[52] Computer Analysis of Nucleic Acid Sequences

By MICHAEL S. WATERMAN

Introduction

The increasing body of nucleic acid sequence data has created interest among many scientists in computational aspects of data storage and data processing. In fact GenBank and other data bases have been created in order to store the data in a useful format, both for archival and analysis purposes. (See Burks *et al.*[1] for a review of GenBank activities.) The value of simply have easy access to all ribosomal RNAs, for example, is not to be underestimated. The purpose of this article is to examine some of the array of tools that have been created in order to look at sequences in a rigorous, systematic way, utilizing the power of modern computers. These analyses began in the early 1970s and are now becoming more focused on specific problems such as consensus patterns in regulatory sequences or RNA folding.

The sections below will outline some methods of computer analysis of sequence data. The intent is to describe the analyses and to give emphasis to the possible utility of the analysis, not to present detailed mathematics or computer science of the techniques. Reference is made to papers where more technical details of equations, pointers, and data structures are to be found.

A great many algorithms have been proposed in recent years. See the issues of *Nucleic Acids Research*[2-4] and the *Bulletin of Mathematical Biology*[5] devoted to computer methods for a good sample of this literature. Frequently ideas such as open reading frame analysis or dot matrices are rediscovered and reimplemented over and over. For this reason I make no attempt to survey the literature. Instead I try to describe some useful and interesting methods of sequence analysis that utilize the power of computers.

Several different questions might be asked about a sequence. One concerns unexpected relationships with other sequences; these discoveries are sometimes made by screening a data base with the sequence. Wilbur

[1] C. Burks, J. W. Fickett, W. B. Goad, M. Kanehisa, F. I. Lewitter, W. P. Rindone, C. D. Swindell, C.-S. Tung, and H. S. Bilofsky, *CABIOS* **1**, 225 (1985).
[2] D. Soll and R. J. Roberts, *Nucleic Acids Res.* **10** (1982).
[3] D. Soll and R. J. Roberts, *Nucleic Acids Res.* **12** (1984).
[4] D. Soll and R. J. Roberts, *Nucleic Acids Res.* **14** (1986).
[5] H. M. Martinez, *Bull. Math. Biol.* **46** (1984).

and Lipman[6] devised a search algorithm based on the computer science technique of hashing and theirs is the method of choice for such questions. Several groups have implemented versions of their technique. Essentially the search looks for exactly matching k-mers (usually $k = 4$ to 6) and reports regions where the test sequence has a high density of matches with the data base. There is no further discussion of data base searches in this article but they should not be overlooked.

Sequence alignment is a popular computer activity. The computer alignments are often based on some explicit optimization function, rewarding matches and penalizing mismatches and insertions and deletions. Sequence alignment can give useful information about evolutionary or functional relationships between sequences. I distinguish two types of alignment: (1) alignment of full sequences and (2) finding segments of sequence that can be well aligned. Full sequence alignment should only be attempted when it is believed that the sequences are related, from one end to the other. If this is not the case, the sequences can be forced into incorrect relationships due to the necessity of matching the "dissimilar" segments. I feel much easier about running a maximum segments analysis, that only finds those segments of the sequences matching at or above some preset level.

Consensus sequence analysis is usually done by "eye" and experiment. Of course we only believe a protein-binding site when it is verified by experiment, but analysis by eye can be biased. So it is useful to have computer methods that can find consensus patterns best fitting explicitly stated criteria. Some algorithms have been developed along these lines,[7,8] and they are described here, along with example output. Consensus repeats and consensus palindromes within a single sequence and among a set of several sequences are analyzed.

Secondary structure of single-stranded nucleic acids is another popular computer analysis. For one sequence, the minimum energy algorithms which employ dynamic programming are the usual method, and I briefly describe them below. For sets of several sequences which fold into a common structure, the comparative or consensus method is very useful. The methods that Woese, Noller, and collaborators[9,10] have so powerfully employed are now included in an explicit algorithm that is described and illustrated on a set of 5S sequences.

[6] W. J. Wilbur and D. J. Lipman, *Proc. Natl. Acad. Sci. U.S.A.* **80,** 726 (1983).
[7] M. S. Waterman, R. Arratia, and D. J. Galas, *Bull. Math. Biol.* **46,** 515 (1984).
[8] D. J. Galas, M. Eggert, and M. S. Waterman, *J. Mol. Biol.* **186,** 117 (1985).
[9] H. F. Noller and C. R. Woese, *Science* **212,** 403 (1981).
[10] H. F. Noller, *Annu. Rev. Biochem.* **53,** 119 (1984).

Computer analysis of sequences has some distinct advantages over analysis by "eye." The computer analysis must be well defined, explicitly, so that the same search can be duplicated elsewhere. Many more cases can be examined by computer, and the calculations are done correctly. Inherent in these advantages are some important disadvantages. The computer will only do exactly what it is programmed to do. If the task is the "wrong" one, massive and correct calculations do not help. In addition, a machine will not notice a pattern it has not been programmed to notice, something at which humans excel. The sequence analyst should be aware of what the computer is and is not doing. Computation might be useful but it should not stand alone.

Sequence Comparisons

The majority of mathematical effort on sequence analysis has been in the area of sequence alignment. One of the reasons for this is the appeal of the basic problem that can be described as follows: Given two sequences $a = a_1 a_2 \ldots a_n$ and $b = b_1 b_2 \ldots b_m$, what is the minimum number of substitutions, insertions, and deletions needed to transform a into b? The obvious application in molecular biology is to find minimal evolutionary pathways between sequences. The correspondences between a and b are usually displayed as alignments where it is easy to see highly conserved regions of sequence.

Applications to many other areas exist and a book has been written on the topic of sequence comparisons.[11] In computer science there is the problem of recognizing mistyped words as well as file comparisons, and a large literature exists in that field on the so-called string matching problem. Some of that literature is parallel to and largely independent of the biological literature. Transpositions of adjacent letters are considered in computer science; in biology inversions pose related problems. Recognizing the relationships between groups of birds is sometimes studied via bird song comparisons, as is the manner in which younger birds acquire vocalization. In geology, an important class of problems relating stratigraphic sequences can best be approached by sequence comparison methods. These and other applications require careful algorithm development, with attention paid to the specifics of the problem settings.

In 1970 Needleman and Wunsch[12] wrote a landmark paper that ap-

[11] D. Sankoff and J. B. Kruskal (eds.), "Time Warps, String Edits, and Macromolecules: The Theory and Practice of String Comparison." Addison-Wesley, London, 1983.

[12] S. B. Needleman and Wunsch, *J. Mol. Biol.* **48**, 444 (1970).

proached sequence comparison (alignment) through a dynamic programming algorithm. Their algorithm finds maximum similarity between two sequences, where matches receive positive weight and mismatches and insertions and deletions receive nonpositive weight. Some mathematicians begin to attempt to define a distance between sequences and so to construct a metric space. This search culminated with Sellers,[13] who obtained the desired results for single insertions and deletions, and with later workers who extended his work to multiple insertions and deletions.[14] While it has been proven that similarity and distance are equivalent concepts when matching full sequences,[15] for certain other problems similarity is superior, and I concentrate on similarity here.

Sellers[16] wrote the first articles on finding the best matching pieces (segments) between two sequences and has continued those efforts. A much simpler approach through similarity was taken by Smith and Waterman,[17] and an extension[18] of that technique is presented below.

Matching or aligning entire sequences should be attempted when the sequences are known or suspected to be closely related. Even when this is the case, an extraordinary number of optimal alignments can result; many of these will differ only slightly from one another. I illustrate this below with some recommendations on how to deal with the situation. Most sequence comparisons will, however, involve sequences only significantly related in pieces, if at all. In those cases a full alignment is not informative and the maximum segments algorithm is the most appropriate. These algorithms can produce segment matchings which are best, second best, and so on; this is shown in examples.

Aligning Full Sequences

As explained above, what is to be presented is a similarity method for sequence alignment. The function $s(a,b)$ is to define similarity between the letters a and b. In the examples below matches receive weight 1 and mismatches receive -1, so that

$$s(a,b) = \begin{cases} +1, & \text{if } a = b \\ -1, & \text{if } a \neq b \end{cases}$$

[13] P. Sellers, *SIAM J. Appl. Math.* **26,** 787 (1974).
[14] M. S. Waterman, T. F. Smith, and W. A. Beyer, *Adv. Math.* **20,** 367 (1976).
[15] M. S. Waterman, *Bull. Math. Biol.* **46,** 473 (1984).
[16] P. Sellers, *J. Algorithms* **1,** 359 (1980).
[17] T. F. Smith and M. S. Waterman, *J. Mol. Biol.* **147,** 195 (1981).
[18] M. S. Waterman and M. Eggert, *J. Mol. Biol.* **197,** 723 (1987).

Deletions of length k receive weight $-x_k$ and below only x_1 is used. The algorithm is based on recursively computing a matrix S. First

$$\begin{aligned} S_{0,0} &= 0, \\ S_{i,0} &= -x_i, \qquad 1 \le i \le n \\ S_{0,j} &= -x_j, \qquad 1 \le j \le m \end{aligned}$$

Then

$$S_{i,j} = \max\{S_{i-1,j-1} + s(a_i, b_j), \max_{k \ge 1}\{S_{i,j-k} - x_k\}, \max_{k \ge 1}\{S_{i-k,j} - x_k\}\}$$

For single insertions and deletions only,

$$S_{i,j} = \max\{S_{i-1,j-i} + s(a_i, b_j), S_{i-1,j} - x_1, S_{i,j-1} - x_1\}$$

The idea of these calculations is that $S_{i,j}$ is the maximum similarity of $a_1 a_2 \ldots a_i$ and $b_1 b_2 \ldots b_j$. This is why for example, $S_{0,j} = -x_j$. The recursion is based on the ways an alignment can end:

$$\begin{matrix} \ldots a_i \\ \ldots b_j \end{matrix} \qquad \text{corresponds to } S_{i-1,j-1} + s(a_i, b_j)$$

and

$$\begin{matrix} \ldots - \\ \ldots b_j \end{matrix} \qquad \text{corresponds to } S_{i,j-1} - x_1$$

and so on.

When $n = m$, the multiple insertion and deletion program runs in time proportional to n^3, while the single insertion and deletion program runs in the much preferred time n^2. Fortunately, when the function x_k is linear in k, $x_k = \alpha + \beta * k$, the running time[15] can still be made n^2. Multiple insertions and deletions are important as adjacent bases are deleted or inserted by one event and should be weighted accordingly.

Alignments can be produced from the matrix by two methods. One is accomplished by saving pointers at each matrix entry that indicate what events were necessary to calculate $S_{i,j}$. Then beginning at $S_{n,m}$ pointers are followed to $S_{0,0}$, producing the optimal alignments. The second technique is, at each matrix entry beginning at $S_{n,m}$, to recompute to see which events produced the entry. Both methods take little time in comparison with the matrix construction.

For the examples, I first align 5S *Escherichia coli* (*rrnB* operon)[19] with

[19] V. A. Erdmann, E. Huysman, A. Vandenbe, and R. De Wachter, *Nucleic Acids Res.* **11**, r105 (1983).

the closely related 5S *Beneckia harveyi*.[20] As mentioned above,

$$s(a,b) = \begin{cases} +1, & \text{if } a = b \\ -1, & \text{if } a \neq b \end{cases}$$

and $x_1 = 2$. ($x_k = \infty$, $k \geq 2$, so that only single insertions and deletions are allowed.) The two sequences have similarity of 76 with alignment

```
UGCCUGGCGGCAGUAGCGCGGUGGUCCCACCUGACC-CCAUGCCGAACUCAGAAGUGAAACGC
UGCUUGGCGACCAUAGCGAUUUGGACCCACCUGACUUCCAUUCCGAACUCAGAAGUGAAACGA

CGUAGCGCCGAUGGUAGUGUGGGGUCUCCCCAUGCGAGAGUAGGGAACUGCCAGGCAU
AUUAGCGCCGAUGGUAGUGUGGGGCUUCCCCAUGUGAGAGUAGGACAUCGCCAGGCUU
```

There is a second optimal alignment, which results from simply changing

$$\begin{matrix} \text{C-} \\ \text{UU} \end{matrix} \quad \text{to} \quad \begin{matrix} \text{-C} \\ \text{UU} \end{matrix}$$

For a second example, I align 5S *E. coli* with the more distantly related 5S *Mycoplasma capricolum*.[21] Here the similarity is 22, with 52,020 different but optimal alignments. One of these 52,020 is given next with the portions common to all alignments in boxes.

```
UGCCUGGCGGCAGUAGCGCGGUGGUCCCACCUGACCCCAUGCCGAACUCAGAAGUGAAACGCC
U---UGGUGGUA-UAGCAUAGAGGUCACACCUGUUCCCAUGCCGAACACAGAAGUUAAGCUCU

GUAGCGCCGAUGGUAGUGUGGGGUCUCCCCAUGCGAGA-G-UAGGGAACUGCCAGGCAU
AUUACG--G-UGAAAAUAU---UACUU---AUGUGAGAAGAUAGCAAGCUGCCAGU--U
```

Obviously 52,020 alignments are too many to look at individually. The idea of displaying features common to all alignments is a minimal approach for dealing with these difficulties. As will be seen in the next section, a maximum segments algorithm will produce much of the same information.

Maximum Segments

This is a preferred method for exploring sequence relationships if computer time is available.[17] See Smith *et al.*[22] for a data base search with the algorithm. It consists of a simple alteration of the preceding method. The

[20] K. R. Luehrsen and G. E. Fox, *J. Mol. Evol.* **17**, 52 (1981).

[21] H. Hori, M. Sawada, S. Osawa, K. Murao, and H. Ishikura, *Nucleic Acids Res.* **9**, 5407 (1981).

[22] T. F. Smith, M. S. Waterman, and C. Burks, *Nucleic Acids Res.* **13**, 645 (1985).

recursively defined matrix here is H and recursion begins with

$$H_{0,0} = 0,$$
$$H_{i,0} = 0, \qquad 1 \le i \le n$$
$$H_{0,j} = 0, \qquad 1 \le j \le m$$

and

$$H_{i,j} = \max\{H_{i-1,j-1} + s(a_i, b_j), \max_{k \ge 1}\{H_{i,j-k} - x_k\}, \max_{k \ge 1}\{H_{i-k,j} - x_k\}, 0\}$$

For single insertions and deletions only,

$$H_{i,j} = \max\{H_{i-1,j-1} + s(a_i, b_j), H_{i-1,j} - x_1, H_{i,j-1} - x_1, 0\}$$

Notice that a simple addition of 0s in the boundary conditions and recursion is the only change from the definition of S. These simple changes have the pleasant effect of causing $H_{i,j}$ to be the maximum similarity of *all possible* segments ending in a_i and b_j.

After the 1981 Smith–Waterman algorithm[17] much work has gone into finding second, third, . . . best segment matches.[23] Fortunately there is also a simple, useful solution to this question. There is no problem with the observation that the best segment matching is associated with the entries where $\max_{i,j} H_{i,j}$ is achieved. Since large entries influence the entries nearby, it is not clear whether or not the second-best matching is near the first or not. One way to make certain of finding the second best is to recalculate the matrix, only this time no match, mismatch, insertion, or deletion from the maximum segment can be used. With single or linear insertion and deletion functions, only a small part of the matrix need be recomputed.

To illustrate the algorithm,[18] I compare the sequences 5S *E. coli* and 5S *Mycoplasma capricolum* as above. There the similarity was 22, with 52,020 different but optimal alignments. Asking for all nonintersecting segment alignments with score 10 or larger produces two alignments, the first with score 29 and the second with score 11:

JGGCGGCAGUAGCGCGGUGGUCCCACCUGACCCCAUGCCGAACUCAGAAGUGAAAUGCGAGA-
JGGUGGUA-UAGCAUAGAGGUCACACCUGUUCCCAUGCCGAACACAGAAGUUAAAUGUGAGAA

G-UAGGGAACUGCCAG
GAUAGCAAGCUGCCAG

[23] P. Sellers, *Bull. Math. Biol.* **46**, 501 (1984).

These two segment matches comprise most of the alignment that is common to all 52,020 optimal full sequence alignments.

A portion of the first matrix calculation is given in Table I. No recalculation has been done so that the reader can check understanding of the method.

As a final example, 5S *E. coli* is compared to *E. coli* Phe-tRNA. This second sequence (from GenBank) is CCCGGAUAGCUCAGUCGG UAGAGCAGGGGAUUGAAAAUCCCCGUGUCCUUGGUUCGAUU CCGAUCCGGGCACCA. There are five segment matches with scores greater than or equal to 6. Each is shown, 5S on top, with sequence position numbers indicated. Two segment matches have score 8:

```
99 AGAGUAGGGAACUG
20 AGAGCAGGGGAUUG

65 UAGCGCCGAUGGUAGUGUGGGG
07 UAGCUCAGUCGGUAGAGCAGGG
```

The remaining three segments have score 6:

```
89 UCCCCAUG
38 UCCCCGUG

13 GUAGCGCGGUGG
18 GUAGAGCAGGGG

14 UAGCGCGGU-GGU
07 UAGCUCAGUCGGU
```

Consensus Patterns

Perhaps the most important, certainly one of the most difficult, tasks of genetic sequence analysis is that of finding unknown patterns that occur imperfectly among a set of sequences or within a single sequence. Usually these searches are for approximately conserved regions that have functional significance. If the patterns are exactly conserved they are easily found; this is frequently not the case. These conserved patterns include the famous boxes such as TATAAT in bacterial promoters and CAAT in eukaryotic promoters, as well as enhancer and other sequences. Generally, gene regulation seems to involve some repeated protein binding sites. The question treated in this section is how to search for these unknown patterns.

When the problem is to locate shorter unknown patterns that occur imperfectly among a set of several sequences, the task seems hopeless due to the combinatorial nature of aligning many sequences. Aligning 10 sequences with 10 possible positions or shifts per sequence gives a total of

TABLE I

MAXIMUM SEGMENTS MATRIX FOR 5S *E. coli* AND 5S *Mycoplasma capricolum*

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

10^{10} possible alignments. This is at the limits of computability of modern computers with no consideration given to sequence length or to analysis of the alignments themselves for consensus patterns; any larger problem is clearly impossible. This is an instance of a so-called combinatorial explosion. Fortunately, recently developed methods[8] are able to find these consensus patterns in reasonable time for large data sets.

The first problem type considered is consensus patterns in a single sequence. Solutions are available for the "best" consensus repeat or palindrome. The found patterns are not allowed to overlap, and all patterns of a chosen word size (k) are considered. Mismatches, insertions, and deletions are allowed; the algorithm can locate a consensus word not actually occurring in the sequence itself. The second problem type is repeating words or palindromes between a set of sequences. In either case the analysis is more often limited by storage than by computation time. All 4^k words (patterns) of length k are stored for DNA and RNA; in proteins the storage is 20^k. Thus this analysis is suitable for smaller patterns.

Single Sequence Patterns

Under consideration first is finding a given length consensus palindrome in a single sequence. The procedure described here will locate the best nonoverlapping palindrome. If overlapping patterns are of interest, an easy modification will allow that option. Actually there are two problems of interest: (a) that of finding nonoverlapping palindromes of some given length, regardless of composition, and (b) that of finding a specific (but unknown) palindrome of given length. It is important to allow some amount of mismatch in these patterns. Examples are

GAGGGCTGTTTATATGAGTGCTACCAATGG (a)

CAACGTTGATACGTTCTTAAACCTTTATCT (b)

where in (a) three nonidentical palindromes (allowing one mismatch) are marked and where in (b) three versions of AACGTT are marked. The second problem is more difficult and is discussed here.

The following method seems natural but has serious flaws: take each word of the fixed length actually occurring in the sequence and, if it is an approximate palindrome, use it as a template along the entire sequence to see if it has any other approximate nonoverlapping occurrences. There are two difficulties with this naive procedure. (1) Such a method takes computing time proportional to the square of the length of the sequence times the pattern length. (2) It is frequently the case that the best consensus palindrome never exactly occurs in the sequence. While (2) is the most important objection, it is overcome by an algorithm which has computer time linear with sequence length.

Initially the length of palindrome of interest must be chosen. Odd-length palindromes of length $2k + 1$ are of the form

$$X_1X_2 \ldots X_kNY_k \ldots Y_2Y_1$$

where $\overline{X}_1 = Y_1$ ($\overline{A} = T$, etc.), $N = \{A, T, G, C\}$, while even lengths of $2k$ do not have the N in the center. Therefore all length $2k + 1$ or length $2k$ palindromes can be encoded by 4^k patterns.

Now the word $w = ATTCCGAT$ is considered to illustrate the ideas. If up to three mismatches (mm) are allowed, the following palindromes are within the neighborhood of w.

Length 6 Palindromes	mm from $w = ATTCCGAT$
ATTCGAAT	2
ATTGCAAT	2
ATCCGGAT	2
ATCGCGAT	2
ATTATAAT	3
ATTTAAAT	3
ATCATGAT	3
ATCTAGAT	3
ATAGCTAT	3
ATGGCCAT	3
ATACGTAT	3
ATGCGCAT	3
ATACGTAT	3

Obviously the situation is more complex if for example the mismatch level is raised to 5. Here, for example, the pair $X_1 * Y_1 = A * T$ can be altered, for example to G * C, giving an additional two mismatches. This general idea of finding a neighborhood of exact palindromes within mm mismatches of an existing word of the sequence is the basis on which the palindrome algorithms are built.

Consider the problem of finding the palindrome pattern of length $2k$ (or $2k + 1$) that occurs most frequently, nonoverlapping, in the given sequence. Of course none of these occurrences needs be exact so a weight $\lambda_w(v)$ can be introduced for a word v that is a fixed number of mismatches from a given palindrome w. Here if there are mm mismatches, $\lambda = \text{mm}/(\text{word length})$. Let $S_w(i)$ be defined by

$$S_w(i) = \max\{\Sigma_v \lambda_w(v)\}$$

when the sum is over nonoverlapping words v of the sequence $a_1a_2 \ldots a_n$. Suppose $S_w(j)$ is known for $1 \leq j \leq i$ and all w. Then to find

palindromes of length $2k$

$$S_w(i+1) = \max\{S_w(i);\ S_w(i-2k) + \lambda_w(a_{i-2k+1}a_{i-2k+2}\ \ldots\ a_i)\}$$

The consensus word has the maximum value of $S_w(n)$.

This formula shows that the calculation is linear with sequence length, an extremely valuable feature. If an overabundance of palindromes (not

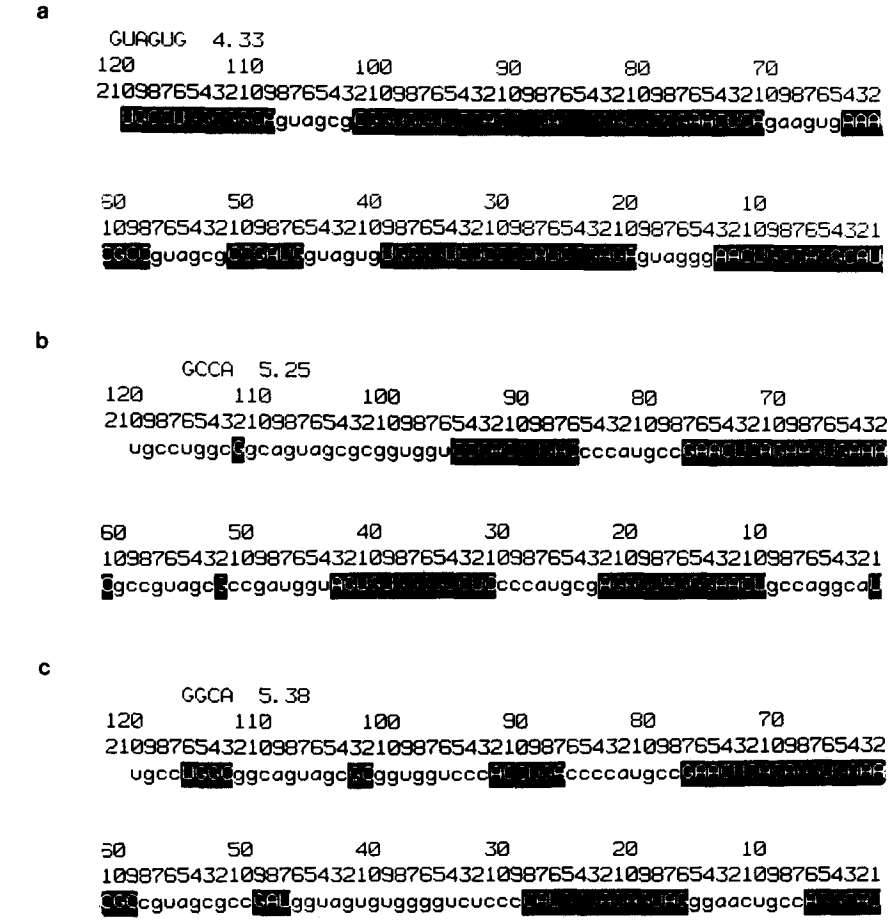

FIG. 1. Consensus patterns in *E. coli* 5S RNA. Base number 120 is the 5′-end of the sequence. (a) The result of a search for 6-letter repeats, allowing one mismatch. The pattern GUAGUG occurs five times with a score of 4.33, and the versions of the pattern are shown in lower case letters. (b) The result of a search for 8-letter palindromes, allowing up to three mismatches. The palindrome GCCAUGGC occurs eight times within the mismatches allowed. (c) The result of a search for 9-letter palindromes, allowing up to three mismatches. The palindrome GGCANUGCC is found seven times.

necessarily identical) is of interest, the above procedure is easily applicable. The neighborhood calculations need not be done, only counting of complementary pairs in the word.

Very similar, although somewhat simpler, concepts can be applied for an algorithm that finds consensus fixed-length repeats in a single sequence. This completes discussion of concepts for a single sequence and these algorithms are now applied to 5S *E. coli*. See Fig. 1.

Multiple Sequence Patterns

Much the same ideas from the single-sequence case are employed to study consensus patterns in a set of sequences.[7,8] The basic parameters of the method are a window width W, word size k, and the level of mismatches allowed. (Insertions and deletions can be included in the method.) With the window positioned on the sequences, the search finds the highest scoring word where each word w is given its highest score $\lambda_i(w)$, occurring within window W in sequence i. The score of w is

$$S(w) = \Sigma_i \lambda_i(w)$$

and the object is to find the winning word w^*

$$S(w^*) = \max_w S(w)$$

For each window position, $S(w^*)$ is plotted, and the resulting data can be graphed and examined for features of interest. Patterns in the sequences that correspond to the score $S(w^*)$ can be displayed.

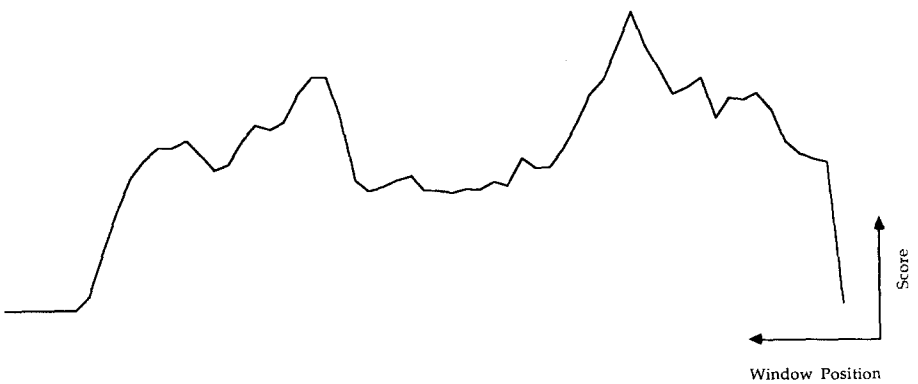

FIG. 2. Consensus pattern scores for a set of 30 *E. coli* promoter sequences aligned on start sites, with window size 9, word size $k = 6$, and up to 2 mismatches allowed. The graph gives score versus position of right edge of window. The position, scores, and patterns creating the scores can be directly related to the sequences themselves, and the graph gives direction to directly studying the sequences. See Fig. 3.

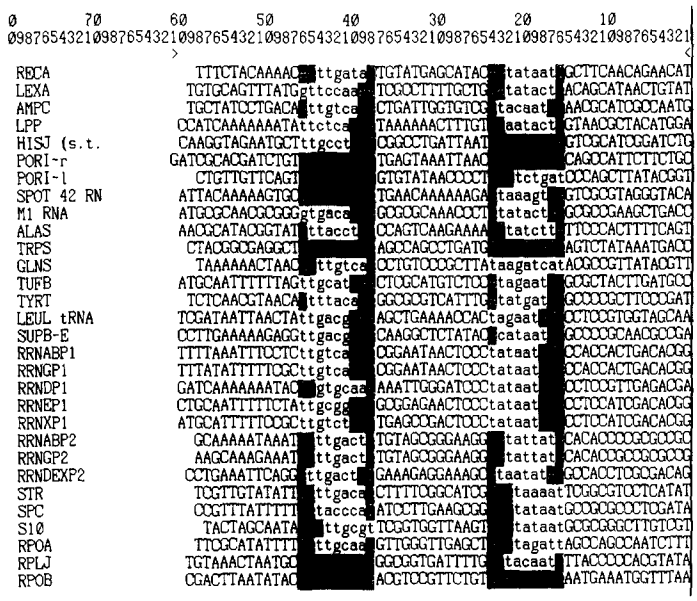

FIG. 3. The sequences and patterns creating the two major peaks in Fig. 2. The right-hand pattern has score 21.83 and is the famous -10 consensus, TATAAT. The left-hand pattern has score 17.17 and is the -35 consensus pattern.

It is convenient to understand these methods with an example. A subset of 30 *E. coli* promoter sequences[24] is aligned on known mRNA start sites. The sequences are approximately 60 bases long. They are analyzed with $W = 9$, $k = 6$, and maximum mismatches mm = 2. The resulting plot appears in Fig. 2. The sequences, their descriptions, and the patterns causing the highest peak appear in Fig. 3. The rightmost pattern is TATAAT (the famous -10 pattern) which has 8 exact occurrences, 11 with 1 mm, and 7 with 2 mm. The leftmost pattern is the -35 pattern, which has its strongest representation in these data as TTGCCA; it has 7 occurrences with 1 mm and 17 with 2 mm.

It is possible to analyze the sequences with the sequences written in the RY, or any other alphabet. There are no patterns found in the RY alphabet which are not found in the usual four-letter alphabet.

Finally, similar ideas can be used to find consensus palindromes in a set of sequences. Since protein-binding sites sometimes are palindromes, this is a useful tool.

[24] D. K. Hawley and W. R. McClure, *Nucleic Acids Res.* **11**, 2237 (1983).

Secondary Structure

The first tRNA sequence by Holley *et al.*[25] in 1965 was published with a secondary structure inferred from the sequence. Since that time many other secondary structures have been presented and the problem of how to estimate structure by experimental as well as theoretical techniques has frequently been addressed. The current structures of 16S and 23S RNA were found by a combination of both approaches.[9,10] Here of course I restrict discussion of computer methods for prediction of secondary structure.

Two major approaches have emerged for the computer analysis. The first employs dynamic programming, in an algorithm closely related to the sequence comparison algorithms presented above. Tinoco *et al.*[26] presented a base pair matrix [with (i,j) entry = 1 if base i will pair base j] which led them to consideration of minimum energy structures. Dynamic programming algorithms came later and are the current method of choice for a single sequence. The second class of methods employs consensus ideas, tracing back to early deduction of a common tRNA structure.[27] Woese, Noller, and colleagues have advanced and refined these methods, and I discuss a mathematical version of this consensus analysis below.

If the data consist of a single sequence then the dynamic programming approach is recommended. On the other hand, if the data are a set of sequences suspected to have common structural elements, then the consensus method can succeed in cases when dynamic programming cannot.

Folding by Dynamic Programming

The application of dynamic programming to secondary structure prediction was begun by two groups. The approach of one group had the advantage of incorporating general loop, bulge, and base pairing free energy functions; the disadvantage was the building up of complex structures from simpler ones.[28,29] The other group did optimization in one pass but only found structures with maximum base pairing.[30]

Since the presentation of the first algorithms, Zuker has become the

[25] R. W. Holley, J. Apgar, G. A. Everett, J. T. Madison, M. Marquisee, S. H. Merrill, J. R. Penswick, and A. Zamir, *Science* **147,** 1462 (1965).
[26] I. Tinoco, O. C. Uhlenbeck, and M. D. Levine, *Nature (London)* **230,** 362 (1971).
[27] M. Levitt, *Nature (London)* **224,** 759 (1969).
[28] M. S. Waterman, *Adv. Math. Suppl. Studies* **1,** 167 (1978).
[29] M. S. Waterman and T. F. Smith, *Math. Biosci.* **42,** 257 (1978).
[30] R. Nussinov, G. Pieczenik, J. R. Griggs, and D. J. Kleitman, *SIAM J. Appl. Math.* **35,** 68 (1978).

leading figure with his useful dynamic programming codes (see Zuker and Sankoff for a review).[31] He has combined realistic energy functions into a single pass algorithm that is quite efficient. His program runs in time proportional to n^3, where n = sequence length, and it requires storage n^2. Fully rigorous prediction takes exponential time (which is unacceptable) and n^2 storage. Recently it was shown[32] that, by increasing storage to n^3, the exponential time can be reduced to n^4. None of this should deeply concern someone with a sequence to fold. Zuker's efficient and useful code is recommended.

To understand why complicated programs are needed to study RNA folding, I briefly consider the number of candidate structures.[33] If $F(n)$ is the number of secondary structures for a sequence of length n, it is required that $F(0) = F(1) = F(2) = F(3) = 1$; that is, there must be at least 2 bases in an end loop. Since secondary structures do not include knotted structures, a recursion is obtained:

$$F(n+1) = F(n) + \sum_{1 \le j \le n-2} F(j-1)F(n-j), \qquad n \ge 3$$

This formula counts all possible structures, forcing pairs between base j and base $n+1$. The recursion can be shown[33] for large n to behave like

$$F(n) \cong [(1+\sqrt{2})/\pi]^{1/2} n^{-3/2} (1+\sqrt{2})^n$$

For $n = 150$, $F(n) \cong 1.2 \times 10^{54}$. Even with all allowances for the overcount as compared to real sequences, these large numbers show that an algorithm is needed.

Surprisingly, the dynamic programming algorithms are based on logic similar to that for the counting. The maximum number of base pairs algorithm goes like this: let $M_{i,j}$ = maximum number of base pairs in the sequence segment from base i to base j. Take $M_{i,j}$ known for $0 \le i,j \le n$. Then add base $n+1$. If, in the optimal structure base $n+1$ is unpaired, $M_{1,n+1} = M_{1,n}$. Otherwise, base $n+1$ is paired to j, where $1 \le j \le n-2$. Then

$$M_{1,n+1} = M_{1,j-1} + 1 + M_{j+1,n}$$

Here the 1 counts the new base pair between j and $n+1$, while the other two terms are optimal for the other two pieces of sequence. To collect this into a recursion, let $\delta_{i,j} = 1$ if bases i and j can pair and 0 otherwise. Then

$$M_{i,j} = \max\{M_{1,n}; \max_{1 \le j \le n-2}\{M_{1,j-1} + 1 + M_{j+1,n}\}\delta_{i,j}\}$$

[31] M. Zuker and D. Sankoff, *Bull. Math. Biol.* **46**, 591 (1984).
[32] M. S. Waterman and T. F. Smith, *Adv. Appl. Math.*, **7**, 455 (1986).
[33] P. R. Stein and M. S. Waterman, *Discrete Math.* **26**, 261 (1978).

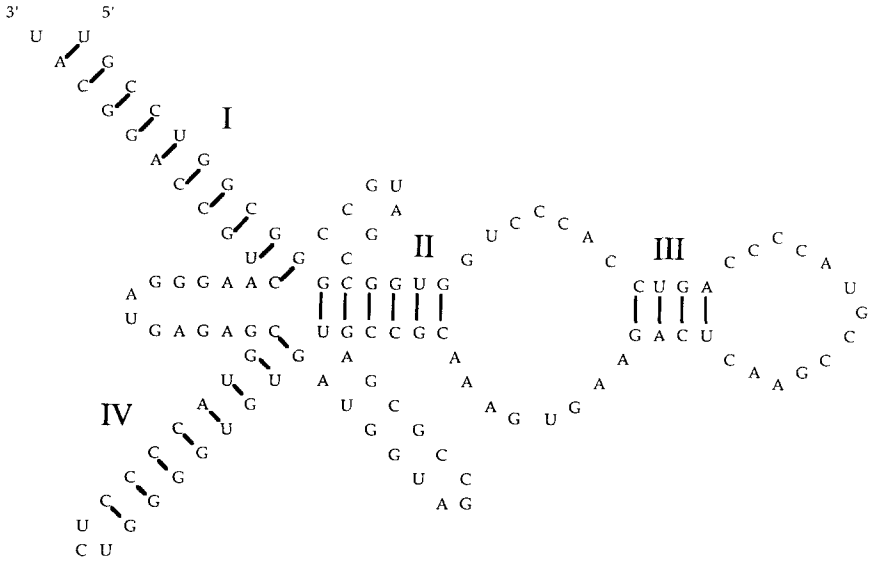

FIG. 4. A secondary structure for *E. coli* 5S RNA.

All the complication of algorithms, coding, and running times comes in converting this simple, elegant idea to handle the various free energy functions associated with base pairs, bulges, interior loops, and multibranch loops. This is a difficult task!

Folding by Consensus

The consensus methods of folding are sometimes referred to as comparative methods. Levitt[27] in 1969 gave an analysis of the known tRNA sequences by this approach. In contrast with his impressive results, the dynamic programming codes currently fold about 50% of tRNAs into a cloverleaf. More recently, comparative methods were used by Woese, Noller, and colleagues[9,10] to solve 16S and 23S structures. I now describe programs and mathematics to fold a set of RNAs. The ideas are based on the insights of Woese and Noller but differ from their methods by being able to perform systematic, complete, and explicitly defined searches.

The algorithms will be illustrated by 34 5S sequences from *E. coli* and related sequences that were obtained from a collection of Olsen and Pace and which can be found in GenBank. A 5S model for *E. coli* is taken from the literature[34] and is shown in Fig. 4 for reference.

[34] B. Lewin, "Genes," 3rd Ed. Wiley, New York, 1987.

This analysis is very different from dynamic programming: here it is desired to find many "common" helices of a certain size and quality. No minimum energy calculations are made. The base pairs A * U, G * C, and G * U are allowed. These helices are allowed to shift in location with reference to the sequences in some fixed alignment. Two windows are placed on the sequences and it is these windows which determine the shifting. For example with

WINDOW 1	WINDOW 2
AUGG***	********CCGU

the 4-base pair helix $\overset{\text{AUGG}}{\text{UGCC}}$ is formed and the patterns could appear anywhere in their windows. Window positions determine approximate helix position while window width determines the amount of shifting allowed. It is not required that the helices in the various sequences be composed of the same base pairs. These features will be illustrated with the 5S sequence set.

The sequences must be aligned initially. Obvious features to align on are the right and left ends of the sequences. This can be done in three ways: (1) align on left ends, (2) align on right ends, and (3) align on both ends, leaving gaps in the center of the shorter sequences. Other features to align on include known biological features or highly significant long patterns common to all the sequences. In our sequences such a pattern (cgaac) will prove useful. These various alignments are explored for common patterns of folding.

In Fig. 5, the longest common pattern is seen to be cga, shown in lowercase letters in the figure. The pattern cgaac is in 32 of 34 sequence while ccgaac is in 31 of 34. Notice the small amount of shifting to achieve the alignment. The expected length E common to 32 of 34 random sequences[15] is given by

$$E = \log\left(\frac{34 * 33}{2} * 120^{32}\right) + \log\left(\frac{3}{4}\right) + 0.577 \log(e) - \frac{1}{2} = 2.77$$

and $\sigma = 0.29$ with all logs to the base 4.[31] Therefore cgaac occurs almost 8 σs above expected.

The first analysis of secondary structure now takes place. There are many ways to place two windows on the sequences. To organize the analysis, first fix the separation of the windows. First with no (0) separation, i.e., adjacent windows, move the two windows across the sequence set. Then increase the separation, moving across the set at each fixed separation until the windows are at the maximum separation. In each position the number of helices found is plotted. Mispairs are allowed. Thus each separation produces a graph. All these graphs are superimposed so that interesting peaks, representing a larger number of helices, can be

FIG. 5. A set of 34 5S sequences aligned on the consensus pattern CGAAC, appearing as lower case letters in the sequences.

located. As is seen below, this is an overwhelming amount of information. To handle these data, then, it is possible to move between graphs and the sequences to observe which sequence patterns produced the peaks.

With the 34 sequences in the alignment of Fig. 5, the algorithm is run with a window of 8, a helix size of 4, and no mispairings (mm = 0). The superimposed graphs are shown in Fig. 6a and several interesting peaks show up. The leftmost of these peaks is the result of helix III (see Fig. 4), and one of the graphs with separation 9 is shown in Fig. 6b. A sequence pattern or set of helices producing the highest peak in this graph has score 31, so that all but three of the sequences have this pattern.

By allowing two mispairings, the alignment shown in Fig. 7 is produced. This method of representing helices in secondary structure by

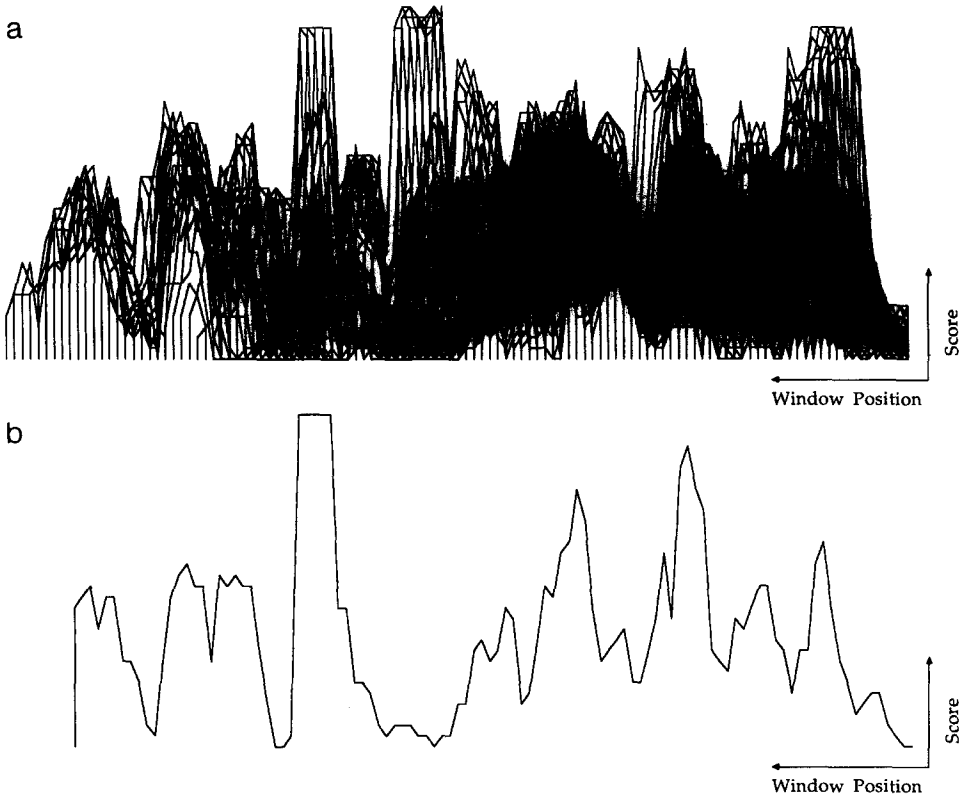

FIG. 6. Graphs of folding scores with window sizes 8, helix size 4, and no mispairing allowed. In a, all separations are plotted; in b, one of these graphs with separation 9 is shown. Sequence patterns causing these scores can be viewed.

FIG. 7. Alignment on the sequence helices causing the highest peak in Fig. 6b. Thirty-one of the sequences have 4-base pair helices, and up to two mispairings finds the folding in all sequences. This corresponds to helix III of Fig. 4.

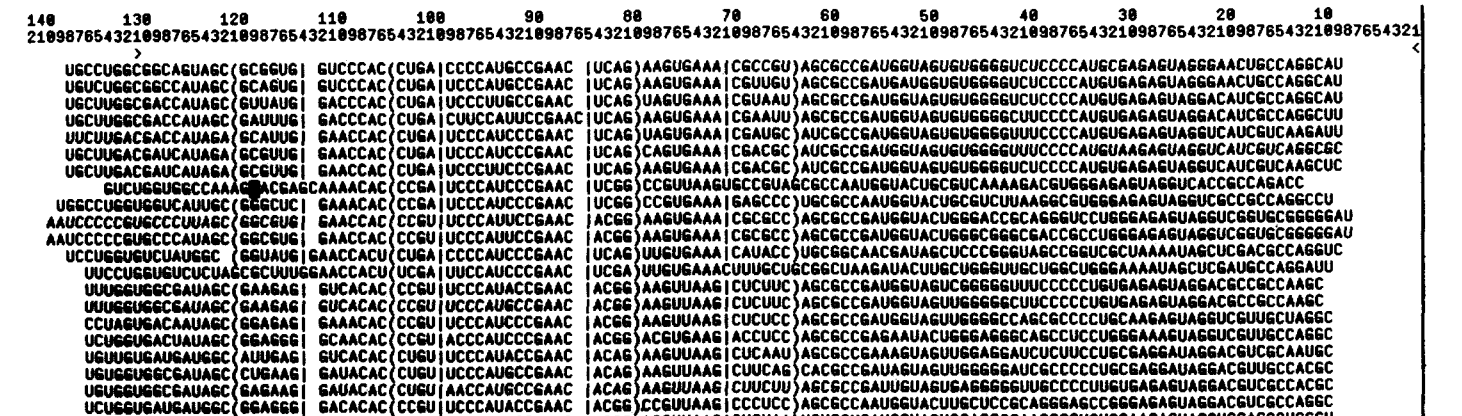

FIG. 8. Additional alignment on a second helix that corresponds to the highest set of peaks in Fig. 6a. The 6-base pair helix is aligned using one or less mispairs, and it corresponds to helix II in Fig. 4.

FIG. 9. Folding with all four consensus helices. The order of folding is helix III, helix II, helix IV, and helix I.

parentheses is unambiguous: for example,

5′ ((( )) ( )) 3′

represents the 5S structure of Fig. 4. The highest peak of Fig. 6a corresponds to helix II and this helix is added to the alignment in Fig. 8. Actually a helix of 6 base pairs fits best and mm = 1 is allowed. Three of the sequences do not obtain a helix according to these criteria and consequently do not have parentheses inserted.

The peaks of Fig. 6a which are the third highest group, those at the rightmost of the plot, correspond to helix I, while the fourth highest group corresponds to helix IV. Helix IV has 8 base pairs and all but one sequence has a helix with three or less mispairs. The final folding is shown in Fig. 9. The folding is achieved in an iterative manner: longest common pattern, helix III, helix II, helix IV, and finally helix I. Frequently finding one pattern assists in finding another.

It must be emphasized that the main concern here has been consensus helices and not simultaneous folding and alignment. These activities should properly be done together or iteratively as Woese and Noller do. Whenever there were multiple choices for a folding pattern the choice here is that giving the "best" consensus alignment. Clearly, additional work is needed to make the criteria more explicit.

What about additional folding patterns? This can be approached using the folded sequences. For a brief look, set $W = 10$ and helix size 4 with no mispairs allowed. The major helices I–IV already discussed are labeled in Fig. 10 and some other interesting patterns are labeled A, B, and C. These

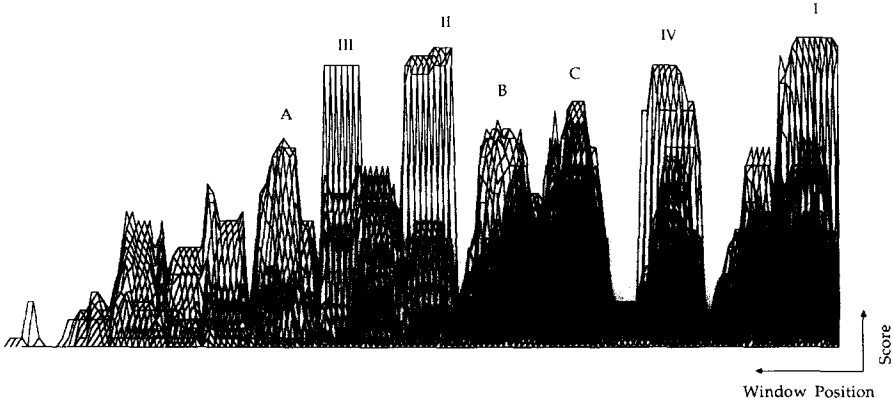

FIG. 10. Graphs of folding scores versus window position on the sequences aligned as in Fig. 9, with window size 10, helix length 4, and no mispairs. Peaks I–IV correspond to the found helices while A, B, and C were not observed earlier in this analysis.

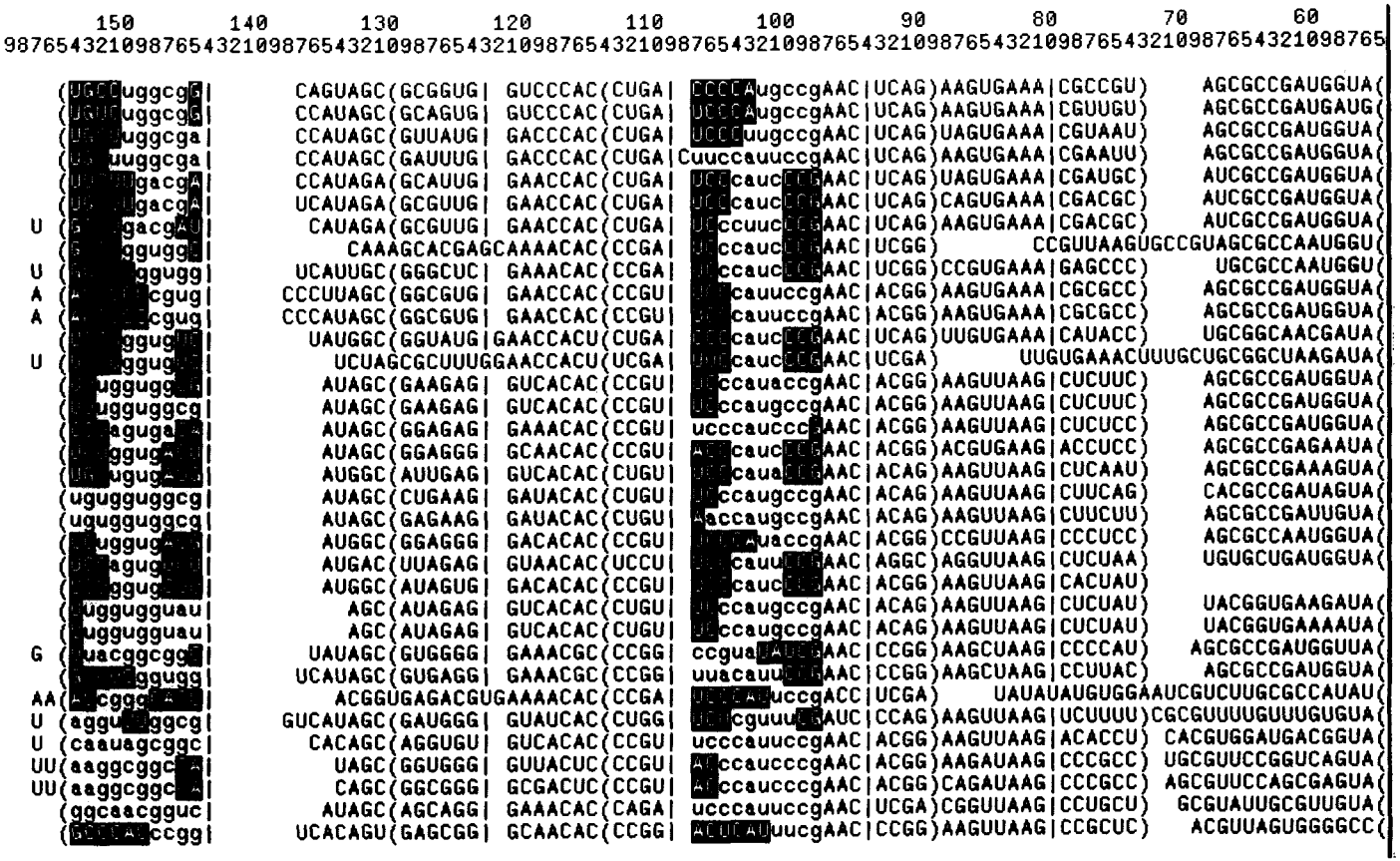

FIG. 11. The sequence patterns for peak A of Fig. 10, allowing one mispair. The 5′ part of the sequences making up helix I is interacting with the loop of helix III.

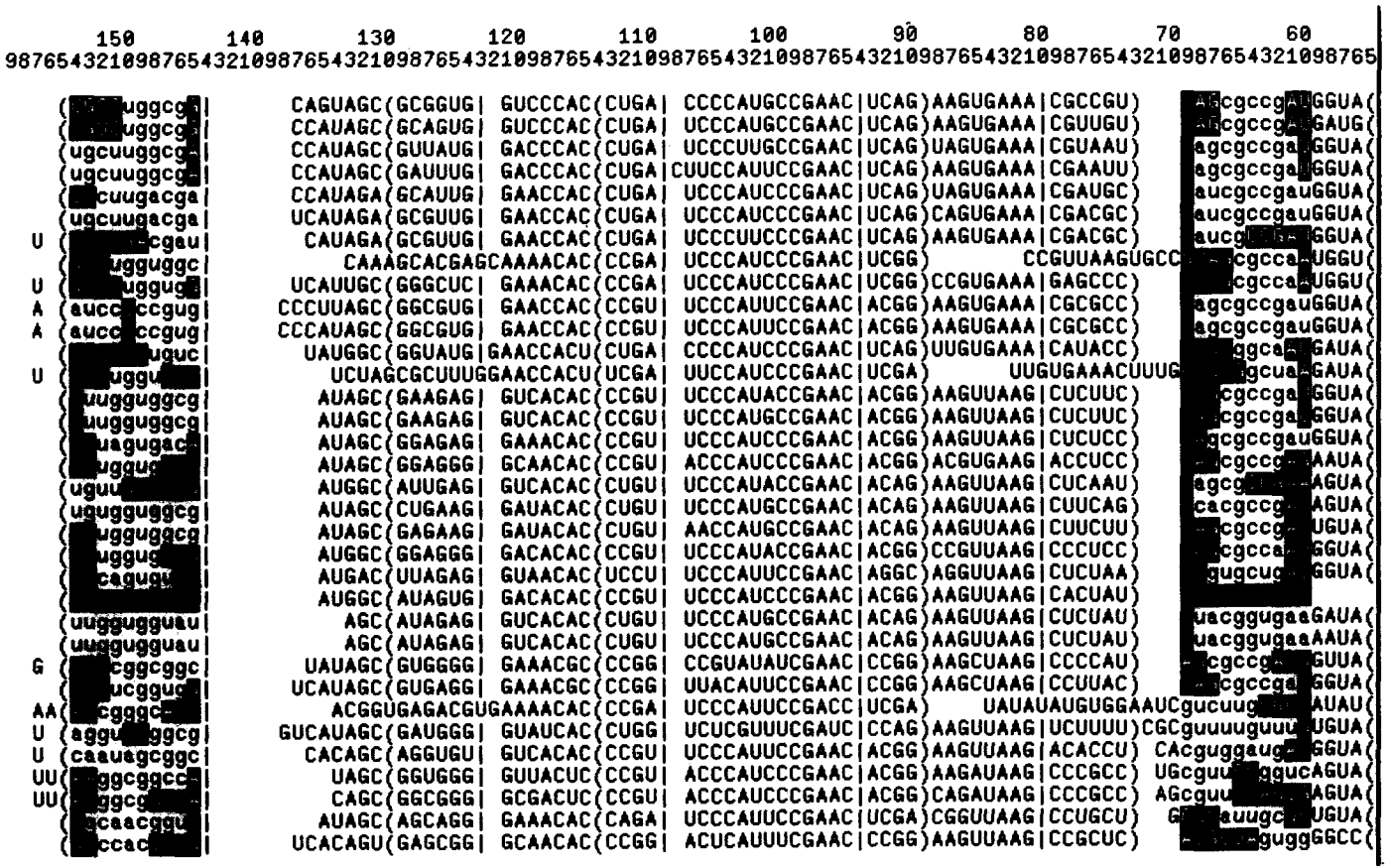

FIG. 12. The sequence patterns for peak B of Fig. 10, allowing one mispair. The 5′ part of the sequences making up helix I is interacting with the arm between helices IV and III.

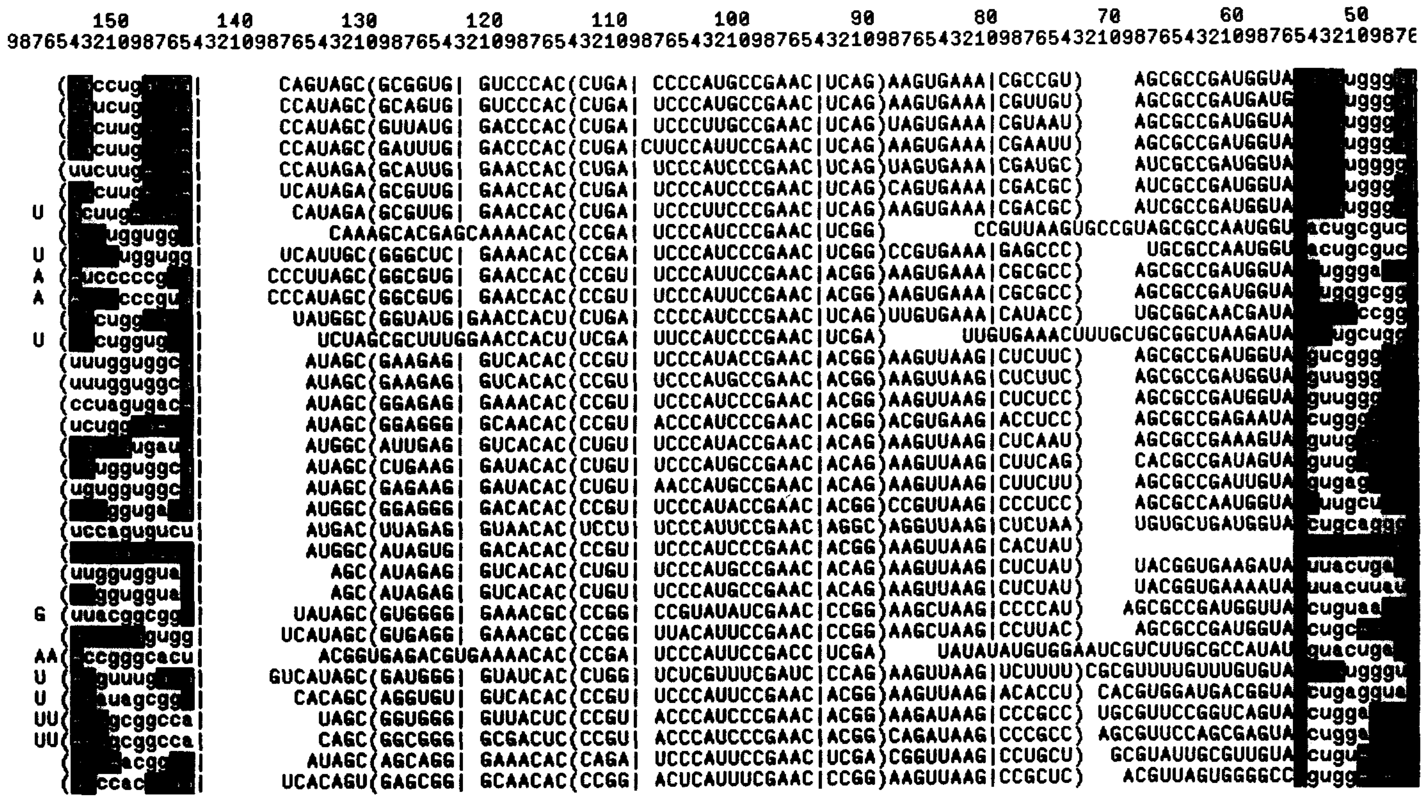

FIG. 13. The sequence patterns for peak A of Fig. 10, allowing one mispair. The 5′ part of the sequences making up helix I is interacting with the 5′ part of the sequences making up helix IV.

three peaks are all produced from interactions with the 5′ part of helix I (see Figs. 11–13). While I do not take the space to do so, an extremely detailed study of tertiary structure is possible.

Conclusions

I have by no means discussed all the topics that are important to computer analysis of sequences. Several others come to mind: statistical approaches to significance, consensus repeats for large patterns, and alignment of many sequences. I will give some thoughts on these topics in this section.

The issue of biological versus statistical significance frequently comes up. Biological significance is the goal here; all that statistics can do is provide hints about what might be taken seriously. Sequences can be viewed as satisfying some model of randomness, such as uniform and independent bases, and the analyst might ask whether some observed pattern is the result of sequence conservation or simply is to be expected from sequences satisfying the model of randomness. Since each person doing the analysis might view randomness differently, there is a proliferation of different simulation techniques. I prefer simple assumptions of randomness, since for maximum segments problems they have been shown to model the matching of unrelated real sequences.[22] There are two theoretical approaches that are useful. The first is the theory of large deviations, which is extensively discussed by Galas *et al.*[8] This theory is appropriate when there are a large number of short sequences. When long sequences are matched, the recently developed extreme value theory provides excellent information about longest matchings between the sequences.[22] This so-called $\log(n)$ theory is discussed in Waterman.[15] While these theoretical approaches can be very useful, simulation is often resorted to because of sequence complications. For example, the *E. coli* promoter sequences have varying sequence composition and this complicates the large deviation theory. Fortunately, the distributions of the quantities of interest, such as highest peak or maximum segment score, do not have large variation, and a few simulations can give a good picture of the maximum expected from random sequences.

The algorithms that produce the consensus repeats analysis reported in this article depend on storing all patterns of interest. The methods do not generalize, for example, to finding 72-base pair repeats. What I can suggest is along the lines of a study in progress with I. Wool, J. McNally, and R. Jones that is concerned with 15- to 25-letter repeats in a large set of ribosomal protein sequences. We use each pattern of appropriate length actually occurring in the sequence set. Then we find the most often repeat-

ing sequence word. We use the found occurrences to modify the pattern and iterate. While this can hardly be said to be a highly efficient search, it is effective and much more informative than the usual approaches that try to analyze many pairwise comparisons.

Finally, I bring up the old problem of the alignment of many sequences. In the realm of dynamic programming, the usual combinatorial explosion sets in, and even three sequences are almost beyond reach. A solution can be based on the consensus word between many sequences, and an algorithm can be constructed to give the maximum sum of scores of consensus words. This practical algorithm[35] is very useful for multiple sequence alignment.

The computer analysis of nucleic acid sequences has produced some interesting mathematics, and some algorithms and programs useful for sequence analysis. As biology continues to gather sequence data at increasing rates and to find new, fundamental questions of interest, the mathematics and computer science to solve related questions will also progress.

Acknowledgments

I appreciate receiving the 5S collection from Gary Olsen and Norm Pace. Much assistance from Mark Eggert and Peter Sugiono is gratefully acknowledged. This work was supported by the System Development Foundation and the National Institutes of Health.

[35] M. S. Waterman, *Nucleic Acids Res.* **14**, 9095 (1986).

[53] Phylogenetic Analysis Using Ribosomal RNA

By GARY J. OLSEN

The inference of phylogenetic relationships from molecular data (i.e., the field of molecular evolution)[1] is contributing greatly to our understanding of the evolution of life on Earth. Although the discussion that follows is directed toward analyses based on rRNA sequences, nearly all of the concepts, and many of the details, are equally applicable to the other DNA, RNA, or protein sequences. The rRNAs will be identified by their typical prokaryotic sedimentation values: 5S, 16S, and 23S. No issue will be made of the fragmentations of these RNAs in some organisms (giving rise to the 5.8S, 4.5S, 2S, etc. rRNAs) or of the absence of 5S rRNA in some mitochondria.

The merits of rRNA for phylogenetic inference have been extensively

[1] E. Zuckerkandl and L. Pauling, *J. Theor. Biol.* **8**, 357 (1965).