Simultaneous Solution of the RNA Folding, Alignment and Protosequence Problems

David Sankoff

SIAM Journal on Applied Mathematics, Vol. 45, No. 5 (Oct., 1985), 810-825.

Stable URL:
http://links.jstor.org/sici?sici=0036-1399%28198510%2945%3A5%3C810%3ASSOTRF%3E2.0.CO%3B2-U

SIAM Journal on Applied Mathematics is currently published by Society for Industrial and Applied Mathematics.

SIAM J. APPL. MATH.
Vol. 45, No. 5, October 1985

008

SIMULTANEOUS SOLUTION OF THE RNA FOLDING, ALIGNMENT AND PROTOSEQUENCE PROBLEMS*

DAVID SANKOFF†

Abstract. The alignment of finite sequences, the inference of ribonucleic acid secondary structures (folding), and the reconstruction of ancestral sequences on a phylogenetic tree, are three problems which have dynamic programming solutions, which we formulate in a common mathematical framework. Combining the objective functions for alignment (parsimony, or minimal mutations) and folding (free energy), we present an algorithm which solves all three problems simultaneously for a set of N sequences of length n in time proportional to n^{3N} and storage n^{2N}. Incorporating a "cutting corners" constraint against biologically unlikely alignments reduces these requirements so that they are proportional to n^3 and n^2, respectively, for fixed N.

1. Introduction. The dynamic programming comparison of sequences finds widespread applications in molecular biology, in the detection and evaluation of similarities between two (or more) nucleic acid sequences or protein sequences. The optimal *alignments* obtained through this method indicate structural and functional similarities among the sequences. When three or more sequences are aligned, another dynamic programming algorithm is available to reconstruct the *protosequence*, or ancestral sequence common to all of them. Dynamic programming is also the best algorithmic approach to a third problem, that of inferring, from a given sequence of terms representing a ribonucleic acid (RNA) molecule, how this molecule "folds" in space, how certain segments come into contact with other regions of the sequence, and chemically bind with them to form a thermodynamically stable *secondary structure.*

In practice, whether or not they make use of formal algorithms, molecular biologists often carry out RNA sequence alignment and folding analysis in conjunction, so that partial information about what regions of two or more sequences are highly similar, and hence aligned, can be used to constrain the search for a common secondary structure—and vice versa—the discovery of common folding possibilities among two or more sequences suggests that the pertinent regions be aligned.

In this paper we incorporate the mutually informative nature of folding and alignment into a single algorithm for producing both, optimizing a linear combination of the objective functions used when the problems are treated separately. An N-dimensional generalization of the alignment method to the problem of protosequence reconstruction allows us to incorporate this latter task as well, so that all three are carried out simultaneously.

Dynamic programming for complex problems tends to be computationally expensive. We avoid this here through the judicious combination of a number of biologically well motivated (and previously studied) constraints on the set of possible alignment and folding solutions. Thus the computation time necessary for our most general algorithm is proportional to n^3K^N where N is the number of sequences being analyzed, n is the length of the longest sequence and K is a small (relative to n) integer constant. Our approach here will be first to develop the case $N=2$, and then to present the algorithm for general N.

In mathematical biology, dynamic programming for sequence comparison was introduced by Needleman and Wunsch (1970), an efficient (quadratic) form of the algorithm by Sankoff (1972), and a general distance function for evaluating alignments

* Received by the editors July 26, 1984, and in revised form January 2, 1985.
† Centre de recherches mathématiques, Université de Montréal, c.p. 6128 Montréal, Canada H3C 3J7

by Sellers (1974a, 1974b). The material in the next section, including Theorem 1, is based on this work. The current state of the field and related areas is represented by the papers in Sankoff and Kruskal (1983) and the review by Waterman (1984). The use of dynamic programming for secondary structure inference was published first by Waterman (1978), Nussinov et al. (1978) and Waterman and Smith (1978); our statements of Theorems 2 and 3 are phrased in terms of Sankoff et al. (1983) and Zuker and Sankoff (1984), who synthesize and advance a series of improvements in algorithmic efficiency including the Nussinov et al. paper, Zuker and Steigler (1981) and Mainville (1981). In the section on protosequences, which deals with generalizations to more than two sequences, recursion (16), Theorem 5 and their applications have previously been discussed by Sankoff et al. (1973), (1976), Sankoff (1975), Sankoff and Rousseau (1975), Sankoff and Cedergren (1983) and Waterman (1984).

2. Alignments. Let $\mathbf{a} = a_1, \cdots, a_m$ and $\mathbf{b} = b_1, \cdots, b_n$ be sequences of terms from any set $\mathcal{A}$. (For RNA, the set is $\{A, C, G, U\}$, but this fact plays no role at the level of generality we shall adopt.) An alignment of **a** and **b** is defined by two integer sequences $1 \leqq i_1 < i_2 < \cdots < i_r \leqq m$ and $1 \leqq j_1 < j_2 < \cdots < j_r \leqq n$. Let s be the number of pairs (i_k, j_k) such that $a_{i_k} \neq b_{j_k}$. Then for any $x > 0$ and $y \geqq \frac{1}{2}x$ we may define the cost of the alignment to be

$$(m+n-2r)y + sx. \tag{1}$$

This may be interpreted, as in Fig. 1, as the cost of converting **a** into **b** by *replacing* each a_{i_k} by b_{j_k} in the s cases in which they are not equal, with unit replacement cost x; by *deleting* the $m-r$ terms from **a** which are not aligned (not indexed by an i_k); and by appropriately *inserting* the $n-r$ terms of **b** which are not aligned (not indexed by a j_k). The cost of each deletion and insertion is y. Note that (1) is also the cost of converting **b** into **a**. This interpretation is biologically meaningful in terms of the explanatory cost of models of evolutionary divergence of the two sequences as measured by the number and kind of mutational events (replacements, insertions and deletions) implied by the alignment to account for the change from **a** to **b**, or vice versa, or from some common ancestor sequence to **a** on one hand and to **b** on the other.

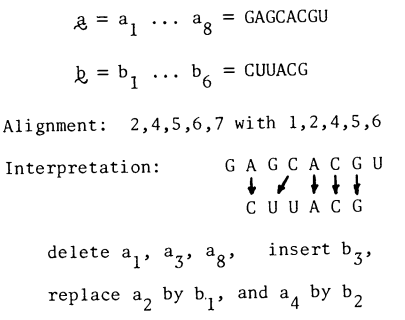

FIG. 1. *Interpretation of an alignment.* $m = 8$, $n = 6$, $r = 5$, $s = 2$.

The *distance* between **a** and **b** is the cost of the optimal alignment, i.e., the minimum alignment cost possible (most parsimonious evolutionary model) for fixed x and y. (That this is a metric on the set of finite sequences is easily verified.) The dynamic programming solution to finding the optimal alignment is summarized as follows.

THEOREM 1. *For $1 \leqq i \leqq j \leqq m$ and $1 \leqq h \leqq k \leqq n$, let $D(i, j; h, k)$ be the minimum alignment cost between the partial sequences $a_i, \cdots, a_j$ and $b_h, \cdots, b_k$. Then for $i < j$*

and/or $h < k$

$$D(i,j;h,k) = \min \begin{cases} D(i,j;h,k-1)+y \\ D(i,j-1;h,k-1)+x & \text{if } a_j \neq b_k, \\ D(i,j-1;h,k-1) & \text{if } a_j = b_k, \\ D(i,j-1;h,k)+y \end{cases} \tag{2}$$

$$= \min \begin{cases} D(i+1,j;h,k)+y \\ D(i+1,j;h+1,k)+x & \text{if } a_i \neq b_h, \\ D(i+1,j;h+1,k) & \text{if } a_i = b_h, \\ D(i,j;h+1,k)+y \end{cases} \tag{3}$$

with initial conditions

$$D(i,i;h,h) = \begin{cases} x & \text{if } a_i \neq b_h, \\ 0 & \text{if } a_i = b_h \end{cases}$$

for $1 \leqq i \leqq m$ *and* $1 \leqq h \leqq n$.

Proof. In the optimal alignment of $a_i, \cdots, a_j$ and $b_h, \cdots, b_k$ suppose there are r aligned pairs, s of which are unequal. Then either

(i) Both a_j and b_k are aligned, in which case they must be aligned with each other, constituting the rth aligned pair. Then from (1),

$$\begin{aligned} D(i,j;h,k) &= (j-i+1+k-h+1-2r)y + sx \\ &= \begin{cases} ((j-1)-i+1+(k-1)-h+1-2(r-1))y + sx \\ ((j-1)-i+1+(k-1)-h+1-2(r-1))y + (s-1)x + x \end{cases} \\ &= \begin{cases} D(i,j-1;h,k-1) & \text{if } a_j = b_k, \\ D(i,j-1;h,k-1)+x & \text{if } a_j \neq b_k \end{cases} \end{aligned} \tag{4}$$

since removing this pair must leave us with an optimal alignment of $a_i, \cdots, a_{j-1}$ and $b_h, \cdots, b_{k-1}$, or

(ii) at most one of a_i or b_h is in the rth aligned pair and

$$\begin{aligned} D(i,j;h,k) &= (j-i+1+k-h+1-2r)y + sx \\ &= ((j+k-1)-i+1-h+1-2r)y + sx + y \\ &= \begin{cases} D(i,j-1;h,k)+y & \text{if } a_j \text{ is not aligned,} \\ D(i,j;h,k-1)+y & \text{if } b_k \text{ is not aligned} \end{cases} \end{aligned} \tag{5}$$

since removing either a_j or b_k if they are not aligned must leave us with an optimal alignment of $a_i, \cdots, a_{j-1}$ and $b_h, \cdots, b_k$, or of $a_i, \cdots, a_j$ and $b_h, \cdots, b_{k-1}$, respectively. This proves (2); recurrence (3) is proved similarly, by focusing on the first aligned pair rather than the rth.

The initial conditions derive from the alignment of the single-term sequences a_i and b_h, so that depending on whether these terms are identical or different, the best alignment will involve exactly zero or one replacement, respectively. □

An algorithm for calculating $D(i,j;h,k)$ for all $1 \leqq i \leqq j \leqq m$ and $1 \leqq h \leqq k \leqq n$, then, consists of applying recurrences (2) and (3) to all m^2n^2 4-tuples (i,j,h,k) in any order (there are many) in which $(i+1,j,h,k)$, $(i+1,j,h+1,k)$, $(i,j,h+1,k)$, $(i,j-1,h,k)$, $(i,j-1,h,k-1)$ and $(i,j,h,k-1)$ are processed before (i,j,h,k). This requires computing time proportional to m^2n^2, or approximately n^4 if $m \sim n$. Later in this paper

we will need values of D for all the 4-tuples calculated in the theorem. In other contexts, however, it is often required only to align the full sequences **a** and **b**. Then only the value of $D(1, m; 1, n)$ is required. For this case, an n^2 (quadratic) algorithm consists of using only (2) to calculate D for all 4-tuples of form $(1, j; 1, k)$ only.

Once $D(1, m; 1, n)$ has been calculated, an optimal alignment can be rapidly (in time proportional to $m+n$) identified by *backtracking* the optimizing steps in the recurrence, starting at (m, n). For example if $D(1, m; 1, n) = D(1, m-1; 1, n)+y$, then an optimal alignment exists where a_m is deleted. If then $D(1, m-1; 1, n) = D(1, m-2; 1, n-1)+x$ and $a_{m-1} \neq b_n$ then in this optimal alignment $i_r = m-1$ and $j_r = n$ for some r, and so on. (N.B. This is a somewhat different notion of backtracking than that used in branch-and-bound and similar algorithms.)

Note that if we were to allow $y < \frac{1}{2}x$, all optimal alignments would involve insertions and deletions only (no replacements), and the aligned pairs would constitute the longest common subsequence of **a** and **b**. These alignments are also optimal in the case $y = \frac{1}{2}x$ though here other optimal alignments involving replacements are possible. (See Hirschberg (1983) for a discussion of this type of problem.)

3. Folding. A *secondary structure* on a sequence $\mathbf{a} = a_1, \cdots, a_n$ is a set S of pairs (i, j) where $1 \leqq i < j \leqq n$ satisfying the *knot* constraint—if $i \leqq i' \leqq j \leqq j'$, then (i, j) and (i', j') cannot be two distinct elements of S.

Suppose that $(i, j) \in S$ and $i < r < j$. If no $(i', j') \in S$ where $i < i' < r < j' < j$, then we say r is *accessible* from (i, j). Also the pair $(p, q) \in S$ is said to be accessible from (i, j) if both p and q are.

The $u \geqq 0$ unpaired terms accessible from a given $(i, j) \in S$ together with the $k-1 \geqq 0$ accessible pairs, constitute the *k-loop closed by* (i, j). Those pairs and unpaired terms in no k-loop are *external.*

LEMMA 1. *Each unpaired term and each pair in a structure belongs to exactly one k-loop or is external.*

Proof. Consider any term r accessible from both (i, j) and (i', j'). By definition of accessibility either $i' < i < r < j' < j$ or $i < i' < r < j < j'$, both of which are prohibited by the knot constraint. □

The decomposition verified in Lemma 1 is biologically meaningful—the k-loops are known to experimentalists as *hairpins* ($k = 1$), *bulges* and *interior loops* ($k = 2$, $u > 0$), *stacked pairs* ($k = 2$, $u = 0$), and *multiple loops* ($k \geqq 3$), as illustrated in Fig. 2.

For $k = 1$ and $k = 2$ experimentally determined values are available (Tinoco et al. (1973), Salser (1977)) for $e(s)$, the relative thermodynamic stability (or instability) of a k-loop s, measured in units of free energy, as a function of k, u, the pair (a_i, a_j), and the values of a_r for the accessible r. The basic hypothesis permitting the purely computational (i.e. nonexperimental) solution of the folding problem is that the secondary structure occurring naturally is the one which, among all possible pairing schemes, is thermodynamically the most stable, in that the sum of the free energies of its k-loops is minimal. Since only stacked pairs contribute negative free energy, stable structures tend to have long series of *nested stacked pairs* $(i, j), (i+1, j-1), \cdots, (i+h, i-h)$, and as few other loops as possible. Note that in the same way as the closing pair (i, j) of a k-loop s is not itself defined to form part of that loop in the decomposition determined by S, the energetic contribution of the pair is not considered part of $e(s)$, but part of that of the k-loop, if any, in which (i, j) is accessible. Thus external pairs contribute zero free energy to a structure. The same holds for external unpaired terms, so that, for example, the structure containing no pairs (i.e. $S = \phi$) has free energy zero.

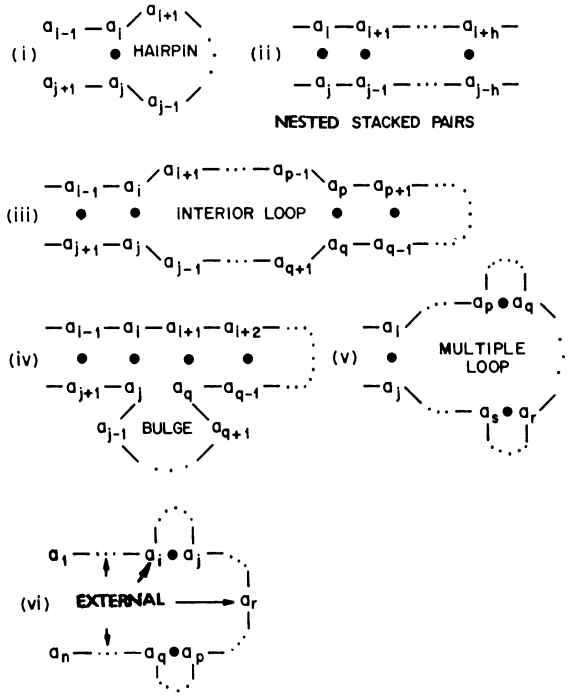

FIG. 2. *Types of k-loop. Heavy dots interposed between two elements of each pair.*

These considerations lead to the following optimization problem: given a sequence **a** and values of $e(s)$ for all possible k-loops s, find the secondary structure S which minimizes the sum of $e(s)$ over all k-loops in the decomposition determined by S.

Before we state and prove a dynamic programming solution to this problem, we will need some further structural consequences of the knot constraint and definitions of accessibility.

LEMMA 2. *If $(i,j) \in S$, then $i<r<j$ if and only if r is accessible from some pair $(i^*,j^*) \in S$, where $i \leqq i^* < r < j^* \leqq j$.*

Proof. Either r is accessible from (i,j) or there is some intervening $(i',j') \in S$ such that $i<i'<r<j'<j$. Then either r is accessible from (i',j'), or there is still another intervening pair, (i'',j'') where $i'<i''<j<j''<j'$, and so on. After at most $r-i$ steps we must arrive at (i^*,j^*) since there can be no more intervening pairs. □

LEMMA 3. *Given a structure S and $(i,j) \in S$, the substructure S_{ij} imposed on (i,j) by S consists of (i,j), its accessible pairs, their accessible pairs, and so on. Then S_{ij} is a secondary structure on $i,\cdots,j$. All its k-loops are k-loops of S and any k-loop of S containing a term in $i+1,\cdots,j-1$ is a k-loop of S_{ij}.*

Proof. The knot constraint holds for S_{ij} by virtue of its holding for all pairs in S, so that S_{ij} is a secondary structure. Since (i,j) is a pair in S and S_{ij}, all terms in $i+1,\cdots,j-1$ must be accessible to some pair in $i,\cdots,j$, by Lemma 2. By Lemma 1 they cannot be accessible to any pair outside of $i,\cdots,j$. Thus the k-loops of S_{ij} and the k-loops of S containing terms in $i+1,\cdots,j-1$, coincide. □

LEMMA 4. *If $(i,j) \in S$ and $i<r<j$, then r is in some k-loop of some substructure $S_{p,q}$ where (p,q) is accessible from (i,j), or else itself is accessible from (i,j).*

Proof. Suppose r is in no k-loop of any such substructure $S_{p,q}$ and r is not accessible from (i,j). Then there is some $(i',j') \in S$ such that $i<i'<r<j'<j$. Now, by Lemma 2 and Lemma 3, r must be in a k-loop of $S_{i',j'}$, so to prevent a contradiction

(i', j') cannot be accessible from (i, j). Then there must be another pair $(i'', j'') \in S$ such that $i < i'' < i'$, and so on. But this can proceed at most $r - i$ steps before we encounter a contradiction. □

The simplest way of formulating a dynamic programming solution is then:

THEOREM 2. *Let $F(i, j)$ be the minimum energy possible for a secondary structure S on the partial sequence $i, \cdots, j$. Let $C(i, j)$ be the minimum given that $(i, j) \in S$, where $C(i, j) = \infty$ if no such structure exists. Then*

$$C(i,j) = \min_{k \geqq 1} \min_{\substack{s \text{ is a } k\text{-loop} \\ \text{closed by } (i,j)}} \left\{ e(s) + \sum_{\substack{(p,q) \\ \text{accessible} \\ \text{from } (i,j)}} C(p,q) \right\} \tag{6}$$

with the initial conditions $C(i, i) = \infty$, and

$$F(i,j) = \min \left\{ C(i,j), \min_{i \leqq h < j} [F(i,h) + F(h+1,j)] \right\}, \tag{7}$$

with the initial conditions $F(i, i) = 0$.

Proof. We first deal with pairs of form (i, i) and then prove that if the recurrences are correct for all (i', j') where $i < i' \leqq j' < j$, then they are also correct for (i, j).

Consider the case of (i, i). Since it is not true that $i < i$, there is no structure containing (i, i), so $C(i, i) = \infty$. A structure with no pairs has free energy zero, so $F(i, i) = 0$.

For any other (i, j), consider an optimal structure S_{ij} on $a_i, \cdots, a_j$ under the constraint $(i, j) \in S$. Let $(p_1, q_1), \cdots, (p_{k-1}, q_{k-1})$ be the pairs accessible from (i, j), and $S_{p_1,q_1}, \cdots, S_{p_{k-1},q_{k-1}}$ the imposed substructures. By previous application of the recurrence, these will be optimal, i.e. they will have energies $C(p_1, q_1), \cdots, C(p_{k-1}, q_{k-1})$. Any k-loop of S_{ij} having terms r such that $p_h < r < q_h$ will also be a k-loop of S_{p_h,q_h}, and vice versa, by Lemma 3. By Lemma 4, any term r not satisfying $p_h < r < q_h$ for any h is accessible from (i, j) and is part of the k-loop s closed by (i, j). Then the energy of S_{ij}, the sum of the energies of its k-loops, is

$$e(s) + \sum_{\substack{(p,q) \text{ accessible} \\ \text{from } (i,j)}} C(p,q).$$

The correctness of (6) then follows from the optimality of S_{ij} and the definition of C as minimal.

As for (7), $F(i, j) \leqq F(i, h) + F(h+1, j)$ for any $i \leqq h < j$ since we can always construct a secondary structure on $i, \cdots, j$ as the union of those on $i, \cdots, h$ and on $h+1, \cdots, j$, whose energy is the sum of theirs. Then it suffices to find one h for which the equality holds. Now, if the optimal structure on $a_i, \cdots, a_j$ contains (i, j), $F(i, j) = C(i, j)$ by definition. Otherwise, if there are no pairs, $F(i, j) = 0 = F(i, h) + F(h+1, j)$ for any $i \leqq h < j$. If there is at least one pair, there must be at least one external pair (p, q), by an argument analogous to those of Lemma 2 and Lemma 4. Either $p \neq i$ or $q \neq j$ or both. Say $p \neq i$. Can $F(i, j) < F(i, p-1) + F(p, j)$? If so, the subset of pairs involving terms in $i, \cdots, p-1$ in the optimizing secondary structure on $i, \cdots, j$ could be used as a secondary structure on $i, \cdots, p-1$, and the remaining ones as a secondary structure on $p, \cdots, j$. (No pair could contain terms i' from $i, \cdots, p-1$ and j' from $p, \cdots, j$, since then $i' < p < j'$, contradicting by Lemma 2 the fact that p is external.) The sum of their energies would be less than $F(i, p-1) + F(p, j)$, a contradiction. □

$F(1, n)$ is then the free energy of the optimal or "true" secondary structure.

To use recurrence equations (6) and (7) to find the optimal structure, we must apply them to pairs (i,j) ordered in such a way as to ensure $C(i,j)$ and $F(i,j)$ are evaluated after $C(i',j')$, and $F(i',j')$, for all $i<i'<j'<j$.

As in the alignment algorithm, it is then an easy matter to backtrack through the (i,j) array to actually construct a structure with free energy $F(1,n)$.

Note that biological constraints such that certain values of a_i and a_j cannot form a pair, or that $u \geqq 3$ for hairpins, can be satisfied simply by assigning $e(s)=\infty$ to k-loops violating these conditions.

Because the calculation of $C(i,j)$ in this algorithm involves evaluating all possible k-loops, it must examine all $2k$-tuples of the form $i<p_1<q_1<\cdots<p_{k-1}<q_{k-1}<j$. Including all pairs (i,j), this takes time proportional to n^{2k}. Since the maximum k possible grows linearly with n, the algorithm requires exponential computing time, and is hence impractical.

In practice, though $e(s)$ may be a complicated function of u for 2-loops, the values of u actually encountered are rarely large, say $u \leqq U$. For multiple loops, on the other hand, large values of u are possible, but here $e(s)$ may be approximated by a linear function of u and k:

$$e(s)=A+(k-1)P+uQ. \tag{8}$$

THEOREM 3. *Let C and F be as in Theorem 2. Then, under the restrictions that $u \leqq U$ in 2-loops and e satisfies (8) for multiple loops,*

$$C(i,j)=\min\begin{cases} e(s), & s \text{ is the hairpin closed by } (i,j), \\ \min\{e(s)+C(p,q)\}, & s \text{ a 2-loop closed by } (i,j) \text{ with } (p,q) \text{ accessible, } u=p-i+j-q-2 \leqq U, \\ \min\limits_{i<h<j-1}\{G(i+1,h)+G(h+1,j-1)+A\}, \end{cases} \tag{9}$$

where

$$G(i,j)=\min\begin{cases} C(i,j)+P, \\ \min\limits_{i \leqq h<j} \min \begin{cases} (G(i,h)+(j-h)Q, \\ G(i,h)+G(h+1,j), \\ (h-i+1)Q+G(h+1,f), \end{cases} \end{cases} \tag{10}$$

and, as in (4),

$$F(i,j)=\min\begin{cases} C(i,j), \\ \min\limits_{i \leqq h<j}\{F(i,h)+F(h+1,j)\}, \end{cases} \tag{11}$$

with initial conditions $C(i,i)=\infty$, $F(i,i)=0$ and $G(i,i)=\infty$.

Proof. We must prove that G enters into the evaluation of multiple loops only, that $e(s)$ for these loops satisfies (8) and that they are optimal. The theorem is then a special case of Theorem 2, with the restriction on 2-loops being satisfied by the second option in (9). Note first that $G(i,j)$ is finite only if there is some pair (i',j') where $i \leqq i'<j' \leqq j$ in the corresponding structure. Thus in (9), G enters into the recurrence for C if and only if the optimizing structure contains at least two pairs accessible to (i,j), one on each side of h, i.e. a 3-loop, or higher-order loop, closed by (i,j).

The recurrence for G ensures that each accessible pair (p,q) in an optimizing loop contributes $C(p,q)+P$ and each unpaired term Q. When the multiple loop option is chosen in (9), the value A is added, so that (8) is satisfied. To prove that the k-loops

concerned are optimal, we must show that the minimization in (10) is taken over all possible accessible configurations. This is the case since any such configuration is an alternation of accessible pairs and (possibly) some string of unpaired terms, and all such alternations are evaluated in (10). □

The linearity condition (8) allows us to piece together k-loops with large k from smaller pieces in (9) in a computationally efficient way—a search among $j-i-1$ values of h rather than all possible $2k$-tuples.

Bounding the search for 2-loops by U in (9) means that the dominant term in assessing the number of steps to completion of the algorithm is contributed by the search for h which is at most linear in n. Since (9)-(11) are applied to all $1 \leqq i < j \leqq n$, the total computing time is proportional to n^3.

It is easily verified that the proof of Theorem 3 carries through with no difficulty when (8) is replaced by

$$e(s) = \mathbf{A}(a_i, a_j) + \sum_{h=1}^{k-1} \mathbf{P}(a_{p_h}, a_{q_h}) + \sum_{h=0}^{k-1} \mathbf{Q}(a_{q_h+1}, \cdots, a_{p_{h+1}-1}) \tag{12}$$

(where $q_0 = i$ and $p_k = j$), so that e depends on the actual values of the sequence terms which close the k-loop s, or form its accessible pairs $(p_1, q_1), \cdots, (p_{k-1}, q_{k-1})$, and also depends on the set of unpaired terms which intervenes between each two accessible pairs (p_h, q_h) and (p_{h+1}, q_{h+1}). In the statement of the theorem, A is replaced by $\mathbf{A}(a_i, a_j)$ in (9), P is replaced by $\mathbf{P}(a_i, a_j)$ in (10), and $(j-h)Q$ and $(h-i+i)Q$ are replaced by $\mathbf{Q}(a_{h+1}, \cdots, a_j)$ and $\mathbf{Q}(a_i, \cdots, a_h)$, respectively, in (10).

While the theorem remains true, the efficiency of the algorithm based on it is sensitive to how the function $\mathbf{Q}(.)$ is calculated. If $\mathbf{Q}(.)$ can be evaluated in a fixed number of steps, as in the original statement of Theorem 3, then we retain an n^3 algorithm. If the time taken to compute $\mathbf{Q}(a_i, \cdots, a_j)$ is linear in $j-i$, on the other hand, the algorithm may require time proportional to n^4.

The basic idea of decomposing multiple loops through the linear search for h is found in Nussinov et al. (1978). The bound on 2-loops was used by Zuker and Steigler (1981) in the first computer implementation for long ($\sim 10^3$) sequences to incorporate realistic loop costs while retaining global optimality. Mainville (1981) introduced linear multiple loop energies, and J. B. Kruskal generalized these and integrated them into the linear search for h (Sankoff et al. (1983)). The synthesis of all these features as has been presented in this section is drawn from the latter paper and from Zuker and Sankoff (1984), except for the generalization proposed above as (12), and its implications for computing efficiency.

4. Integrating alignment and folding. When biologists consider two RNA sequences to have the same secondary structure, this refers to the general "shape" of the structure—it does not mean that they are identical in all respects. For example, where there are nested stacked pairs $(i, j), (i+1, j-1), \cdots, (i+h, j-h)$, the number of pairs may vary somewhat from case to case, without greatly affecting the overall appearance (or function) of the structure. The same is true for nested 2-loops in general. In some cases the number of consecutive unpaired terms, in k-loops or external, is variable though the structures are considered basically the same.

One aspect of a structure which is an invariant part of its identity is its *branching configuration*, determined by the external pairs and multiple loops. More precisely, let $i_1 < i_2 < \cdots < i_v$ be the positions in sequence **a** of all terms which form part either of an external pair or of an accessible pair in a multiple loop in structure A. Let $j_1 < j_2 < \cdots < j_w$ be the same for structure B on sequence **b**. For A and B to be *equivalent*

in terms of branching configurations, we require $v = w$ and $(i_f, i_g) \in A$ if and only if $(j_f, j_g) \in B$.

The definition allows complete freedom as to the number and type of 2-loops nested in each external pair and in each multiple loop accessible pair, as well as to the number of unpaired external terms and unpaired terms in any k-loop. Thus, while equivalence between structures is a necessary aspect of finding a common folding of two sequences, it is not sufficient, as illustrated in Fig. 3. We should require the same

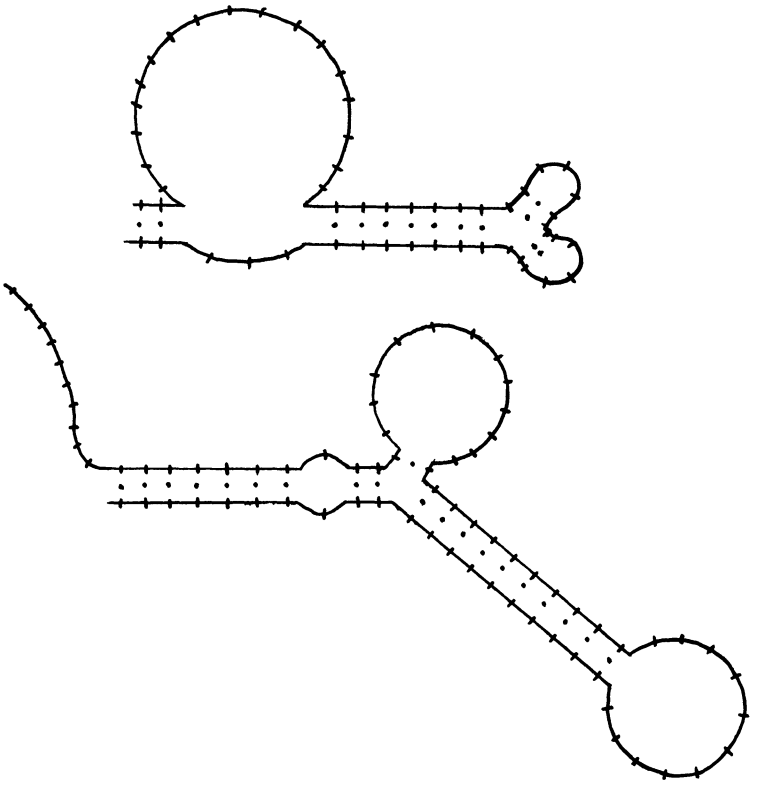

FIG. 3. *Two equivalent, but highly dissimilar secondary structures. Sequence terms represented by bars on solid line. Dots interposed between two elements of each pair.*

secondary structures for two sequences to show more similarities than just equivalent branching configurations. To evaluate how similar two equivalent structures are, it is natural to introduce the notion of alignment as discussed earlier in this paper. One way of doing this, although not the only one, is as follows: First, the equivalence requirement itself is guaranteed by constraining any alignment so that i_1 is aligned with j_1, i_2 with $j_2, \cdots, i_v$ with j_w. Also, for simplicity's sake, we will require that any k-loop be aligned with a single k-loop in the other structure, or be inserted or deleted *in toto*. Thus, for example, we cannot have elements of the same k-loop in one structure aligned with elements from two or more k-loops in the other structure. Since all external pairs and accessible pairs in multiple loops are aligned and not deleted or inserted, each multiple loop and external region will have its counterpart in the other structure, but 2-loops need not, there being no constraint against deleting or inserting their accessible pairs. As for hairpins, it can be shown that equivalent structures will always

have the same number of hairpins in the same general locations vis-a-vis the constrained pairs of the structures (i.e. if there is a hairpin between i_f and i_g in one structure, there will also be one between j_f and j_g in the other where i_f is constrained to be aligned with j_f and i_g with j_g). We will require that such corresponding hairpins be aligned, rather than allow one to be deleted in its entirety and the other to be inserted in its entirety. Since there is no constraint on their deletion or insertion, the number of 2-loops remains free to vary between one structure and the other. Note that these alignment constraints are not necessary for equivalence and could be relaxed in a more general analysis.

The goal then becomes one of finding equivalent structures and a constrained alignment such that the entire configuration of structure and alignment is optimal in some sense. Now, we cannot expect that there will always exist equivalent structures which are thermodynamically optimal for both sequences separately, and even if we did find such a case, a suitably constrained alignment of them would not necessarily be the minimum cost constrained alignment over all possible pairs of equivalent structures.

Thus we introduce a new objective function to optimize, representing a trade-off between free energy and alignment cost, namely their weighted sum. Without loss of theoretical generality, we may use the unweighted sum since alignment cost is in terms of arbitrary units x and y. In practice, of course, these units would have to be calibrated using known secondary structures.

We extend the definition of $D(i_1, j_1; i_2, j_2)$, the minimizing cost of an alignment between $a_{i_1}, \cdots, a_{j_1}$ and $b_{i_2}, \cdots, b_{j_2}$, as follows: if $i_1 > j_1$, then $D(i_1, j_1; i_2, j_2)$ will refer to the cost of inserting the entire sequence $b_{i_2}, \cdots, b_{j_2}$, and if $i_2 > j_2$, it will refer to the cost of deleting $a_{i_1}, \cdots, a_{j_1}$. In addition, we make the notational convention, for the empty loop ϕ containing no elements, $e(\phi) = 0$.

Then we can find the optimizing structure and alignments using two-sequence generalizations of arrays C, G and F.

THEOREM 4. *Let $F(i_1, j_1; i_2, j_2)$ be the minimum cost possible for a pair of equivalent secondary structures S_1 and S_2, restricted as in Theorem 3, on positions $i_1, \cdots, j_1$ and $i_2, \cdots, j_2$ of sequences $\mathbf{a}^{(1)}$ and $\mathbf{a}^{(2)}$ respectively, where the cost is the sum of the free energies and a constrained alignment cost.*

Let $C(i_1, j_1; i_2, j_2)$ be the minimum cost given that $(i_1, j_1) \in S_1$ and $(i_2, j_2) \in S_2$ without considering the costs of aligning $a^{(1)}_{i_1}$, $a^{(1)}_{j_1}$, $a^{(2)}_{i_2}$ and $a^{(2)}_{j_2}$. If no such pair of structures exists, set $C = \infty$. Then

$$
\begin{aligned}
(13)\quad C(i_1, j_1; i_2, j_2) = \min \Bigg\{ & e(s_1) + e(s_2) + D(i_1+1, j_1-1; i_2+1, j_2-1), \quad s_1, s_2 \text{ hairpins closed by } (i_1, j_1), (i_2, j_2) \text{ respectively,} \\
& \min \{ e(s_1) + e(s_2) + C(p_1, q_1; p_2, q_2) \\
& \qquad + D(i_1+1, p_1; i_2+1, p_2) + D(q_1, j_1-1; q_2, j_2-1) \}, \\
& \qquad s_1, s_2 \text{ are 2-loops closed by } (i_1, j_1), (i_2, j_2) \text{ with } (p_1, q_1), (p_2, q_2) \text{ accessible,} \\
& \qquad p_1 - i_1 + j_1 - q_1 - 2 \leqq U,\ p_2 - i_2 + j_2 - q_2 - 2 \leqq U, \\
& \qquad \text{or one of } \begin{cases} s_1 = \phi & \text{and } (p_1, q_1) = (i_1, j_1) \\ s_2 = \phi & \text{and } (p_2, q_2) = (i_2, j_2), \end{cases} \\
& \min_{\substack{i_1 < h_1 < j_1 - 1 \\ i_2 < h_2 < j_2 - 1}} \{ G(i_1+1, h_1; i_2+1, h_2) \\
& \qquad + G(h_1+1, j_1-1; h_2+1, j_2-1) + 2A \}, \Bigg.
\end{aligned}
$$

where

$$G(i_1, j_1; i_2, j_2)$$

$$= \min \begin{cases} C(i_1, j_1; i_2, j_2) + 2P + D(i_1, i_1; i_2, i_2) + D(j_1, j_1; j_2, j_2), \\ \min\limits_{\substack{i_1 < h_1 < j_1 \\ i_2 < h_2 < j_2}} \min \begin{cases} G(i_1, h_1; i_2, h_2) + (j_1 - h_1 + j_2 - h_2)Q \\ \qquad + D(h_1 + 1, j_1; h_2 + 1, j_2), \\ G(i_1, h_1; i_2, h_2) + G(h_1 + 1, j_1; h_2 + 1, j_2), \\ (h_1 - i_1 + 1 + h_2 - i_2 + 1)Q \\ \qquad + G(h_1 + 1, j_1; h_2 + 1, j_2) + D(i_1, h_1; i_2, h_2), \end{cases} \end{cases} \tag{14}$$

and

$$F(i_1, j_1; i_2, j_2) = \min \begin{cases} C(i_1, j_1; i_2, j_2) + D(i_1, i_1; i_2, i_2) + D(j_1, j_1; j_2, j_2), \\ \min\limits_{\substack{i_1 \leqq h_1 < j_1 \\ i_2 \leqq h_2 < j_2}} \{F(i_1, h_1; i_2, h_2) + F(h_1 + 1, j_1; h_2 + 1, j_2)\}, \\ D(i_1, j_1; i_2, j_2), \end{cases} \tag{15}$$

with initial conditions $C(i_1, i_1; i_2, i_2) = \infty$ *and* $G(i_1, i_1; i_2, j_2) = G(i_1, j_1; i_2, i_2) = \infty$.

Proof. We first show that C and F can be finite only if they pertain to equivalent structures. Now $G(i_1, j_1; i_2, j_2)$ is finite only if the two corresponding structures have the same number of external pairs. This is obvious if the first option in (14) (exactly one pair) holds, and all of the other options conserve this property in building up successively more complex structures. Then the last option of recurrence (13) can incorporate only multiple loops of the same order into the two structures. This same argument applies recursively to show that the orderings of terms in all accessible pairs satisfies the definition of equivalence.

The last and first options in (15) refer to the cases of zero external pairs in both structures, and one such pair in both structures, respectively; so these structures are trivially equivalent, or because (13) guarantees equivalence, respectively. By recursion then, the middle option can only piece together structurally equivalent parts of both sequences.

We next note that corresponding accessible pairs in multiple loops are aligned, by the first option of (14), and the same for external pairs, by the first option of (15). The second and fourth options in (14) and the last option in (15) assure that corresponding multiple loops are aligned and corresponding external regions are aligned. The first two options in (13) require that all terms in a 1- or 2-loop are aligned with a corresponding 1- or 2-loop or, in the case of a 2-loop, that the whole loop is inserted or deleted. In the latter case, the extended definition of D becomes pertinent.

Finally, we must show that, subject to equivalence and constrained alignment, the recurrences correctly identify optimal structures. To do this it suffices first to fix i_2 and j_2 and to prove their correctness as recurrences on i_1 and j_1 only, and second to do the same with the roles of (i_1, j_1) and (i_2, j_2) reversed. When this is done, the theorem becomes identical to Theorem 3 except that there are different values of $e(s)$ for k-loops and nonzero costs for external terms. For the external terms, the correctness of (15) can be proved by the same type of argument as for (7) in Theorem 2. For 1- and 2-loops, the arbitrariness of $e(s)$ in Theorem 3 allows us to consider, say $e'(s_1) = e(s_1) + e(s_2) + D(i_1 + 1, j_1 - 1; i_2 + 1, j_2 - 1)$ for fixed s_2, as a new energy function for hairpins. For multiple loops, the inclusion of the alignment costs in (14) means that the loop cost is no longer as in Theorem 3, a linear function of u, the number of unpaired terms in the loop. However, Theorem 3 remains true when (8) is replaced

by (12), where e.g. $\mathbf{Q}(a_{h_1+1}, \cdots, a_{j_1}) = (j_1 - h_1 + j_2 - h_2)Q + D(h_1+1, j_1; h_2+1, j_2)$ for fixed $b_{h_2+1}, \cdots, b_{j_2}$. Thus the entire proof of Theorem 3 carries over to the present case. □

To assess the computational effort necessary to calculate (13)-(15) for all appropriate (i_1, j_1, i_2, j_2), we point out that the considerations following Theorem 3 apply independently to S_1 and S_2, so that the algorithm becomes n^6. It is important to note that in this algorithm recurrences (2) and (3) for D in Theorem 1 must be calculated at the same time, according to the same ordering of 4-tuples as for (13)-(15) of Theorem 4, taking the fixed number of steps per 4-tuples indicated in (2) and (3), and not otherwise affecting the complexity of computing C, F and G.

5. Protosequences. When N nucleic acid sequences are aligned simultaneously, they are generally from N organisms whose evolutionary relationships are represented by a tree diagram with or without an identified root, as in Fig. 4. The given sequences

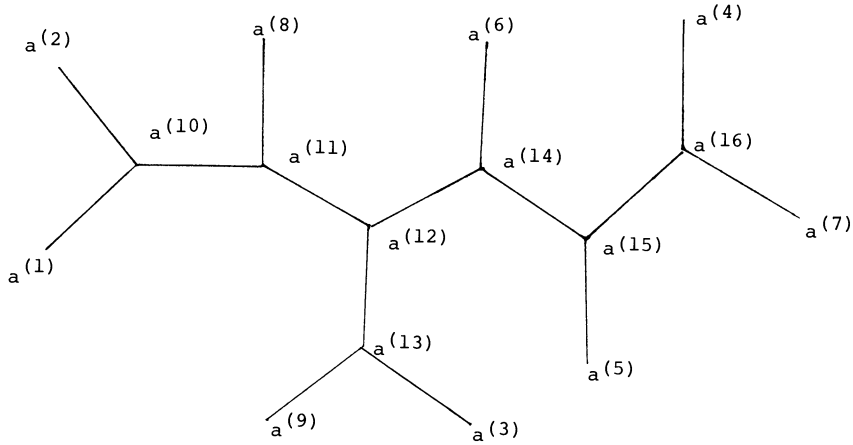

FIG. 4. *Unrooted tree diagram of evolutionary relationships among nine observed (or known) nucleic acid sequences* $\mathbf{a}^{(1)}, \cdots, \mathbf{a}^{(9)}$ *associated with the terminal vertices. Hypothetical ancestor sequences* $\mathbf{a}^{(10)}, \cdots, \mathbf{a}^{(16)}$ *associated with nonterminal vertices.*

$\mathbf{a}^{(1)}, \cdots, \mathbf{a}^{(N)}$ are associated with the terminal vertices $v_1, \cdots, v_N$ of the tree and the nonterminal ones $v_{N+1}, v_{N+2}, \cdots$ represent hypothetical ancestral sequences (protosequences) which are reconstructed in the course of the alignment.

We generalize (2) as follows (the generalization of (3) is analogous)

$$D(\overrightarrow{i,j}) = \min_{\text{not all } j'_r = j_r} \{D(\overrightarrow{i',j'}) + \gamma(\overrightarrow{a_j(j-j')})\} \tag{16}$$

where $j'_r = j_r$ or $j'_r = j_r - 1$, for $r = 1, \cdots, N$ and $D(\overrightarrow{i,j})$ represents the alignment of the partial sequences $a^{(1)}_{i_1}, \cdots, a^{(1)}_{j_1}; \cdots; a^{(N)}_{i_N}, \cdots, a^{(N)}_{j_N}$. We interpret $\gamma(\overrightarrow{a_j(j-j')})$ as follows: The argument of γ is an N-vector whose rth component is $a^{(r)}_{j_r}$ or ϕ depending on whether $j'_r = j_r - 1$ or $j'_r = j_r$, respectively. This value ($a^{(r)}_{j_r}$ or ϕ) is associated with vertex v_r. We are free to assign any values to all the nonterminal vertices. We then count the number s of branches (edges) of the tree with different non-ϕ values at the two endpoints, and the number t of branches with one ϕ and one non-ϕ value at the endpoints. Then $ty + sx$ represents the explanatory cost of generating the configuration of values at the terminal vertices given the values we chose to assign to the nonterminal vertices. We define γ to be the minimum cost possible for any such assignment (most parsimonious explanation).

How can we find this optimum cost? The vertices of the tree must first be assigned *heights.* This is done only once, before starting application of recurrence (16). We single out any nonterminal vertex as the root of the tree. (This choice may be suggested by biological considerations but for our purpose it is arbitrary.) The N terminal vertices are all assigned height zero, and successively for $H = 1, 2, \cdots$, the height H is assigned to any vertex (except the root) which has not already been assigned a height and for which all but one of the neighbouring vertices have all been assigned height less than H. It can be shown that when this rule can no longer be applied, only the root will remain without an assigned height. This is assigned a height one greater than any of its neighbours.

To evaluate γ, at each application of the recurrence, one defines a number of functions $\gamma_v(.)$ on $\mathscr{A} \cup \{\phi\}$ where $\mathscr{A}$ is the set or alphabet from which the sequence terms are drawn. One such function is defined for each vertex v of the tree and the γ_v are calculated recursively, starting with those v of height zero and continuing with those of height $1, 2, \cdots$.

THEOREM 5. *For each $\alpha \in \mathscr{A} \cup \{\phi\}$, we set $\gamma_v(\alpha) = 0$ if the rth component of $\overrightarrow{a_j(j-j')}$ is α, and $\gamma_v(\alpha) = \infty$ otherwise, for all vertices of height zero (terminal vertices). For vertices of height $H = 1, 2, \cdots$*

$$\gamma_v(\phi) = \sum_{\substack{v' \text{ neighbour of } v \\ \text{of height less than } H}} \min \begin{cases} \gamma_{v'}(\phi), \\ y + \min\limits_{\alpha \in \mathscr{A}} \gamma_{v'}(\alpha), \end{cases}$$

$$\gamma_v(\alpha) = \sum_{\substack{v' \text{ neighbour of } v \\ \text{of height less than } H}} \min \begin{cases} \gamma_{v'}(\alpha), \\ y + \gamma_{v'}(\phi), \\ x + \min\limits_{\substack{\beta \in \mathscr{A} \\ \beta \neq \alpha}} \gamma_{v'}(\beta), \end{cases} \tag{17}$$

for $\alpha \neq \phi$, and $\gamma = \min_{\alpha \in \mathscr{A} \cup \{\phi\}} \gamma_{\text{root}}(\alpha)$.

Proof. We will show that for a vertex v of height H, $\gamma_v(\alpha)$ is the cost of the subtree containing v, the vertices of height less than H to which it is connected, the vertices to which the latter are connected and so on, under the condition that α is assigned to v.

If v is of height one, all its ν neighbours but one are terminal vertices. It is easily verified that $\gamma_v(\alpha)$, for $\alpha \neq \phi$, is $ty + sx$ where t is the number of these terminal vertices with ϕ assigned, and s is the number with any $\beta \neq \alpha, \beta \in \mathscr{A}$ assigned. Similarly, $\gamma_v(\phi) = (\nu - t - 1)y$.

For v of height greater than one, (17) will have been proven to give the cost function for each of its neighbours but one. The subtree determined by v, these neighbours, etc. consists of their individual subtrees plus the $\nu - 1$ edges from v to its neighbours. Thus (17) gives the required cost for v. □

Backtracking applied to this algorithm gives the actual assignments of the minimizing values to each of the nonterminal vertices.

The evaluation of γ requires computational effort proportional to N and to the square of the size of the alphabet.

Given Theorem 5, the correctness of recurrence (16) in finding the most parsimonious alignment (and reconstruction of the protosequences) can be proved using the same arguments as for Theorem 1.

We are now in a position to state the theorem which encompasses all the previous ones, and whose proof is a straightforward generalization of that of Theorem 4, with the N-dimensional generalization of D given by (16) and (17).

THEOREM 6. *Let $F(\overrightarrow{i,j})$ be the minimum cost possible for N equivalent secondary structures $S_1, S_2, \cdots, S_N$ on positions $i_1, \cdots, j_1; i_2, \cdots, j_2; \cdots; i_N, \cdots, j_N$, of sequences $\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \cdots, \mathbf{a}^{(N)}$, respectively, this cost being the sum of the free energies and the*

alignment cost over a given phylogeny (tree). Let $C(\overrightarrow{i,j})$ *be the minimum cost under the constraints* $(i_1, j_1) \in S_1, \cdots, (i_N, j_N) \in S_N$ *but excluding the costs of aligning the elements of these closing pairs. If no such N-tuple of structures exists, then* $C = \infty$. *Then*

$$(18)\quad C(\overrightarrow{i,j}) = \min \begin{cases} \sum e(s_r) + D(\overrightarrow{i+1, j-1}), & s_r \text{ the hairpin closed by } (i_r, j_r), \\ \min\limits_{\substack{\text{not all} \\ (p_r, q_r) = (i_r, j_r)}} \sum e(s_r) + C(p, q) + D(\overrightarrow{i+1, p}) + D(\overrightarrow{q, j-1}), & \\ & s_r \text{ the 2-loop closed by } (i_r, j_r) \text{ with} \\ & (p_r, q_r) \text{ accessible, } p_r - i_r + j_r + q_r - 2 \leqq U, \\ \min\limits_{i_r < h_r < j_r - 1} G(\overrightarrow{i+1, h}) + G(\overrightarrow{h+1, j-1}) + NA, & \end{cases}$$

where

$$(19)\quad G(\overrightarrow{i,j}) = \min \begin{cases} C(i,j) + NP + \gamma(a_{i_1}^{(1)}, \cdots, a_{i_N}^{(N)}) + \gamma(a_{j_1}^{(1)}, \cdots, a_{j_N}^{(N)}), \\ \min\limits_{i_r < h_r < j_r} \min \begin{cases} G(\overrightarrow{i,h}) + \sum (j_r - h_r) Q + D(\overrightarrow{h+1, j}), \\ G(\overrightarrow{i,h}) + G(\overrightarrow{h+1, j}), \\ \sum (h_r - i_r + 1) Q + G(\overrightarrow{h+1, j}) + D(\overrightarrow{i,h}), \end{cases} \end{cases}$$

and

$$(20)\quad F(\overrightarrow{i,j}) = \min \begin{cases} C(\overrightarrow{i,j}) + \gamma(a_{i_1}^{(1)}, \cdots, a_{i_N}^{(N)}) + \gamma(a_{j_1}^{(1)}, \cdots, a_{j_N}^{(N)}), \\ \min\limits_{i_r \leqq h_r < j_r} \{F(\overrightarrow{i,h}) + F(\overrightarrow{h+1, j})\}, \\ D(\overrightarrow{i,j}), \end{cases}$$

with initial conditions $C(\overrightarrow{i,i}) = \infty$, *and* $G(\overrightarrow{i,j}) = \infty$ *if any* $i_r = j_r$.

An analysis of the computing time necessary to carry out the algorithm summarized in (18)–(20) shows that allowing all i_r, h_r and j_r to vary freely requires time proportional to n^{3N}, for fixed N. Computing time is also proportional to U^{2N} in the search for 2-loops. These considerations would seem to make the algorithm impractical, given that the n^3 algorithms presently available to find a single secondary structure are themselves unwieldy for large n.

However, in practice there is little reason to allow a free range of variation to all of $i_1, \cdots, i_N$. We can expect the optimal secondary structures to align positions relatively close to each other, so that for each $i = 1, \cdots, n$, we need consider only the case where for all i_r,

$$(21)\quad i - \frac{W}{2} \leqq i_r \leqq i + \frac{W}{2}$$

and similarly for h and j. (See Kruskal and Sankoff (1983) for a discussion of this way of "cutting corners" in dynamic programming.) Computing time is then proportional to $n^3(U^2 W)^N$, which is in the range of feasibility.

We point out that like Theorem 4, Theorem 6 is easily generalized to the case where (8) is replaced by (12), with attendant computational difficulties only if the calculation of $\mathbf{Q}$ is costly.

6. Discussion. In our analysis of the algorithms we have concentrated on time rather than memory requirements. Because we must store the $2N$-dimensional arrays

C, D, F and G as we calculate them, for use in later applications of the recurrences, memory requirements are of the order of n^{2N}. In cutting corners, however, this is reduced to $n^2(U^2W)^N$.

Promising directions for future work on these problems includes the weakening or elimination of the alignment constraints on equivalent structures—it should not be necessary to impose the condition that corresponding accessible and external pairs be aligned. Also the requirement that corresponding hairpins be aligned seems too strong, as does the requirement that elements of each k-loop never be mapped onto elements from more than one corresponding k-loop in another structure. Finally, it would be desirable to weaken the notion of equivalence itself, to allow some degree of structural difference if this produces much better energies and alignments.

REFERENCES

D. HIRSCHBERG (1983), *Recent results on the complexity of the common-subsequence problem*, in Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison, D. Sankoff and J. B. Kruskal, eds., Addison-Wesley, Reading, MA, pp. 325–330.

J. B. KRUSKAL AND D. SANKOFF (1983), *An anthology of algorithms and concepts for sequence comparison*, in Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison, D. Sankoff and J. B. Kruskal, eds., Addison-Wesley, Reading MA, pp. 265–310.

S. MAINVILLE (1981), *Comparaisons et auto-comparaisons de chaînes finies*, Ph.D. thesis, Université de Montréal, Montreal.

S. E. NEEDLEMAN AND C. D. WUNSCH (1970), *A general method applicable to the search for similarities in the amino-acid sequence of two proteins*, J. Molecular Biol., 48, pp. 443–453.

R. NUSSINOV, G. PIECZENIK, J. R. GRIGGS AND D. J. KLEITMAN (1978), *Algorithms for loop matchings*, this Journal, 35, pp. 68–82.

W. SALSER (1977), *Globin messenger-RNA sequences—analysis of base-pairing and evolutionary implications*, in Cold Spring Harbor Symposium on Quantitative Biology, 42, pp. 985–1002.

D. SANKOFF (1972), *Matching sequences under deletion-insertion constraints*, Proc. Natl. Acad. Sci. USA, 69, pp. 4–6.

———, (1975), *Minimal mutation trees of sequences*, this Journal, 78, pp. 35–42.

D. SANKOFF AND R. J. CEDERGREN (1983), *Simultaneous comparison of three or more sequences related by a tree*, in Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison, D. Sankoff and J. B. Kruskal, eds., Addison-Wesley, Reading, MA, pp. 253–263.

D. SANKOFF, R. J. CEDERGREN AND G. LAPALME (1976), *Frequency of insertion-deletion, transversion, and transition in the evolution of 5S ribosomal RNA*, J. Molecular Evolution, 7, pp. 133–149.

D. SANKOFF AND J. B. KRUSKAL, eds. (1983), *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, Addison-Wesley, Reading, MA.

D. SANKOFF, J. B. KRUSKAL, S. MAINVILLE AND R. J. CEDERGREN (1983), *Fast algorithms to determine RNA secondary structures containing multiple loops*, in Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison, D. Sankoff and J. B. Kruskal, eds., Addison-Wesley, Reading, MA, pp. 93–120.

D. SANKOFF, C. MOREL AND R. J. CEDERGREN (1973), *Evolution of 5S RNA and the nonrandomness of base replacement*, Nature New Biology, 245, pp. 232–234.

D. SANKOFF AND P. ROUSSEAU (1975), *Locating the vertices of a Steiner tree in an arbitrary metric space*, Math. Programming, 9, pp. 240–246.

P. H. SELLERS (1974a), *An algorithm for the distance between two finite sequences*, J. Combin. Theory, 16, pp. 252–258.

———, (1974b), *On the theory and computation of evolutionary distances*, this Journal, 26, pp. 787–793.

I. TINOCO, Jr., P. N. BORER, B. DENGLER, M. D. LEVINE, O. C. UHLENBECK, D. M. CROTHERS AND J. GRALLA (1973), *Improved estimation of secondary structure in ribonucleic acids*, Nature New Biology, 246, pp. 40–41.

M. S. WATERMAN (1978), *Secondary structure of single-stranded nucleic acids* in Studies in Foundations and Combinatorics, Advances in Mathematics Supplementary Studies, Vol. 1, Academic Press, New York, pp. 167–212.

———, (1984), *General methods of sequence comparison*, Bull. Math. Biol., 46, pp. 473–500.

M. S. WATERMAN AND T. F. SMITH (1978), *RNA secondary structure: a complete mathematical analysis*, Math. Biosci., 42, pp. 257-266.

M. ZUKER AND P. STIEGLER (1981), *Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information*, Nucleic Acids Res., 9, pp. 133-148.

M. ZUKER AND D. SANKOFF (1984), *RNA secondary structures and their prediction*, Bull. Math. Biol., 46, pp. 591-621.