Proc. Natl. Acad. Sci. USA
Vol. 80, pp. 3123–3124, May 1983
Applied Mathematical Sciences

Sequence alignments in the neighborhood of the optimum with general application to dynamic programming

(algorithms/discrete dynamic programming/Kth shortest paths/sequence distance/sequence homology)

MICHAEL S. WATERMAN

Departments of Mathematics and Biological Sciences, University of Southern California, Los Angeles, California 90089-1113

Communicated by Mark Kac, December 15, 1982

ABSTRACT **When applying dynamic programming techniques to obtain optimal sequence alignments, a set of weights must be assigned to mismatches, insertion/deletions, etc. These weights are not predetermined, although efforts are being made to deduce biologically meaningful values from data. In addition, there are sometimes unknown constraints on the sequences that cause the "true" alignment to disagree with the optimum (computer) solution. To assist in overcoming these difficulties, an algorithm has been developed to produce all alignments within a specified distance of the optimum. The distance can be chosen after the optimum is computed, and the algorithm can be repeated at will. Earlier algorithms to solve this problem were very complex and not practical for any case involving sequences with significant time or storage requirements. The algorithm presented here overcomes these difficulties and has application to general, discrete dynamic programming problems.**

The first dynamic programming algorithm to yield optimal sequence alignments was due to Needleman and Wunsch (1). Extensions were made by Sellers (2, 3) and Waterman *et al.* (4). Relationships between these algorithms are discussed in the review by Smith *et al.* (5). These algorithms find the minimum weighted changes to convert one sequence into another.

Recently Fitch and Smith (6) considered the effect of the weights that are assigned to the events of mismatch and insertion/deletion. They chose short sequences of chicken hemoglobin α and β chains, where the correct alignment is assumed to be known. Interestingly, they conclude that the only algorithm to give the correct alignment is that discussed in Smith *et al.* (5). Basically the reason is that insertion/deletion events must be included and correctly weighted.

The subject of this paper is an algorithm that yields all alignments within a user-specified distance of the optimum alignment value. Even with "incorrect" alignment weights, there is reason to believe that the "correct" alignment is not too far from the optimum for alignments of truly related sequences. In referring to dynamic programming solutions of the minimum free energy RNA secondary structure problem—an analogous problem—Dumas and Ninio (7) comment, "unfortunately, they give by design only one structure so that the biologically important folding may be missed." Their comments, in part, motivated the present paper.

The dynamic programming/operations research literature has considered these problems, beginning with Bellman and Kalaba (8). Since then, several authors have studied what is referred to as the *K*th best-path problem (9, 10). To find the *K*th best path, the original problem and the preceding $(K-1)$ best-path problems must be solved. Storage and calculation for each path is somewhat more expensive in time and storage than the preceding one.

The publication costs of this article were defrayed in part by page charge payment. This article must therefore be hereby marked "*advertisement*" in accordance with 18 U.S.C. §1734 solely to indicate this fact.

When problems are solved where the initial optimum takes significant time and storage, the *K*th best-path methods of operations research are not easy to implement. In particular, new methods are needed for the sequence alignment problem. Fortunately, the approach of this paper allows solution of this problem in a simple manner. A paper with Thomas Byers (11) considers in detail the application of this algorithm to general, discrete dynamic programming problems.

OPTIMAL ALIGNMENTS

The algorithm of Sellers gives the minimum weighted sum of substitutions and insertion/deletions to convert the sequence $\mathbf{A} = a_1a_2 \ldots a_n$ into $\mathbf{B} = b_1b_2 \ldots b_m$. A distance $d(a,b)$ (or weight) is given, and an insertion/deletion of a is given weight $d(a,\Delta) = d(\Delta,a)$, where Δ is our symbol for a blank or space. For simplicity we restrict discussion in this section to single insertion/deletions, but later an example is presented in which insertions/deletions of length greater than one are included.

To find the distance between **A** and **B**, set up a matrix D. Initialize by $D_{k,0} = k$ $(0 \le k \le n)$ and $D_{0,\ell} = \ell$ $(0 \le \ell \le m)$. Values of $D_{i,j}$ are interpreted as the distance between $a_1a_2 \ldots a_i$ and $b_1b_2 \ldots b_j$. The values are obtained by

$$D_{i,j} = \min \{D_{i-1,j} + d(a_i,\Delta), D_{i-1,j-1} + d(a_i,b_j), D_{i,j-1} + d(\Delta,b_j)\}.$$

This procedure calculates $D_{n,m}$, the value associated with the optimal alignment(s). To produce these alignments, begin at $D_{n,m}$. The optimal step(s) leading to $D_{n,m}$ reveals the right-hand end of the alignment. Succeeding steps (back) through the matrix will produce the remainder of the optimal alignment(s) in time proportional to the length of the longest sequence.

For illustration, take **A** = A-U-A-A-A and **B** = A-U-G-G-A-A-A, where matches have 0 weight while mismatches and insertion/deletions have weight 1. The resulting D and traceback are given in Fig. 1. The corresponding alignment with $D_{5,7} = 2$ is

A-U-G-G-A-A-A
A-U-Δ-Δ-A-A-A.

NEAR OPTIMAL ALIGNMENTS

The problem is to find all alignments with score within e of $D_{n,m}$, the best possible score. For $e \ge 0$, this set will include the optimal alignments. The approach here is to exploit the meaning of the entries $D_{i,j}$ and to modify the traceback procedure in an appropriate manner.

Suppose we are at position (i,j) and that we are performing a traceback that can result in an alignment with score less than or equal to $D_{n,m} + e$. Let the score of the alignment from (n,m) to but not including (i,j) be equal to $T_{i,j}$. $T_{i,j}$ is the sum of values of each step to reach (i,j) and, as such, can include several steps that are not optimal. There are three possible steps from (i,j),

	Δ	A	U	G	G	A	A	A
Δ	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
U	2	1	0	1	2	3	4	5
A	3	2	1	1	2	2	3	4
A	4	3	2	2	2	2	2	3
A	5	4	3	3	3	2	2	2

FIG. 1. $D_{i,j}$ matrix resulting from application of recursion to sequences A-U-A-A-A and A-U-G-G-A-A-A. The lines indicate the traceback.

namely $(i-1,j)$, $(i-1,j-1)$, and $(i,j-1)$. These steps are taken if, respectively,

$$T_{i,j} + d(a_i,\Delta) + D_{i-1,j} \leq D_{n,m} + e$$

$$T_{i,j} + d(a_i,b_j) + D_{i-1,j-1} \leq D_{n,m} + e$$

$$T_{i,j} + d(\Delta,b_j) + D_{i,j-1} \leq D_{n,m} + e.$$

The traceback algorithm is based on the reasoning that an alignment score can be decomposed into three parts: the score to the current position ($T_{i,j}$), the weight of the next step (d(,)), and the score of the "remaining" alignment. The best possible value for the "remaining" alignment is the corresponding value of D and, for this reason, that value of D appears in the test criteria.

It is necessary to save pointers at all locations where multiple steps are possible. In our program, we always execute the lower, left paths first and stack the pointers in a "last-in-first-out" manner. This implies that, for alignments with initial regions in common, the initial regions are only found once.

To illustrate the traceback methodology, consider the example of Fig. 1 with $D_{5,7} = 2$ and $e = 1$. One of the near optimal alignments is shown in Fig. 2. At position (2, 3) (a_2 = U and b_3 = G), $T_{2,3} = 1$, resulting from the alignment

G-A-A-A
Δ-A-A-A.

The three candidates are

$$1 + d(\mathrm{U},\Delta) + D_{1,3} = 1 + 1 + 2 = 4$$

$$1 + d(\mathrm{U},\mathrm{G}) + D_{1,2} = 1 + 1 + 1 = 3$$

$$1 + d(\Delta,\mathrm{G}) + D_{2,2} = 1 + 1 + 0 = 2.$$

Assume the lower path has been followed already and see that we can move to position (1, 2) with $T_{1,2} = 2$, resulting from $T_{1,2} = 1 + T_{2,3}$, the alignment being

G-G-A-A-A
U-Δ-A-A-A.

	Δ	A	U	G	G	A	A	A
Δ	0	1	2	3	4	5	6	7
A	1	0	1	2	3	4	5	6
U	2	1	0	1	2	3	4	5
A	3	2	1	1	2	2	3	4
A	4	3	2	2	2	2	2	3
A	5	4	3	3	3	2	2	2

FIG. 2. Nonoptimal alignment with score 3. The alignment is
A-U-G-G-A-A-A
A-Δ-U-Δ-A-A-A.

For $D_{5,7} = 2$ and $e = 1$, there are eight alignments with the score of 3 and one alignment (the optimum) with the score of 2.

α AND β CHAINS OF CHICKEN HEMOGLOBIN

A study of sequence alignment sensitivity to weights and the use of multiple insertion/deletions has been carried out by Fitch and Smith (6) and was discussed earlier in this paper. The sequences displayed below are chicken hemoglobin mRNA sequences, nucleotides 115–171 from the β chain (upper sequence) and 118–156 from the α chain (lower sequence).

U-U-U-G-C-G-U-C-C-U-U-U-G-G-G-A-A-C-C-U-
U-U-U-C-C-C-C-A-C-U-U-C-G A-U-C-U-

C-U-C-C-A-G-C-C-C-C-A-C-U-G-C-C-A-U-C-C-
U-U-G-U-C-A-C-A-C

G-G-C-A-A-C-C-C-C-A-U-G-G-U-C
G-G-C-U-C-C-G-C-U-C-A-A-A-U-C

This alignment is presumed correct from an analysis of the many known amino acid sequences for which such RNA sequences code.

While the algorithm for sequence alignment presented above included only single-base insertion/deletions, longer insertion/deletions, such as displayed in the alignment above, are unlikely to be the "sum" of single-base events. The Sellers' algorithm was extended by Waterman *et al.* (4) to include these events and allows the user to choose weights for each insertion/deletion event of length 1,2,3,...; Fitch and Smith (6) show, for the example of this section, that the correct alignment cannot be obtained without this capability.

By using a mismatch weight of 1 and a multiple insertion/deletion function of $2.5 + k$, where k is the insertion/deletion length, the correct alignment is found among the 14 optimal alignments. (This is region Q of the Fitch–Smith paper.) To indicate the size of neighborhoods in this example, there are 14 alignments within 0% of the optimum, 14 within 1%, 35 within 2%, 157 within 3%, 579 within 4%, and 1,317 within 5%.

A mismatch weight of 1 and a multiple insertion/deletion function of $2.5 + 0.5\,k$ is in region P of Fitch and Smith; accordingly, the correct alignment is not in the list of two optimal alignments. It is necessary to go to the list of 31 alignments with 4% of optimal alignment to find the correct alignment. This example illustrates the sensitivity of alignment to weighting functions and illustrates the utility of the method presented in this paper.

The author thanks Thomas Byers for programming assistance and useful comments. The University of California, San Francisco Departments of Biochemistry and Biochemistry and Biophysics and the System Development Foundation are each gratefully acknowledged for providing partial support of this work.

1. Needleman, S. & Wunsch, C. (1970) *J. Mol. Biol.* **42**, 245–261.
2. Sellers, P. (1974) *J. Comb. Theory Ser. A* **16**, 253–268.
3. Sellers, P. (1974) *J. SIAM* **26**, 787–793.
4. Waterman, M., Smith, T. & Beyer, W. (1976) *Adv. Math.* **20**, 367–387.
5. Smith, T., Waterman, M. & Fitch, W. (1982) *J. Mol. Evol.* **18**, 38–46.
6. Fitch, W. S. & Smith, T. F. (1983) *Proc. Natl. Acad. Sci. USA* **80**, 1382–1386.
7. Dumas, J. P. & Ninio, J. (1982) *Nucleic Acids Res.* **10**, 197–206.
8. Bellman, R. & Kalaba, R. (1960) *J. SIAM* **8**, 582–588.
9. Lawler, E. (1972) *Manage. Sci.* **18**, 401–405.
10. Lawler, E. (1976) *Combinatorial Optimization: Networks and Matroids* (Holt, Reinhart, & Winston, New York).
11. Byers, T. & Waterman, M. (1983) *Oper. Res.*, in press.