CABIOS

Vol.6. no.4 1990
Pages 309 – 318

Comparing multiple RNA secondary structures using tree comparisons

Bruce A.Shapiro and Kaizhong Zhang[1]

Abstract

In a previous paper, an algorithm was presented for analyzing multiple RNA secondary structures utilizing a multiple string alignment algorithm. In this paper we present another approach to the problem of comparing many secondary structures by utilizing a very efficient tree-matching algorithm that will compare two trees in $O(|T_1| \times |T_2| \times L_1 \times L_2)$ in the worst case and very close to $O(|T_1| \times |T_2|)$ for average trees representing secondary structures. The result of the pairwise comparison algorithm is then used with a cluster algorithm to produce a multiple structure clustering which can be displayed in a taxonomy tree to show related structures.

Introduction

In a previous paper (Shapiro, 1988), an algorithm was presented that utilizes a multiple sequence homology algorithm (Sobel and Martinez, 1986) to compare many RNA secondary structures. The object of such analysis is to be able to determine how structural similarity is related to functional similarity. This approach is analogous to the philosophy behind sequence homology programs (Needleman and Wunsch, 1970; Sankoff, 1972; Sellers, 1980; Dumas and Ninio, 1982; Goad and Kanehisa, 1982; Nussinov, 1983; Wilbur and Lipman, 1983; Fickett, 1984) where portions of sequence that show homology are assumed to behave in a similar way in a functional sense. Recent biological experiments (Hall *et al.*, 1982; Altuvia *et al.*, 1987; Berkhout *et al.*, 1987; Deckman and Draper, 1987; Thomas and Nomura, 1987; Tuerk *et al.*, 1988) have indicated how in several instances structural motifs appear to have similar functionality. It is also becoming evident that certain sequence motifs appear to be reproducible in structures that are in themselves reproducible, e.g. the recent paper by Tuerk *et al.* (1988) that seems to indicate that the CUUCGG motif appears quite often in structural hairpins that apparently occur in termination sites. The presumption, of course, is that structures that are functionally similar will have similar structures. This concept of determining similarity may also be useful for structure prediction of phylogenetically related structures.

Thus, the ability to compare secondary structures and substructures can elucidate potentially interesting sites and structures (Konings and Hogeweg, 1989; Margalit *et al.*, 1989). The methodology discussed in this paper has the ability to examine structural motifs at various levels because of a very general cost function. It is possible, for example, to determine high-level structural similarity in a manner similar to that discussed in Shapiro (1988). That is, one may determine that structures are similar if they have a similar topological arrangement of loops. This does not account for loop sizes or helical stem sizes. At another level, it is possible to incorporate loop and stem sizes so that the cost function will distinguish between two structures that may have the same looping structure but may differ in the actual sizes of the loops or helical stems. Obviously, this level is able to pick up more subtle differences among the structures than the previous level. At another level it is possible to include loop component sizes. That is, similarity is determined by not just the general morphology but also by the sizes of the two parts of an internal loop or the sizes of the multiple parts of a multibranch loop. At yet another level, it would be possible to include actual sequence data. This would distinguish for example, between structures that have the same shape and size but may differ in the nucleotides that comprise a given substructure. This would determine whether, for example, a compensatory base change has conserved the structure. The general cost function could also contain information such as energy of loops or regions or any other properties that one may desire.

RNA secondary structure and trees

If one examines a typical secondary structure such as the one depicted in Figure 1(a) it becomes apparent that such a structure may be represented as a tree (see Figure 1b). One may represent the helical stems and loops as nodes in such a tree. In Shapiro (1988) the stems were not considered as nodes. Only the loops were represented as nodes. This permitted a higher level of abstraction but also reduced the amount of information available for comparisons. In the current representation, the helical stems may be included as nodes so that information such as stem and loop size may be included in the cost function. The tree shown is considered an ordered tree where the ordering is imposed based upon the 5′ to 3′ nature of the molecule. That is, one may visit the nodes in a very definite order that has implications on the way the tree may be represented. In this paper a tree is represented by a fully parenthesized notation where the root

Image Processing Section, Laboratory of Mathematical Biology, Division of Cancer Biology and Diagnosis, and [1]Advanced Scientific Computer Laboratory Program Resources, Inc., National Cancer Institute, Frederick Cancer Research Facility, National Institutes of Health, Frederick, MD 21701, USA

of every subtree precedes all the nodes contained in its subtree. Thus, the tree depicted in Figure 1(b) may be represented in the fully parenthesized form as

(N(R(I(R(M(R(B(R(M(R(H)) (R(H)))))) (R(H)))))))

where hairpin nodes are represented by H, internal loops are represented by I, bulge loops are represented by B, multibranch loops are represented by M, helical stem regions are represented by R (shown here as connecting arcs) and N is a special node that is used to make sure the tree is connected.

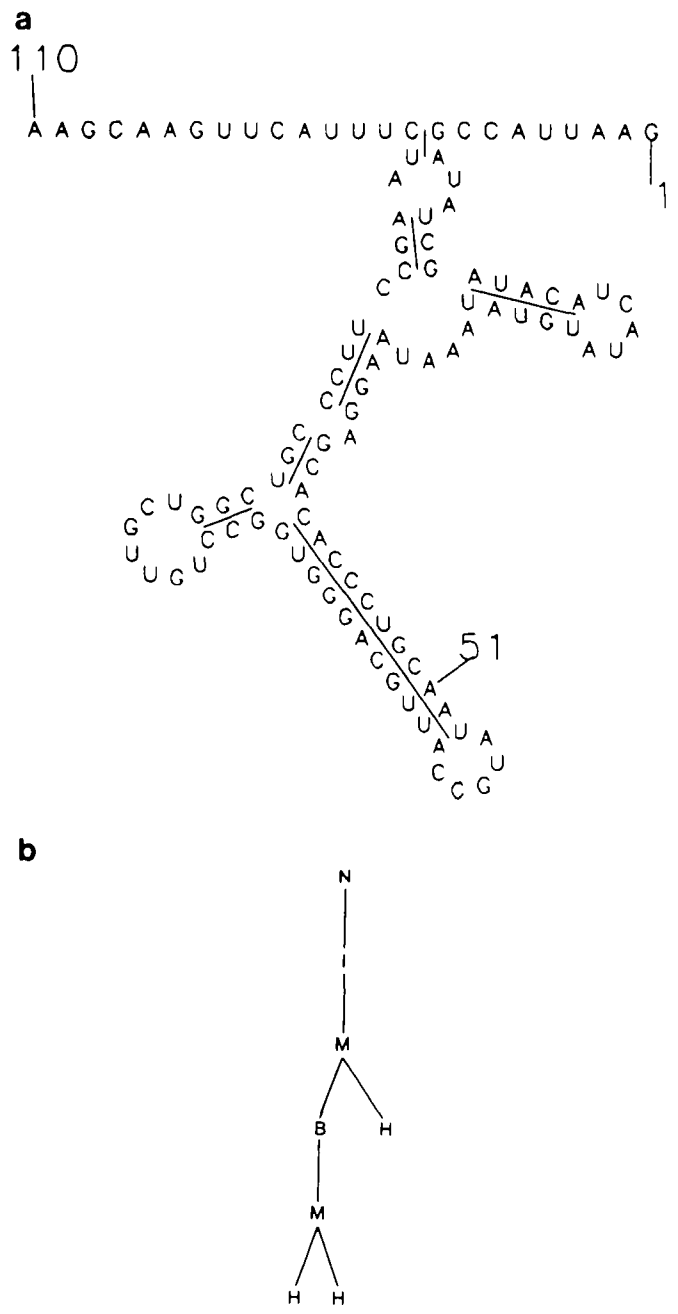

Fig. 1. Illustration of a typical RNA secondary structure and tree representation. (a) normal polygonal representation of the structure; (b) tree representation of the structure.

Following the tree representation presented above is a list of properties for each node. This property list may be empty if one is interested in only the general loose topology of the conformation. The property list is depicted in preorder. That is, the root node is depicted before any of its subtrees and the subtrees are traversed in a left to right order. Thus, the preorder traversal of the parenthesized expression shown above would be the same as the left to right order of the nodes shown in the expression above except the parentheses would be removed. Thus, the properties are represented in such an order with their property name associated with its value. For example, a property list for total size comparison for the structure shown in Figure 1 will look as follows,

Nsize 2; Rsize 2; Isize 3; Rsize 3; Msize 5; Rsize 4; Bsize 1; Rsize 3; Msize 0; Rsize 3; Hsize 7; Rsize 11; Hsize 5; Rsize 5; Hsize 5;

The property list shown only contains the size property per node. It is possible as indicated above that more information may be provided per node, e.g. sizes of loop components, sequence information and energy. Each node is delimited by a semicolon. Input to the algorithm generally consists of a file containing a series of parenthesized tree representations followed by the property lists associated with each tree. The property lists may be empty if size information is not of interest. The information illustrated above is all obtained from a standard region table used for representing the secondary structure of a molecule. The region table for the above example (shown in Table I) consists of a quadruplet for each helical stem in the structure. The components of the quadruplets are the 5′ start position of the region, its 3′ ending position, the region's size and its energy. Table I contains the information required to construct the drawing shown in Figure 1(a) (Shapiro *et al.*, 1984) and the tree representations above.

Tree comparison algorithm

Definitions and background

The algorithm described in this section (Zhang and Shasha, 1990) is a generalization of the algorithms used for determining

Table I. Region table used to generate the secondary structure drawing in Figure 1(a) and the tree in Figure 1(b)

Region no.	5′ start	3′ stop	Size	Energy
1	9	187	2	−2.3
2	13	184	3	−4.3
3	16	121	5	−5.9
4	126	180	4	−5.5
5	131	176	3	−5 2
6	134	160	11	−20 2
7	161	173	3	−6.3

the edit distances between sequences. Three basic edit operations are used to determine the distance between two trees. They are relabel, delete and insert a node. That is, the object of the algorithm is to determine that sequence of tree edit operations that will change a tree T_1 into a tree T_2 with minimum cost. Relabeling a node means changing the label on node n. Deleting a node means making the children of node n become the children of the parent of n and then removing n. Inserting is the complement of delete. This means that inserting n as the child of n' will make n the parent of all the nodes in one of the subtrees of n' including the possibility of an empty subtree. Figure 2 illustrates these editing operations.

An edit operation (Tai, 1979; Zhang and Shasha, 1990) may be represented as a pair $(a,b) \neq (\lambda,\lambda)$, sometimes written $a \rightarrow b$. $a \rightarrow b$ is called a relabeling operation if $a \neq \lambda$ and $b \neq \lambda$; a delete operation if $b = \lambda$; and an insert operation if $a = \lambda$. We will assume a cost function defining the cost of $a \rightarrow b$, denoted by $\gamma(a \rightarrow b)$, as c if $a \neq b$ and 0 otherwise. Let S be a sequence $s_1,\ldots,s_k$ of edit operations that change tree T_1 into tree T_2. T_1 may be changed to T_2 by considering a sequence of trees $A_0,\ldots,A_k$ such that $A = T_1$, $T_2 = A_k$, and $A_{i-1} \rightarrow A_i$ via s_i for $1 \leq i \leq k$. Thus, the cost of a sequence is simply the sum of the cost function applied to each element in the sequence. The distance between T_1 and T_2 is simply the minimum cost sequence taking T_1 to T_2.

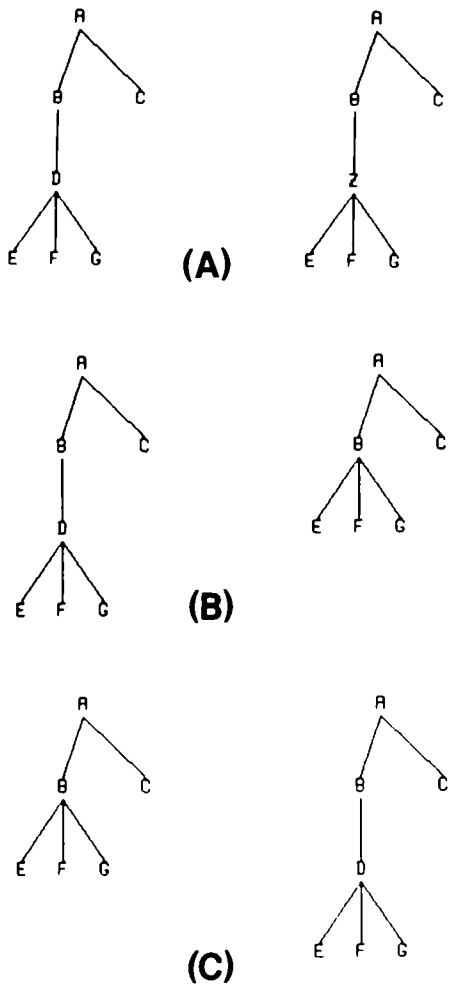

Fig. 2. Illustration of the three tree-editing functions. (a) relabeling, (b) deletion and (c) insertion.

The editing operations described above correspond to a mapping which is a graphical specification showing what edit operations apply to each node in the two trees (or in a more general case two ordered forests). The mapping shown in Figure 3 shows a way to transform T_1 to T_2. It corresponds to the two edit operations delete node with label f ($f \rightarrow \lambda$), followed by insert node with label f ($\lambda \rightarrow f$).

Formally a mapping from T_1 to T_2 is a triple (M,T_1,T_2), where M is any set of integers (i,j) satisfying the following conditions:

1. $1 \leq i \leq |T_1|$, $1 \leq j \leq |T_2|$, where $|\cdot|$ represents the number of nodes in the indicated tree.
2. For any pair of (i_1,j_1) and (i_2,j_2) in M:
   (a) (one-to-one) $i_1 = i_2$ iff $j_1 = j_2$;
   (b) (ancestor) $T_1[i_1]$ is an ancestor of $T_1[i_2]$ iff $T_2[j_1]$ is an ancestor of $T_2[j_2]$;
   (c) (sibling) $T_1[i_1]$ is to the left of $T_1[i_2]$ iff $T_2[j_2]$ is to the left of $T_2[j_2]$.

The notation M instead of (M,T_1T_2) will be used. The cost of M, denoted by $\gamma(M)$, is the cost of nodes to be inserted (i.e. those in T_2 that are not touched by a mapping line) plus the cost of the nodes to be deleted (i.e. those in T_1 not touched by a mapping line) plus the cost of nodes to be relabeled (i.e. those pairs of nodes related by a mapping line with differing labels).

Mappings can be composed. Let M_1 be a mapping from T_1 to T_2 and let M_2 be a mapping from T_2 to T_3. A composed mapping of M_1 and M_2 may be defined as $M_1 \bullet M_2 = \{(i,j) | \exists k \ni (i,k) \in M_1 \text{ and } (k,j) \in M_2\}$. It can be proved (not shown here) that a cost function applied to a composed mapping

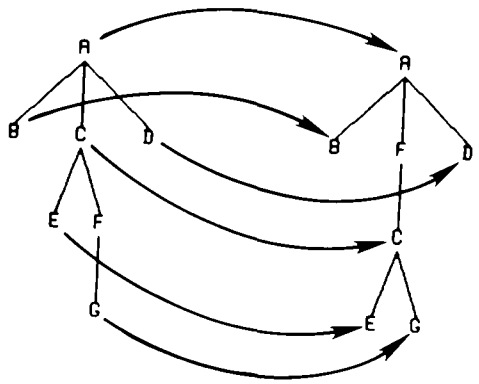

Fig. 3. Illustration of mapping tree T_1 into tree T_2. This mapping corresponds to deleting the node labeling 'f' followed by insertion of this node elsewhere.

satisfies metric properties. It can also be proved (not shown here) that given S, a sequence of edit operations from T_1 to T_2, there exists a mapping M from T_1 to T_2 such that $\gamma(M) \leq \gamma(S)$ and conversely, for any mapping M, there exists a sequence of editing operations such that $\gamma(S) = \gamma(M)$.

Notation

In the algorithm to be discussed a left-to-right postorder traversal of the trees being compared will be used (Knuth, 1973). Some notation will now be defined to facilitate the understanding of the algorithm. It is assumed that the nodes in the postorder traversal are numbered from 1 to n where 1 represents the first node in the traversal order and n represents the last. Thus, let $T[i]$ be the ith node in the tree according to the left-to-right postorder numbering. $l(i)$ is the number of the leftmost leaf descendant of the subtree rooted at $T[i]$. When $T[i]$ is a leaf, $l(i) = i$. $T[i...j]$ is an ordered subforest of T induced by the nodes numbered i to j inclusive. If $i > j$, then $T[i...j]$ is the empty set. The distance between $T_1[i'...i]$ and $T_2[j'...j]$ is denoted by $fdist(T_1[i'...i],T_2[j'...j])$ or $fdist(i'...i,j'...j)$ if the context is clear. The distance between the subtree rooted at i and the subtree rooted at j will be denoted by $tdist(i,j)$.

The algorithm

To compute the edit distance between two trees T_1 and T_2 it suffices to compute $fdist(l(i)...i_1,l(j)...j_1)$ for $l(i) \leq i_1 \leq i$ and $l(j) \leq j_1 \leq j$ where $1 \leq i \leq |T_1|$ and $1 \leq j \leq |T_2|$. This distance may be recusively computed as the minimum of the following three cases for a given (i,j) pair:

1. $T_1[i_1]$ is not touched by a line in M. So,

$$fdist(l(i)...i_1,l(j)...j_1) = fdist(l(i)...i_1-1,l(j)...j_1) + \gamma(T_1[i_1]\rightarrow\gamma)$$

2. $T_2[i_1]$ is not touched by a line in M. So,

$$fdist(l(i)...i_1,l(j)...j_1) = fdist(l(i)...i_1,l(j)...j_1-1) + \gamma(\lambda\rightarrow T_2[j_1]).$$

3. $T_1[i_1]$ and $T_2[j_1]$ are touched by lines in M. By the ancestor and sibling conditions on mappings (i_1,j_1) must be in M. By the ancestor condition on mapping, any node in the subtree rooted at $T_1[i_1]$ can only be touched by a node in the subtree rooted at $T_2[j_1]$. Hence,

$$fdist(l(i)...i_1,l(j)...j_1) = fdist(l(i)...l(i_1)-1,l(j)...l(j_1)-1) + fdist(l(i_1)...i_1-1,l(j_1)...j_1-1])+\gamma(T_1[i_1]\rightarrow T_2[j_1])$$

Since these three cases express all the possible mappings yielding $fdist(i_1,j_1)$, the minimum edit distance is the minimum of these three costs.

When either $l(i_1) \neq$ leftmost child of $T_1[i]$ or $l(j_1) \neq$ leftmost child of $T_2[j]$ [i.e. $l(i_1) \neq l(i)$ or $l(j_1) \neq l(j)$] the equation $fdist(l(i)...i_1,l(j)...j_1) = fdist(l(i)...l(i_1)-1,l(j)...l(j_1)-1) + tdist(i_1,j_1)$ can be used in place of the third case. This is true because $tdist(i_1,j_1) \leq fdist(l(i_1)...i_1-1,l(j_1)..j_1-1) + \gamma(T_1[i_1] \rightarrow T_2[j_1])$. It is also known that $fdist(l(i)...i_1,l(j)...j_1) \leq fdist(l(i)...l(i_1)-1,l(j)...l(j_1)-1) + tdist(i_1,j_1)$. Strict inequality occurs when the minimum distance is not due to the third case.

The algorithm may be further improved by checking for the current trees being considered in the algorithm, rooted at i and j, whether $l(i_1) = l(i)$ and $l(j_1) = l(j)$ as i_1 and j_1 run through the subtrees of i and j. If this is so the third case may be further simplified to $fdist(l(i)...i_1-1,l(j)..j_1-1) + \gamma(T_1[i_1]\rightarrow T_2[j_1])$ by just substituting $l(i_1)$ for $l(i)$. Two sets $KEYROOTS(T_1)$ and $KEYROOTS(T_2)$ are computed, where $KEYROOTS$ is defined as $KEYROOTS(T) = \{k|$ no $k'>k$ such that $l(k) = l(k')\}$. The computation of these sets limits the computation of $tdist$ to the roots of all subtrees that have left siblings and to the root of the trees. The subtree roots that are not in these sets will be captured within the computation of the roots that are within the sets. This last step permits the comparison of two trees that are linear to be performed in a time proportional to a standard

Table II.

	N	I	B	H	M	R	Null
N	0	∞	∞	∞	∞	∞	∞
I		0	3	8	8	∞	5
B			0	8	8	∞	5
H				0	8	∞	100
M					0	∞	75
R						0	5
Null							0

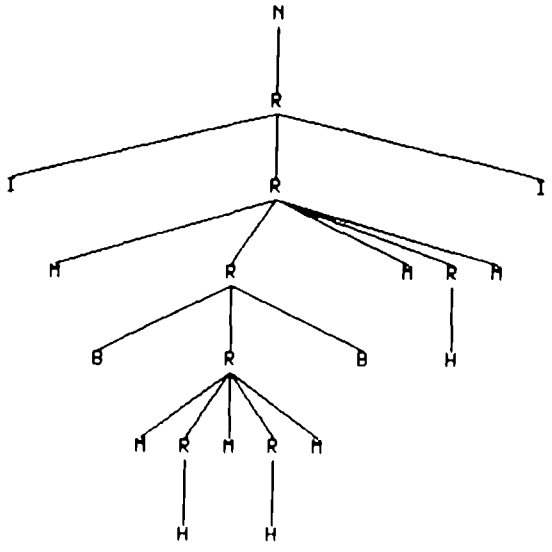

Fig. 4. Example of how one might represent individual loop components in a tree representation.

string comparison algorithm. It can be shown that, in the general case, the time required to compute the comparison of two trees is $O(|T_1| \times |T_2| \times L_1 \times L_2)$ where L_1 and L_2 are the depths of trees T_1 and T_2. Again, if the trees are linear then the algorithm basically runs in $O(|T_1| \times |T_2|)$ time. The fewer the number of nodes that have siblings, the closer to this running time one gets.

The algorithm below fills a two-dimensional array called $TDIST(i,j)$ where $1 \leq i \leq |T_1|$ and $1 \leq j \leq |T_2|$ which represents the distance between all subtrees in T_1 rooted at i and all subtrees in T_2 rooted at j. To facilitate the speed of the algorithm $l(k)$ is precomputed for each tree (see above). Also, the two sets $KEYROOTS(T_1)$ and $KEYROOTS(T_2)$ are precomputed.

The algorithm then proceeds as follows:

```
/*Precompute l(k), KEYROOTS(T1) and KEYROOTS(T2)*/
/*main loop*/

for i' = 1 to |KEYROOTS(T1)|
  for j' = 1 to |KEYROOTS(T2)|
    i = KEYROOTS(T1)[i']
    j = KEYROOTS(T2)[j']
      Compute treedist(i,j)

  Procedure treedist(i,j)

  FDIST(0,0) = 0

  for i1 = l(i) to i
    FDIST(T1[l(i)...i1],0) = FDIST(T1[l(i)...i1-1,0])
    + γ(T1[i1]→λ)

  for j1 = l(j) to j
    FDIST(0,T2[l(j)...j1]) = FDIST(0,T2[l(j)...j1-1])
    + γ(T2[j1]→λ)

  for i1 = l(i) to i
    for j1 = l(j) to j
      if l(i1) = l(i) and l(j1) = l(j) then
        FDIST(T1[l(i)...i1],T2[l(j)...j1]) = min{
          FDIST(T1[l(i)...i1-1],T2[l(j)...j1])
          + γ(T1[i1]→λ),
          FDIST(T1[l(i)...i1],T2[l(j)...j1-1])
          + γ(λ→T1[j1]),
          FDIST(T1[l(i)...i1-1],T2[l(j)...j1-1)
          + γ(T1[i1]→T1[j1])}
        /*store in array permanently*/
        TDIST(i1,j1) = FDIST(T1[l(i)...i1],T2[l(j)...j1])
      else
        FDIST(T1[l(i)...i1],T2[l(j)...j1]) = min{
          FDIST(T1[l(i)...i1-1],T2[l(j)...j1]) +
          γ(T1[i1]→λ),
          FDIST(T1[l(i)...i1],T2[l(j)...j1-1]) +
          γ(λ→T1[j1]),
          FDIST(T1[l(i)...l(i1)-1],T2[l(j)...l(j1)-1]
          + TDIST(i1,j1)}
```

The cost function

Currently, the cost function used for comparing secondary structures is based upon the idea that the structures can be viewed as levels of abstraction. The cost function varies depending upon the amount of information that is included within the nodes of the trees. The simplest one just compares morphology, in which case the only node types that are being compared are N, M, I, B and H. Table II indicates the current heuristic costs that are used when comparing structures. It should be emphasized that it is fairly easy to alter these costs.

If an insertion or deletion edit is performed, the cost assigned is shown under the 'Null' column. The insertion or deletion of a hairpin is highly penalized since the morphologic difference produced by such an operation is quite substantial. The insertion or deletion of a multibranch loop will lead to an even greater penality since this will eventually cause a hairpin insertion or deletion. The cost of comparing an I and a B labeled node is 3 (considered to be a relabeling operation as opposed to a deletion followed by an insertion). The smaller penality was chosen since the difference between an I and a B may be as little as one nucleotide. An M compared to anything else but an M and an R receives a cost of 8. This is being treated as a relabeling of the nodes rather than a deletion followed by an insertion. This heuristic interpretation is chosen because when the node M is compared to an arbitrary node X, the best cost up to this point has already been computed and therefore the distance between the two forests is the best that it could be. If a stem-loop structure were inserted,—as would happen, for example, if an internal loop were being relabeled to a multibranch loop—the cost of the insertion would be captured by the insertion of the hairpin rather than the insertion of a multibranch loop. Thus, the process should be one of relabeling rather than the more difficult physical process of deletion and insertion, which has already been accounted for. However, since an M compared with something else is different than other comparisons, its cost should not be as low as the other relabelings described, thus the value 8 (as opposed to 5 or 3) is used. An H compared to anything non-H is currently also given a relabeling cost of 8. There can never be relabelings of R's (regions). Regions, however, may be inserted or deleted. Obviously two identical nodes receive a cost of 0.

When dealing with total size comparative costs, the same weights as indicated above are used except that the absolute difference in size of the loops and helical stems are also added in determining the cost. A further refinement breaks the total size cost into its component loop parts. The cost is then

determined by taking the sum of the absolute differences of the component parts of one node compared with the component parts of another node, maintaining the order of the component parts within the nodes and also adding in the relabeling and insertion/deletion penalties described above. If the number of component parts being compared are not the same, the best sum of the differences is chosen. Note that bulge loops may be represented as two components within a node, a 0 and the loop size. The position of the 0 in a sense represents where the stacking across the loop occurs (5′ or 3′ side). This last point raises another issue when comparing loops with an unequal number of components. It would be more ideal if instead of choosing the best sum of the differences, the components that make the most sense morphologically to be compared are compared. For example, if the multibranch loop structure has components sizes say (3,4,5) and it is being compared to an internal loop with component sizes (4,5), instead of the difference based on component size being 3 it might actually make more sense for the difference to be 5. To fit the formalism discussed in this paper, this could be accomplished by making each loop component another node in the tree (see Figure 4). Another refinement to be considered is the incorporation of sequence data. This could be accomplished by making each quartet of stacked bases a node in the tree for helical stems and specifying the subsequence associated with the loop components. The tree algorithm will handle the stem nodes with the current paradigm and the component sequence within the loops may be handled by a standard string homology algorithm.

Cluster algorithm

Once all the pairwise structure costs are computed, the next step is to cluster together those structures that are most similar. This is accomplished by a clustering algorithm (Hong and Tan, 1989) to be discussed. The output of the cluster algorithm is a taxonomy tree whose terminal nodes contain the name of the secondary structures and the non-terminal nodes represent a metric measure of the distance between the structures contained in the subtrees.

The metric used to construct the taxonomy tree is defined as follows:

$$D(a,b) = \min_{a_0,\ldots,a_n} \{ \max_{0 \le i < n} \{ d(a_i,a_{i+1}) | a_0 = a, a_n = b, a_0,\ldots,a_n \} \}$$

where $d(x,y)$ represents the pairwise tree cost distance between two trees (structures). If one assumes that x and y represent two structures, $a_0,\ldots,a_n$ represents a path from x to y going through all possible combinations of pairs of structures. Thus, if there where three structures x, y, z being considered, there would be two possible paths connecting x and y, i.e. $x \rightarrow y$ and $x \rightarrow z \rightarrow y$. One chooses the maximum pairwise distance from each path connecting x and y and then chooses the minimum value from amongst these maximums. This minimum value represents the metric distance D between these two structures. The values of the non-terminal nodes in the taxonomy tree define a lower bound on the distance d between terminal nodes in its subtrees. The maximum distance d between two nodes in the subtree below the non-terminal node is the sum of the node costs times the number of children of a node minus 1 for each non-zero non-terminal node from the current root of interest for all subtrees of the current root of interest. Figure 5 gives an example of the compressed form (see below) of a taxonomy tree based upon the above metric for 100 tRNA structures.

A front-end system has been developed that runs on a Symbolics 3675 workstation that generates the appropriate input and output (display of taxonomy trees) for the algorithms described above. This system is mouse and window driven, making user interaction easy.

The taxonomy tree is generated in a parenthesized form on a Sun. This representation is very similar to the parenthesized representation for structures discussed earlier. The root of each subtree forms an element at the beginning of a list with the cost value D with its subtrees nested within. The terminal nodes in the tree (and in the parenthesized expressions) are the names of the structures. This representation is then passed to a Symbolics 3675 computer and displayed in its graphical form. This operation is initiated by clicking the mouse on an appropriate entry in a command menu on the Symbolics. It should be noted that even though the tree is generated on a remote node tied to the Symbolics by ethernet with TCP/IP, the operation is basically transparent to the user. The Symbolics acts as a master controller, thereby making it easy to activate programs that physically reside on different computer nodes in the network. The philosphy is to integrate software into the secondary structure analysis system (B.A.Shapiro, in preparation) without worrying too much about the language or the machine upon which the original algorithm was developed.

It should be noted that an option exists within the system to compress a taxonomy tree that is too large to fit within the display limits. All terminal nodes that are zero distance apart can be collapsed into a new terminal node. Thus, more of the tree becomes visible. There still remains the option of scrolling the graphical presentation of the taxonomy tree to see those parts that are still not in view. One can also decompress the tree whenever desired or ask the system to find a specific structure (terminal node) within the collasped tree. The node that contains the specific structure will be decompressed.

Interaction with the taxonomy tree is also possible by specifying the name of a structure (terminal node); the display algorithm will locate the structure, center it in the display and highlight it. This obviously becomes quite useful when dealing with a large tree. Another option allows one to point at a specific structure within the tree and a display of the structure will be depicted.

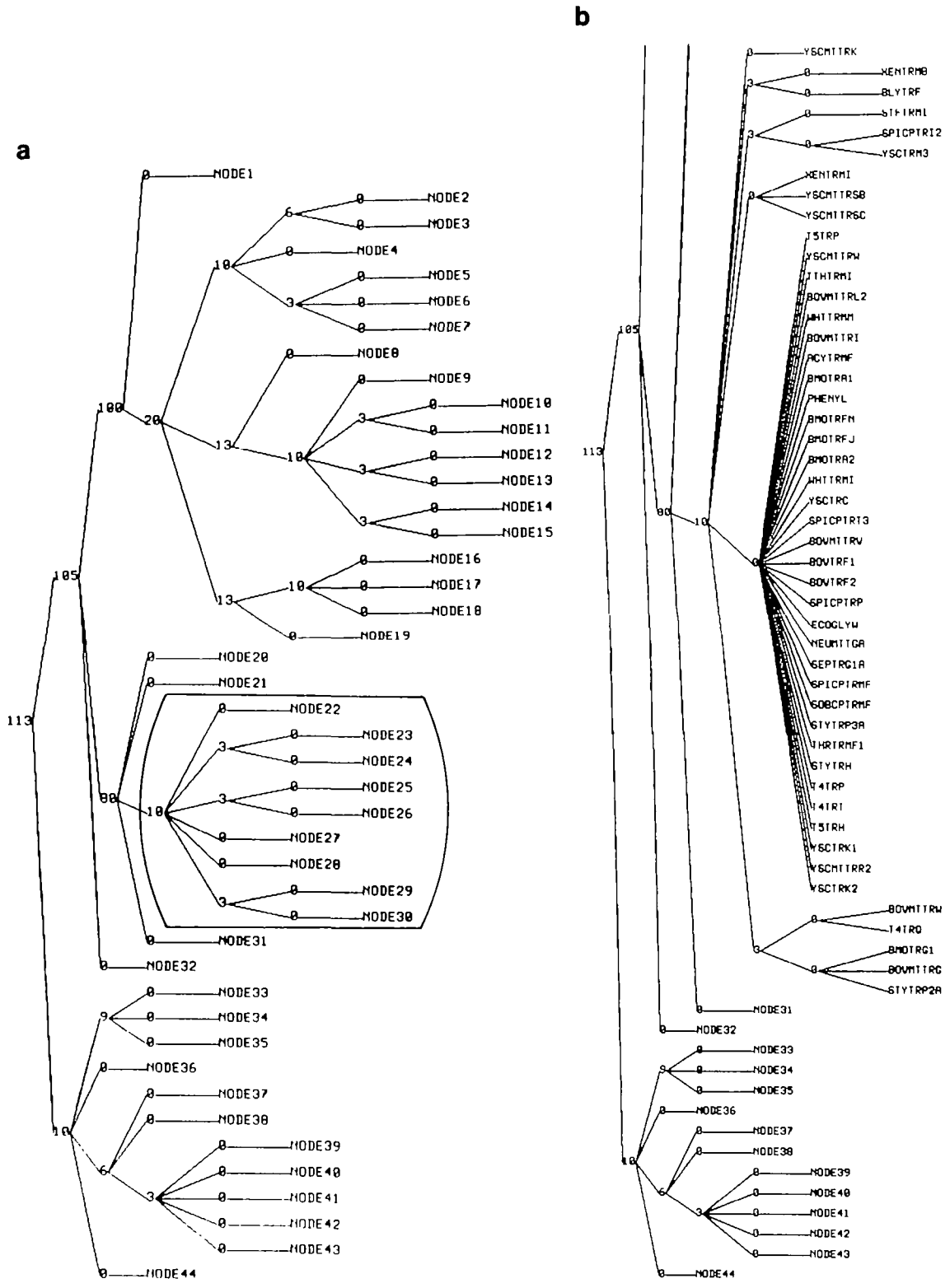

Fig. 5. Examples of taxonomy trees (a) A compressed taxonomy tree representing structural relationships for 100 tRNAs using only morphologic data, i.e. only loop and region entities are used. The compressed form takes up less space. The area within the outlined region represents those nodes that contain structures that are cloverleaf-like. (b) The same 100 tRNAs using the same cost function except that now the 47 cloverleaf-like structures are shown by name. Note the 33 that have identical top-level morphologic structures.

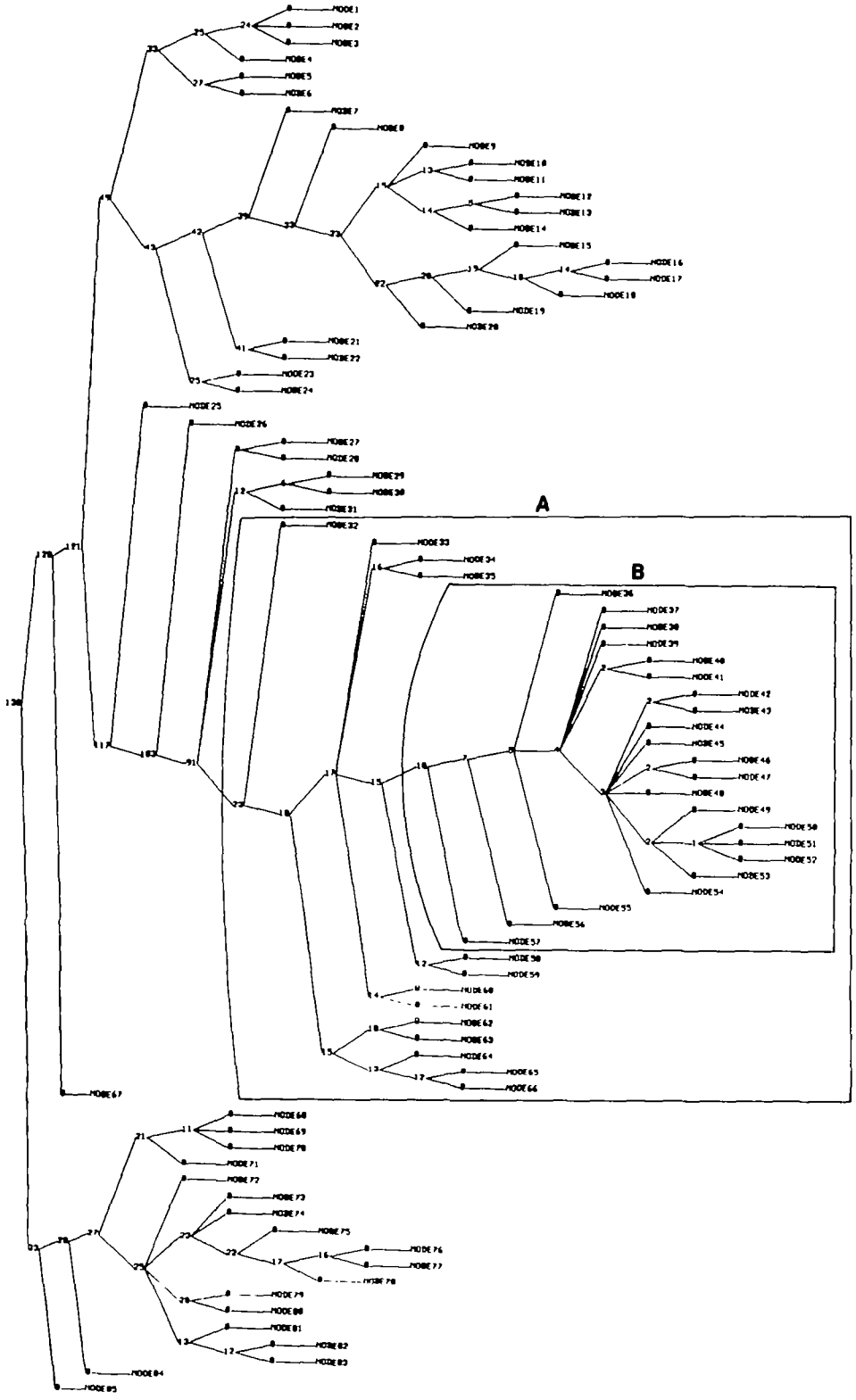

Fig. 6. The compressed taxonomy tree of the same 100 tRNAs illustrated in Figure 5 except that here region and loop sizes are taken into account. The area within A contains the 47 tRNAs that have cloverleaf-like structures. The area within B contains the 33 structures that were identical with the looser cost function of Figure 5 but are now separated by the more refined cost function. The predetermined structure, PHENYL, is in node 45.

Discussion

The pairwise tree comparison algorithm presented here was run on 197 structures. At first a cost function that just distinguished between the loop types was used. This did not include any information concerning the size of loops or regions. The results of these comparisons were then used as input to the clustering algorithm mentioned above. The output was a taxonomy tree where the terminal nodes represented the names of the individual structures and the non-terminal nodes represented the minimum of the maximum distances of the terminal nodes in its subtrees. This result was compared with the algorithm presented in Shapiro (1988). The results looked generally identical in that the groupings of the structures seemed to be similar. The actual time required to accomplish the tree comparison and the cluster analysis on the 197 structures were comparable to the time required to do the string comparison method described in Shapiro (1988). The actual wall time was ~30 min.

A comparison was also run with 100 tRNA structures chosen randomly from the sequence database. These sequences are thought to have cloverleaf-like structures (Sussman and Kim, 1976; Wolfram, 1984). The sequences were folded using the Zuker–Stiegler algorithm (Zuker and Stiegler, 1981) with refined energy rules of Freier *et al.* (1986). In addition, a determined secondary structure (Cantor and Schimmel, 1980) for the tRNA of yeast phenylalanine, named PHENYL, was included. Figure 5(a) illustrates the taxonomy tree that is generated using just the morphologic information (no stem or loop sizes). The tree is presented in its condensed form to allow it to fit on the page. Nodes 22–30 have been expanded in Figure 5(b) to depict 47 cloverleaf-like structures. It will be noted that 33, from node 28, are morphologically identical. There are 14 remaining cloverleaf-like structures that are somewhat different than the previous 33. Figure 6 depicts the condensed taxonomy tree using helix size and loop size in addition to the morphology to compute the distances. The area contained within the outer-ring indicates those structures that are cloverleaf-like, i.e. the 47 that are discussed above. The area within the inner ring contains the 33 structures that were identical under the looser morphologic cost only but are now more refined when taking into account the helix and loop sizes. The predetermined structure of yeast phenylalanine is contained in node 45.

It should also be noted that the multiple alignment program GENALIGN (Sobel and Martinez, 1986) was run on the 100 sequences to test how sequence homology is related to structural homology. The 47 cloverleaf structures were scattered throughout the 100 multiple sequence alignments. Thus, the sequence homology alone is not sufficient for determining structural homology. As a matter of fact it has been shown that there is a fair amount of sequence variability in tRNAs, especially in base-paired regions. The invariance occurs predominately in the loops (Wolfram, 1984). This does, however, serve to illustrate a use for the algorithm by forming the basis of a phylogenetic study of the structure for a class of sequences. The combination of using computed structural similarities in conjunction with sequence alignments can serve as a powerful tool in this regard (see also Konings and Hogeweg, 1989).

The wall time for computing the structural comparisons was <5 min. About 2 min were required to convert the computed region tables for the structures into the representation that the tree comparison algorithm requires. This operation was done using the Symbolics 3675 reading files created and residing on a VAX 8600 and writing the converted files to a sun. The tree comparison took ~2 min on the sun. The results of the comparison were displayed on the symbolics.

The algorithm has proven useful in searching for those compensatory base changes that may preserve function. This was accomplished by automatically generating double mutations in sequences under controlled circumstances and then clustering structures with the wild-type (Margalit *et al.*, 1989). In addition, it may also prove to be useful for searching through a series of suboptimal conformations for consensus structures.

Further experiments are required with the cost function to make it reflect the true costs of changing from one conformation to another. Such a cost function should probably include a measure of the energy required to change such conformations. A weighted cost matrix should also be included at the level of sequence comparisons so that, for example, differences brought about by compensatory base changes have a different cost than those brought about by other nucleotide differences.

In addition, work is currently underway concerning the analysis of taxonomy trees derived from multiple foldings of a single sequence. Such circumstances may arise from suboptimal folding algorithms (Zuker, 1989). The results of this work will be reported upon in a future paper.

The algorithms described above for comparing secondary structure trees and generating taxonomy trees were written in C and currently run on Sun workstations running Unix. A front-end system has also been developed that runs on a Symbolics 3675 workstation. In addition, this system is being ported to a Sun workstation to allow more general access. These systems contain the algorithms to convert region tables to the appropriate intermediate files. This general RNA structure analysis system will be described in a future paper.

Acknowledgements

This project has been funded at least in part with Federal funds from the Department of Health and Human Services under contract number NO1-CO-74102.

References

Altuvia,S., Locker-Giladi,H., Koby,S., Ben-Nun,O. and Oppenheim,A.B. (1987) RNase III stimulates the translation of the cIII gene of bacteriophage lambda. *Proc. Natl Acad. Sci. USA*, **84**, 1–5.

Berkhout,B., Schmidt,B.F., Strien,A., Boom,J., Westrenen,J. and Duin,J.

(1987) Lysis gene of bacteriophage MS2 is activated by translation termination at the overlapping coat gene. *Proc. Natl. Acad. Sci. USA*, **195**, 517–524.
Cantor,C.R. and Schimmel,P.R. (1980) *Biophysical Chemistry Part I The Conformation of Biological Macromolecules*. W.H.Freeman, San Francisco, p. 188.
Deckman,I.C. and Draper,D.E. (1987) S4-alpha mRNA translation regulation complex. *J. Mol. Biol.*, **196**, 323–332.
Dumas,J.P. and Ninio,J. (1982) Efficient algorithm for folding and comparing nucleic acid sequences. *Nucleic Acids Res.*, **10**, 197–206.
Fickett,J.W. (1984) Fast optimal alignment. *Nucleic Acids Res.*, **12**, 175–180.
Freier,S.M., Kierzek,R., Jaeger,J.A., Sugimoto,N., Caruthers,M.H., Neilson,T. and Turner,D.H. (1986) Improved free-energy parameters for predictions of RNA duplex stability. *PNAS*, **83**, 9373–9377
Goad,W.B. and Kanehisa,M.I. (1982) Pattern recognition in nucleic acid sequences. I. A general method for finding local homologies and symmetries. *Nucleic Acids Res.*, **10**, 247–263.
Hall,M.N., Gabay,J., Debarbouille,M. and Schwartz,M. (1982) A role for mRNA secondary structure in the control of translation initiation. *Nature*, **295**, 616–618.
Hong,J. and Tan,X. (1989) The taxonomy-based learning: algorithms and applications, University of Massachusetts at Amherst, COINS Technical Report.
Konings,D.A.M. and Hogeweg,P. (1989) Pattern analysis of RNA secondary structure—similarity and consensus of minimal energy folding. *J. Mol. Biol.*, **207**, 597–614
Knuth,D. (1973) *Fundamental Algorithms. The Art of Computer Progamming*. Addison-Wesley, Reading, MA, Vol 1.
Margalit,H., Shapiro,B.A., Oppenheim,A.B. and Maizel,J V.,Jr (1989) Detection of common motifs in RNA secondary structure. *Nucleic Acids Res.*, **17**, 4829–4845
Needleman,S.B. and Wunsch,C.D. (1970) A general method appliable to the search of similarities in the amino acid sequence of two proteins. *J Mol. Biol.*, **48**, 444–453.
Nussinov,R. (1983) An efficient code searching for sequence homology and DNA duplication. *J. Theor. Biol.*, **100**, 319–328.
Sankoff,D. (1972) Matching sequences under deletion/insertion constraints. *Proc. Natl. Acad. Sci. USA*, **69**, 4–6.
Sankoff,D. (1985) Simultaneous solution of the RNA folding, alignment and protsequence problems. *SIAM J. Appl. Math.*, **45**, 810–825.
Sellers,P.H. (1980) The theory and computation of evolutionary distances: pattern recognition. *J. Algorithms*, **1**, 359–373.
Shapiro,B.A. (1988) An algorithm for comparing multiple RNA secondary structures. *Comput. Applic. Biosci* , **4**, 387–393.
Shapiro,B.A., Maizel,J., Lipkin,L.E., Currey,K. and Whitney,C. (1984) Generating non-overlapping displays of nucleic acid secondary structure. *Nucleic Acids Res.*, **12**, 75–88.
Sobel,E. and Martinez,H. (1986) A multiple sequence alignment program. *Nucleic Acids Res.*, **14**, 363–374.
Sussman,J.L. and Kim,S.H. (1976) Three dimensional structure of transfer RNA in two crystal forms. *Science*, **192**, 853.
Tai,K. (1979) The tree-to-tree connection problem *Journal of the Association for Computing Machinery*, **26**, 422–433.
Thomas,M.S. and Nomura,M. (1987) Translational regulation of the L11 ribosomal protein operon of Escherichia coli: mutations that define the target site for repression by L1. *Nucleic Acids Res.*, **15**, 3085–3096.
Tuerk,C., Gauss,P., Thermes,C., Groebe,D R., Gayle,M., Guild,N., Stormo,G , D'Aubenton-Carafa,Y., Uhlenbeck,O.C., Tinoco,I.,Jr, Brody,E.N. and Gold,L. (1988) CUUCGG hairpins: extraordinarily stable RNA secondary structures associated with various biochemical processes. *Proc. Natl. Acad. Sci. USA*, **85**, 1364–1368.
Wilbur,W.J. and Lipman,D. (1983) Rapid similarity searches of nucleic acid and protein data banks. *Proc. Natl. Acad. Sci. USA*, **80**, 726–730.
Wolfram,S. (1984) *Principles of Nucleic Acis Structure*. Springer-Verlag, New York, pp. 331–332.
Zhang,K. and Shasha,D. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, in press.
Zuker,M. and Stiegler,J. (1981) Optimal computer folding of large RNA sequences using thermodynamics and auxillary information. *Nucleic Acids Res.*, **9**, 133–148.
Zuker,M. (1989) On finding all suboptimal foldings of an RNA molecule. *Science*, **244**, 48–52.

Received on September 15, 1989; accepted on June 12, 1990