SIAM J. APPL, MATH.
Vol. 35, No. 1, July 1978

ALGORITHMS FOR LOOP MATCHINGS*

RUTH NUSSINOV,† GEORGE PIECZENIK,‡ JERROLD R. GRIGGS¶
AND DANIEL J. KLEITMAN§

Abstract. A simplified (two-base) version of the problem of planar folding of long chains (e.g., RNA and DNA biomolecules) is formulated as a matching problem. The chain is prescribed as a loop or circular sequence of letters A and B, n units long. A matching here means a set of A-B base pairings or matches obeying a planarity condition: no two matches may cross each other if drawn on the interior of the loop. Also, no two adjacent letters may be matched. We present a dynamic programming algorithm requiring $O(n^3)$ steps and $O(n^2)$ storage which computes the size of the maximum for the given A-B base sequence and which also allows reconstructing a particular folded form of the original string which realizes the maximum matching size. The algorithm can be adapted to deal with sequences with larger alphabets and with weighted matchings.

An algorithm is also presented for a modified problem closer to the biochemical problem of interest: We demand that every match must be adjacent to another match, forcing groups of two or more parallel matches.

Some results on the expected maximum matching size are presented. As $n \to \infty$, at least 80% of the vertices can be matched on the average on an A-B string of size n.

We briefly discuss the practical application of the algorithm by using contracted versions of very long molecules with a preliminary block construction. A maximum matching is presented for the J-gene of the ϕX174 DNA virus. We conclude by stating some problems requiring further study.

1. Introduction.

The recent advances in sequencing methods have led by now to the complete determination of the nucleotide sequences of the MS2 RNA [8] and ϕX174 single stranded DNA [16] viruses containing 3,569 and 5,473 units (bases) respectively, and the sequencing of much longer chains seems imminent. Certain portions of these chains constitute the various "genes". These genes determine—via the universal genetic code which assigns to each triplet of bases an amino acid—the various proteins necessary for the viral life-cycle.

It is apparent that the nucleotide sequence carries in addition another type of information which plays a crucial role in the regulation of gene expression. The sequence of nucleotides determines the actual shape of the biologically active three-dimensional form (or forms) of the molecule. By folding into a particular shape, certain "initiation sites" of genes which should be rarely decoded can be shielded, and conversely, initiation sites of frequently decoded genes maximally exposed.

It is generally believed and tested in the case of small t-RNA molecules, that the sequence of nucleotides ("primary" structure) determines a shape ("secondary" and "tertiary" structures) or several folded shapes in such a way that the interactions minimize the energy.

A simplistic consideration of nucleotide self-interaction is base pairing, like the classical C-G (cytosine-guanine) and A-T (adenine-thymine, or, in RNA, A-U, adenine-uracil) pairing in the Watson–Crick model of the double helix. In RNA the

* Received by the editors February 16, 1977, and in revised form July 20, 1977.

† Biochemistry Department, Nelson Biological Laboratories, Rutgers University, New Brunswick, New Jersey. Now at Department of Structural Chemistry, Weizmann Institute of Science, Rehovot, Israel. This work is in partial fulfillment of Ph.D. requirements.

‡ Biochemistry Department, Nelson Biological Laboratories, Rutgers University, New Brunswick, New Jersey 08903. This research was supported in part under N.I.H. Grant GM-27-2861.

¶ Department of Mathematics, Massachusetts Institute of Technology, Cambridge, Massachusetts. Now at Department of Mathematics, California Institute of Technology, Pasadena, California, 91125. This work is in partial fulfillment of Ph.D. requirements.

§ Department of Mathematics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139. This research was supported in part by the Office of Naval Research under Contract PNDGN-00014-76-C-036.

existence of A-U, G-C, and G-U base pairs has been confirmed directly from x-ray diffraction work by Klug et al. [1] and Kim et al. [2], [3].

The folding of an N nucleotide chain is then specified by a symmetric $N \times N$ matrix M with $M_{ij} = 1$ if the (compatible) bases in the ith and jth locations along the chain have been paired in the shape considered and $M_{ij} = 0$ otherwise. An arbitrary matrix like this is likely to correspond to a shape which is not realizable because of the spatial hindrance and the stiffness of double helical (paired) regions of bases.

For that reason and for facilitating the visual representation, only (what we loosely refer to as) secondary structure of long polynucleotide chains has been considered to date. These structures are topologically equivalent to planar graphs obtained by closing the nucleotide chain into a planar loop and (imagining that all bases are oriented towards the interior of the loop) forming nonintersecting pairings of (compatible) bases.

Even with these restrictions the combinatorial dimensions of the folding problem for $N \sim 10^3 - 10^4$ are immense, and the simple visual matrix method and existing computer programs appear inadequate for treating it.

In the first method one draws an $N \times N$ matrix P with all potential pairings ($P_{ij} = 1$ if the bases at locations i and j can pair, and $P_{ij} = 0$ otherwise). Then by looking for lines of 1's running diagonally across the matrix, one attempts to identify potential sections which can be matched ("blocks") and piece them together into an optimal planar folded shape.

While this simple method has been extremely useful for suggesting folded shapes of small RNA molecules, $N \sim 50$–100, we found that it is impractical to extend it to larger N. The other approach known to us, the computer programs of Pipas and McMahon [17], cannot deal with longer chains either. In fact their approach is quite elaborate and attempts to identify and weigh energetically with appropriate weights various types of components in folded shapes such as "hairpin"-like sections and "bulges".

The main purpose of this paper is to formulate and to solve by an optimizing dynamic programming matching algorithm the planar folding problem described above.

While the time requirement of $O(n^3)$ (on the order of n^3 elementary operations) of the resulting algorithm is reasonable even for values of n of a few thousand, the $O(n^2)$ memory requirements are somewhat of a practical difficulty.

Another basic shortcoming of the simple algorithm from the biological point of view is the neglect of "base-stacking" effects. The underlying assumption that base pairing *per se* was sufficient to define the actual stability of secondary structure in RNA was brought into strong question by Crothers et al. [4], who showed the importance of base stacking. This is the configuration of adjacent bases. This shifted the emphasis from interaction of independent bases in the polymer to the context of the adjacent bases as well. Base-stacking interactions allow the stabilization of single-stranded elements in nucleic acids as can be seen in the anti-codon region of the x-ray crystal structure of t-RNA.

In order to correct for this deficiency we have

(a) algorithmically solved a related matching problem in which we allow only pairings of two (or more) consecutive bases, say (B_i, B_{i+1}) to, say (B_j, B_{j-1}), (see § 5), and,

(b) introduced the blocks approach.

In this blocks approach one identifies at the outset potential perfect matches between sufficiently long sections of the polynucleotide chain ("blocks"). The original algorithm can then be applied with a large gain in storage time and biological relevance to the blocks rather than to matchings of single bases. In fact, the blocks approach has been used in a different more intuitive approach by two of us to fold the complete ϕX174 virus [9].

The present work also includes an illustration of the application of the simple version of the algorithm to fold the J-gene, the smallest (113 nucleotides) among the nine genes of the ϕX174. Gene J-codes for a small basic protein, whose exact function is still unknown.

Finally we discuss briefly the question of the expected planar matching fraction for a random chain.[1]

2. The model. We represent the secondary structure of long polynucleotide chains by closing the nucleotide chain into a cyclic graph or planar *loop* in which the vertices are indentified with the bases in the chain. Suppose that the loop has size n and that the vertices are numbered consecutively from 1 to n around the loop. Each vertex is labeled by the base it represents, e.g., "A", "C", "G", or "U". The base pairings in the folded form of the nucleotide chain correspond to edges (that is, unordered pairs of vertices of the form $\{i, j\}$) which link nonadjacent vertices with compatible labels, e.g., C-G, in the interior of the loop.

We demand that the folded chain be planar so that the edges in the loop cannot cross each other. No base may be paired twice, so that each vertex belongs to at most one of the interior edges. Thus we are really considering a special matching of the vertices on the loop. With this in mind, we define a planar matching (or *matching* for short) as a set of edges of the loop satisfying the following conditions:

1. Each edge contains vertices whose labels are compatible according to the base pairing rules.
2. No vertex is in more than one edge in the matching.
3. The edges can all be drawn in the interior of the loop without crossing, i.e., $\{g, h\}$ and $\{i, j\}$ with $g<h$ and $i<j$ cannot both belong to the matching if $i<g<j<h$ or $g<i<h<j$. We also exclude edges $\{i, i+1\}(\mathrm{mod}\, n)$ connecting neighboring vertices.

We seek folded shapes for given nucleotide chains in which the pairing interactions minimize the energy, where each possible compatible pairing $\{i, j\}$ is assigned a negative energy, $-E_{i,j}$. That is, we want matchings of maximum weight, where the edges are assigned weights $E_{i,j} \geqq 0$.

First we consider a simplified model in which each vertex is labeled "A" or "B" and the only compatible pairing is A-B, so that every edge in a matching contains one vertex labeled A and one labeled B. Each such edge is assigned weight one. Thus the weight of a matching here means the number of edges it contains, which we call its *size*. We now look for the maximum size of the matchings on a given loop (the maximum matching size: M.M.S. for short) and for the matchings which realize this maximum.

For simple cases with small n, the edges in the maximum matching sets and the M.M.S. can be easily found by inspection; e.g., for the loop with the labels shown (Fig. 1) M.M.S. = 7, and the maximum matchings realizing it are the sets of edges {1, 14},

[1] An example of an earlier interdisciplinary biomathematical effort in a somewhat related area is the Goel–Bremermann and collaboration work on the codon frequency problem. See, e.g., Bremermann and King, *Determination of codon frequencies and sequence structure of polynucleotides*, Lectures on Mathematics in the Life Sciences, no. 5, J. D. Cowan, editor, American Mathematical Society, Providence, RI, 1973.

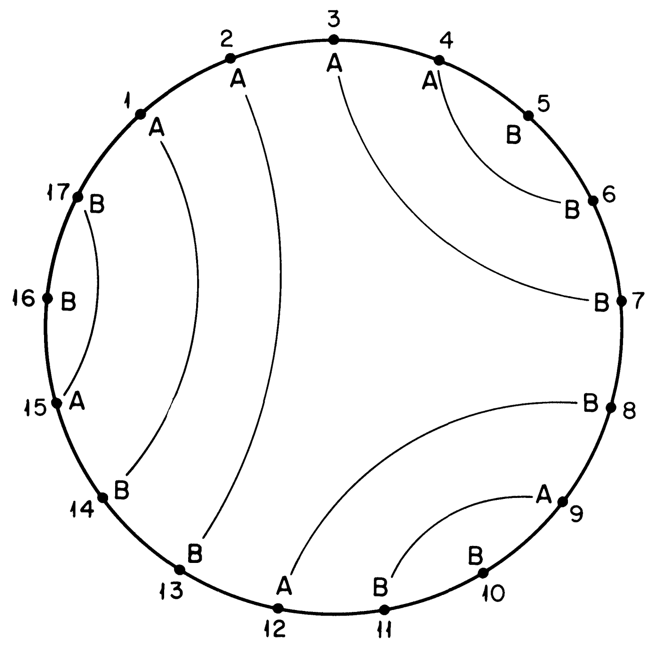

FIG. 1. *An example of a maximum matching on a small chain* ($n = 17$). *Here* M.M.S. = 7.

$\{17, 15\}$, $\{2, 13\}$, $\{3, 7\}$, $\{4, 6\}$, $\{8, 12\}$, $\{11, 9\}$ and $\{1, 14\}$, $\{17, 15\}$, $\{2, 8\}$, $\{3, 7\}$, $\{4, 6\}$, $\{12, 10\}$, $\{13, 9\}$.

However, the main interest is in very large n ($\sim 10^3$–10^4). In the following we provide an efficient algorithm for computing M.M.S. practical for $n \leqq 300$. For larger n some modifications are required, and in particular only approximate M.M.S. and maximum matchings can be found within the given computational limits.

This algorithm can be easily adapted to the more general problem in which the alphabet of labels, the pairing rules, and the edge weights are arbitrary.

3. The algorithm. Given a loop of size n, labeled with A's and B's, we find the M.M.S. and a maximum matching by computing it for short strings, building up to longer strings, and finally to the entire loop. This is similar to the dynamic programming type algorithm of Wagner and Fischer [12] for the longest common subsequence problem.

For $1 \leqq i < j \leqq n$ let $S_{i,j}$ be the string of vertices i, $i+1, \cdots, j$. Assume the maximum matching size M.M.S. (i, j) and at least one maximum matching are known for all strings $S_{i,j}$ of length$<p$, i.e., with $j < i + p$, and $1 \leqq p < n$. Set M.M.S.$(i, j) = 0$ for $j \leqq i + 1$.

We now form the M.M.S.(i, j) and at least one maximum matching for all strings with length p as follows: Add the vertex $j = i + p$ to the string $S_{i,j-1}$. A maximum matching for $S_{i,j}$ might be obtained by joining vertex j to a vertex k with $i \leqq k \leqq j - 2$ and label $(j) \neq$ label (k) (see Fig. 2). Alternatively, j may belong to no edge in the maximum matching for $S_{i,j}$.

Optimizing over all these possibilities, we find M.M.S.(i, j) by

$$\text{(1)} \qquad \text{M.M.S.}(i,j)=\text{MAX}\begin{cases}\text{M.M.S.}(i,j-1);\\ \text{M.M.S.}(i,k-1)+\text{M.M.S.}(k+1,j-1)+1,\\ \quad k\colon i\leqq k\leqq j-2 \text{ and}\\ \quad \text{label}(k)\neq\text{label}(j).\end{cases}$$

We compute through $p = n-2$ and apply

$$\text{(2)} \qquad \text{M.M.S.}=\text{MAX}\begin{cases}\text{M.M.S.}(1,n-1);\\ \text{M.M.S.}(1,k-1)+\text{M.M.S.}(k+1,n-1)+1,\\ \quad k\colon 1<k<n-1 \text{ and}\\ \quad \text{label}(k)\neq\text{label}(n)\end{cases}$$

to obtain the M.M.S. for the whole loop.

Why does this algorithm work? Equation (2) can be justified as follows: Let M be a maximum matching on a given loop with n vertices. Suppose the edge $\{k, n\}$ is in M. So label$(k) \neq$ label(n), i.e., one is "A" and the other is "B", and the edges of M on the string $S_{1,k-1}$ are a matching attaining M.M.S.$(1, k-1)$. (Else these edges could be replaced by a maximum matching on $S_{1,k-1}$, which would increase the size of M, a contradiction.) Similarly, the edges of M on $S_{k+1,n-1}$ form a maximum matching there. Note that the planarity condition 3 implies that no edge in M lies between $S_{1,k-1}$ and $S_{k+1,n-1}$ because such edges would cross $\{k, n\}$. Thus, in this case,

$$\text{M.M.S.} = \text{M.M.S.}(1, k-1) + \text{M.M.S.}(k, n-1) + 1.$$

The other possibility is that the vertex n lies in no edges in M. In this case, M is a maximum matching on $S_{1,n-1}$ and

$$\text{M.M.S.} = \text{M.M.S.}(1, n-1).$$

Hence, M.M.S. is no larger than the right side of (2). Conversely, this value for the M.M.S. is actually attained by matching n to a k attaining this maximum and taking maximum matchings on $S_{1,k-1}$ and $S_{k+1,n-1}$. Equation (1) is justified by similar arguments. Equation (2) differs from (1) only because we do not allow the edge $\{1, n\}$ in a matching. Equations (1) and (2) compute the M.M.S. on the loop from values of the M.M.S. on shorter strings, which were previously computed, so the algorithm works.

To obtain the edges in some maximum matching, we maintain an $n \times n$ array $\mathbf{A}$ where $\mathbf{A}(i, j)$ stores a value of k which attains the maximum in the equation (1) for M.M.S.(i, j). We may store 0 if M.M.S.(i, j) = M.M.S.$(i, j-1)$. We can backtrack after computing the M.M.S. to obtain a maximum matching. We keep a list of string endpoint pairs, S, and a list of edges, E, both initially empty. First we obtain a k which attains the maximum in (2). if $k > 0$, we insert $(1, k-1)$ and $(k+1, n-1)$ into S, and $\{n, k\}$ into E. Otherwise, we insert $(1, n-1)$ into S.

We then take the first element out of S, call it (i, j). If $\mathbf{A}(i, j) = k > 0$, we insert $(i, k-1)$ and $(k+1, j-1)$ into S, and $\{j, k\}$ into E. If $\mathbf{A}(i, j) = 0$, we insert $(i, j-1)$ into S. If $j \leqq i+1$, we simply remove the string from S. When S is empty, E is a maximum matching.

The algorithm requires roughly $n^2/2$ maximizing steps, which each maximize over no more than n numbers. So the algorithm is $O(n^3)$ with storage $O(n^2)$. The algorithm can be easily adapted to more general alphabets and sets of allowed matches, e.g., the RNA problem with letters A, C, G, U, and matches A-U, C-G, G-U.

4. Further comments and improvements.

Continuity arguments can be used to reduce the number of steps in searching the maximum in (1).

Clearly,

$$0 \leqq \text{M.M.S.}(i, j) - \text{M.M.S.}(i, j-1) \leqq 1. \tag{3}$$

Also, shifting the edge $\{k, j\}$ to $\{k+1, j\}$ (or vice versa) can disrupt at most one other edge so that terms from (1)

$$T_k = \text{M.M.S.}(i, k-1) + \text{M.M.S.}(k+1, j-1) + 1$$

for consecutive k's differ at most by one.

Hence, if for some k_0, $i < k_0 < j-1$, $T_{k_0} \leqq \text{M.M.S.}(i, j-1) - h$, i.e., it undershoots the (lower) maximum value by h, then we have to move at least $h+1$ steps to right or left to achieve the maximal T_k. Thus, we can exclude the interval $[k_0 - h, k_0 + h]$ from our search.

This suggests the following modification of our algorithm for faster running time in computing M.M.S.(i, j). Initially set $k = i$. Let $h = \text{M.M.S.}(i, j-1) - T_k + 1 = \text{M.M.S.}(i, j-1) - \text{M.M.S.}(i, k-1) - \text{M.M.S.}(k+1, j-1)$.

If $h > 0$, replace k by $k + h$ and repeat the above computation after first checking if the new value of k exceeds $j-2$, in which case we halt with M.M.S.$(i, j) =$ M.M.S.$(i, j-1)$. If $h = 0$, compare label(j) with label(k). If label $(j) =$ label(k) no edge is allowed between vertices j and k, so we replace k by $k+1$ and repeat the above steps. Finally, if label$(j) \neq$ label(k) we have found a vertex k such that there exists a maximum matching of size M.M.S.$(i, j-1) + 1$ containing edge $\{k, j\}$, so we halt with M.M.S.$(i, j) =$ M.M.S.$(i, j-1) + 1$.

This modification speeds up the algorithm because it skips over many vertices on the string and because it stops when it first finds a vertex k such that there exists a maximum matching on $S_{i,j}$ with edge $\{k, j\}$.

We call a small nested fold with parallel edges that contains no other matches a *flower*. For example, the matches {15, 17}, {1, 14}, and {2, 13} in Fig. 1 form such a flower. The match {6, 8} (A-T) in Fig. 6 is another flower. By the nonadjacency condition on matchings, a flower must contain at least one unmatched vertex (e.g., vertex 16 in Fig. 1 and vertex 7 (A) in Fig. 6). Thus we expect maximum matchings to match long stretches of distant vertices, rather than forming many small flowers, in order to match more vertices. So we would expect h to get fairly large, speeding up the algorithm. In those strings $S_{i,j}$ which have several vertices k which can match j in maximum matchings, the algorithm selects the k farthest from j so that it should produce a maximum matching with larger paired areas which are more interesting to biochemists than matchings with many flowers. In § 5 we modify the problem in order to find matchings with fewer small flowers by not allowing any isolated matches.

In a more general problem in which we have an arbitrary alphabet the possible edges are all assigned weights (e.g., energy), and we seek the matching of maximum *weight* (not necessarily size), the algorithm of § 3 applies directly by adding the weight of the edge $\{k, j\}$ rather than one in (1) and (2). We have to look at every vertex between i and j, so the continuity argument above does not directly apply. However, in specific applications, the assigned energies may very well have some special structure (such as only taking on a limited range of values) which permits an improvement in a similar way.

It is interesting to ask what happens if we relax the restriction that no two adjacent vertices may be matched. In the simple A-B problem the result is clear: if our given loop contains, say j A's and k B's, $j \geqq k$, then we can find a (maximum) matching of size k, that is, we can use up all of the B's. We can find such a matching in only $O(n)$ steps: Start at vertex 1, say it is an A, and check $2, 3, \cdots$, until a B is found. Say it is at i. Then match i and $i-1$. If $i=2$, we start all over at 3; if $i>2$, continue to scan $i+1, i+2, \cdots$, for a B. Always match to the highest scanned unmatched A vertex, e.g., $i+3$ would match $i+2$, and $i+2$ would match $i-2$. The matching we obtain is clearly planar. So it appears that the adjacency condition is what makes the problem difficult: We cannot match locally, independent of the rest of the labeling, and hope to end up with a maximum matching, because the flowers waste too many vertices.

For more general alphabets and edge weights, removing the adjacency condition does not appear to help as much. A simple example comes from our RNA problem with matches A-C, C-G, G-U. Consider a loop of size four, labeled A, C, G, U, in that order. A local approach might match C to G. But this cannot be extended to the maximum matching, which has two matches: A-C, and G-U. However, the M.M.S. algorithm presented in § 3 does work for this general case with the modification that we search up to $k=j-1$ in (1) and use (1) also to compute M.M.S. = M.M.S.$(1, n)$, ignoring (2).

A different modification of the problem that may be useful in different applications comes by relaxing the condition that vertices lie in only one edge in the matching. We allow any number of edges per vertex so long as they do not cross each other, which would violate the planarity condition. Our algorithm can be adapted to this problem by computing M.M.S.(i, j) as the maximum of terms of the form M.M.S.(i, k) + M.M.S.(k, j), $i+1 \leqq k \leqq j-1$. The same idea works if we also allow adjacent matches.

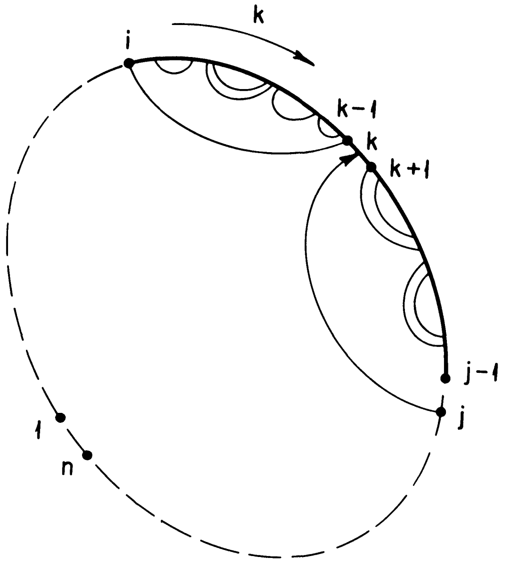

FIG. 2. *The search procedure for optimizing k.*

5. A related problem. In the real problem the energy is lowered not so much by the base, say (A-B) pairing, but rather by the helical stacking of consecutive base pairs. It is clear that a vertex matching configuration realizing the M.M.S. is likely to include also many neighboring $\{i, j\}$, $\{i+1, j-1\}$ edges.

Therefore, a better approximation to the real problem is obtained if we add a fourth requirement to our definition of matchings.

4. No edge lies alone, i.e., there must be an immediately parallel edge. This means that if $\{i, j\}$ is in the matching, then so is $\{i-1, j+1\}$ or $\{i+1, j-1\}$, where we consider the vertices here as numbered mod n.

We present an algorithm to calculate the M.M.S. similar to that of § 3, except that we must now compute matchings on strings of three types. As above, the M.M.S. of each type can be computed from information about shorter strings.

Let M.M.S.(i, j, t) for $1 \leqq i \leqq j \leqq n$ and $t \in \{1, 2, 3\}$ be the M.M.S. of type t on the string of vertices $i, i+1, \cdots, j$, taken from the loop with the same A and B labels, where the types are defined as follows (see Fig. 3):

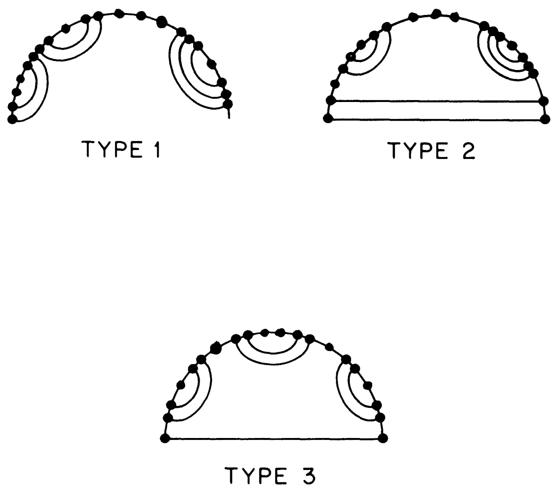

FIG. 3. *The three types of matchings.*

Type 1: Every edge has a neighboring edge (satisfying condition 4) and $\{i, j\}$ is *not* in the matching.

Type 2: Every edge has a neighbor and $\{i, j\}$ is in the matching.

Type 3: $\{i, j\}$ is in the matching and is the only edge without a neighbor.

Initially set M.M.S.$(i, j, t) = 0$ for $j = i$ or $i+1$ and all t. For $t = 2$ or 3 and any i, j, M.M.S.$(i, j, t) = 0$ means no matching of type t exists on the string. As the matching with no edges is type 1, type 1 matchings exist for every string.

Given M.M.S.(i, j, t) for all t and for all i, j, with $i \leqq j < i+p$ for $p > 1$, calculate M.M.S.(i, j, t) for $j = i+p$ as follows:

$$\text{M.M.S.}(i, j, 1) = \text{MAX}\begin{cases} \text{MAX}(\text{M.M.S.}(i, j-1, t)), \\ t = 1, 2; \\ \text{MAX}(\text{M.M.S.}(i, k-1, t) + \text{M.M.S.}(k, j, 2)), \\ t = 1, 2, \\ \quad k\colon i < k < j-3 \text{ and } \text{M.M.S.}(k, j, 2) > 0; \\ 0 \quad \text{otherwise}. \end{cases}$$

$$\text{M.M.S.}(i,j,2)=\begin{cases}\text{MAX}(\text{M.M.S.}(i+1,j-1,t))+1,\\ t=2,3,\\ \quad \text{if M.M.S.}(i+1,j-1,3)>0\\ \quad \text{and label}(i)\neq\text{label}(j);\\ 0 \quad \text{otherwise.}\end{cases}$$

$$\text{M.M.S.}(i,j,3)=\begin{cases}\text{M.M.S.}(i+1,j-1,1)+1,\\ \quad \text{if label}(i)\neq\text{label}(j);\\ 0 \quad \text{otherwise.}\end{cases}$$

Calculate all M.M.S.(i, j, t) through $p = n - 2$. Then find M.M.S.:

$$\text{M.M.S.}=\text{MAX}\begin{cases}\text{MAX M.M.S.}(1,n-1,t),\\ t=1,2;\\ \text{MAX}(\text{M.M.S.}(1,k-1,t)+\text{M.M.S.}(k,n,2)),\\ t=1,2,3,\\ \ k\colon 1<k<n-2 \text{ and M.M.S.}(k,n,2)>0;\\ \text{MAX}(\text{M.M.S.}(1,k-1,t)+\text{M.M.S.}(k,n,3)),\\ t=2,3,\\ \ k\colon 3<k<n-1 \text{ and M.M.S.}(k,n,3)>0\\ \ \text{and M.M.S.}(1,k-1,3)>0.\end{cases}$$

Again we have an $O(n^3)$ algorithm requiring $O(n^2)$ storage. A backtracking procedure will again retrieve a maximum matching.

We now sketch a proof of the validity of the algorithm. Let X be a maximum matching and look at vertex n. If n belongs to no edge in X, then X on the string from 1 to $n-1$ must satisfy all conditions, and hence be of types 1 or 2. Else $\{n, k\}\in X$ for some k: $1<k<n-1$. If $\{n-1, k+1\}\notin X$, then the matching on $k, k+1, \cdots, n$ must be type 3. Edge $\{1, k-1\}$ must be in X to satisfy condition 4. Hence $1, \cdots, n$ is type 2 and $1, \cdots, k-1$ is type 1, 2 or 3. Conversely, maximizing over all of these possibilities must yield the M.M.S. Conditions like "M.M.S.$(k, n, 2)>0$" ensure that a matching of this type exists for this k. Combining matchings on the string in these ways gives rise to a matching on the loop satisfying all of the conditions. Similarly analyzing how we could have obtained matchings on strings of types 1, 2, and 3, we obtain the equations for them. For example, if a matching on $i, \cdots, j$ of type 1 contains edge $\{k, j\}$ then $\{k+1, j-1\}$ must also belong to satisfy condition 4. Hence the matching on $k, \cdots, j$ is type 2. Further, the matching on $i, \cdots, k-1$ must satisfy all conditions, so it is type 1 or 2.

6. The "blocks" approach.

At the cost of slight further complication, we can have algorithms in which we demand any number of neighboring parallel edges.

An alternative approach is to initially locate matching stretches of vertices no shorter than some arbitrary minimal length. These objects will be called *blocks* and the correct computation of their energies, including stacking effects, can make this a more realistic approximation to the biochemical problem.

We note that at this stage of preparing the blocks we have to allow intersecting stretches, i.e., vertices which will simultaneously be assigned to more than one block.

If we want to make the following modification of the algorithm applicable for the choice of minimal energy planar folding by choosing a consistent set of blocks, we will have to separate such intersections into four smaller blocks as indicated in Fig. 4.

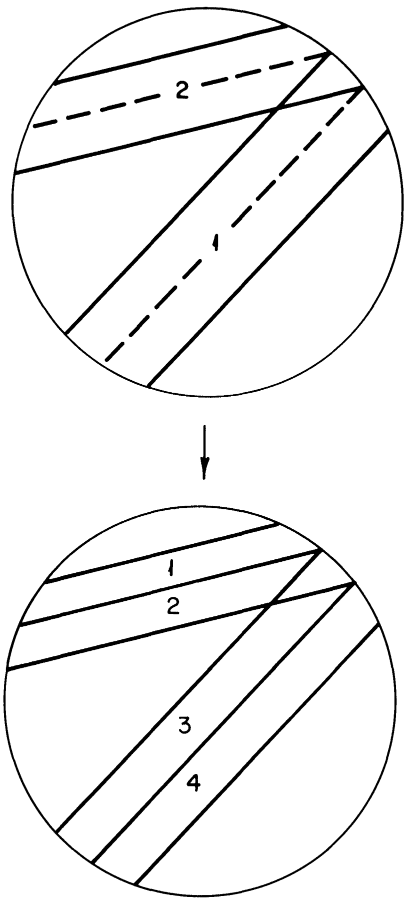

FIG. 4. *A configuration with overlapping stretches and their separation.*

The blocks can then be ordered according to the location of the vertices in the matching stretches in the original string yielding a "contracted" description of the situation (Fig. 5) in terms of many fewer vertices $\bar{n}$ (~ 10 in the example given). Each effective edge is now labeled by the energy $E_{i,j}$ associated with it.

The procedure of selecting the set of nonintersecting edges with the maximal sum $\sum E_{i,j}$ is done using an algorithm exactly like the first one with the following modifications:

(a) Since vertex j is already prefixed to at most one (or some small number, independent of $\bar{n}$, if we start from a configuration such as Fig. 4) of the k's $i \leqq k < j$, no long k search has to be made, and the algorithm will be only $O(\bar{n}^2)$.

(b) Instead of adding one as in the right-hand side of (1), if some $1 \leqq k \leqq j-1$ is chosen, we have to add $E_{k,j}$.

An advantage of this "coarse grained" block description is that storage requirements which are $O(n^2)$ are substantially reduced to $O(\bar{n}^2)$. In the real problem, n may be 10^4, while $\bar{n}$ may be 300.

Clearly an important element in this approach is an efficient method of block construction. We will not go into its details here, but rather refer to the work by two of us [9].

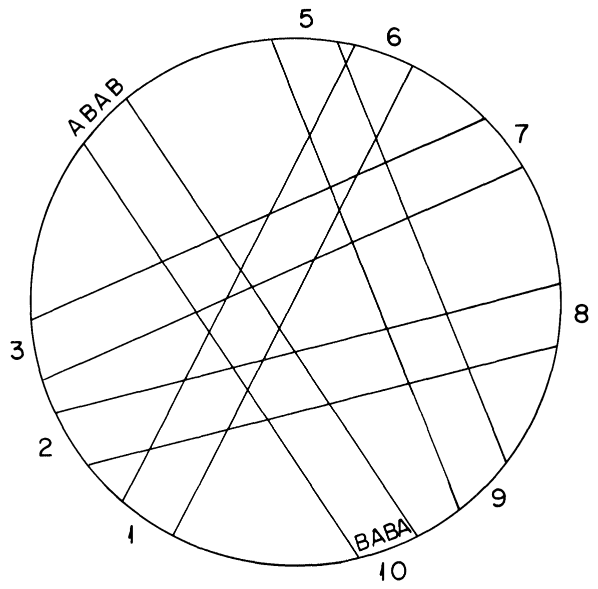

FIG. 5. *The contracted representation of matching blocks. One such block is shown: the* ABAB *string, contracted to vertex* 4, *is matched to the* BABA *string at vertex* 10.

In this work the block approach was used extensively (without establishing, however, a firm algorithmic background) to fold long m-RNA molecules. The method used there for actual block formation was very simple and fast.

More sophisticated approaches for finding given patterns in a string of symbols exist in the literature [14] and hopefully can be incorporated in future developments.

An alternative to the above "floating block" formation is a conceptually much simpler "fixed block formation". In that approach, which is also studied at present, the original long string is divided into, say, m fixed substrings of length $\simeq n/m$ each. One then finds the best matching of each pair of substrings obtaining an $m \times m$ matrix with entries the resulting lowest possible energies (that is, maximum matching size) between the substrings. This contracted version serves as a second stage in which a maximal planar subset of the matched string is chosen. In either stage the original algorithm or a simple modification of it can be used.

7. Programming the algorithm and some preliminary results.

Given the simple algorithm of § 3, it is very straightforward to program it. In fact, the FORTRAN program that we wrote, which incorporated the jump-procedure of § 4, consists of only ~90 cards.

We ran it for different small sections of the recently sequenced (by Sanger and his group [16]) ϕX174 single stranded DNA virus, which in total has 5470 nucleotides. The shape obtained this way for folding separately the J-gene of the ϕX174 with 113 nucleotides is presented in Fig. 6. It took only 1.3 seconds to generate this folded form on the IBM 360/91 computer.

While by construction, this shape is guaranteed to have the maximal number of allowed pairings, it is somewhat deficient from the biological standpoint since quite a few of the bonds are single.

To have a better idea about the systematics of the time involved, we ran the same program also on segments 30 nucleotides long (0.15 sec) and 200 nucleotides long (6.4 sec). While these data are insufficient for an accurate extrapolation of the time

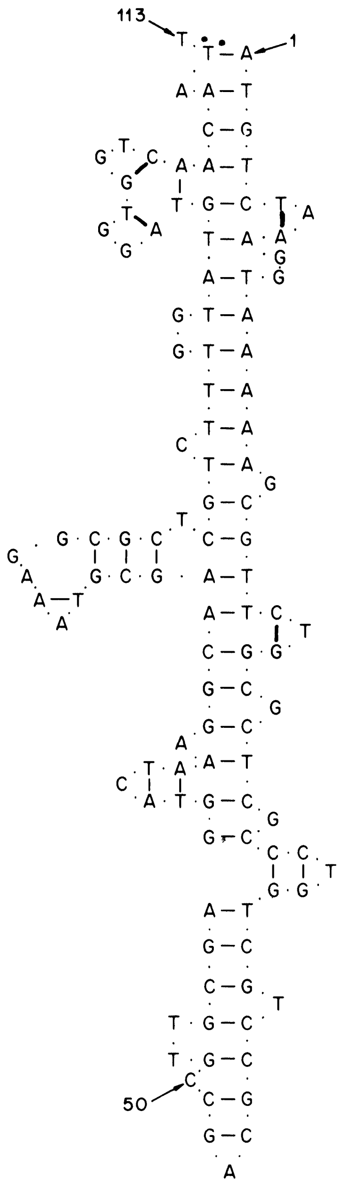

FIG. 6. *A maximum matching for the* J*-gene of* ϕX174, *drawn in its folded form. Here* $n = 113$, *the allowed matches are* A-T and C-G, and M.M.S. = 42.

estimate for very large n (say $n \sim 1{,}000$–$2{,}000$), they strongly indicate that the estimate of $O(n^3)$ asymptotic growth is far too conservative. The jump procedure does in fact shorten significantly the execution time. In any case, the algorithm would take ~ 1 minute for $n = 500$, and 5–10 minutes for $n = 1{,}000$—very reasonable time requirements. This should be contrasted with other folding procedures which exist in the literature [17], which can be applied only to sections shorter than 200 nucleotides and which are ~ 10 times slower.

At present we are trying to surmount the two remaining obstacles in the way of massive applications of the algorithm to very long chains—the need to incorporate stacking effects and the strong storage requirements. It appears that both objects can be obtained by adapting the joint block approach and algorithm described in § 6 above.

We do not have as yet completely satisfactory codes for that approach which will be discussed further in forthcoming works.

8. Expected matching size. We now consider the problem of finding the expected maximum matching size. The matchings here are simple, satisfying conditions 1, 2, and 3 of § 2. The vertices are each labeled either A or B with a probability of $\frac{1}{2}$ each. Then $E(n)$ is defined to be twice the expected size of a maximum matching of such a labeling of the loop with n vertices. That is, $E(n)$ is the mean number of vertices matched in a maximum matching among the 2^n labelings of the loop. Thus $E(n)/n$ is the average proportion of the vertices which are matched. If $E(n)/n$ were known one would have an idea how successful are algorithms which find large matchings that are not necessarily maximum. Such algorithms require less time and storage than the M.M.S. algorithm. As n is often very large (say 10^2–10^4), information on the behavior of $E(n)/n$ as $n \to \infty$ is useful.

Chvátal and Sankoff [10] studied the expected size of the longest common subsequence (LCS) of two sequences in k-symbols of length n. Just as in their problem, it is true for loop matchings that $E(n)$ is superadditive, that is, for $m, n > 0$, $E(m+n) \geqq E(m) + E(n)$, from which it follows by Fekete's theorem [11] that

$$\lim_{n\to\infty} \frac{E(n)}{n} = \sup_n \frac{E(n)}{n}.$$

Thus $E(n)/n$ has a limit as $n \to \infty$—call it $\bar{c}$—which it never exceeds. Clearly $\bar{c} \leqq 1$; no better upper bound has been found yet. For a lower bound we may split the loop into two strings of length $n/2$ and take the LCS as a matching. Thus $\bar{c} \geqq \frac{8}{11}$, using the lower bound obtained by Chvátal and Sankoff [10] for LCS with $k = 2$. We have devised a simple algorithm (time and storage $O(n)$) which finds matchings of expected size approaching $(.8)n$ as $n \to \infty$, so that $\bar{c} \geqq .8$, which is the best known lower bound on $\bar{c}$. The analysis of this algorithm uses Markov chains, in one approach, and difference equations in another approach. (See [15] for details.)

Similarly, for the RNA problem (A, C, G, U) the mean proportion of vertices matched has a limit; call it $\bar{d}$. Applying the lower bound from [10] for LCS with $k = 4$, we obtain $\bar{d} \geqq .451$. As above the simple algorithm improves the lower bound, to $\frac{1}{2}$ for $\bar{d}$. However, we required no G-U bonds to achieve these lower bounds, so $\bar{d}$ should be considerably larger than $\frac{1}{2}$.

9. Areas requiring further study. Given the present algorithm we can apply it to the real m-RNA data and also to a randomized string with the same length and the same base proportions. If we find that the real data tend to yield consistently larger maximal matching sizes, then it would mean that favoring folding potentialities is one of the considerations which determines the choice of coding in the higher degenerate genetic code.

In the present approach the only constraints of the folding process were the base pairing rules and the assumption of planarity. There is some experimental information which was not incorporated into the present algorithm and which in principle could considerably shorten the search time. This information involves cleavage data, i.e., the facility with which certain enzymes bisect the bond between two consecutive bases at i and $i+1$. If in the folded form the two specific bases $i, i+1$ are paired against say, $j-1, j$, then the original $(i, i+1)$ and $(j, j-1)$ bonds get strengthened. It may turn out eventually that the cleavage data will conflict with the planarity rule. An open and interesting question is to find an algorithmic procedure which will optimally incorporate the cleavage data and planarity requirements.

We would certainly like more information about expected matching sizes, particularly $\bar{c}$ and $\bar{d}$. From a mathematical standpoint it would be interesting to learn more about labeling with larger alphabets, arbitrarily assigned probabilities (i.e., a vertex has label A_i with probability p_i), and arbitrary sets of allowed matchings pairs (generalizing the A-U, C-G, G-U problem).

The computational complexity of these problems is still open. For the related problem of finding the longest common subsequence of two sequences over a finite alphabet of lengths m and n, respectively, there is an $O(mm/\log(\max(m, n)))$ algorithm [13]. It may be that a similar $O(n^3/\log n)$ algorithm can be found for the A-B loop matching problem.

We have briefly indicated various generalizations continuing the present work. We hope that they will lead to more realistic and feasible approaches to the rather difficult combinatorial problem of RNA folding and to new insights both in biochemistry and in algorithmic approaches to combinatorial problems.

Acknowledgments. Two of us (R.N. and G.P.) would like to thank C.C.I.S., Hill Center, at Rutgers University for computer facilities, and members of Biochemistry Group, in particular, Gary Gaber, for their interest. We also express our thanks to Anthony Duncan for helpful discussion.

REFERENCES

[1] J. D. ROBERTUS, J. E. LADNER, J. T. FINCH, D. RHODES, R. S. BROWN, B. F. C. CLARK AND A. KLUG, *Structure of yeast phenylalanine tRNA at* 3 *A resolution*, Nature, 250 (1974), pp. 546–551.

[2] S. H. KIM, G. J. QUIGLEY, F. L. SUDDATH, A. MCPHERSON, D. SNEDEN, J. J. KIM, J. WEINZIERL, P. BLATTMANN AND A. RICH, *The three-dimensional structure of yeast phenylalanine transfer RNA: Shape of the molecule at* 5.5 *A resolution*, Proc. Nat. Acad. Sci. USA, 69 (1972), pp. 3746–3750.

[3] F. L. SUDDATH, G. J. QUIGLEY, A. MCPHERSON, D. SNEDEN, J. J. KIM, S. H. KIM AND A. RICH, *Three-dimensional structure of yeast phenylalanine transfer ENA at* 3.0 *A resolution*, Nature, 248 (1974), pp. 22–24.

[4] J. GRALLA, J. A. STEITZ AND D. M. CROTHERS, *Direct physical evidence for secondary structure in an isolated fragment of R*17 *bacteriophage mRNA*, Ibid., 248 (1974), pp.·204–208.

[5] G. PIECZENIK, *Genetic-code constraints on amino acids and nucleotide sequences*, Ph.D. thesis, New York University, New York City, 1972. *Note*: This dissertation is listed as 1973 in Dissertation Abstracts.

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

[7] J. GRALLA AND C. DELISI, *mRNA is expected to form stable secondary structures*, Nature, 248 (1974), pp. 330–332.

[8] W. FIERS, R. CONTRERAS, F. DUERINCK, G. HAEGEMAN, D. ISERENTANT, J. MERREGAERT, W. MIN JOU, F. MOLEMANS, A. RAEYMAEKERS, A. VAN DEN BERGHE, G. VOLCKAERT AND M. YSEBAIERT, *Complete nucleotide sequence of bacteriophage MS*2 *RNA: Primary and secondary structure of the replicase gene*, Ibid., 260 (1976), pp. 500–507.

[9] R. NUSSINOV AND G. PIECZENIK, *Folding m-RNA molecules with computer codes*, Rutgers University, manuscript in preparation 1976.

[10] V. CHVÁTAL AND D. SANKOFF, *Longest common subsequences of two random sequences*, J. Appl. Probability, 12 (1975), pp. 306–315.

[11] M. FEKETE, *Verteilung der Wurzeln bei gewissen algebraischen Gleichungen mit ganzzahlingen Koeffizienten*, Math. Z., 17 (1923), pp. 228–249.

[12] R. A. WAGNER AND M. J. FISCHER, *The sıring to string correction problem*, J. Assoc. Comput. Mach., 21 (1974), pp. 168–173.

[13] W. J. MASEK, *A fast algorithm for the string editing problem and decision graph complexity*, M.S. thesis, Mass. Inst. of Tech., Cambridge, 1976.

[14] D. E. KNUTH, J. H. MORRIS, JR. AND V. R. PRATT, *Fast pattern matching in strings*, Computer Science Department Rep. STAN-CS-74-440, Stanford Univ., Stanford, CA, Aug. 1974.

[15] J. R. GRIGGS, *Symmetric chain orders, Sperner theorems, and loop matchings*, Ph.D. thesis, Mass. Inst. of Tech., Cambridge, 1977.

[16] F. SANGER, G. M. AIR, B. G. BARREL, N. L. BROWN, A. R. COULSON, J. C. FIDDES, C. A. HUTCHISON, III, P. N. SLOCOMBE AND M. SMITH, *Nucleotide sequence of bacteriophage $\phi X174$ DNA*, Nature, 265 (1977), pp. 687–695.

[17] J. M. PIPPAS AND J. E. MCMAHON, *Method for predicting RNA secondary structure*, Proc. Nat. Acad. Sci. U.S.A., 72 (1975), pp. 2017–2021.