SIAM J. APPL. MATH.
Vol. 48, No. 5, October 1988

007

THE MULTIPLE SEQUENCE ALIGNMENT PROBLEM IN BIOLOGY*

HUMBERTO CARRILLO†‡ AND DAVID LIPMAN†

Abstract. The study and comparison of sequences of characters from a finite alphabet is relevant to various areas of science, notably molecular biology. The measurement of sequence similarity involves the consideration of the different possible sequence alignments in order to find an optimal one for which the "distance" between sequences is minimum. By associating a path in a lattice to each alignment, a geometric insight can be brought into the problem of finding an optimal alignment. This problem can then be solved by applying a dynamic programming algorithm. However, the computational effort grows rapidly with the number N of sequences to be compared ($\mathbf{O}(l^N)$, where l is the mean length of the sequences to be compared).

It is proved here that knowledge of the measure of an arbitrarily chosen alignment can be used in combination with information from the pairwise alignments to considerably restrict the size of the region of the lattice in consideration. This reduction implies fewer computations and less memory space needed to carry out the dynamic programming optimization process. The observations also suggest new variants of the multiple alignment problem.

Key words. sequence comparison, biological sequences, dynamic programming

AMS(MOS) subject classifications. 49, 68, 92

1. Introduction.

The comparison of two or more sequences of numbers or letters is common in several fields, such as molecular biology, speech recognition, and computer science (see Sankoff and Kruskal [11] for an overview of the area). Sequence comparison is particularly important in molecular biology where it has been critical in the study of evolution, the control of gene expression, and in the analysis of protein structure/function relationships.

Let us consider the problem of comparing two related proteins: sequences whose elements are taken from an alphabet of twenty different amino acids. In the course of divergence from a common ancestor, mutations may occur involving amino acid replacements, insertions, and deletions. Lacking knowledge of the original correspondence of amino acids, we may consider all possible transformations of the sequences using insertions, deletions, and replacements and choose a configuration that is optimal with respect to the resulting intersequence correspondence of amino acids. The criterion of optimality may also consider an associated cost of the transformations involved so as to choose the most "likely" set of transformations given some model of protein evolution. For example, some amino acid replacements may be more costly than others and, because several consecutive amino acids may be inserted in a single mutational event, the cost of adjacent insertions may not increase in a strictly additive fashion (see Fitch and Smith [3]).

In problems such as the construction of an evolutionary tree based on sequence data, or in protein engineering, where a multiple alignment of related sequences may often yield the most helpful information in the design of a new protein, a molecular biologist must compare more than two sequences simultaneously.

To solve the optimization problems of sequence comparison that appear in molecular biology and other areas such as speech processing and computer science,

* Received by the editors May 25, 1987; accepted for publication (in revised form) September 18, 1987.

† Mathematical Research Branch, National Institute of Diabetes and Digestive and Kidney Diseases, National Institutes of Health, Bethesda, Maryland 20892.

‡ This author is on leave from the Universidad Nacional Autónoma de México.

the methods of dynamic programming have proven to be useful (see Kruskal [7]). The dynamic programming approach has the limitation that its complexity scales up greatly with dimension (see Sankoff et al. [12]). In the following, we make observations on the problem of aligning N sequences and that of aligning subsets of these sequences that reveal constraints of the problem that will prove useful in reducing computation in the dynamic programming method.

2. Sequences. We assume here that an alphabet α of n characters is given:

$$\alpha = \{\alpha_1, \cdots, \alpha_n\}.$$

A *sequence* of k characters, or a *k-sequence*, of this alphabet is a subset of

$$\alpha^k = \prod_{i=1}^{k} \alpha$$

where $\prod$ stands for the cartesian product of sets. Thus a k-sequence S is a set of the form

$$S = (\alpha_{n_1}, \cdots, \alpha_{n_k})$$

where for each $j = 1, \cdots, k$, n_j is a natural number that satisfies that $1 \leqq n_j \leqq n$. Together with α we will consider another alphabet β which is obtained from α by adding the blank character "$-$":

$$\beta = \{-\} \cup \{\alpha_1, \cdots, \alpha_n\}.$$

To simplify notation in the present discussion we will assume that α is the set of the first n natural numbers $\alpha = \{1, \cdots, n\}$.

3. Alignments. An *alignment* of the sequences $S_1, \cdots, S_n$ is another set of sequences, $\bar{S}_1, \cdots, \bar{S}_n$, such that each sequence $\bar{S}_i$ is obtained from S_i by inserting blanks in positions where some of the other sequences have a nonblank character.

More precisely, if the sequences are

$$S_1 = (n_1^1, \cdots, n_{k_1}^1),$$
$$\cdots$$
$$S_N = (n_1^N, \cdots, n_{k_N}^N),$$

then the alignment $(\bar{S}_1, \cdots, \bar{S}_N)$ is an element of the set

$$\bigcup_{i=\max\{k_1,\cdots,k_N\}}^{k_1+\cdots+k_N} \prod_{j=1}^{N} \beta^i$$

that satisfies the following conditions. Assuming that for each $i = 1, \cdots, N$,

$$\bar{S}_i = (m_1^i, \cdots, m_l^i),$$

with

$$\max\{k_1, \cdots, k_N\} \leqq l \leqq k_1 + \cdots + k_N,$$

then there are increasing functions

$$\nu_i: \quad \{1, \cdots, k_i\} \to \{1, \cdots, l\}$$

such that:

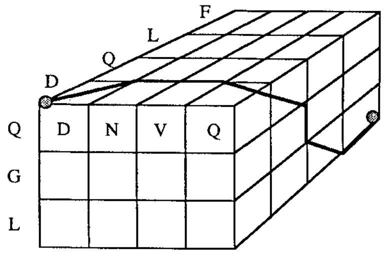
(a) $m^i_{\nu_i(j)} = n^i_j$ for $j = 1, \cdots, k_i$.

(b) If some $j = 1, \cdots, l$ is not in the image of the function ν_i (i.e., $j \neq \nu_i(k)$ for all $k = 1, \cdots, k_i$), then m^i_j is a blank.

Finally, it is also required that for each $j = 1, \cdots, l$, there is at least one value of i for which m^i_j is not a blank. Each alignment of the sequences $S_1, \cdots, S_N$ encodes a set of different insertions, deletions, and replacements that transform one sequence into another. As an illustration of the notion of an alignment, we present a simple example. Consider the short amino acid sequences, $S_1 = DQLF$, $S_2 = DNVQ$, and $S_3 = QGL$, where each capital letter corresponds to a different amino acid (note that most proteins are at least 100 amino acids in length). A possible alignment of these sequences is shown in Fig. 1.

4. Paths. To any given set of N sequences $S_1, \cdots, S_N$ of length $k_1, \cdots, k_N$, we will associate a *lattice* $L(S_1, \cdots, S_N)$ in N-dimensional space. This lattice consists of the N-dimensional hypercubes (from here on, referred to simply as cubes) that are obtained by making the Cartesian product of N strings of squares. Each of these strings is considered to be associated to a particular sequence and has as many squares as characters in the corresponding sequence (see Fig. 1). All cubes in the lattice form an N-dimensional parallelepiped. The corner of this parallelepiped that corresponds to the first character of all sequences is called the *original corner.* The corner of the parallelepiped farthest away from the original corner, corresponding to the last character of all sequences, is called the *end corner.* The corner of a sublattice L' of L that is closest to the original corner of L will be called the original corner of L'. Similarly, the corner that is closest to the end corner of L will be called the end corner of L'. When $N = 2$ the lattice $L(S_1, S_2)$ is just a net of squares or a "matrix."

A *path* $\gamma(S_1, \cdots, S_N)$ between the sequences $S_1, \cdots, S_N$ is a connected broken line joining the original corner to the end corner. The segments of this broken line join vertices of the lattice that belong to one cube. We also require a path to be such that each intersection of it with a plane parallel to the faces of the parallelepiped has to contain only one point or the union of several edges of the cubes of the lattice. An alternative expression of this latter property is to say that if a point is moving along the broken line, starting at the original corner, then each time it travels a segment, it gets closer to the end corner. Paths in an N-dimensional lattice encode alignments of the N sequences that form the lattice. Reciprocally, each alignment of the sequences $S_1, \cdots, S_N$ determines a unique path in the corresponding N-dimensional lattice. This one-to-one relationship between alignments and paths is very convenient because it provides us with a geometrical insight to study the problem of sequence comparison. For example, a path in an N-dimensional lattice, $\gamma(S_1, \cdots, S_N)$, can be projected



```
D - - Q - L F
D N V Q - - -
- - - Q G L -
```

FIG. 1

into any of the planes formed by each pair of sequences (S_i, S_j). The projected path, which we denote $p_{ij}(\gamma(S_1, \cdots, S_N))$, represents an alignment of the sequences S_i and S_j.

5. Optimal paths. Each alignment of a set of sequences is to be understood as a pattern of comparison of these sequences. If a set of scoring rules that penalize each individual deletion, insertion, or replacement is given, then we may assign a score to an alignment corresponding to the sum of the scores of each of the transformations that it encodes. This gives rise to the problem of finding optimal alignments in the sense that they have a minimum score associated with them. Conceivably, there are many different ways in which scoring rules can be established. The penalty for a particular transformation might be dependent on the position where it occurs and on the transformations that have been made previously. For example, in the evolution of proteins and nucleic acids, the probability of two adjacent deletions is not the square of the probability of a single deletion because a single mutational event may have been responsible for both deletions.

Since each path is associated with a unique alignment, in this way the scoring rules allow us to assign to any given path γ a *measure* that we denote by $M(\gamma)$. The number $M(\gamma)$ is a measure of the similarity among the sequences $S_1, \cdots, S_N$ when they are compared according to the alignment associated to the path γ. In some particular cases M will make a metric space out of the set of all paths (Sellers [13]), but this is not true for all biologically interesting measures.

Corresponding to each measure $M(\gamma)$ there is at least one path, $\gamma^*(S_1, \cdots, S_N)$, such that M attains a minimum value at γ^*. Such a path is called an *optimal path*. The optimal paths in the lattice $L(S_1, \cdots, S_N)$ can be found by dynamic programming methods. In what follows we give a brief description of how the algorithm proceeds, (see Needleman and Wunsch [9] for the first application of dynamic programming to biological sequence comparison and Waterman [17] for a recent review).

Consider the N-dimensional lattice $L(S_1, \cdots, S_N)$. Each vertex in L may be thought of as the end corner of the sublattice $L(S_1(i_1), \cdots, S_N(i_N))$, where for each sequence S, $S(i)$ denotes the sequence consisting of the first i elements of S. The idea of the method is to recursively find optimal paths for all these sublattices of $L(S_1, \cdots, S_N)$. Initially, we compute the score of each of the possible paths on the cube that has a vertex at the original corner. Next, using this information, we compute the minimum score needed to reach from the original corner to the vertices of the adjacent cubes through a valid path. This process is repeated until we calculate the minimum score needed to reach the end corner, i.e., the optimal measure of a path in $L(S_1, \cdots, S_N)$. If together with the optimal score of each vertex of the lattice we keep in memory a pointer to mark, in each step of the recursion, the segment of minimal contribution, then we can trace back through the pointers, constructing an optimal path.

It can be shown for sequences $\{S_1, \cdots, S_N\}$ of lengths $\{k_1, \cdots, k_N\}$, that this computation of γ^* takes on the order of $\prod_{i=1}^N k_i$ steps. We remark that the above applies only to scoring rules for which the penalty associated to the transformations that take place in a given position of the alignment is independent of the transformations that have taken place at previous positions. See Gotoh [4], and Waterman [16] for discussion of a similar procedure which applies to useful special cases of scoring rules with memory, and also requires $\mathbf{O}(\prod_{i=1}^N k_i)$ steps.

6. Constraints on multiple alignments. We proceed now to make some observations on the problem of determining $\gamma^*(S_1, \cdots, S_N)$ where $N > 2$, that will result in significantly fewer computations. The essence of a multiple alignment of the sequences $S_1, \cdots, S_N$ is to create a collective pattern of comparison that allows us to measure simultaneously the similarity among them. In particular, a multiple alignment implies $\binom{N}{2}$ pairwise alignments of these sequences. On the other hand, $\binom{N}{2}$ arbitrarily given pairwise alignments do not determine a multiple alignment unless they satisfy a compatibility condition that allows the simultaneous comparison of the N sequences.

A natural measure for the similarity of a set of sequences would be some nondecreasing function of the sum of the measures of pairwise similarity (see Murata et al. [8] and Gotoh [5]). To include the case in which the similarity of the pairs could be considered in a nonuniform way in measuring the similarity of the N sequences, we consider that the similarity between the sequences S_i and S_j is given by the measure μ_{ij} of two-dimensional paths in the lattice $L(S_i, S_j)$. Thus the measure M of an N-dimensional path γ is a nondecreasing function of

$$\sum_{i<j}^{N} \mu_{ij}(p_{ij}(\gamma)) \tag{1}$$

where p_{ij} are the projections of γ on the plane determined by the sequences S_i and S_j, i.e., M must be of the form

$$M(\gamma)=H\left(\sum_{i<j}^{N} \mu_{ij}(p_{ij}(\gamma))\right)$$

where H is a nondecreasing function. In particular, this condition is satisfied by any N-dimensional measure that is related to a two-dimensional measure by

$$M(\gamma)=\sum_{i<j}^{N} F_{ij}(\mu(p_{ij}(\gamma)))$$

where F_{ij} are nondecreasing functions.

In a specific application the scoring rules that have produced the measure in consideration may have been designed such that $M(\gamma(S_1, \cdots, S_N))$ measures similarity between the sequences $S_1, \cdots, S_N$ or, alternatively, that $M(\gamma(S_1, \cdots, S_N))$ is the sum of the edges of an evolutionary tree whose leaves correspond to the sequences of interest (see Sankoff [10] and Sankoff et al. [12]). In the following discussion we will assume that the measure is of the form expressed in (1); however, the approach presented here may also be extended to the case of sequences related by an evolutionary tree (Altschul and Lipman [2]).

Let us consider any N-dimensional path γ^e that will be called *γ-estimated.* By optimality,

$$M(\gamma^e)-M(\gamma^*)\geqq 0,$$

and then

$$\sum_{i<j}^{N} \mu_{ij}(p_{ij}(\gamma^e))-\sum_{i<j}^{N} \mu_{ij}(p_{ij}(\gamma^*))\geqq 0.$$

Let γ_{ij}^* denote an optimal path for the measure μ_{ij} in the two-dimensional lattice determined by the sequences S_i and S_j. Since $\mu_{ij}(p_{ij}(\gamma^*))\geqq \mu_{ij}(\gamma_{ij}^*)$, we have that

$$\sum_{i<j}^{N} \mu_{ij}(p_{ij}(\gamma^e))-\sum_{\substack{i<j\\(i,j)\neq(k,l)}}^{N} \mu_{ij}(\gamma_{ij}^*)\geqq \mu_{kl}(p_{kl}(\gamma^*)).$$

Let us define U_{kl} as

$$U_{kl}=\sum_{i<j}^{N} \mu_{ij}(p_{ij}(\gamma^e))-\sum_{\substack{i<j\\(i,j)\neq(k,l)}}^{N} \mu_{ij}(\gamma_{ij}^*).$$

For each k, $l=1,\cdots,N$, this positive number U_{kl} is an upper bound for the measure of the projection of any N-dimensional optimal path into the plane determined by the sequences S_k and S_l. Then, when looking for γ^* we need only consider those paths γ in $L(S_1,\cdots,S_N)$ that satisfy that $\mu_{kl}(p_{kl}(\gamma))\leqq U_{kl}$. We will call this set of paths X_{kl}.

Thus the paths γ in the set

$$X=\bigcap_{i<j}^{N} X_{ij}$$

are the only possible candidates to be an optimal path. To consider only paths in X means having to apply the dynamic programming procedure to find γ^* only in some subregion Y of $L(S_1,\cdots,S_N)$.

Let Y_{ij} be the set of squares of $L(S_i, S_j)$ whose end corners are traversed by some path of measure smaller or equal to U_{ij}. Let Y_{ij}^{-1} be the set of points $x\in L(S_1,\cdots,S_N)$ such that $p_{ij}(x)\in Y_{ij}$, where p_{ij} is the projection into the plane $L(S_i, S_j)$. The set Y_{ij}^{-1} contains all paths in X_{ij} and the set

$$Y=\bigcap_{i<j}^{N} Y_{ij}^{-1}$$

contains all the paths in X. Y is a region of $L(S_1,\cdots,S_N)$ that may not be connected but the connected component of it that contains the original corner will contain all the optimal paths. Generally Y is such that $p_{ij}(Y)$ is a proper subset of Y_{ij}.

The above reasoning proves that it is unnecessary to apply the dynamic programming method to the entire lattice $L(S_1,\cdots,S_N)$, but it suffices to consider just the subregion Y.

The regions Y_{ij} can be determined using a simple variation of the dynamic programming method previously described. Recall that in that algorithm, we compute recursively the optimal path scores of each sublattice of L which shares the original corner of L. This recursion could be done in the opposite direction (end corner to original corner), computing optimal path scores for each sublattice which shares the end corner of L. Then, for any vertex in L, we know an optimal path from that vertex to the original corner, and an optimal path to the end corner. Therefore we know the optimal path score associated with each vertex of the lattice, which is exactly the information we need to decide whether a particular point is an element of Y_{ij}. It can be shown that, in this way, all regions Y_{ij} may be computed in $\mathbf{O}(\sum_{i<j}^{N} k_ik_j)$ steps, where $k_1,\cdots,k_N$ are the lengths of the sequences $S_1,\cdots,S_N$. See Altschul and Erickson [1] and Zuker [19] for a complete discussion of this method.

7. Estimated paths. The smaller the region Y, the smaller the amount of computation necessary to find an optimal path. To obtain a small region Y we would need to produce an estimated path γ^e with measure close to the measure of an optimal path. This is because the minimum upper bounds U_{ij} occur when $M(\gamma^e)$ equals the measure of an optimal path. One approach used to deal with this problem is based on the following simple observation: If there is a path γ such that $p_{ij}(\gamma)=\gamma_{ij}^*$ for all i, j, where γ_{ij}^* denotes an optimal path on the lattice $L(S_i, S_j)$, then γ is an optimal path for the N-dimensional problem. Therefore, since $M(\gamma)$ is a nondecreasing function of $\sum \mu_{ij}(p_{ij}(\gamma))$, the idea is to construct γ^e in such a way that its projections $p_{ij}(\gamma^e)$ are as close as possible to the optimal paths γ_{ij}^*.

Given $\binom{N}{2}$ two-dimensional paths, it is not always possible to find an N-dimensional path γ that will have them as projections. This is because, being part of an alignment of N sequences, the alignment encoded in the two-dimensional paths $p_{ik}(\gamma)$ and $p_{kj}(\gamma)$ imply some restrictions on the alignment of the sequences S_i and S_j encoded in $p_{ij}(\gamma)$. Therefore a given set of $\binom{N}{2}$ two-dimensional optimal paths $\{\gamma_{ij}^*\}$ is the set of projections of an N-dimensional path only when the paths γ_{ij}^* are properly compatible. If $N-1$ two-dimensional paths are given, it can be guaranteed that there is at least one N-dimensional path of which they are all projections if each of the N dimensions is represented in at least one of the two-dimensional paths.

Because of the possible insertion of gaps, a set of $N-1$ paths γ_{ij}^*, chosen in a compatible way do not necessarily determine a unique N-dimensional path that will project into them. For instance, given the paths $\gamma_{1,2}^*, \cdots, \gamma_{1,N}^*$, it could be that for $k=2, \cdots, N$, the regions $\Gamma_{1,k} \subset L(S_1, \cdots, S_N)$ with the property that

$$\Gamma_{1,k} = p_{1,k}^{-1}(\gamma_{1,k}^*) = \{x \in L(S_1, \cdots, S_N) \mid p_{1,k}(x) \in \gamma_{1,k}^*\}$$

are such that the set

$$\Gamma = \bigcap_{k=2}^{N} \Gamma_{1,k}$$

is not a path but contains an m-dimensional region with $1 < m < N$. To determine a convenient path through this lower-dimensional subregion of $L(S_1, \cdots, S_N)$, Γ, another optimization problem must be solved. Note that this problem can, in turn, be reduced in the same way that the original problem was to obtain a region of smaller dimension. Repeating this procedure, the region where the optimization must be carried out will eventually be two-dimensional. This method of producing an estimated path is heuristic and there is no guarantee that the resulting path will always have a measure close to that of an optimal path.

Another approach is to start with a guess of an estimated measure M^e which, unlike $M(\gamma^e)$, may not necessarily correspond to the measure of an actual γ. We may then determine all upper bounds using M^e and proceed to compute an "optimal" path γ^{e^*} derived from this initial estimate. It is then trivial to show that if $M(\gamma^{e^*}) \leqq M^e$, then γ^{e^*} is a true optimal path. If $M(\gamma^{e^*}) > M^e$, then $M(\gamma^{e^*})$ can be used to determine the upper bounds required in the computation of γ^*.

8. Discussion. We have demonstrated an important relationship between the alignment of N sequences and the problem of aligning subsets of these sequences. Although we examined the case where the lower dimension was two, our results may be generalized for any dimension less than N. This relationship can be exploited to develop multiple alignment algorithms with reduced computational requirements. Though we have not proposed a detailed algorithm, it is clear that the computational requirements of such an algorithm would be a function of the size of the subregion Y in the lattice $L(S_1, \cdots, S_N)$ plus the number of computations necessary to generate it. The region Y contains only those paths whose projections p_{ij} have scores less than their respective upper bounds U_{ij} and is contained in the region

$$Y_i = \bigcap_{j \neq i}^{N} Y_{ij}^{-1},$$

where $i \in \{1, \cdots, N\}$. We can easily estimate the size of the region Y_i if we know the average number of cubes in the j dimension of Y_{ij}. Here we consider the size of a subregion $\mathbf{R}$ of L to be the number of hypercubes that it contains, and we denote it by $|\mathbf{R}|$. Let us define f_{ij}, the size of Y_{ij} relative to the size of $L(S_i, S_j)$, as

$$f_{ij}=\frac{|Y_{ij}|}{k_i k_j},$$

where k_i and k_j are the lengths of the sequences S_i and S_j (i.e., $k_i k_j = |L(S_i, S_j)|$). Then the average number of cubes in the j dimension of Y_{ij} is $f_{ij}k_j$, and

$$|Y_i| \approx k_i \prod_{j\neq i}^{N} f_{ij}k_j.$$

To determine the regions Y_{ij} requires $\mathbf{O}(\sum_{i<j}^{N} k_i k_j)$ computations, a figure which does not grow exponentially with N. Given the regions Y_{ij}, the region Y_i can be generated and the optimization can be carried out within in

$$\mathbf{O}\left(k_i \prod_{j\neq i}^{N} f_{ij}k_j\right)$$

steps; the memory requirements are of the same order. Since we are not using the information from all the Y_{ij}'s, the size of Y_i is an upper bound on the size of Y. Whether it would be more efficient to carry out the optimization in the region Y or in the region Y_i would depend on the particular application. Clearly there are a number of strategies for using the information that the regions Y_{ij} provide.

Complementary to the above reasoning, further computational improvements may be achieved by noting that not all points in the subregion Y must be stored in the computation of an optimal path γ^*. This is because, in performing the dynamic programming recursion, if we can determine that the lower bound cost of any path through some point in Y is greater than $M(\gamma^e)$, then that point is not necessary in the ongoing computation. For simplicity, let us consider the case where $M(\gamma)=\sum_{i<j}^{N} \mu_{ij}(p_{ij}(\gamma))$. Recall that in the dynamic programming algorithm we recursively compute the minimal path score from the original corner to each point in the subregion Y. Recall also that in computing the Y_{ij}'s, we have determined a lower bound for the cost of a projeced path from any point in the lattice to the end corner of the lattice. Then a lower bound on the measure of the paths through a given point of Y is calculated by adding the minimal path score from the original corner (which would be computed at this step of the recursion) to the sum of the lower bounds of the projected paths to the end corner. This approach would be expected to be particularly useful when dealing with gap functions of the sort discussed by Gotoh [4] and Waterman [16].

Ukkonen [15] has described a method for decreasing the computational cost of pairwise alignments when the measure of distance is very simple (e.g., identities have distance zero, all other transformations contribute a distance of one). It is based on the observation that a priori $M(\gamma^*)\geqq 0$, and if $M(\gamma^*)=0$, the sequences are identical and the location of γ^* is on the main diagonal of the plane. Therefore there is a relation between the score of a sublattice end corner and its distance from the main diagonal. In computing an optimal path, we proceed recursively through the two-dimensional lattice, computing the scores of sublattice end corners on diagonals at successively greater distance from the main diagonal only when they could potentially lead to paths with better scores. This approach can be generalized to N dimensions. Here the subregion of the lattice $L(S_1, \cdots, S_N)$, considered in the optimization, expands in a roughly symmetrical fashion around the a priori optimal path of measure zero, until it contains all sublattice end corners with optimal path scores less then or equal to $M(\gamma^*)$.

If the sequences to be compared differ by only a very few insertions, deletions, and replacements, a generalized Ukkonen algorithm would have computational advantages over an algorithm based on our observations. In comparisons where the sequences are more divergent (which is generally the case in biological sequence comparisons), we might consider using the Ukkonen algorithm, but only on those points which project within the regions Y_{ij}. However, Ukkonen's method becomes quite complicated and less efficient with more realistic measures of distance.

We have described a method of efficiently determining an estimated path that, however, is not guaranteed to always find paths close in measure to an optimal path. Other methods of determining estimated paths have been described (Sobel and Martinez [14], Johnson and Doolittle [6] and Waterman [18]) and may have advantages in some cases. Clearly, the problem of computing good estimated paths requires further investigation since this information can now be usefully applied to the problem of computing an optimal path.

Our results lend themselves to a view of the N sequence multiple alignment problem as one of finding an optimal set of pairwise alignments (or any dimension less than N) which satisfy certain compatibility conditions. In other words, we must find an n-dimensional path whose projections are optimal. This view suggests new forms of the problem. For example, we may consider a nonuniform weighting of the pairwise comparisons to force the multiple alignment such that some sequences are closer to their pairwise optima than others. As another example, using pairwise distance measures μ_{ij} specific to certain subsets of sequences may be more appropriate when doing interfamily sequence comparisons. Understanding the relationship between higher- and lower-dimensional alignments can therefore lead to more efficient algorithms, and to variants of the original problem which may have useful applications.

REFERENCES

[1] S. F. Altschul and B. W. Erickson, *Locally optimal subalignments using nonlinear similarity functions*, Bull. Math. Biol., 48 (1986), pp. 633-660.

[2] S. F. Altschul and D. J. Lipman, *Trees, stars, and multiple biological sequence alignment*, SIAM J. Appl. Math., to appear.

[3] W. M. Fitch and T. F. Smith, *Optimal sequence alignments*, Proc. Nat. Acad. Sci. U.S.A., 80 (1983), pp. 1382-1386.

[4] O. Gotoh, *An improved algorithm for matching biological sequences*, J. Mol. Biol., 163 (1982), pp. 705-708.

[5] ——— *Alignment of three biological sequences with an efficient traceback procedure*, J. Theoret. Biol., 121 (1986), pp. 327-337.

[6] M. S. Johnson and R. F. Doolittle, *A method for the simultaneous alignment of three or more amino acid sequences*, J. Mol. Evol., 23 (1986), pp. 267-278.

[7] J. B. Kruskal, *An overview of sequence comparison: time warps, string edits and macromolecules*, SIAM Rev., 25 (1983), pp. 201-237.

[8] M. Murata, J. S. Richardson, and J. L. Sussman, *Simultaneous comparison of three protein sequences*, Proc. Nat. Acad. Sci. U.S.A., 82 (1985), pp. 3073-3077.

[9] S. B. Needleman and C. D. Wunsch, *A general method applicable to the search for similarities in the amino acid sequences of two proteins*, J. Mol. Biol., 48 (1970), pp. 444-453.

[10] D. Sankoff, *Simultaneous solution of the RNA folding alignment and protosequence problems*, SIAM J. Appl. Math. 45 (1985), pp. 810-825.

[11] D. Sankoff and J. B. Kruskal, eds., *Time Warps, String Edits and Macromolecules: the Theory and Practice of Sequence Comparison*, Addison-Wesley, London, 1983.

[12] D. Sankoff, C. Morel, and R. J. Cedergren, *Evolution of 5S ribosomal RNA*, Nature New Biology, 245 (1973), pp. 232-234.

[13] P. Sellers, *On the theory and computation of evolutionary distances*, SIAM J. Appl. Math., 26 (1974), pp. 787-793.

[14] E. SOBEL AND H. MARTINEZ, *A multiple sequence alignment program*, Nucleic Acids Research, 14 (1986), pp. 363-374.

[15] E. UKKONEN, *On Approximate String Matching*, Proc. Internat. Conference on Foundation of Comput. Theory, Lecture Notes in Computer Science 158, Springer-Verlag, Berlin, 1983, pp. 487-496.

[16] M. S. WATERMAN, *Efficient sequence alignment algorithms*, J. Theoret. Biol., 108 (1984), pp. 333-337.

[17] ———, *General methods of sequence comparison*, Bull. Math. Biol., 46 (1984), pp. 473-500.

[18] ———, *Multiple sequence alignment by consensus*, Nucleic Acids Research, 14 (1986), pp. 9095-9102.

[19] M. ZUKER, *Suboptimal sequence alignments*, manuscript in preparation.