Computer methods for locating kinetoplastid cryptogenes

Arndt von Haeseler¹, Beat Blum², Larry Simpson², Nancy Sturm² and Michael S. Waterman¹,³*

¹Department of Mathematics, University of Southern California, Los Angeles, CA 90089-1113,
²Department of Biology, University of California, Los Angeles, CA 90024 and ³Department 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 protozoa 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·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 a = a1a2... an and b = b1b2... bm. 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 5. In the above notation s(-,b) = s(a,-) = -δ, for all a and b. A convenient way to view alignments is to insert gaps or ‘-‘s into each sequence and score the implied alignment. The sequence a(b) becomes a* = a*1a*2... a*l(b*1b*2... b*m) where the subsequence obtained by removing a*1 = − is the original sequence a. The best scoring alignments have score

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

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

\[ S_{ij} = S(a_1a_2... a_i, b_1b_2... b_j), S_{0j} = j\delta, S_{i0} = 0. \]

Then

\[ S_{ij} = \max\{S_{i-1,j-1} + s(a_i,b_j), S_{i-1,j} - \delta, S_{i,j-1} - \delta\} \]

(1)

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

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:

\[ H(a,b) = \max\{S(a_1a_2... a_i, b_1b_2... b_j) : 1 \leq k \leq i, n \leq l \leq j\}. \]

The algorithm to find \( H(a,b) \) recursively computes

\[ H_{i,j} = \max\{0, S(a_1a_2... a_i, b_1b_2... b_j) \} : 1 \leq k \leq i, 1 \leq l \leq j\].

Set \( H_{ij} = 0 \) if \( i \cdot j = 0 \). Then

\[ H_y = \max\{H_{i-1,j-1} + s(a_i,b_j), H_{i-1,j} - \delta, H_{i,j-1} - \delta, 0\}. \]

(2)

The best scoring segments satisfy

\[ H(a,b) = \max_{1 \leq i \leq m, 1 \leq j \leq m} H_{ij}. \]

Again, these best scoring alignments can be obtained by backtracks from the scoring matrix \( (H_{ij}) \). This algorithm is known as a local algorithm.

gRNA algorithms

Let the cryptogene sequence be \( a = a_1a_2... a_n \). We will search for gRNA genes in \( b = b_1b_2... b_m \). For example could be an entire maxicircle sequence. To be clear, both a and 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

\[
\sigma(-,A) = \sigma(-,G) = c_3 \geq -\infty,
\]

and

\[
\sigma(-,T) = \sigma(-,C) = -\infty.
\]

Of course \( \sigma(a,-) = -\infty \) 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 \( T\cdot G \) 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 include 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 backtracks require the same basepair, we output only the one alignment with largest score. An iterative recombination 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 backtracks 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 \( b = b_1 b_2 \ldots b_n \) 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 \( a = a_1 a_2 \ldots a_m \) be the cryptogene region, with \( P(a_j = \alpha) = p_\alpha, \alpha \in \{A,T,G,C\} \). 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 \( p_1 = p_{G\cdot T} + p_T q_A + p_C q_G + p_G q_C \) be the probability a Watson--Crick pair and \( p_2 = p_{G\cdot U} + p_U q_G \) 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 \( p_1 \):

\[
P(l_a \geq l \text{ and } k = 0) = (1 - p_1 - p_2) p_1^l \tag{1}
\]

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

\[
P(l_a \geq l) = (1 - p_1 - p_2) \sum_{k = 0}^{3} \left( \frac{1}{k} \right) p_1^{l-k} p_2^k \tag{2}
\]

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

\[
P(L_a \geq l) = (1 - p_1 - p_2) (p_1 + p_2)^l, \tag{4}
\]

so that

\[
P(L_a = l) = (1 - p_1 - p_2)^2 (p_1 + p_2)^l. \tag{5}
\]

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: \( A\cdots A\cdots A\cdots G\cdots G\cdots G\cdots C\cdots \)
- gRNA gene: \( A?\cdots G?\cdots G?\cdots C?\cdots \)

After 'editing' these sequences become

\[
\begin{align*}
&tA\cdots \quad tA\cdots \quad tG\cdots \quad tG\cdots \quad tC\cdots \\
&A?\cdots \quad G?\cdots \quad A?\cdots \quad G?\cdots \quad A?\cdots
\end{align*}
\]
Then Perron–Frobenius theory implies $6 \implies \text{exit is from } 6$ since $j < 8$. We need to approximate the distribution of the number of steps.

Since we wish to find the maximum of a large number of such steps, we need to create a basepair or to create another editing state.

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 editing state. Recall that Rule 5 requires there be no more than 3 consecutive basepairs after the anchor. Therefore we track an $A$, $G$, or $C$ to the right. At this point, we observe another editing state. Again, we track another editing state.

The transition matrix $M = (p_{ij})$ 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 $M_{ij}^k = \text{probability of moving from state } i \text{ to state } j \text{ in } k \text{ steps}$. 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 $k > 4$. See (24) for a discussion of this theory. Then the largest eigenvalue $\lambda$ of $M$ is positive and the left $(1 = (1, 1, 2, \ldots, 8))$ and right $(r = (r_1, r_2, \ldots, r_8))$ eigenvectors also can be chosen to be positive.

Normalizing them so that $\sum_{i=1}^{8} r_i = 1$ then Perron–Frobenius theory implies $M_{ij} = \lambda r_i r_j$.

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 to 5 while Rule 6 implies exit is from 6 to 8. We need

$$P(L_e = k) \text{begin at } i \in S) = \sum_{j=6}^{8} \sum_{s=1}^{8} M_{ij}^k (1 - \sum_{s=1}^{8} M_{ij}^s)$$

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 $a = a_1 a_2 \cdots a_n$ encodes a segment of a homologous protein $y = y_1 y_2 \cdots y_m$ the edited cryptogene should be closely related to $y$. A similarity function such as one of the Dayhoff matrices or the minimum mutation matrix is used in the following way. Let $s(a i_{a 1} + i_{a 2} + 2, y)$ be the similarity function between the amino acid encoded by the triplet $a a_{a 1} \cdots a_{a 1 + 2}$ and the amino acid $y$. Define $H_{ij}$ to be the best score of any segment of $y$ ending at $y_j$, or any edited segment of $y$ 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).

$$H_{ij} = \max \{0, H_{i-1,j} - \delta_N, H_{i-1,j-1} + s(a_i, a_{i+1}, y_{j+1}), H_{i-1,j-1} + s(a_{i+1}, y_{j+1}), H_{i-1,j} + s(a_{i+1}, y_j), H_{i-1,j} + s(a_i, a_{i+1}, y_{j+1})\}$$

Here we have $\delta_N$ and $\delta_A$ the cost of deleting a nucleotide or amino acid, respectively. Normally in protein/DNA sequence
We first illustrate our algorithms by a small example with
cryptogene a = AAAATAAGGAAAGGAGGGCAAAT (5' to
3') and potential guide sequences in b = TGGGTTCC-
AAATTGTACTCATAAT (3' to 5'), compliment. For
this illustrative example we choose the naive weights where
\( \sigma(a,b) = \begin{cases} 
1 & \text{if } a \text{ and } b \text{ form a basepair (including } G\cdot U) \text{ and } \infty \text{ otherwise.} \\
\sigma(-,A) = \sigma(-,G) = \frac{1}{2} & \text{otherwise} 
\end{cases} 
\)

For this example, \( (a,b) \) = aA, bC and \( \sigma(a,b) = \frac{1}{2} \).

The results are quite robust regarding the values of \( \sigma(-,-) \),
extcept 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
we illustrate the number of contiguous U's inserted for maximicircles
and for both maximicircles 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.
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-U basepairs. We refer to this basepaired region as the anchor.

**Rule 2.** There can be no more than 3 G-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.

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

\[ (4.1986 \times 10^9)(0.0021)(0.7666)^t = 1 \]
the cytochrome-b cryptogene as our target. The related protein genome. The maximum score is 182, which comes from aligning
L tarentolae cytochrome-b in

The matrix is chosen, with deletion penalties \(d_s = 12\) and \(d_p = 4\). A protein because portions of the corresponding region of
no. J01476). We chose the first 40 amino acids of the yeast is cytochrome-b of the mitochondria of yeast (GenBank accession

It is empirical evidence that these scores are being obscured by the statistical problems 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.

or \(t = 34.18\). Therefore, the average or expected length of the longest Z is about \(t = 35\). If the cryptogene is localized to a region of length 1000 rather than 100, we then obtain \(t = 43\). 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 kinetoplastid 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 \(d_s = 12\) and \(d_p = 4\). 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 \(\geq 160\).)

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 Ts. 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 results.
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—Frobenius 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