Some simple computational methods to improve the folding of large RNAs

A.B.Jacobson, L.Good, J.Simonetti* and M.Zuker+

Department of Microbiology, and *Computing Center, SUNY, Stony Brook, NY 11794, USA, and +Division of Biological Sciences, National Research Council of Canada, Ottawa, K1A OR6, Canada

Received 19 July 1983

ABSTRACT

Computational methods are described which increase the efficiency of the RNA folding algorthm described by Zuker and Stiegler. Bit addressing has been used to reduce the memory requirements from 2NxN to NxN/2. The order in which the nucleotide sequence is examined internally has been altered, and some additional short arrays which carry temporary information have been introduced. These changes optimize the management of the large data arrays generated by the algorithm. The methods were developed for use with a UNIVAC 1100/82 computer. They are, however, easily adaptable to other computers; especially those with virtual memory capabilities. The analysis of sequences up to 1000 nucleotides long are relatively routine, and larger searches are also feasible. Some limitations and applications of the algorithm are also discussed.

INTRODUCTION

The rapid proliferation of nucleic acid sequences in recent years has led to the development of a number of algorithms which attempt to predict the two dimensional structure of large single-stranded RNAs (1-13). Although the published methods do not, as yet, predict any known structures correctly, the combinatoric method which these algorithms provide, when used in conjunction with other information provide a powerful tool for the analysis of RNA structure.

Although the most recent algorithms which have been described incorporate dynamic programming methods and are therefore computationally efficient, the number of alternative possibilities which need to be considered, even for very short nucleic acid sequences is very large and the analysis of sequences long enough to be of biological interest quickly exhaust the core memory of large computers. Thus simple procedures which optimize the utilization of computer memory are of interest.

The methods described here have been introduced into the algorithm described by Zuker and Stiegler (8) in order to adapt it to run efficiently on a Univac 1100/82 computer. The requirement for core memory has been cut by

© IRL Press Limited, Oxford, England.

a factor of 4. In addition, multibanking techniques have been introduced to enable us to exchange banks of data to and from main memory in an efficient manner. The methods we have used have general application and can be adapted rather easily to other computers, especially those with virtual memory.

METHODS

The algorithm used in the search for an optimal RNA structure has been described previously (8). Briefly, dynamic programming methods are used to identify the optimal structure for each subsequence S from i to j within a sequence N nucleotides long, where $1 \leq i < j \leq N$. The relative stability of alternate structural features within S are evaluated using published thermodynamic data (14-19). The main working arrays employed in the search are V and W. V(i,j) contains the lowest free energy value that can be computed for the section S when i and j pair with one another. W(i,j) contains the lowest free energy which can be computed for the section S whether or not i and j pair with each other. The identification of the optimal structure at each i,j pair requires a recursive search.

The computer memory requirements for the arrays V and W are both NxN/2. For large sequences,the entire arrays cannot be stored in directly addressable core memory,and must therefore be divided into sections. The methods we have developed optimize the amount of information which can be stored in each section, and further increase the efficiency with which sections are "swapped" to and from core memory.

To optimize storage we have used bit addressing to store all arrays carrying information for a given i,j pair in one word. Further space saving is achieved by storing all values calculated by by the program in integer form. The number of bits required for the storage of information in the arrays V and W can be calculated from the data shown in Tables 1 and 2. Table 1 shows free energy values which can be calculated for GC rich sections, 200

Table 1. Calculated Δ G values for 200 nucleotide sections of several different RNAs. Note that the structures that form in regions of similar base composition can vary significantly in stability.

Sequence	Section	Δ G kcal/mole	%GC
MS2	3100-3299	-106.0	57.5
SV40	500-699	-79.6	50.0
Polio	200-399	-87.8	57.5
T4-Gene 32	950-1149	-39.8	39.0
E. coli 23S	800-999	-89.5	55.5

Table 2. Base pairing energies expressed in tenths of a kilocalorie/mole. The stacking energies are identical with those described by Salser (18). The loop destabilizing energies are described in Cech et al., (19). The value 999 has been inserted to prohibit the formation of unwanted structures (see discussion).

Stacking Energies (UG = GU)

	GU	AU	UA	CG	GC
GU	-3	-3	-3	-13	-13
AU	-3	-12	-18	-21	-21
UA	-3	-18	-12	-21	-21
CG	-13	-21	-21	-48	-43
GC	-13	-21	-21	-30	-48

Loop Destabilizing Energies

Size	1	2	3	4	5	6	7	8	9	10	12	14	16	18	20	25	30
Bulge loop	28	61	67	72	74	75	77	78	79	80	81	83	84	85	86	87	89
Hairpin loop																	
Closed by:																	
CG	999	999	70	45	41	43	45	46	48	49	50	52	53	54	55	57	59
AU	999	999	70	45	41	43	45	46	48	49	50	52	53	54	55	57	59
Interior loop																	
Closed by:																	
CG-CG	999	1	9	16	21	25	26	27	28	29	31	32	33	34	35	37	39
CG-AU	999	10	18	25	30	34	35	36	37	38	39	40	41	42	43	45	47
AU-AU	999	18	26	33	38	42	43	44	45	46	48	49	50	51	52	54	56

nucleotides long, of several different RNAs. Table 2 provides a compilation of the most recent free energy values for calculating structure (19) as they are implemented in our program. All values for loop free energies have been rounded to two significant digits and all values are stored as integers. The rounding generally has little or no effect on the structures chosen as optimal by the program.

From the data presented in Table 1 it can be seen that the structures produced by folding sequences 1000 nucleotides long will generally have a calculated free energy value greater than -500.0 kilocalories/mole. Since our energy values are carried in integer form we allocate 13 bits each for the storage of the arrays V and W. Two additional arrays, C(i,j) and C(j,i), are used by the program. These are bookkeeping arrays which track base composition of nucleotide pairs, and the properties of multibranch loops. The value of C(i,j) never exceeds 3 and the value of C(j,i) need not exceed 93.

Thus these arrays are easily accommodated in the remaining bits of the Univac 1100 36 bit word. Further space saving can be achieved by carrying the values in V(i,j) as the difference between V(i,j) and W(i,j). This method has not been implemented, but would be appropriate to computers with smaller word size.

The most important feature of the new program concerns the pattern in which the arrays V(i,j) and W(i,j) are filled. As discussed previously (4,6,8), dynamic programming methods proceed in a sequential fashion, solving the structure for the smallest nucleotide sections first. The exact pattern that is followed will vary according to the architecture of the computer used. Figure 1A shows three different search procedures which can be employed. The first of these is the simplest to visualize. Starting with the smallest sections first, all sections of equal size are examined sequentially. This results in a fill in procedure along the diagonal of the array. This method is most appropriate for computers with array processing such as the Cray where several intervals of the same size can be evaluated simultaneously. Figures 1B and 1C show alternate, mathematically equivalent searches. The search shown in Figure 1C is the one used in this program version. The analysis starts with short sections at the end of the sequence and procedes backwards. For each value of i, the program evaluates all sections from i+4 to j in the order of increasing length.

The advantage of this procedure becomes apparent, when one considers the manner in which sections of the large data arrays are "swapped" to and from main memory. For our Univac program, the data arrays are divided into 3 large banks of 200,000 words each. These are indicated schematically in Figure 1c. The fill in pattern minimizes the number of time the banks need to be exchanged as the analysis proceeds. For any section $S_{i..j}$, the values referenced most frequently are the W and lower C values for the subsections i..i', i'+1..j, i+1..i', i'+1..j-1, where i' lies anywhere within the section i..j. With the search procedure shown in figure 1c the required values for i'+1..j are always present in the columns of the appropriate bank. The values for i'+1..j-1, are always present in the appropriate bank as well, except for the case where j is located in the first column of the bank. The values i..i', and i+1..i' are located in a horizontal row across all three banks. However, the fill in procedure in Figure 1C proceeds in a horizontal and sequential fashion. This feature enables us to store the required values of i in two temporary small arrays which are available to the programs regardless of the bank in which the current i..j values are stored.

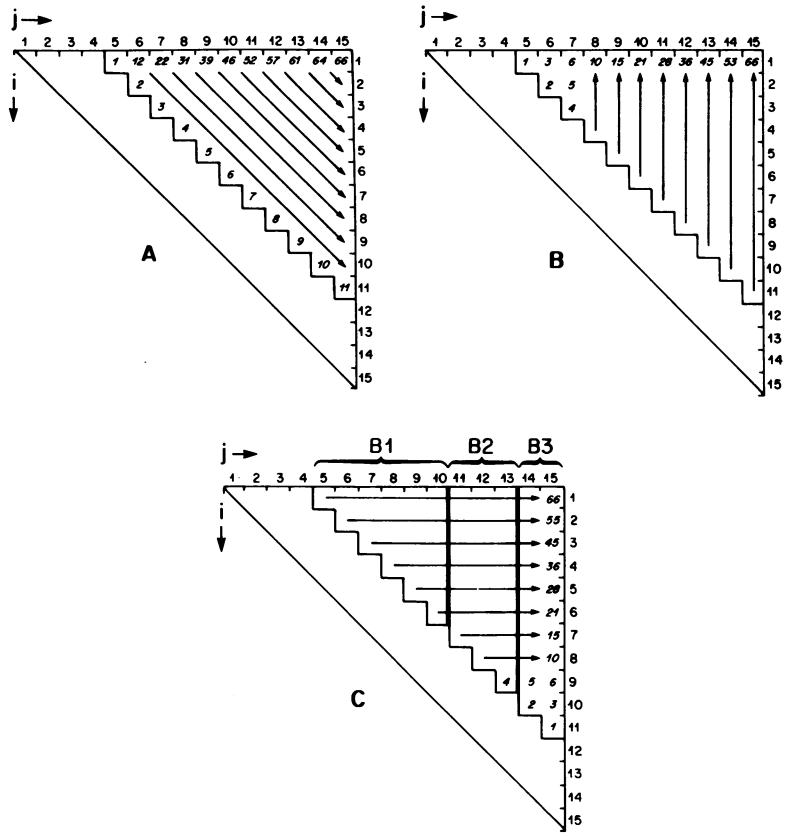

Figure 1. Three graphs showing patterns which can be used in storing data generated by the RNA folding algorithm. A) Filling along the diagonal of the matrix. Sections of the nucleotide sequence of equal size are examined sequentially. B) Filling columns in the matrix. C) Filling rows in the matrix. In each instance the smallest sections of the sequence are evaluated first, followed by larger sections as required by the algorithm. The position of the three data banks, B1, B2, and B3, are indicated schematically in Figure 1C.

In practice the data banks are allowed to overlap by 31 nucleotides. This overlap allows a small additional increase in the optimization of the program. This optimization is not of general use, and is not described in detail here.

In addition to the features described above, we have employed multi-banking to optimize our program for the Univac environment (20). This method is used when more than 262K words (18 bit addressing space) of data need to

be accessed. Multibanking is a technique which allows the user to move banks of data to and from main memory without incurring I/O overhead. If a bank is "based" it can be addressed directly by the program. If the bank is "unbased" it may continue to reside in main memory or be swapped out into secondary storage at the discretion of the operating system. The use of this method required the preparation of two very short routines in assembly language which could be implemented easily on other computers. In general, however, the virtual memory management systems of many of the large computers will handle the program adequately without these extra instructions. Recently, a version of this program has been implemented on a Vax computer with a VMS operating system. The time required for paging was reduced by a factor of 6.

DISCUSSION

In this paper we have described two computational methods which allow us to increase our capability of using computers to predict the structure of large nucleotide sequences. On the Univac 1100 runs of 800 to 1000 nucleotides are routine. When these methods are fully implemented for some of the smaller computers with virtual memory, runs of this type will become rather commonplace. It should be noted however, that the information obtained from computer runs of this type still suffer from a number of limitations.

Computer modeling studies have been performed on a variety of RNA sequences for which the two dimensional structure is thought to be known. In general, the structures which can be predicted by the modeling are not very satisfactory. Using the newest energy rules on a class of 92 tRNA molecules,only 24% of the sequences were were folded into perfect cloverleaf structures. An additional 27% were folded into structures quite similar to but not identical with the standard cloverleaf structure. The remainder of the sequences appeared in a variety of structures whose calculated free energy was lower than that of the cloverleaf form (unpublished results). The source of the difficulties which the alogorithms experience in predicting known structures correctly is not fully understood, although it is thought to lie primarily with the energy rules used in the prediction. In particular, no information is available for the appropriate treatment of multibranched loops. Since these structures are a conspicuous feature of transfer RNA molecules, it is not surprising that the structures are not predicted correctly by the folding algorithms. Some effort has been directed toward improving energy rules for structure prediction by direct modeling with tRNA sequences (5) and viroid RNA (21). These studies have only had limited

success. Possibly, some consideration of tertiary structure will be required before the algorithms are able to predict structure reliably.

In addition to the problems discussed above, the dynamic programming method itself has an important intrinsic limitation, since it provides only one optimal structure for each nucleotide sequence. Alternate equivalent structures, and suboptimal structures are not identified. These alternate structures may be of particular interest in phylogentic comparisons of related RNAs, or in situations where alternate conformations of the RNA may be implicated in the regulation of gene expression. As with the tRNA structures, the optimal folding identified by the algorithm need not reflect the structure of greatest biological interest.

Despite these limitations, the folding algorithms provide a powerful tool which can be used in conjunction with other methods for the analysis of the two dimensional structure of single-stranded RNAs. A number of subprograms have been developed which can be used in various ways in conjunction with biochemical, phylogenetic, and/or electron microscopic data. In general these programs identify a particular suboptimal solution which is defined by local parameters introduced into the program. For example, a simple alteration in the traceback procedure allows us to simulate the folding of an RNA molecule during its biosynthesis. Specific double-stranded and single-stranded regions can be introduced into the structure by the appropriate use of bonus and penalty weights. It is possible to cut and splice sequences or to force the folding of particular subsections of the sequence to conform with known long range interactions. The optimization of short loops can be achieved by setting limits to the size of loop which is allowed to form during the calculation. The details of these and other program modifications will be presented elsewhere (manuscript in preparation). It is generally useful to employ several different programs and different constraints in the analysis of each sequence in order to gain insight into suboptimal structures which may have functional significance.

ACKNOWLEDGEMENT

The first three authors would like to acknowledge their indebtedness to John Milazzo of the Stony Brook Computing Center. Without his advice and support this project could not have been completed. The research has been supported in part by a grant from the National Institutes of Health (AI15273) to ABJ.

REFERENCES

1. Pipas,J.M. and McMahon, J.E. (1975) Proc.Nat.Acad. Sci. USA 72,2017-2021
2. Studnicka, G.M., Rahn, G.M., Cummings, I.W. and Salser, W.A. (1978) Nucleic Acids Res. 5, 3365-3387.
3. Waterman, M.S. and Smith, T.F. (1978) Math. Biosc. 42, 257-266.
4. Nussinov, R., Pieczenik,G., Griggs, J.R., and Kleitman, D. (1978) Siam J. Appl. Math. 35,68-82.
5. Ninio,J. (1979) Biochimie 61, 1133-1150.
6. Nussinov, R. and Jacobson, A.B. (1980) Proc. nat. Acad. Sci. USA 77, 6309-6313.
7. Howell, J.A., Smith, R.F., and Waterman, M.S. (1980) Siam J. Appl. Math 39, 119-133.
8. Zuker, M. and Stiegler, P (1981) Nucleic Acids Res. 9, 133-148.
9. Studnicka, G.M., Eiserling, F.A. and Lake, J.A. (1981) Nucleic Acids Res. 9,1885-1904.
10. Dumas, J.P. and Ninio, J. (1982) Nucleic Acids Res. 10, 197-206.
11. Auron, P.E., Rindone, W.P., Vary, C.P.H., Celentano, J.J., and Vournakis, J.N. Nucleic Acids Res. 10,403-419.
12. Goad,W.E. and Kanehisa, M. (1982) Nucleic Acids Res. 10, 265-278.
13. Shapiro, B.A. and Lipkin, L.E. (1983) Computing in Biological Science (Geisow and Barrett eds.) pp 233-271.
14. Tinoco, I.,Jr., Borer, P.N., Dengler, B., Levine, M.D., Uhlenbeck, O.C. Crothers, D.M., and Gralla, J. (1973) Nature New Biology, 246, 40-41.
15. Borer, P.N., Dengler, B., Tinoco, I.,Jr., and Uhlenbeck, O.C.,(1974) J. Mol. Biol. 86,843-853.
16. Wickstrom,E., and Tinoco, I.,Jr. (1974) Biopolymers 13, 2367-2383.
17. Yuan, R.C., Steitz, J.A., Moore, P.B., and Crothers, D.M. (1979) Nucleic Acids Res. 7, 2399-2418.
18. Salser.W (1977) Cold Spring Harbor Sumposium on Quantitative Biology, 42, 985-1002.
19. Cech, T.R., Tanner, N.K., Tinoco, I.,Jr., Weir, B.R., Perlman, P.F. and Zuker,M, (1983) Proc. Nat. Acad Sci. In Press.
20. Doesschate, P.C. (1981) USE Inc. Technical Papers, Spring Conference pp 501-504.
21. Riesner, D., Steger, G., Schumacher, J., Gross, H.J., Randles, J.W., and Sanger, H.L. (1983) Biophys. Struct. Mech. 9,145-170.