequence obtained by removing $a^*_i = -$ is the original sequence $\mathbf{a}$. The best scoring alignments have score

$$S(\mathbf{a},\mathbf{b}) = \max\left\{\sum_{i=1}^{L} s(a_i^*,b_i^*) : \text{all } a^* \text{ and } b^*\right\}.$$

The number of ways of inserting gaps into sequences grows exponentially with sequence lengths. The score can be calculated by a straightforward recursion. Let

$$S_{i,j} = S(a_1a_2 \ldots a_i, b_1b_2 \ldots b_j),\ S_{0,j} = j\delta, S_{i,0} = i\delta, S_{00} = 0.$$

Then

$$S_{i,j} = \max\{S_{i-1,j-1} + s(a_i,b_j),\ S_{i-1,j} - \delta,\ S_{i,j-1} - \delta\} \quad (1)$$

To compute $S_{n,m}$ takes computing time proportional to nm, the product of sequence lengths ($O(nm)$). Corresponding alignments can be found by a procedure called tracebacks.

The algorithm (1) was modified by Smith and Waterman (1) to find best scoring subsequence alignments. Here we wish to find 'intervals' or segments of sequence that have high scoring alignments:

$$\mathrm{H}(\mathbf{a},\mathbf{b}) = \max\{S(a_ka_{k+1} \ldots a_i, b_lb_{l+1} \ldots b_j) \mid 1 \le k \le i \le n,\ 1 \le l \le j \le m\}.$$

The algorithm to find $H(\mathbf{a},\mathbf{b})$ recursively computes

$$H_{i,j} = \max\{0;\ S(a_ka_{k+1}\} \ldots a_i,b_lb_{l+1}\} \ldots b_j): 1 \le k \le i,\ 1 \le l \le j\}.$$

Set $H_{i,j} = 0$ if $i \cdot j = 0$. Then

$$H_{ij} = \max\{H_{i-1,j-1} + s(a_i,b_j),\ H_{i-1,j} - \delta,\ H_{i,j-1} - \delta,\ 0\} \quad (2)$$

The best scoring segments satisfy

$$H(\mathbf{a},\mathbf{b}) = \max_{\substack{1 \le i \le n \\ 1 \le j \le m}} H_{ij}.$$

Again, these best scoring alignments can be obtained by tracebacks from the scoring matrix ($H_{i,j}$). This algorithm is known as a local algorithm.

gRNA algorithms

Let the cryptogene sequence be $\mathbf{a} = a_1a_2 \ldots a_n$. We will search for gRNA genes in $\mathbf{b} = b_1b_2 \ldots b_m$. $\mathbf{b}$ for example could be an entire maxicircle sequence. To be clear, both $\mathbf{a}$ and $\mathbf{b}$ will be represented as DNA sequences. This is convenient since the cryptogene is actually DNA and the potential gRNA genes reside in the mitochondrial DNA genome of the trypanosome. Therefore while the biological editing process takes place with RNA sequences, we will search DNA sequences.

The known examples of editing in the introduction make it clear that, even when the cryptogene region is known precisely, we cannot assume one gRNA gene covers the cryptogene regions of editing. Of course it is also true that gRNA genes are very small compared to the entire maxicircle. Therefore a local algorithm along the lines of Smith–Waterman is appropriate.

We will replace the similarity function $s(a,b)$ by a new function $\sigma(a,b)$ which will embody changes necessary for the new problem. First of all we wish to reward basepairs since gRNAs base pair with transcribed cryptogenes.

$$\sigma(a,b) = \begin{cases} c_1 \text{ if } (a,b) \in \{(A,T), (T,A)\} \\ c_2 \text{ if } (a,b) \in \{(G,C), (C,G)\} \\ c_3 \text{ if } (a,b) \in \{(G,T), (T,G)\} \\ c_4 \text{ otherwise.} \end{cases}$$

Here we will take $c_1 > 0$ and $c_2 > 0$. The two canonical basepairs, $A \cdot T$ and $G \cdot C$ are listed separately since they have different free energies. It is important to allow for the possibility of $c_1 \neq c_2$. In RNA structures $G \cdot U$ basepairs are common; we

take $c_3 \geq 0$. Generally we wish c_1, c_2, c_3 to reflect the free energies of base pairs. It is possible to include stacking energies but stacking energies are not implemented. In fact, we make $c_3 > 0$ but small relative to c_1 and c_2. For non-basepairs, it is possible to allow for non-pairing, but they should be accounted for in bulge or interior loop functions. Therefore $c_4 = -\infty$ is usually our choice.

Recall that we want to find good scoring regions between **a** and **b** with possible *U*'s (*T*'s in our DNA version) inserted into the sequence **a**. If we allow no bulges or interior loop in our structure, this reduces to

$$\sigma(-,A) = \sigma(-,G) = c_5 \geq -\infty,$$

and

$$\sigma(-,T) = \sigma(-,C) = -\infty.$$

Of course

$$\sigma(a,-) = -\infty \text{ for all } a.$$

In our implementation, $c_3 = \sigma(G,T) > 0$ while $c_5 = 0$. This allows us to prefer $G \cdot T$ basepairs over the insertion of *T*'s to form $T \cdot G$ *or* $T \cdot A$ basepairs. It will be seen in the next section that it is necessary to introduce heuristic rules suggested by the biology to limit the candidate gRNA genes.

Editing can also includes the deletion of *U*'s in RNA, so that it is of some interest to devise an algorithm to include these events. All that is required is:

$$\sigma(T,-) = c_6 \geq -\infty,$$

and

$$\sigma(a,-) = -\infty \text{ all } a \neq T.$$

Finally, we describe the method of traceback used to produce candidate gRNA genes. We begin by choosing the largest score. Whenever there is a tie between inserting a *T* or making a basepair in the traceback, we always take the basepair. Multiple alternate gRNA candidates could be produced but we only output one gRNA. In addition, if two tracebacks require the same basepair, we output only the one alignment with largest score. An iterative recomputation scheme along the lines of Waterman and Eggert (19) could be implemented to produce the *k*-best candidate gRNA genes that do not share any basepairs in common, but we have not found this to be necessary. The tracebacks are generally unambiguous.

Rules

As will be detailed in Results, the gRNA algorithm presented above does not place known gRNAs among the highest scoring potential gRNAs. To improve this situation, we studied the known gRNAs located in *L.tarentolae* maxicircles and abstracted 6 rules to eliminate spurious candidates for gRNAs. In Results we will discuss these rules in more detail, but simply list them here without further discussion.

Rule 1. There must be at least 5 basepairs at the 3′ end of the cryptogene region, including $G \cdot U$ basepairs. We refer to this basepaired region as the *anchor*.

Rule 2. There can be no more than 3 $G \cdot U$ pairs in the anchor.

Rule 3. There must be at least 4 contiguous Watson–Crick basepairs in the anchor.

Rule 4. No more than 8 contiguous *U*'s can be inserted.

Rule 5. There are no more than 3 basepairs between inserted *U*'s.

Rule 6. There must be at least 1 basepair at the 5′ end of a cryptogene.

It is necessary to present our rules at this point since we next turn to the statistical distribution of gRNA gene lengths in random DNA sequences, motivated by these rules.

Statistical distribution of random gRNA gene lengths

We model our sequences as independent, identically distributed nucleotides. Let $\mathbf{b} = b_1 b_2 \ldots b_n$ be the maxicircle, or the DNA to be searched for the presence of gRNA genes. Set $P(b_j = \alpha) = q_\alpha$, $\alpha \in \{A,T,G,C\}$. Similarly let $\mathbf{a} = a_1 a_2 \ldots a_m$ be the cryptogene region, with $P(a_i = \alpha) = p_\alpha$, $\alpha \in \{A,T,G,C\}$. We model the editing process with three steps: (1) we more precisely define anchor, modifying Rules 1–3 somewhat and calculate the probability of this event; (2) we describe the 5 ways to enter editing from an anchor; (3) finally we give the matrix of transition probabilities between the 8 editing states. Rule 4 would unduly complicate the analysis and is not included.

Recall that an anchor must begin with a mispair, must have at least 5 consecutive basepairs, can have no more than 3 $G \cdot U$ basepairs, and must have at least 4 Watson–Crick consecutive basepairs. Let $p_1 = p_A q_T + p_T q_A + p_C q_G + p_G q_C$ be the probability a Watson–Crick pair and $p_2 = p_G q_U + p_U q_G$ be the probability of a $G \cdot U$ pair. Therefore the probability of a mispair is

$$P(\text{mispair}) = 1 - p_1 - p_2 \tag{3}$$

Let $L_a = l$ be the anchor length, that is the number of basepairs in the anchor. The event that $L_a = l$ can be decomposed into 4 disjoint events: $L_a = l$ with k $G \cdot U$ pairs, $0 \leq k \leq 3$. The probability of the first of these events is elementary to compute since it is equivalent to coin tossing with success probability p_1:

$$P(L_a \geq l \text{ and } k = 0) = (1 - p_1 - p_2)\, p_1{}^l$$

The remaining cases are easily derived but the formulas are cumbersome. However, for $16 \leq l$,

$$P(L_a \geq l) = (1 - p_1 - p_2) \sum_{k=0}^{3} \binom{l}{k} p_1{}^{l-k}\, p_2{}^k.$$

For computational convenience, we model the anchor as requiring at least 5 basepairs of either type. Therefore

$$P(L_a \geq l) \approx (1 - p_1 - p_2)\,(p_1 + p_2)^l,$$

so that

$$P(L_a = l) \approx (1 - p_1 - p_2)^2 (p_1 + p_2)^l$$

Having calculated the anchor probabilities we now turn to the transition from anchor to editing. We visualize the cryptogene written above the gRNA gene. While insertion events take place in RNA, it is very convenient to represent insertion events in the genes. Therefore we will show 'inserted *U*'s' by 'inserted lowercase *t*'s'. The anchor can only be terminated by an editing event, which is generated by a non-basepair of the following type:

cryptogene	$A\cdots$	$A\cdots$	$G\cdots$	$G\cdots$	$C\cdots$
gRNA gene	$A?\cdots$	$G?\cdots$	$A?\cdots$	$G?\cdots$	$A?\cdots$

After 'editing' these sequences become

$tA\cdots$	$tA\cdots$	$tG\cdots$	$tG\cdots$	$tC\cdots$
$A?\cdots$	$G?\cdots$	$A?\cdots$	$G?\cdots$	$A?\cdots$

Table 1. The transition matrix M

	1	2	3	4	5	6	7	8
1	q_A	q_G	0	0	0	q_T	0	0
2	q_A	q_G	0	0	0	q_T	0	0
3	0	0	q_A	q_G	0	$q_T + q_C$	0	0
4	0	0	q_A	q_G	0	$q_T + q_C$	0	0
5	0	0	0	0	q_A	q_G	0	0
6	p_Aq_A	p_Aq_G	p_Gq_A	p_Gq_G	p_Cq_A	0	$p_1 + p_2$	0
7	p_Aq_A	p_Aq_G	p_Gq_A	p_Gq_G	p_Cq_A	0	0	$p_1 + p_2$
8	p_Aq_A	p_Aq_G	p_Gq_A	p_Gq_G	p_Cq_A	0	0	0

Therefore we use states 1 to 5 to denote the occurrence of $\overset{A}{\cdot}$, $\overset{A}{G}$, $\overset{G}{\cdot}$, $\overset{G}{C}$, and $\overset{C}{\cdot}$. The probability of occurrence of states 1 to 5 can easily be calculated: $P(\text{State 1}) = p_Aq_A$, $P(\text{State 2}) = p_Aq_G$, $P(\text{State 3}) = p_Gq_A$, $P(\text{State 4}) = p_Gq_G$, $P(\text{State 5}) = p_Cq_A$. These states induce editing and the shift of the cryptogene sequence one position to the right. Of course this is illustrated above. Our next task is to find the transition probabilities of moving from state i to state j:

$$p_{ij} = P(\text{State } j | \text{State } i).$$

When we perform an editing operation (states 1 to 5) we shift an A, G, or C to the right. At this point, we observe another nucleotide from the gRNA gene sequence. To continue matching, we need to create a basepair or to induce another editing state. If a basepair is created, we can then create another basepair or another editing state. Recall that Rule 5 requires there be no more than 3 consecutive basepairs after the anchor. Therefore we track the number of consecutive basepairs by states 6, 7, and 8. The transition matrix $M = (p_{i,j})$ for this study is presented in Table 1.

We must now study the length L_e of the edited portion of the random gRNA gene. The powers of M, M^k, give the values M^k_{ij} = probability of moving from state i to state j in k steps. Since we wish to find the maximum of a large number of such gRNA gene lengths, it is sufficient to consider M^k for large values of k. We employ the Perron–Frobenius theory of positive matrices; this theory applies since M^k is positive for $k \geq 4$. See (24) for a discussion of this theory. Then the largest eigenvalue λ of M is positive and the left ($\mathbf{l} = (l_1, l_2, \ldots, l_8)$) and right ($\mathbf{r} = (r_1, r_2, \ldots, r_8)$) eigenvectors also can be chosen to be positive. Normalize them so that

$$\sum_{i=1}^{8} l_i r_i = 1$$

Then Perron–Frobenius theory implies

$$M^k_{i,j} \approx \lambda^k r_i l_j$$

Next let $S = \{1,2,\ldots,8\}$ be our set of states. The task is to approximate the distribution of the number of steps L_e from entry to exit from S. Entry is from states $1 \leq i \leq 5$ while Rule 6 implies exit is from $6 \leq j \leq 8$. We need

$$p(L_e = k | \text{begin at } i \in S) = \sum_{j=6}^{8} M^k_{ij} \left(1 - \sum_{s=1}^{8} M_{i,s}\right)$$

$$\approx \sum_{j=6}^{8} \lambda^k r_i l_j \left(1 - \sum_{s=1}^{8} M^k_{j,s}\right).$$

Then

$$P(\text{Enter } S \text{ and } L_e = k) \sum_{i=1}^{5} P(\text{State } i)\, P(L_e = k | \text{begin at } i \in S)$$

$$\approx \lambda^k \left(\sum_{i=1}^{5} r_i P(\text{State } i)\right)\left(\sum_{j=6}^{8} l_j \left(1 - \sum_{s=1}^{8} M_{j,s}\right)\right)$$

$$= c\lambda^k.$$

Recall that $P(\text{State } i) > 0$ only for $1 \leq i \leq 5$.

Let Z = the random gRNA gene length, L_a = random anchor length, and L_e = random editing region length. Earlier we computed $P(L_a = l)$. Clearly $Z = L_a + L_e$, and

$$P(Z = t) = P(L_a + L_e\, t)$$

$$= \sum_{l=5}^{t} P(L_a = l) P(L_e = t - l) \qquad (4)$$

The dominant feature of (4) is that $P(Z = t)$ will look like a power of λ or of $(p_1 + p_2)$, depending on whichever is larger. It is then possible to obtain detailed information about the maximum value of Z. The approximate expected maximum can be obtained by solving

$$nmP(Z = t) = 1$$

for t. For an estimation of statistical significance, not utilized in this paper, it is possible to apply the Chen–Stein method (Arratia *et al.* [25]).

This section has presented an analysis of the maximum length of edited regions for random DNA. A more difficult analysis for general scoring schemes could be based on work of Arratia *et al.* (26) and Karlin and Altschul (27) where the distribution of the best scoring segment is derived. Those analyses do not include insertions and deletions but if editing were restricted to one sequence we believe that a rigorous analysis is possible. Our conclusions below do not require this more difficult and sophisticated mathematics.

Protein sequences from other organisms

The approach we take is based on the observation that if a segment of cryptogene sequence $\mathbf{a} = a_1a_2 \cdots a_n$ encodes a segment of a homologous protein $\mathbf{y} = y_1y_2 \cdots y_m$, the edited cryptogene should be closely related to $\mathbf{y}$. A similarity function such as one of the Dayhoff matrices or the minimum mutation matrix is used in the following way. Let $s(a_ia_{i+1}a_{i+2}, y)$ be the similarity between the amino acid encoded by the triplet $a_ia_{i+1}a_{i+2}$ and the amino acid y. Define H_{ij} to be the best score of any segment of $\mathbf{y}$ ending at y_j, or any edited segment of $\mathbf{a}$ ending at a_i, or 0, whichever is larger. As in the algorithm in Section 2.2, we will not charge for the insertion of U's (T's).

$$\begin{aligned} H_{ij} = max\, \{&0, H_{i-1,j} - \delta_N, H_{ij-1} - \delta_A, \\ &H_{i-3,j-1} + s(a_{i-2}a_{i-1}a_i, y_j), \\ &H_{i-2,j-1} + s(ta_{i-1}a_i, y_j), \\ &H_{i-2,j-1} + s(a_{i-1}ta_i, y_j), \\ &H_{i-2,j-1} + s(a_{i-1}a_it, y_j), \\ &H_{i-1,j-1} + s(tta_i, y_j), \\ &H_{i-1,j-1} + s(ta_it, y_j), \\ &H_{i-1,j-1} + s(a_itt, y_j), \\ &H_{i,j-1} + s(ttt, y_j)\}. \end{aligned}$$

Here we have δ_N and δ_A, the cost of deleting a nucleotide or amino acid, respectively. Normally in protein/DNA sequence

	A	A	A	A	T	A	A	G	G	A	A	A	G	A	G	A	G	G	C	G	A	A	A	A	T
T	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0
T	1	2	2	2	0	1	2	2	2	2	2	2	2	2	2	2	2	2	0	1	2	2	2	2	0
G	1	2	2	2	3	1	2	2	2	2	2	2	2	2	2	2	2	2	3	1	2	2	2	2	3
G	1	2	2	2	3	1	2	2	2	2	2	2	2	2	2	2	2	2	3	1	2	2	2	2	3
T	1	2	3	3	0	4	2	3	3	3	3	3	3	3	3	3	3	3	0	4	2	3	3	3	0
T	1	2	3	4	0	1	5	3	4	4	4	4	4	4	4	4	4	4	0	1	5	3	4	4	0
T	1	2	3	4	0	1	2	6	4	5	5	5	5	5	5	5	5	5	0	1	2	6	4	5	0
C	0	0	0	0	0	0	0	3	7	0	0	0	6	0	6	0	6	6	0	1	0	0	0	0	0
C	0	0	0	0	0	0	0	1	4	0	0	0	1	0	1	0	1	7	0	1	0	0	0	0	0
A	0	0	0	0	1	0	0	1	4	0	0	0	1	0	1	0	1	7	0	1	0	0	0	0	1
A	0	0	0	0	1	0	0	1	4	0	0	0	1	0	1	0	1	7	0	1	0	0	0	0	1
A	0	0	0	0	1	0	0	1	4	0	0	0	1	0	1	0	1	7	0	1	0	0	0	0	1
T	1	1	1	1	0	2	1	1	2	5	1	1	1	2	1	2	1	2	0	1	2	1	1	1	0
T	1	2	2	2	0	1	3	2	2	3	6	2	2	2	3	2	3	2	0	1	2	3	2	2	0
G	1	2	2	2	3	1	3	2	2	3	6	2	2	2	3	2	3	2	3	1	2	3	2	2	3
T	1	2	3	3	0	4	2	4	3	3	4	7	3	3	3	4	3	4	0	4	2	3	4	3	0
A	1	2	3	3	4	4	2	4	3	3	4	7	3	3	3	4	3	4	0	4	2	3	4	3	4
C	0	0	0	0	0	0	0	3	5	0	0	0	8	0	4	0	5	4	0	1	0	0	0	0	0
T	1	1	1	1	0	1	1	1	4	6	1	1	1	9	1	5	1	6	0	1	2	1	1	1	0
C	0	0	0	0	0	0	0	2	2	0	0	0	2	0	10	0	6	2	0	1	0	0	0	0	0
T	1	1	1	1	0	1	1	1	3	3	1	1	1	3	1	11	1	7	0	1	2	1	1	1	0
T	1	2	2	2	0	1	2	2	2	4	4	2	2	2	4	2	12	2	0	1	2	3	2	2	0
T	1	2	3	3	0	1	2	3	3	3	5	5	3	3	3	5	3	13	0	1	2	3	4	3	0
T	1	2	3	4	0	1	2	3	4	4	4	6	6	4	4	4	6	4	0	1	2	3	4	5	0
A	1	2	3	4	5	1	2	3	4	4	4	6	6	4	4	4	6	4	0	1	2	3	4	5	6
T	1	2	3	4	0	6	2	3	4	5	5	5	7	7	5	5	5	7	0	1	2	3	4	5	0
A	1	2	3	4	5	6	2	3	4	5	5	5	7	7	5	5	5	7	0	1	2	3	4	5	6
T	1	2	3	4	0	6	7	3	4	5	6	6	6	8	8	6	6	6	0	1	2	3	4	5	0

Figure 1. Cryptogene (horizontal sequence) and potential gRNA (vertical sequence) alignment.

comparison it is desirable to delete nucleotides in groups of 3 to stay in frame. With RNA editing, a single nucleotide deletion might correspond to at least a three letter deletion in the edited cryptogene.

RESULTS

We first illustrate our algorithms by a small example with cryptogene $\mathbf{a} = AAAATAAGGAAAGAGAGGCGAAAAT$ (5′ to 3′) and potential guide sequences in $\mathbf{b} = TTGGGTTCCAAATTGTACTCTTTTATAT$ (3′ to 5′, compliment). For this illustrative example we choose the naive weights where $\sigma(a,b) = 1$ if a and b form a basepair (including $G \cdot U$) and $-\infty$ otherwise. $\sigma(-,A) = \sigma(-,G) = 0$, and $-\infty$ for all other cases involving '−'. The matrix H appears as Figure 1. The largest entry is 13 and the traceback yields the following alignment where inserted T's are denoted by t:

```
A A G G t t t A A t A t G A G A G G
T T C C A A A T T G T A C T C T T T
```

Application to known cryptogenes

Now that we have established our model, it is necessary to test its predictive power. Our ultimate goal is to find new and unknown cryptogene/gRNA gene pairs. Before that is possible we must be able to locate gRNA gene sequences from known cryptogene sequences. In this section we describe a first step of that process: searching the *Leishmania tarentolae* maxicircle DNA for gRNA genes that correspond to parts of the cryptogenes Cytochrome b, Murf2, ND7, and COII. The gRNA gene for COII appears immediately 3′ of the COII gene. Therefore we expect this gRNA to have atypical properties. In all instances we search with cryptogenes where gRNA sequences have been found experimentally (Blum *et al.* (15)).

With a fixed scoring scheme, it is a statistical problem to determine how many genomic segments will score at or above a given cutoff when compared with the cryptogene. The cutoff score can be determined by the score of the known gRNA gene. Our implicit assumption in deriving the gRNA algorithms is that real gRNAs will score noticeably better than other genomic sequence. With our naive scoring methods this assumption turns out to be unfounded. We derived some statistical results that relate to the length of the longest match using our algorithm. Here length is the simplest possible score. We will discuss the statistics in detail below, but in most cases there are a large number of gRNA-like matches expected from random sequence comparisons.

Table 2 gives the ranking by score of each gRNA gene in the list of candidate gRNA genes produced by our algorithm. The gRNA gene names given are those appearing in the literature. Since many potential gRNAs have the same score, the ranking of the gRNAs are ambiguous. The length in Table 2 is defined to be the length of the longest maxicircle segment that is basepaired with the cryptogene (with inserted U's). The matches are scored with the following parameters

$$\sigma(a,b) = \begin{cases} 500, \text{ if } (a,b) \in \{(A,T),(T,A)\} \\ 1000, \text{ if } (a,b) \in \{(G,C),(C,G)\} \\ 1, \text{ if } (a,b) \in \{(G,T),(T,G)\} \\ -\infty, 0 \text{ otherwise}, \end{cases}$$

$$\sigma(-,A) = \sigma(-,G) = 0,$$

$$\sigma(-,T) = \sigma(-,C) = -\infty,$$

and

$$\sigma(a,-) = -\infty \text{ for all } a \in \{A,G,C,T\}.$$

The results are quite robust regarding the values of $\sigma(\cdot,\cdot)$, except that we feel it is important to score $G \cdot C$ basepairs more strongly than $A \cdot T$ base pairs. For example, scoring by $\sigma(A,T) = 1$, $\sigma(G,C) = 3$, and $\sigma(G,T) = 1$ does not change the results of the search.

Except for the ND7/gRNA-5′ match, ranking by score or by length is not useful. Since we have experimental examples of gRNAs, we examined them for common patterns that might be of assistance in our computational approaches. Artificial intelligence methods or consensus sequence methods are unlikely to yield significant insights into this problem. The reason for this is biological: each of the many known gRNAs basepair with a different portion of a nucleic acid coding region for different proteins. Therefore universal conservation of patterns of letters in cryptogene/gRNA gene matches seem unlikely. The heuristic rules we have derived are all in terms of the number of bases or number of basepairs. We feel these patterns are associated with the editing machinery. These patterns were obtained from the 7 known maxicircle gRNAs for *L. tarentolae*. In figure 2 data for the number of contiguous U's inserted for maxicircles and for both maxicircles and minicircles. Except for one 9 letter insertion in a minicircle gRNA, all regions are less than or equal to 8. Figure 3 presents the maxicircle data for the number of basepairs between inserted U's. In maxicircle gRNAs, this number never exceeds 3. This number 3 is often exceeded in minicircles however and we require 3 or less in our Rule 5.

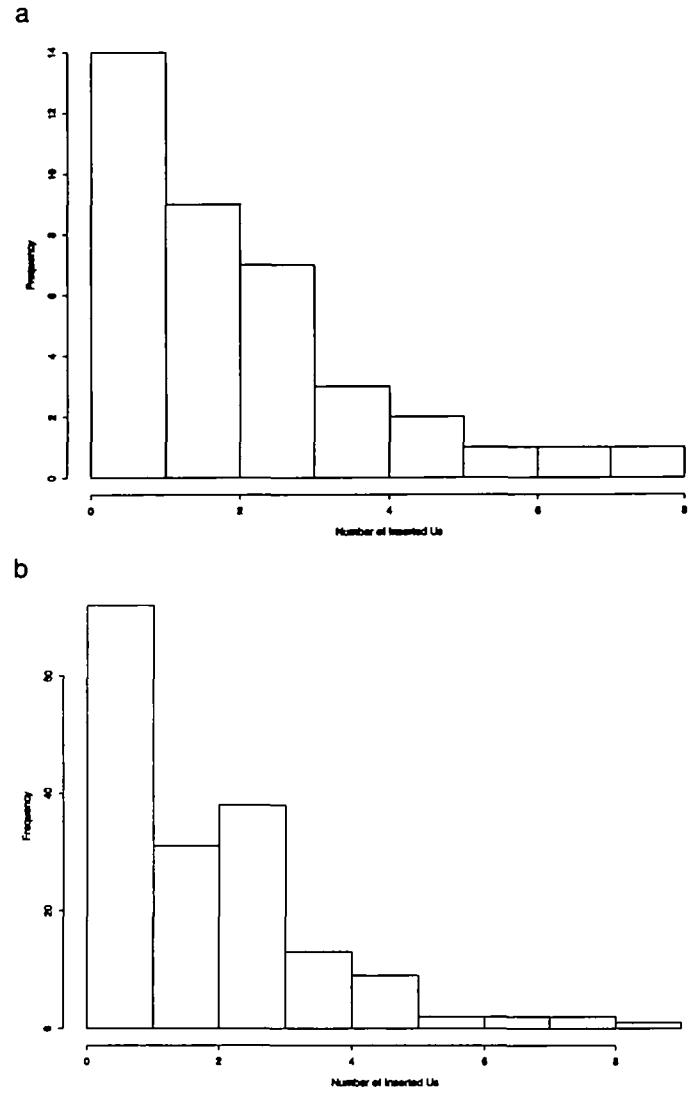

Figure 2. Frequency histograms of inserted Us. Figure 2a is the data for maxicircle gRNAs and figure 2b is the data for maxicircle gRNAs.

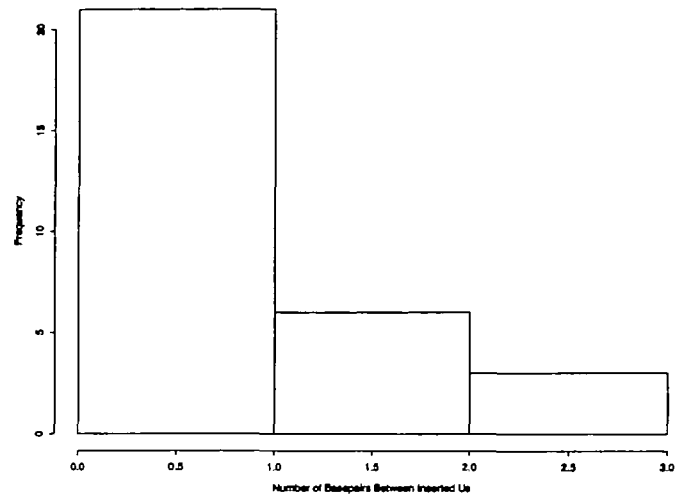

Figure 3. Frequency histogram of the number of basepairs between inserted Us for maxicircle gRNAs.

Relaxation of Rule 5 would make the real maxicircle gRNAs even lower in our list. Our 6 rules are as given below. The rules describe properties imposed on the region of the cryptogene that is basepaired with a gRNA.

Rule 1. There must be at least 5 basepairs at the 3′ end of the cryptogene region, including $G \cdot U$ basepairs. We refer to this basepaired region as the *anchor*.

Rule 2. There can be no more than 3 $G \cdot U$ pairs in the anchor.

Rule 3. There must be at least 4 contiguous Watson–Crick basepairs in the anchor.

Rule 4. No more than 8 contiguous *U*'s can be inserted.

Rule 5. There are no more than 3 basepairs between inserted *U*'s.

Rule 6. There must be at least 1 basepair at the 5's end of a cryptogene.

Recall that editing proceeds 3′ to 5′ along the cryptogene. Rules 1–3 all concern the 3′ anchor region, which we expect is necessary to stabilize the interaction between cryptogene and gRNA and thereby allow the editing machinery to proceed. Rules 5 and 6 are also stability rules at the 5′ end of the cryptogene and we think of these as stopping rules. The remaining Rule 4 restricts the number of contiguously inserted *U*'s during the editing process. Table 3 repeats the analysis of cryptogenes from *L. tarentolae* with known gRNAs with application of Rules 1–6.

Statistical distribution of gRNA gene lengths

In Tables 2 and 3 we presented two different rankings of 7 gRNA gene scores when the entire *L. tarentolae* maxicircle is searched with 4 cryptogenes. Without our rules applied to the search, all but one of the 7 gRNA genes fall far below the highest scoring regions of the maxicircle. Even with the rules applied, two of the gRNA genes rank first, one ranks third, another eighth, while the remaining ranks 18th and below. This leads us to pose the question of the statistical distribution of the longest candidate edited region or, using the language from the statistical section, the longest matched region. To obtain this distribution, we first model the editing process as a Markov chain. Applying results from Perron–Frobenius theory, we can obtain the approximate distribution of the length of the candidate edited regions. Finally, we use these results to obtain the statistical distribution of the length of longest candidate edited region.

In the known *L. tarentolae* maxicircle sequence of $n = 20{,}993$ basepairs, $p_A = q_A = 0.3835$, $p_G = q_G = 0.1160$, $p_C = q_C = 0.0949$, and $p_T = q_T = 0.4056$. This yields $p_1 = 0.3331$, $p_2 = 0.0941$, $c = 0.0154$, and $\lambda = 0.7666$. Summing the series (2.4), we obtain

$$P(Z = t + 1) = 0.0005\ (0.7666)^t$$

We return to the example of Table 2. There we compared cryptogene regions of length 100 with both strands of the maxicircle (of length $2 \times 20{,}993$). The goal here is to give the expected length of the best Z from all the possible Z's encountered in such a search. Since there should be 1 'longest Z' and since there are (100) $(2 \times 20{,}993)$ ways to begin Z's, we solve

$$(100)\ (2 \times 20{,}993) P(Z \geq t) = 1$$

for t. This gives

$$(4.1986 \times 10^6)\ (0.0021)\ (0.7666)^t = 1$$

Table 2. Ranking by score of known gRNAs when searching the *L.tarentolae* maxicircle with 4 cryptogenes

Cryptogene	gRNA	Length	Position in the list of suboptimal gRNA
cytochrome b	gRNAI	32	27–30
	gRNAII	54	36–43
Murf2	gRNAI	14	271–306
	gRNAII	46	1286–1327
ND7	gRNA-5′	44	2
	gRNA-FS	19	92–93
COII	gRNA-FS	16	88–100

Table 3. Ranking by score of known gRNAs when searching the *L.tarentolae* maxicircle with 4 cryptogenes. Rules 1–6 have been applied.

Cryptogene	gRNA	ranking with Rule 1–6 applied
Cytochrome b	gRNAI	3
	gRNAII	1
Murf2	gRNAI	78–84
	gRNAII	18–21
ND7	gRNA-5′	1
	gRNA-FS	8
CO-II	gRNA-FS	86–99

or $t = 34.18$. Therefore, the average or expected length of the longest Z is about $t \approx 35$. If the cryptogene is localized to a region of length 1000 rather than 100, we then obtain $t \approx 43$. This is quite consistent with Tables 2 and 3, except for gRNAI of Murf2 (which has an unusual base composition). Tables 2 and 3 of course reflect the scoring schemes of equation (5), while we have studied the simpler score of length in this section.

Using protein sequences from other organisms

In this section protein sequences are used to motivate other dynamic programming algorithms for locating cryptogenes. Once the sequence of an edited cryptogene is known, there are two obvious directions to proceed. First gRNA gene sequences can be searched for that complement edited regions of the cryptogenes. Second the translated protein sequence can be used to search a protein data base. Thus it is possible to infer the protein family in cases where homologies are discovered. In the case of Murf4, Bhat *et al.* (13) have argued by sequence comparison that the cryptogene encodes subunit 6 of ATPase. Instead of pursuing such problems where standard algorithms already exist, we are interested here in other questions. If a potential cryptogene does not have a known edited RNA, we wish to identify potential homologous proteins encoded by the cryptogene. On the other hand if a protein family should have a gene in kinetopastid DNA but the gene is not known, it should be possible to search for potential cryptogenes.

To test the algorithm presented for protein sequences, we chose the cytochrome-b cryptogene as our target. The related protein is cytochrome-b of the mitochondria of yeast (GenBank accession no. J01476). We chose the first 40 amino acids of the yeast protein because portions of the corresponding region of cytochrome-b in *L.tarentolae* are edited. The Dayhoff PAM250 matrix is chosen, with deletion penalties $\delta_A = 12$ and $\delta_N = 4$. The penalty for aligning stop codons with an amino acid is $-\infty$. We searched bases 1–15,000 in the *L.tarentolae* maxicircle genome. The maximum score is 182, which comes from aligning a 73 base region of *L.tarentolae* to all 40 amino acids from the yeast protein. The region of the maxicircle known to be edited to encode cytochrome-b of *L.tarentolae* is not among the first 191 top scores. (There are 191 regions scoring ≥ 160.)

To further explore the utility of the algorithm, we compared 71 bases of the cryptogene with the same 40 amino acid sequence from yeast. The highest scoring alignment was 130, but unfortunately this alignment does not imply the editing known by experiment to be correct. The first 15 alignments have the same property.

The above algorithm does not exhaust the comparisons possible between protein and kinetoplastid DNA, since potential gRNA genes are not been incorporated into the algorithm. One way to remedy this is to create an algorithm to simultaneously compare protein sequences, potential cryptogenes, and potential gRNA genes. Of course, the computational complexity is the product of the three sequence lengths, and another critical problem is the incomplete nature of kinetoplastid DNA sequences. The problem could be approached by pairwise comparisons to reduce computational complexity but there are further difficulties. Because of sequence evolution, the result of the above protein/cryptogene comparison is likely to be distinct from the edited cryptogene, thereby compounding the already difficult search for gRNA genes. If gRNA/cryptogene searches are made first, problems of incomplete data and statistical problems arise.

DISCUSSION

Our goal was to derive new computational algorithms to search for unknown cryptogenes and gRNA genes. Dynamic programming algorithms have been described that apply to these problems. Our so-called gRNA algorithms search for possible gRNA genes that are complementary to candidate cryptogenes. We imposed Rules 1–6, derived from known gRNA/cryptogene relationships, to limit the spurious matches. In addition, we presented an algorithm that matches a segment of a cryptogene with a segment of a homologous protein. To obtain the possible cryptogene product, we translate the cryptogene sequence edited in all possible ways with T's. As discussed above it is possible (but very computationally expensive) to bring potential gRNA sequences into a 3-way comparison. In summary, the methods presented in this paper illustrate the flexibility and versatility of dynamic programming algorithms. While retaining the rigorous foundations, the appropriate searches can be designed to fit the biological problem.

Unfortunately known gRNA sequences do not match cryptogenes with scores that are unusual when compared with scores from a maxicircle of random DNA. Therefore our algorithms might be useful but not now essential for unraveling the complexities of RNA editing. In table 2, we show the ranking by score of known gRNA genes when we search the maxicircle of *L.tarentolae*. Even after Rules 1–6 are applied the rankings of 3 out of the 7 known gRNA genes fall below 10 in the ranking. This is empirical evidence that these scores are being obscured by the statistical distribution of scores from unrelated genomic DNA. To study this situation more carefully, we model the genome by independent nucleotides and the editing process by a Markov chain. From this model we learn the expected longest match in random genomic gRNA is consistent with the lengths observed in our real gRNA sequences. The behavior of this expected longest match is captured in a simple formula and explains the results of *L.tarentolae* comparisons.

Are there any ways to greatly improve the situation? Improvements in the scoring schemes for matching are unlikely to help much. We have studied a simple scheme that counts basepairs as well as one where realistic free energies are assigned. Neither of these performs very differently. Real gRNAs are in or very close to the statistical noise. Note that a single sequencing error can cause our algorithm to fail. However if we relax the stringent conditions now required for a 'match' the real gRNAs would be even more obscured by the random matches. This leads to the natural question of how the biological system finds the correct gRNAs. The answer seems to be an abundance of gRNA molecules that dominates the population of transcribed RNA molecules. If we could recognize these promoters of abundantly transcribed genes, we could of course limit our search space and consequently the expected lengths of 'random gRNAs' would be greatly decreased. This would allow the sample size to be appropriate to the biology. Unfortunately we do not know reliably how to recognize promoters in genomic DNA.

The last paragraph suggested promoter recognition as one approach to improving the results of our search methods. Other possibilities include codon usage, comparisons between kinetoplastid genomes, and application to genomes such as *T. brucei*. In general we feel it is important to develop computational search techniques for novel biological features that are difficult to identify, such as RNA editing.

ACKNOWLEDGEMENTS

We are grateful to Richard Arratia of USC for valuable discussion about Perron–Frobeius theory. This research was supported by Deutsche Forschungsgemeinschaft (A.v.H.), the National Science Foundation (A.v.H., M.S.W), and the National Institutes of Health (B.B., A.v.H., L.S., N.S. and M.S.W.).

REFERENCES

1. Smith,T.F. and Waterman,M.S (1981) *J. Mol. Biol.* **147**, 195–197.
2. Simpson,L. and Shaw,J. (1989) *Cell* **57**, 355–366.
3. Feagin,J.E., Abraham,J. and Stuart,K. (1988) *Cell* **53**, 413–422.
4. Shaw,J., Campbell,D. and Simpson,L. (1989) *Proc. Natl. Acad. Sci.* **86**, 6220–6224.
5. Shaw,J., Feagin,J.E., Stuart,K. and Simpson,L. (1988) *Cell* **53**, 401–411.
6. Benne,R. (1989) *Biochem. Biophys. Acta* **1007**, 131–139.
7. Benne,R., DeVries,B., Van den Burg,J. and Klaver,B. (1983) *Nucleic Acids Res.* **11**, 6925–6941.
8. Benne,R., Van den Burg,J., Brakenhoff,J., Sloof,P., Van Boom,J. and Tromp,M. (1986) *Cell* **46**, 819–826.
9. Van der Spek,H., Speijer,D., Arts,G.-J., Van den Burg,J., Van Steeg,H., Sloof,P. and Benne, R. (1990) *EMBO J.* **9**, 257–262.
10. Van der Spek,H., Van den Burg,J., Croiset,A., Van den Broek,M., Sloof,P. and Benne,R. (1988) *EMBO J.* **7**, 2509–2514.
11. Feagin,J., Jasmer,D. and Stuart,K. (1987) *Cell* **49**, 337–345.
12. Stuart,K., Feagin,J. and Abraham,J. (1989) *Gene* **82**, 155–160.
13. Bhat,G.J., Koslowsky,D.J., Feagin,J.E., Smiley,B.L. and Stuart,K. (1990) *Cell* **61**, 885–894.
14. Koslowsky,D.J., Bhat,G.J., Perrollaz,A.L., Feagin,J.E., and Stuart,K. (1990) *Cell* **62**, 901–911.
15. Blum,B., Bakalara,N. and Simpson,L. (1990) *Cell* **60**, 189–198.
16. Feagin,J.E., Shaw,J.M., Simpson,L. and Stuart,K. (1988) *Proc. Natl. Acad. Sci. USA* **85**, 539–543.
17. Simpson,L. (1987) *Annu. Rev. Microbiol.* **41**, 363–382.
18. Sturm,N.R. and Simpson,L. (1990) *Cell* **61**, 879–884.
19. Waterman,M.S. and Eggert,M. (1987) *J. Mol. Biol.* **197**, 723–728.
20. Waterman,M.S. (1984) *Bull. Math. Biol.* **46**, 473–500.
21. Needleman, S. B. and Wunsch, C. D. (1970) *J. Mol. Biol.* **48**, 443–453.
22. Waterman,M.S. (1978) *Studies in Foundations and Combinatorics — Advances in Mathematics Supplementary Studies* 1, 167–212.
23. Zuker,M. (1989) Waterman,M.S. (1989) In Waterman,M.S. (ed.), *Mathematical methods for DNA sequences*, CRC Press, Boca Raton, FL, pp. 53–92.
24. Karlin,S. and Taylor,H.M. (1975) A First Course in Stochastic Processes (2nd Edition). Academic Press, New York.
25. Arratia,R.A., Goldstein,L. and Gordon,L. (1989) *Ann. Prob.* **17**, 9–25.
26. Arratia,R.A., Morris,P. and Waterman,M.S. (1988) *J. Appl. Prob.* **25**, 106–119.
27. Karlin,S. and Altschul,S. (1990) *Proc. Natl. Acad. Sci. USA* **87**, 2264–2268.