0028–825X/93/3103–0269 $2.50/0 © The Royal Society of New Zealand 1993

Designer algorithms for cryptogene searches

MICHAEL S. WATERMAN
Departments of Mathematics and Biological Sciences
University of Southern California
Los Angeles, California 90089–1113, U.S.A.

ARNDT VON HAESELER
Lehrstuhl f. allgemein Biologie
Institut f. Zoologie
Universitaet Muenchen
Luisenstr. 14
W–8000 Muenchen 2, Germany

Abstract RNA editing in the mitochondria of kinetoplastid protozoa describes the insertion and (or) deletion of precise numbers of uridines at precise locations in the transcribed RNA. Such genes are known as cryptogenes. We describe dynamic programming algorithms to search for unknown cryptogenes and for the sequences that template the editing, gRNAs. Results of applying the cryptogene-gRNA algorithm to known cryptogene sequences from *Leishmania tarentolae* are presented.

Keywords algorithm; dynamic programming; RNA editing; cryptogene

INTRODUCTION

The nucleic acid sequence database is rapidly increasing, doubling approximately every two years. The GenBank release number 73 (Spring 1993) contains approximately 100 million nucleotides. The database is very useful for finding genes related by structure or evolutionary history. Much effort has gone into developing algorithms to rapidly (FASTA (Lipman & Pearson 1985) and BLAST (Altschul et.al. 1990)) or accurately (Smith & Waterman 1981) search the database. In this article we will take this technology in a somewhat different direction. Our interest is in modifying existing algorithms so that we might gain insight into a newly discovered biological phenomena, RNA editing.

The term RNA editing was introduced to describe the modification of mRNA sequences within coding regions (see Simpson & Shaw 1989 for a review). In the mitochondria of kinetoplastid protozoa, RNA editing involves the addition (and more rarely the deletion) of precise numbers of uridine (U) residues at a few or many locations, as otherwise the resulting mRNA would not encode the correct protein. A gene that undergoes editing is refered to as a cryptogene. RNA editing in kinetoplastid protozoa is more disruptive to the central dogma of the co-linearity of DNA and protein sequences than the phenomena of introns and exons. How does the mitochondria accomplish this intricate job of editing?

The answer to the mystery of RNA editing lies in genes known as guide RNA (gRNA) genes. These genes are transcribed to small gRNAs that mediate or guide the editing process (Blum et. al. 1990). It has been determined that gRNAs form short helical or basepaired "anchor" regions immediately downstream (3′) of the pre-edited region, forming the first event of RNA editing. The remainder of the gRNA forms a perfectly paired region with the edited mRNA, given that we allow G·U basepairs. The idea is that the editing machinery proceeds 3′ to 5′ on the gene (5′ to 3′ on the gRNA), moving past basepairs until a mispair is encountered. Then a U is inserted into the cryptogene to basepair with the unpaired G or A in the gRNA. Then the machinery continues. Several gRNA can be involved to give a mature mRNA.

In this paper we describe and extend some of our earlier work (von Haeseler et al. 1992) where we developed computational methods for locating kinetoplastid cryptogenes. Our dynamic programming algorithms "detect" possible gRNA genes and cryptogenes among genomic DNA. A key insight is to allow free insertion of U into potential cryptogenes to increase the number of basepairs with the potential

B93008
Received 19 January 1993; accepted 2 June 1993

gRNA. The algorithms are based on the algorithm for best subsequence alignment of Smith & Waterman (1981).

CRYPTOGENE-gRNA ALGORITHMS

The local similarity search algorithms as presented by Smith & Waterman (1981) is to find the best matching regions or segments between two sequences. Take $\mathbf{x} = x_1 x_2 \ldots x_n$ and $\mathbf{y} = y_1 y_2 \ldots y_m$ to be the two sequences. Define a similarity between the letters x and y to be $s(x,y)$. It is convenient to require that $s(x,y)$ take on both positive and negative values, and $s(x,x) > 0$ usually holds. Since we will not need to weight gaps other than proportional to length, it is sufficient to extend $s(.,.)$ to handle these cases, so set $s(-,y) = s(x,-) = -\delta < 0$ for all letters x and y. "–" represents the insertion/deletion of y or x, respectively, and there is a negative score of $-\delta$ for each inserted/deleted letter.

The global alignment of **x** and **y** is defined to be

$$S(\mathbf{x},\mathbf{y}) = \max\ \{\sum_{i\geq 1} s(x_i^*, y_i^*) : x_i^* \text{and } y_i^* \text{ are just } x_1 \ldots x_n \text{ and } y_1 \ldots y_m \text{ with "}-\text{" inserted.}\}$$

$S(\mathbf{x},\mathbf{y})$ is the maximum scoring alignment over all $\binom{n+m}{n}$ possible alignments, where all letters of **x** and **y** must be accounted for in the alignment.

Now we define the local score by

$$H(\mathbf{x},\mathbf{y}) = \max\{S(x_i x_{i+1} \ldots x_j, y_k y_{k+1} \ldots y_l) : 1 \leq i \leq j \leq n,\ 1 \leq k \leq l \leq m\}.$$

This is the maximum of the global alignment scores of all pairs of intervals from **x** and **y**. While H appears to be $\binom{n}{2}\binom{m}{2}$ global alignment problems, there is an elementary $O(nm)$ algorithm, known as the Smith-Waterman algorithm. Set $H_{i,j} = 0$ if $i{\cdot}j = 0$. Then recursively find

$$H_{i,j} = \max\ \{H_{i-1,j-1} + s(x_i,y_j),\ H_{i-1,j} + s(x_i,-),\ H_{i,j-1} + s(-,y_j), 0\}.$$

It can be shown that $H(\mathbf{x}, \mathbf{y}) = \max\{H_{i,j}: 1 \leq i \leq n, 1 \leq j \leq m\}$.

In Waterman & Eggert (1987), it is shown that a simple recalculation procedure allows us to compute the k-best alignments, which do not share a match or mismatch. This modification is important in practice but is not required for the algorithm we next present.

Now we modify the Smith-Waterman algorithm to make a new algorithm suitable for finding cryptogenes. For definiteness let $\mathbf{x} = x_1 \ldots x_n$ contain the potential cryptogene and $\mathbf{y} = y_1 \ldots y_m$ contain potential gRNA sequences. The reason for employing local algorithms such as Smith-Waterman is the unknown locations of cryptogenes and/or gRNAs.

The changes are easy to make once the motivation is clear. We wish to use the basic algorithm presented above but with modified scoring. Basepairs must be rewarded, for example

$$s(a,b) = \begin{cases} 1.1 \text{ if } (a,b) \in \{(A,U),(U,A)\}, \\ 2.1 \text{ if } (a,b) \in \{(G,C),(C,G)\}, \\ 0.1 \text{ if } (a,b) \in \{(G,U),(U,G)\}, \\ -\infty \text{ otherwise} \end{cases}$$

Here non basepairs are not allowed by setting $s(A,C) = -\infty$, for example.

These are just possible numbers that are motivated by the free energies of basepairs. Stacking energies can easily be added. It is not clear the free energy is the correct motivation for this scoring. Our results in practice do not seem to be very dependent on scoring.

To cover the insertion of U, we need to set $s(-,A) = s(-,G) = c > -\infty$. In our program we used $s(-,A) = s(-,G) = 0$ for the "free" insertion of U's. To prevent all other insertions we set

$$s(-,U) = s(-,C) = -\infty$$

as we do

$$s(x,-) = -\infty \text{ for all } x \in \{A,U,G,C\}.$$

It should be emphasised that the search here is for a precise molecular structure, that of the cryptogene-gRNA interaction. The motivations of evolutionary related subsequences is invalid here. If the precise basepair and insertion process does not take place, the resulting mRNA will not encode the correct amino-acid sequence.

Tracebacks from the max H_{ij} usually involve multiple "alignments" but for these gRNA problems the multiplicity is never an issue in practice. In addition, Waterman & Eggert (1987) describe a subtle algorithm to produce the k-best subsequence alignments that do not share any matches or mismatches. It did not turn out to be necessary to bring these ideas into producing the k-best candidate gRNA-cryptogene interactions. Generally the tracebacks are unambiguous and non-intersecting.

RESULTS

To test the power and utility of our approach we considered the following problems. There are seven gRNAs experimentally known to edit four cryptogenes (cytochrome b, Murf2, ND7, and COII) in *Leishmania tarentolae*. Our test problem was to take these known cryptogenes and search the *L. tarentolae* maxicircle (20 993 basepairs, both strands) and see what the algorithm produces. In Table 1 these results are shown and, except for the ND7-gRNA 5′ alignment, the results were not useful. Varying scoring did not improve the rankings. Then we studied the known examples of editing for useful (common) sequence patterns that could improve the search. Since each of the known gRNAs basepair with a different nucleic acid coding region, consensus methods are unlikely to give useful insights. Instead, our observations involved the numbers of bases and basepairs, features that might be employed by the editing machinery. Here are the features we imposed on our search:

1. There must be at least five basepairs in the anchor.
2. There cannot be more than three G·U basepairs in the anchor.
3. There must be at least four adjacent Watson-Crick basepairs in the anchor.
4. The number of adjacent inserted U's cannot exceed eight.
5. Between inserted U's there cannot be more than three (non-edited) basepairs at the 5′ end of the cryptogene.

These rules improve the results as seen in the last column of Table 1 but they are not good enough to suggest that we can find unknown cryptogene-gRNA pairs.

To study why our results were so weak, we modelled the genome as a random sequence and we derived by Markov chains and Perron-Frobenius theory the statistical distribution of the length of the longest candidate edited region. If Z is the length of a random editing event, we show that

$$P(Z = t) = 0.0005(0.7666)^t$$

for the maxicircle sequence with $p_U = 0.3835$, $p_C = 0.1160$, $p_G = 0.0949$, and $p_A = 0.4056$. Recall that the maxicircle sequence is 20 993 and that we search both strands, so $m = 2 \times 20\,993$. For a cryptogene $n = 100$, the "longest Z" should be approximately the solution of

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

or $t = 34.18$. If $n = 1000$, $t \approx 43$. Therefore, random genomic sequence gives potential gRNAs between 35 and 45, just the sizes of the **real** gRNA sequences in Table 1. This statistical distribution explains the lack of power of our searches.

PROTEIN-CRYPTOGENE SEARCHES

A distinct approach is to see if a subsequence of $\mathbf{x} = x_1x_2 \ldots x_n$ encodes a subsequence of a homologous protein $\mathbf{z} = z_1z_2 \ldots z_m$. We could, for example, use the cytochrome-b sequence from yeast for $\mathbf{z}$. The computational algorithm is derived from asking if free insertion of U's into $\mathbf{x}$ will encode a protein that is evolutionarily related to $\mathbf{z}$. As usual we need to specify the scoring. Let $s(x_i\, x_{i+1}\, x_{i+2}, z_j)$ be the similarity (e.g., as from the Dayhoff matrix) between the amino acid encoded by $x_i\, x_{i+1}\, x_{i+2}$ and z_j. Define H_{ij} to be the best score of any (edited) subsequence ending at x_i and at z_j, or 0, whichever is larger. We must allow for deletion of nucleotides (δ_N) or aminoacids (δ_A) as well as the free insertion of U's. The resulting algorithm is

Table 1 Ranking by score of known gRNAs when searching the maxicircle with four cryptogenes.

Cryptogene	gRNA	Length	Rank in the list of suboptimal gRNA	Rank after applying rules
cytochrome b	gRNAI	32	27–30	3
	gRNAII	54	36–43	1
Murf2	gRNAI	14	271–306	78–84
	gRNAII	46	1286–1327	18–21
ND7	gRNA-5'	44	2	1
	gRNA-FS	19	92–93	8
COII	gRNA-FS	16	88–100	86–99

$$H_{i,j} = \max \left\{ \begin{array}{l} 0, \\ H_{i-1,j} - \delta_{N}, \\ H_{i,j-1} - \delta_{A}, \\ H_{i-3,j-1} + s(x_{i-2}x_{i-1}x_{i}, z_j), \\ H_{i-2,j-1} + s(x_{i-1}x_{i}U, z_j), \\ H_{i-2,j-1} + s(x_{i-1}Ux_{i}, z_j), \\ H_{i-2,j-1} + s(Ux_{i-1}x_{i}, z_j), \\ H_{i-1,j-1} + s(x_{i}UU, z_j), \\ H_{i-1,j-1} + s(Ax_{i}U, z_j), \\ H_{i-1,j-1} + s(UUx_{i}, z_j), \\ H_{i,j} + s(UUU, z_j) \end{array} \right\}$$

After the $n \times m$ matrix $H = (\mathrm{H}_{i,j})$ is calculated, then $\max\{H_{ij} : 1 \le i \le n,\ 1 \le j \le m\}$ gives the best matching segment between possible cryptogenes encoded proteins in **x** and in the protein **z**. This method was not a success in practice for fairly obvious reasons. Namely, the sources of error in all possible cryptogenes and in the evolutionary distance to **z** dominate any biological matchings.

Clearly the knowledge that the cryptogene must be edited to a coding sequence can be used, although it is not quite as straightforward as the above algorithms. The algorithm we give next has not appeared elsewhere. Let **x** be the potential cryptogene sequence, **y** be the sequence containing potential gRNAs, and **z** be the potentially homologous amino acid sequence. Among first approaches that might be considered is to run a three-sequence alignment algorithm with time and space complexity equal to the products of the three sequences. While the method we give does involve three-sequence alignment, it handles the cryptogene-gRNA alignment first and is a practical way to greatly limit the search, and should be much more powerful than the (H_{ij}) method described above.

First we find all potential cryptogene-gRNA matches. Define F_{ij} = maximum number of amino acids encoded (by the genetic code) in the gRNA ending at y_j and in a cryptogene ending at x_i. Clearly we initialise the algorithm by $F_{ij} = 0$ if $i \le 2$ or $j \le 2$. Let $S = \{A,G\}$, the nucleotides that pair with inserted U's.

$$F_{i,j} = \begin{cases} \left(F_{i,j-3} + 1\right) \times I\left\{y_{j-2}y_{j-1}y_j \in S^3\right\}, \\ \left(F_{i-1,j-3} + 1\right) \times I\{y_{j-2}y_{j-1}y_j \in \\ \qquad \{x_iSS, Sx_iS, SSx_i\} \cap T^c\} \\ \left(F_{i-2,j-3} + 1\right) \times I\{y_{j-2}y_{j-1}y_j \in \\ \qquad \{x_ix_{i-1}S, x_iSx_{i-1}, Sx_ix_{i-1}\} \cap T^c\} \\ \left(F_{i-3,j-3} + 1\right) \times I\{y_{j-2}y_{j-1}y_j = \\ \qquad \{x_{i-2}x_{i-1}x_i \in T^c\} \end{cases}$$

where T = set of terminator or stop codons. Notice that $F_{ij} = 0$ unless it starts or extends a cryptocoding segment.

Each $F_{ij} > 0$ corresponds to F_{ij} amino acids encoded in the gRNA. Usually these will be only one path achieving F_{ij} but that is of course not necessarily the case. A simple modification of the algorithm for F gives the path that maximises the x_i's used in the alignment, which is equivalent to minimise the number of inserted U's. This seems to us to be the most reasonable choice for the choice of cryptogene.

Therefore, the DNA encodes possible protein sequence segments and we take only those with $F_{ij} \ge c$, that is, with at least c residues. It is elementary to maximally extend these possible coding intervals in **x** by adding triplets of nucleotides (codons) from **x** or from another coding interval, maintaining amino acid encoding. Adding a penalty for changing intervals is also easy to do. Two maximal extensions are either identical or disjoint.

Now we match these extensions with **z**, the homologous protein sequence. The extension sequences can be indexed by the increasing sequence $x_{i1}\ x_{i2}\ \dots\ x_{in}$ when i_l is the largest x_i in the l-th "codon". If this set is empty omit x_{il} in the sequence. Define

$G_{i,k}$ = Best score aligning any segment of **z**, ending at residue z_k, with any extension ending at x_i.

Then

$$G_{i,j} = \begin{cases} G_{i,k-1} - \delta_A, \\ G_{i-1,k-1} + s(y,z_k), \text{ all extensions using} \\ \qquad\qquad \text{only } x_i \text{ to code } y, \\ G_{i-1,k-1} - \delta_N - \delta_A, \\ G_{i-2,k-1} + s(y,z_k), \text{ all extensions using} \\ \qquad\qquad \text{only } x_{i-1}x_i \text{ to code } y, \\ G_{i-2,k-1} - 2\delta_N - \delta_A, \\ G_{i-3,k-1} + s(y,z_k), \text{ all extensions using} \\ \qquad\qquad \text{only } x_{i-2}x_{i-1}x_i \\ \qquad\qquad \text{to code } y, \\ G_{i-3,k-1} - 3\delta_N - \delta_A, \\ 0 \end{cases}$$

Our technique of putting intervals together to match a sequence has some history. Bement & Waterman (1977) analysed geological data with a related technique. Auger & Lawrence (1989) studied identification of segment neighbourhoods in protein sequences. Sankoff (1992) looked at a decomposition of a sequence into segments, each belonging to some template in an inventory. Here the set of maximal extensions can be viewed as his inventory. This method has not yet been tested on data.

ACKNOWLEDGMENTS

This research was supported by grants from the National Institutes of Health and the National Science Foundation.

REFERENCES

Altschul, S.; Gish, W.; Miller, W.; Myers, E.; Lipman, D. 1990: Basic local alignment search tool. *Journal of molecular biology 215*: 403–410.

Auger, I. E.; Lawrence, C. E. 1989: Algorithms for optimal identification of segment neighborhoods. *Bulletin of mathematical biology 51*: 39–54.

Bement, T. R.; Waterman, M. S. 1977: Locating maximum variance segments in sequential data. *Mathematical geology 9*: 55–61.

Blum, B.; Bakalara, N.; Simpson, L. 1990: A model for RNA editing in kinetoplastid mitochondria: "guide" RNA molecules transcribed from maxicircle DNA provide the edited information. *Cell 60*: 189–198.

Lipman, D. J.; Pearson, W. R. 1985: Rapid and sensitive protein similarity searches. *Science 227*: 1435–1441.

Sankoff, D. 1992: Efficient optimal decomposition of a sequence into disjoint regions, each matched to some template in an inventory. *Mathematical bioscience 111*: 279–293.

Simpson, J.; Shaw, J. 1989: RNA editing and the mitochondria cryptogenes of kinetoplastid protozoa. *Cell 57*: 355–366.

Smith, T. F.; Waterman, M. S. 1981: Identification of common molecular subsequences. *Journal of molecular biology 147*: 195–197.

von Haeseler, A.; Blum, B.; Simpson, L.; Sturm, N.; Waterman, M. S. 1992: Computer methods for locating kinetoplastid cryptogenes. *Nucleic acids research 20 (11)*: 2717–2724.

Waterman, M. S.; Eggert, M. 1987: A new algorithm for best subsequence alignments with application to tRNA-rRNA comparisons. *Journal of molecular biology 197*: 723–728.