Fast Folding and Comparison of RNA Secondary Structures

(The Vienna RNA Package)

By

IVO L. HOFACKER[a,*], WALTER FONTANA[c], PETER F. STADLER[a,c], L. SEBASTIAN BONHOEFFER[d], MANFRED TACKER[a] AND PETER SCHUSTER[a,b,c]

[a]Institut für Theoretische Chemie der Universität Wien
A-1090 Wien, Austria

[b]Institut für Molekulare Biotechnologie
D-07745 Jena, Germany

[c]Santa Fe Institute
Santa Fe, NM 87501, U.S.A.

[d]Department of Zoology, University of Oxford
South Parks Road, Oxford OX1 3PS, UK

[*]Mailing Address:
Institut für Theoretische Chemie der Universität Wien
Währingerstraße 17, A-1090 Vienna, Austria
Email: ivo@otter.itc.univie.ac.at
Phone: **43 1 40 480 678, Fax: **43 1 40 28 525

Abstract

Computer codes for computation and comparison of RNA secondary structures, the `Vienna RNA package`, are presented, that are based on dynamic programming algorithms and aim at predictions of structures with minimum free energies as well as at computations of the equilibrium partition functions and base pairing probabilities.

An efficient heuristic for the inverse folding problem of RNA is introduced. In addition we present compact and efficient programs for the comparison of RNA secondary structures based on tree editing and alignment.

All computer codes are written in `ANSI C`. They include implementations of modified algorithms on parallel computers with distributed memory. Performance analysis carried out on an Intel Hypercube shows that parallel computing becomes gradually more and more efficient the longer the sequences are.

1. Introduction

Recent interest in RNA structures and functions was caused by their catalytic capacities (Cech & Bass 1986, Symons 1992) as well as by the success of selection methods in producing RNA molecules with perfectly taylored properties (Beaudry & Joyce 1992, Ellington & Szostak 1990). Conventional structure analysis concentrates on natural molecules and closely related variants which are accessible by site directed mutagenesis. Several current projects are much more ambitious (particularly encouraged by ready availability of random RNA sequences) and aim at the exploration of sequence-structure relations in full generality (Fontana *et al.* 1991, 1993a,b). The new approach turned out to be successful on the level of RNA secondary structures. In order to be able to do proper statistics millions of structures derived from arbitrary sequences have to be analyzed. In addition folding of long sequences becomes more and more important as well. Both tasks call for fast and efficient folding algorithms available on conventional sequential computers as well as on parallel machines. The need arises to compare the performance of sequential and parallel implementations in order to provide information for the conception of optimal strategies for given tasks.

The inverse folding problem is one of several new issues brought up by recent developments in rational design of RNA molecules: given an RNA secondary structure, which are the RNA sequences that form this structure as a minimum free energy structure. The information about many such "structurally neutral" sequences is the basis for tayloring RNA molecules which are suitable candidates for multi-functional molecules. More and more sequence data becoming currently available call for efficient comparisons either directly or on the level of their minimum free energy structures. Conventional alignment techniques are supplemented by new approaches like statistical geometry (Eigen *et al.* 1988) and split decomposition (Bandelt & Dress 1992).

In this paper we introduce a package for computation, comparison and analysis of RNA secondary structures and properties derived from them, the Vienna RNA Package. The core of the package consists of compact codes to compute either minimum free energy structures (Zuker & Stiegler 1981, Zuker & Sankoff 1984) or partition functions of RNA molecules (McCaskill 1990). Both use the idea of dynamic programming originally applied by Waterman (Waterman 1978, Waterman & Smith 1978, Nussinov & Jacobson 1980). Non-thermodynamic criteria of structure formation like maximum matching (the maximal number of base pairs; Nussinov *et al.* 1978) or various versions of kinetic folding (Martinez 1984) can be applied as alternative options. An inverse folding heuristic is implemented to determine sets of structurally neutral sequences. A statistics package is included which contains routines for cluster analysis, statistical geometry, and split decomposition. This core is now avaliable as library as well as a set of stand alone routines.

In a forthcoming version the package will include routines for secondary structure statistics (Fontana *et al.* 1993b), statistical analysis of RNA folding landscapes as well as sequence-structure maps (Schuster *et al.* 1993). Further options will be available for RNA melting kinetics, in particular for the computation of melting curves and their first derivatives (Tacker *et al.* 1993). Extensions of the package provide access to computer codes for optimization of RNA secondary structrues according to predefined criteria, as well as simulations of molecular evolution experiments in flow reactors (Fontana & Schuster 1987, Fontana *et al.* 1989).

In section 2 we present the core codes for folding as well as some I/O-routines that can be used for stand-alone applications of the folding programs. Section 3 introduces a variant of the folding program which is suitable for implementation on a parallel computer with hypercube architecture. Section 4 is dealing with the inverse folding problem. Section 5 decribes codes for comparing RNA secondary structures as well as base-pairing matrices derived from partition functions. Basic to our routines is a tree representation of RNA secondary structures introduced previously (Fontana *et al.* 1993b). Some examples of selected applications of the Vienna RNA package are given in section 6.

2. RNA Folding Programs

A secondary structure on a sequence is a list of base pairs i, j with $i < j$ such that for any two base pairs i, j and k, l with $i \leq k$ holds:

$$\begin{aligned} i = k &\iff j = l \\ k < j &\implies i < k < l < j \end{aligned} \tag{1}$$

The first condition implies that each nucleotide can take part in not more that one base pair, the second condition forbids knots and pseudoknots. The latter restriction is necessary for dynamic programming algorithms. A base pair k,l is *interior* to the base pair i,j, if $i < k < l < j$. It is *immediately interior* if there is no base pair p,q such that $i < p < k < l < q < j$. For each base pair i,j the corresponding loop is defined as consisting of i,j itself, the base pairs immediately interior to i,j and all unpaired regions connecting these base pairs. The energy of the secondary structure is assumed to be the sum of the energy contributions of all loops. (Note that a stacked basepair constitutes a loop of zero size.) As a consequence of the additivity of the energy contributions, the minimum free energy can be calculated recursively by dynamic programming (Waterman 1978, Waterman & Smith 1978, Zuker & Stiegler 1981, Zuker & Sankoff 1984).

Experimental energy parameters are available for the contribution of an individual loop as functions of its size, of the type of its delimiting basepairs, and partly of the sequence of the unpaired strains. These are usually measured for $T = 37°C$ and $1M$ sodium chloride solutions (Freier *et al.* 1986, Jaeger *et al.* 1989). For the base pair stacking the enthalpic and entropic contributions are known separately. Contributions from all other loop types are assumed to be purely entropic. This allows to compute the temperature dependence of the free energy contributions:

$$\Delta G_{\mathrm{stack}} = \Delta H_{37,\mathrm{stack}} - T\Delta S_{37,\mathrm{stack}} \qquad \Delta G_{\mathrm{loop}} = -T\Delta S_{37,\mathrm{loop}} \tag{2}$$

We use a recent version of the parameter set published by Freier *et al.* (1986), which was supplied in an updated version by Danielle Konings. In the current implementation we do not consider dangling ends. The essential part of the energy minimization algorithm is shown in table 1.

The structure (list of base pairs) leading to the minimum energy is usually retrieved later on by “backtracking” through the energy arrays.

Table 1: Pseudo Code of the minimum free energy folding algorithm.

```
for(d=1...n)
   for(i=1...d)
      j=i+d
      C[i,j] = MIN(
          Hairpin(i,j),
          MIN( i<p<q<j : Interior(i,j;p,q)+C[p,q] ),
          MIN( i<k<j  : FM[i+1,k]+FM[k+1,j-1]+cc )        )
      F[i,j] = MIN( C[i,j], MIN(i<k<j  : F[i,k]+F[k+1,j]))
      FM[i,j]= MIN( C[i,j]+ci, FM[i+1,j]+cu, FM[i,j-1]+cu,
                  MIN( i<k<j  : FM[i,k]+FM[k+1,j] )        )
free_energy = F[1,n]
```

Remark. `F[i,j]` denotes the minimum energy for the subsequence consisting of bases i through j. `C[i,j]` is the energy given that i and j. pair. The array `FM` is introduced for handling multiloops. The energy parameters for all loop types except for multiloops are formally subsumed in the function `Interior(i,j;p,q)` denoting the energy contribution of a loop closed by the two base pairs $i{-}j$ and $p{-}q$. We have assumed that multi-loops have energy contribution `F=cc+ci*I+cu*U`, where `I` is the number of interior base pairs and `U` is the number of unpaired digits of the loop. The time complexity here is $\mathcal{O}(n^4)$. It is reduced to $\mathcal{O}(n^3)$ by restricting the size of interior loops to some constant, say 30.

The partition function for the ensemble of all possible secondary structures can be calculated analogously

$$Q = \sum_{\text{all structures } S} e^{-\dfrac{\Delta G(S)}{RT}} \tag{3}$$

(McCaskill 1990). A pseudocode is given in table 2.

Clearly the algorithm of table 2 does not predict a secondary structure, but one can calculate the probability P_{kl} for the formation of a base pair (k,l):

$$P_{kl} = \frac{Q_{1,k-1} Q^b_{kl} Q_{l+1,n}}{Q} + \sum_{\substack{i,j\\ i<k<l<j}} \frac{P_{ij} Q^b_{kl} \mathtt{EInterior}(i,j;k,l)}{Q^b_{ij}} + \tag{4}$$

$$\mathtt{Ecc}\cdot\mathtt{Eci} \sum_{\substack{i,j\\ i<k<l<j}} \frac{P_{ij}}{Q^b_{kl}} \left[\mathtt{Ecu}^{k-i-1} Q^m_{l+1,j-1} + Q^m_{i+1,k-1} \mathtt{Ecu}^{j-l-1} + Q^m_{i+1,k-1} Q^m_{l+1,j-1}\right],$$

where the symbols are defined in the caption of table 2. In this case the backtracking has time complexity $\mathcal{O}(n^3)$ just as the calculation of the partition function itself.

Table 2: Pseudocode for the calculation of the partition function.

```
for(d=1...n)
   for(i=1...d)
      j=i+d
      QB[i,j] = EHairpin(i,j) +
         SUM( i<p<q<j : EInterior(i,j;p,q)*QB[p,q] ) +
         SUM( i<k<j : QM[i+1,k-1]*QM1[k,j-1]*Ecc )
      QM[i,j] =
         SUM( i<k<j : (Ecu^(k-i)+QM[i,k-1])*QM1[k,j] )
      QM1[i,j]= SUM( i<k<=j : QB[i,k]*Ecu^(j-k)*Eci )
      Q[i,j]  = 1 + QB[i,j] +
         SUM( i<p<q<j : Q[i,p-1]*QB[p,q] )
partition_function = Q[1,n]
```

Remark. Here Ex:=exp$(-x/RT)$ denotes the Boltzmann weights corresponding to the energy contribution x. Q[i,j] denotes the partition function Q_{ij} of the subsequence i through j. The array QM contains the partition function Q^b_{ij} of the subsequence subject to the fact that i and j form a base pair. QM and QM1 are used for handling the multiloop contributions. x y means x^y. For details see McCaskill (1990).

Both folding algorithms have been integrated into a single interactive program including postscript output of the minimum energy structure and the base pairing probability matrix.

The program requires $6\,n^2$ bytes of memory for the minimum energy fold and $10\,n^2$ bytes for the calculation of the partition function on machines with 32 bit integers and single precision floating points. In order to overcome overflows for longer sequences we rescale the partition function of a subsequence of length ℓ by a factor $\tilde{Q}^{\ell/n}$, where $\tilde{Q}$ is a rough estimate of the order of magnitude of the partition functions:

$$\tilde{Q} = \exp\left(\frac{-184.3 + 7.27(T - 37.)}{RT}\right) \tag{5}$$

The performance of the algorithms reported here is compared with Zuker's (Zuker 1989) more recent program `mfold 2.0` (available via anonymous ftp from `nrcbsa.bio.nrc.ca`) which computes suboptimal structures together with the minimum free energy structure in table 3. The computation of the minimum free energy structure including the entire matrix of base pairing probabilities is considerably faster with the present package (although we do not provide information on individual suboptimal structures). Secondary structures are represented by a string of dots and matching parentheses, where dots symbolize unpaired bases and matching parentheses symbolize base pairs. An example is seen in the sample session shown in in figure 1.

Table 3: Performance of implementations of folding algorithms. CPU time is measured on a SUN SPARC 2 Workstation with 32M RAM. Data are for random sequences.

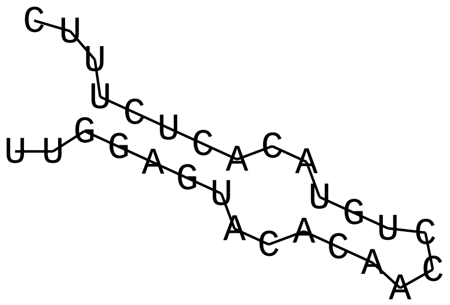
n	CPU time per folding [s]		
	RNAfold 1.0		mfold 2.0
	MFE	MFE+PF	
100	2.0	6.2	24.5
200	10.9	34.7	129.6
300	32.2	97.4	354.4
500	96.6	312.4	1258.3
690	228.1	743.9	3105.1

Because of the simplifications in the energy model and the uncertainties in the energy parameters predictions are not always as accurate as one would like. It is, therefore, desirable to include additional structural information from phylogenetic or chemical data.

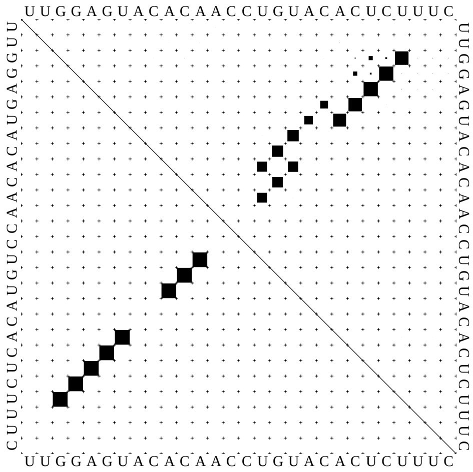
Our minimum free energy algorithm allows to include a variety of constraints into the secondary structure prediction by assigning bonus energies to structures honoring the constraints. One may enforce certain base pairs or prevent bases from pairing. Additionally, our algorithm can deal with bases that have to pair with an unknown pairing partner. A sample session is described in figure 2.

```
tram> RNAfold -T 42 -p1
Input string (upper or lower case); @ to quit
.........1.........2.........3.........4.........5.........6.........7.........
UUGGAGUACACAACCUGUACACUCUUUC
length = 28

UUGGAGUACACAACCUGUACACUCUUUC
..(((((..(((...)))..)))))...
minimum free energy = -3.71
..(((((([[(,,...)))..))))))...
free energy of ensemble = -4.39
frequency of mfe structure in ensemble 0.337231
```



rna.ps



dot.ps

Figure 1: Interactive example run of RNAfold for a random sequence. When the base pairing probability matrix is calculated the symbols . , | { } ( ) are used for bases that are essentially unpaired, weakly paired, strongly paired without preferred direction, weakly upstream (downstream) paired, and strongly upstream (downstream) paired, respectively. Apart from the console output the two postscript files `rna.ps` and `dot.ps` are created. The lower left part of `dot.ps` shows the minimum energy structure, while the upper right shows the pair probabilities. The area of the squares is proportional to the binding probability.

3. Parallel Folding Algorithm.

We provide an implementation of the folding algorithm for parallel computers. In the following we present a parallelized version of the minimum energy folding algorithm for message passing systems. Since all subsequences of equal length can be computed concurrently, it is advisable to compute the

```
Input string (upper or lower case); @ to quit
.........1.........2.........3.........4.........5.........6.........7.........

CACUACUCCAAGGACCGUAUCUUUCUCAGUGCGACAGUAA
.((.......<<..........||............))..
length = 40
CACUACUCCAAGGACCGUAUCUUUCUCAGUGCGACAGUAA
.((((((..(((((.....)))))...))).....)))..
minimum free energy = 0.83
```

a)

```
CACUACUCCAAGGACCGUAUCUUUCUCAGUGCGACAGUAA
((((.....(((((.....)))))...)))).........
minimum free energy = -1.52
```

b)

Figure 2: a) Example Session of `RNAfold -C`. The constraints are provided as a string consisting of dots for bases without constraint, matching pairs of round brackets for base pairs to be enforced, the symbols ’<’ and ’>’ for bases that are paired upstream and downstream, respectively, and the pipe symbol ’|’ denoting a paired base with unknown pairing partner. **b)** shows minimum free energy structure without contraints for comparison.

arrays F, C and FM (defined in table 1) in diagonal order, dividing each subdiagonal into P pieces. Figure 3 shows an example for 4 processors. Our algorithm stores the arrays F and FM both as columns and rows, while the C array is stored in diagonal order. The maximal memory requirement occurs at $d = n/2$, where we need $n^2/(2P)$ integers each for F and FM, while the array C needs only $\mathcal{O}(n)$ storage. Since the length of the rows and columns increases, one needs to reorganize the storage after each diagonal. If one allocates twice the minimum memory, storage has to be reorganized only once and the total requirement is the same as for the sequential algorithm. After completing one subdiagonal each processor has to either send a row to or receive a column from its right neighbour, and it has to either receive a row from or send a column to its left neighbour.

Since we do not store the entire matrices, we cannot do the usual backtracking to retrieve the structure corresponding to the minimum energy. Instead, we write for each index pair (i, j) two integers to a file, which identify

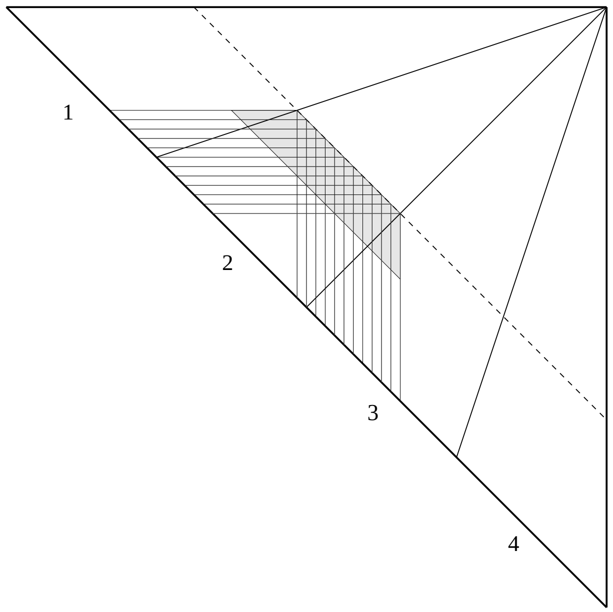


Figure 3: Representation of memory usage by the parallel folding algorithm. The triangle representing the triangular matrices `F`, `C`, and `FM`, respectively, is divided into sectors with an equal number of diagonal elements, one for each processor. The computation proceeds from the main diagonal towards the upper right corner. The information needed by processor 2 in order to calculate the elements of the dashed diagonal are highlighted. To compute its part of the dashed diagonal processor 2 needs the horizontally and vertically striped parts of the arrays `F` and `FM`, and the shaded part of the array `C`. The shaded part does not extend to diagonal, because we have restricted the maximal size of interior loops.

the term that actually produced the minimum. The backtracking can then be done with $\mathcal{O}(n)$ readouts. All in all we need $\mathcal{O}(n)$ communication and I/O steps each transferring $\mathcal{O}(n)$ integers, while the computational effort is $\mathcal{O}(n^3)$. The communication overhead therefore becomes negligible for sufficiently long sequences.

On the Intel `Ipsc/2` the advantage of storing `F` and `FM` in rows and columns outweighs the I/O overhead for sequences longer than some 200 nucleotides. The efficiency of the parallelization as a function of sequence length and number of processors can be seen in figure 4.

The partition function algorithm can be parallelized analogously.

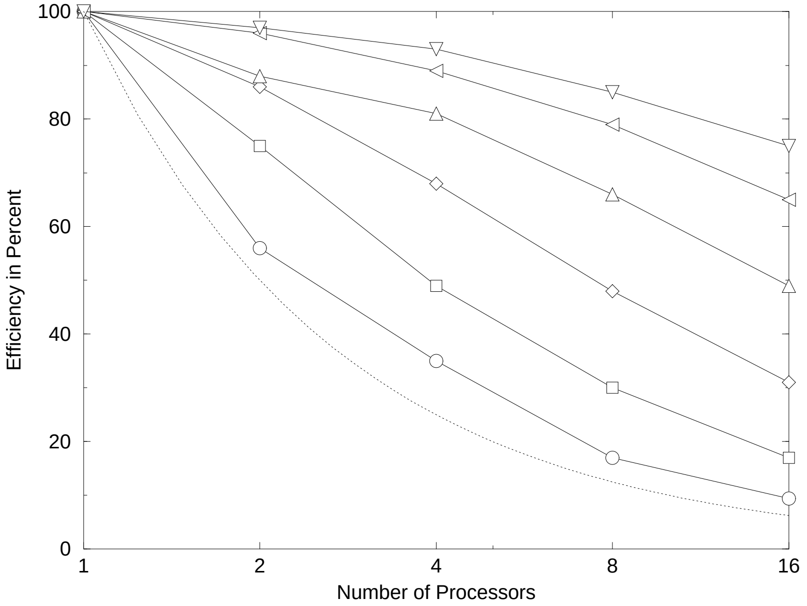


Figure 4: Performance of parallel algorithm for random sequences of length 50 ○, 100 □, 200 ◇, 400 △, 700 ◁, 1000 ▽. Efficiency is defined as speedup divided by the number of processors. The dotted line is $1/n$ corresponding to no speedup at all.

4. Inverse Folding

Inverse folding is highly relevant for two reasons: (1) to find sequences that form a predefined structure under the minimum free energy criterium, and (2) to search for sequences whose Boltzmann ensembles of structures match a given predefined structure more closely than a given threshold. The first aspect is directly addressed by the inverse folding algorithm presented here. The second issue is somewhat more realistic. In the case of compatible sequences, that is sequences which *can* form a base pair wherever it is required in the target structure, we shall often find the desired structure among suboptimal foldings with free energies close to the minimal value. Matching base pairing probabilities can be approached by the same inverse folding heuristic using the base pairing matrices of the partition function algorithm rather than minimum free energy structures.

Only compatible sequences are considered as candidates in the inverse folding procedure. Clearly, a compatible sequence can but need not have the target structure as its minimum free energy structure.

Our basic approach is to modify an initial sequence I_0, such as to minimize a cost function given by the "structure distance" $f(I) = d(S(I), \mathcal{T})$ between the structure $S(I)$ of the test sequence I and the target structure $\mathcal{T}$. A set of possible distance measures is discussed in section 5. The actual choice of this distance measure is not critical for the performance of the algorithm.

This procedure requires many evaluations of the cost functions and thereby many executions of the folding algorithm. Instead of running the optimization directly on the full length sequence, we optimize small substructures first, proceeding to larger ones as shown in the flow chart (figure 5). This reduces the probability of getting stuck in a local minimum, and more important, it reduces the number of foldings of full length sequences. This is possible because substructures contribute additively to the energy. If $\mathcal{T}$ is an optimal structure on the sequence S and contains the base pair (i, j) then the substructure $\mathcal{T}_{i,j}$ must be optimal on the subsequence $S_{i,j}$. It is likely then, but by no means necessary, that the converse also holds: A structure that is optimal for a subsequence will also appear with enhanced probability as a substructure of the full sequence.

For the actual optimization (denoted by 'FindStructure' in figure 5) we use the simplest possibility, an adaptive walk. In general, an adaptive walk will try a random mutation, and accept it iff the cost function decreases. A mutation consists in exchanging one base at positions that are unpaired in the target structure $\mathcal{T}$, or in exchanging two bases, while retaining compatibility, if their corresponding positions pair in T. If no advantageous mutation can be found, the procedure stops, and restart again with a new initial string

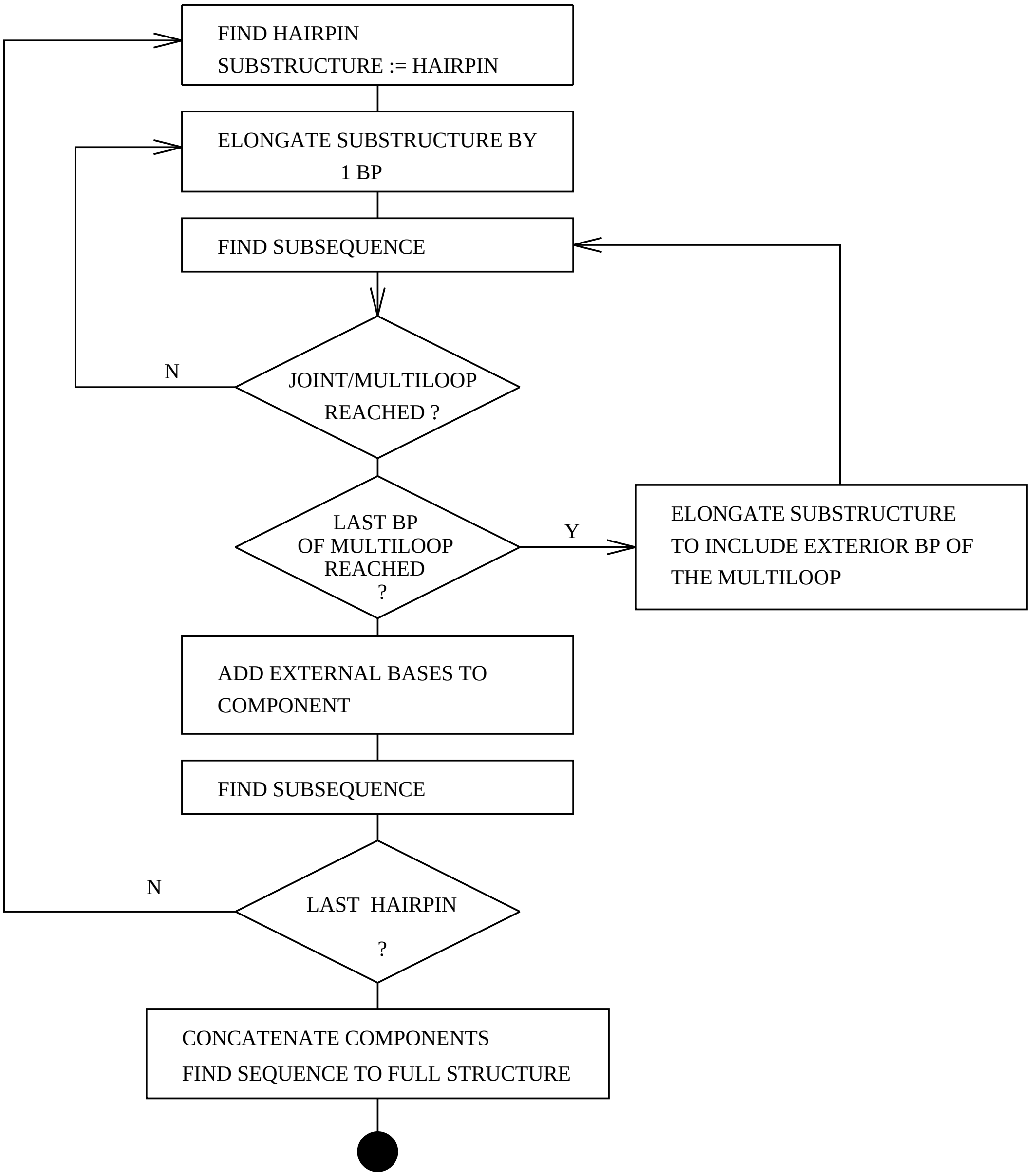

FIND HAIRPIN
SUBSTRUCTURE := HAIRPIN
ELONGATE SUBSTRUCTURE BY 1 BP
FIND SUBSEQUENCE
N
JOINT/MULTILOOP REACHED ?
LAST BP OF MULTILOOP REACHED ?
Y
ELONGATE SUBSTRUCTURE TO INCLUDE EXTERIOR BP OF THE MULTILOOP
ADD EXTERNAL BASES TO COMPONENT
FIND SUBSEQUENCE
N
LAST HAIRPIN ?
CONCATENATE COMPONENTS
FIND SEQUENCE TO FULL STRUCTURE

Figure 5: Flow chart of the inverse folding algorithm.

I_0. Optionally, we can restrict the search to bases that do not pair correctly. This slightly increases the probability that no sequence can be found, but greatly reduced the search space.

Sequences found by the inverse folding algorithm will often allow alternative structures with only slightly higher energies. To find sequences with

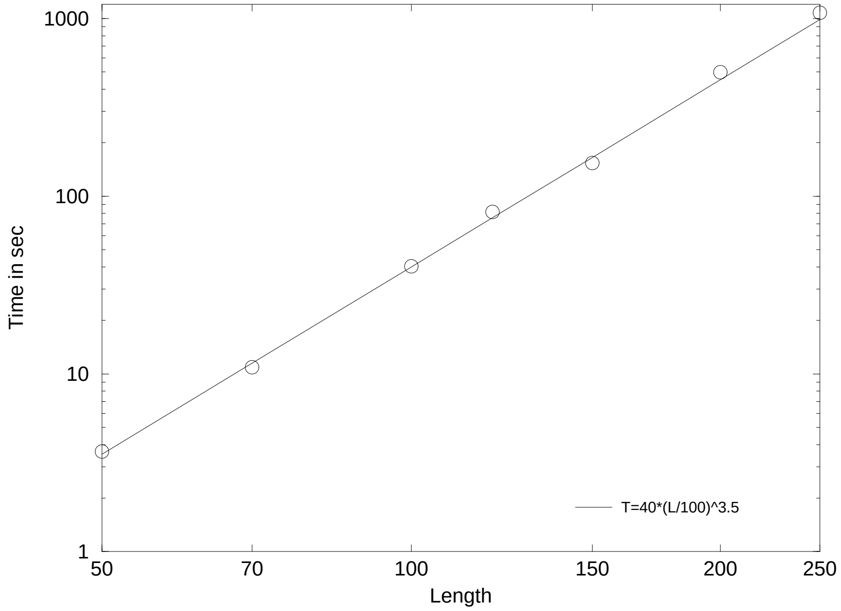


Figure 6: Performance of inverse folding. full line: $T{=}4{\cdot}10^{-6}n^{3.5}$.

clearly defined structures, the partition function algorithm can be used to maximize the probability

$$P(\mathcal{T}) = \frac{1}{Q}\exp(-\Delta G(\mathcal{T})/RT) \tag{6}$$

of the desired structure in the ensemble. This procedure is much slower, since the optimization is done with the full length sequence.

5. Comparison of Secondary Structures

RNA secondary structures can be represented as trees (Zuker & Sankoff 1984, Shapiro 1988, Shapiro & Zhang 1990). The tree representation was used, for example, to obtain coarse grained structures revealing the branching pattern or the relative positions of loops. More recently we proposed a tree representation at full resolution (Fontana *et al.* 1993b). A secondary structure S_k is converted one to one into a tree T_k by assigning an internal node to each base pair and a leaf node to each unpaired digit (Figure 8). The

```
tram> RNAinverse -Fm -R -3
Input structure & start string (lower case letters for const positions)
@ to quit, and 0 for random start string
.........1.........2.........3.........4.........5.........6.........7.........
(((((((..((((........)))).(((((.......))))).....(((((.......))))))))))))....
0
length = 76
CUAUACUACGAGGAUAAUCUGCCUUUUGCCAAAGAGGGUGGCAUUUCAUCAGCUCCGAAUGCUGAGGUAUAGCGAA 20
AGCUCUGAUAUCUCUACGAAUAGAUCCUUUAUAUCUCUUAAAGCGUGUCUGGAAGAUAACUCCAGCAGAGCUUGUG 25
UUCUCCUGUAGUCGACUUUAGGACUCGAGGCCGUAUUUGCCUCACGGAAAUGUUUACAAUGCAUUAGGAGGAGUGC 29
```

A

```
tram> RNAinverse -Fp -R 3
Input structure & start string (lower case letters for const positions)
 @ to quit, and 0 for random start string
.........1.........2.........3.........4.........5.........6.........7.........
(((((((..((((........)))).(((((.......))))).....(((((.......))))))))))))....
0
length = 76
GCUAGCGUUGGGCUUUUUUUCGCCCUGCCGCAAAACCCGCGGCUUCUCGCUACAUCUCUCGUAGCCGCUAGCAAAA 50
(0.844786)
GCGUUACAAGCGCAAUCCCCCGCGCAGCGUCAAAACCCGACGCCAACAGCUACAAAACCCGUAGCGUAACGCAAAA 55
(0.859519)
GCGCGCCAAGCGCAAAAAAAAGCGCAGCCGCAAAACACGCGGCAAAAAGCGGCAGAAAAAGCCGCGGCGCGCAAAA 49
(0.85046)
```

B

Figure 7: Sample session of `RNAinverse`. A: Minimum free energy. B: Partition function.
Data on the natural tRNA$^{\text{phe}}$ sequence for comparison: the clover leaf structure occurs only with a probability of 0.172511 and with even smaller probabilities for the three sequences found by the `RNAinverse -Fm` (0.0891717, 0.0329602, and 0.0335713, respectively).

conversion starts with a root which does not correspond to a physical unit of the RNA molecule. It is introduced to prevent the formation of a tree forest for RNA structures with external elements (For details of the interconversion of secondary structures and trees see also Fontana *et al.* 1993b).

As shown in (Fontana *et al.* 1993b) the trees T_k can be rewritten as *homeomorphically irreducible trees* (HITs), see appendix B. The transformation from the full tree to the HIT retains complete information on the structure. Secondary structure, full tree as well as HIT are equivalent.

Tree editing induces a metric in the space of trees (see appendix A), and in the space of RNA secondary structures. A tree is transformed into another

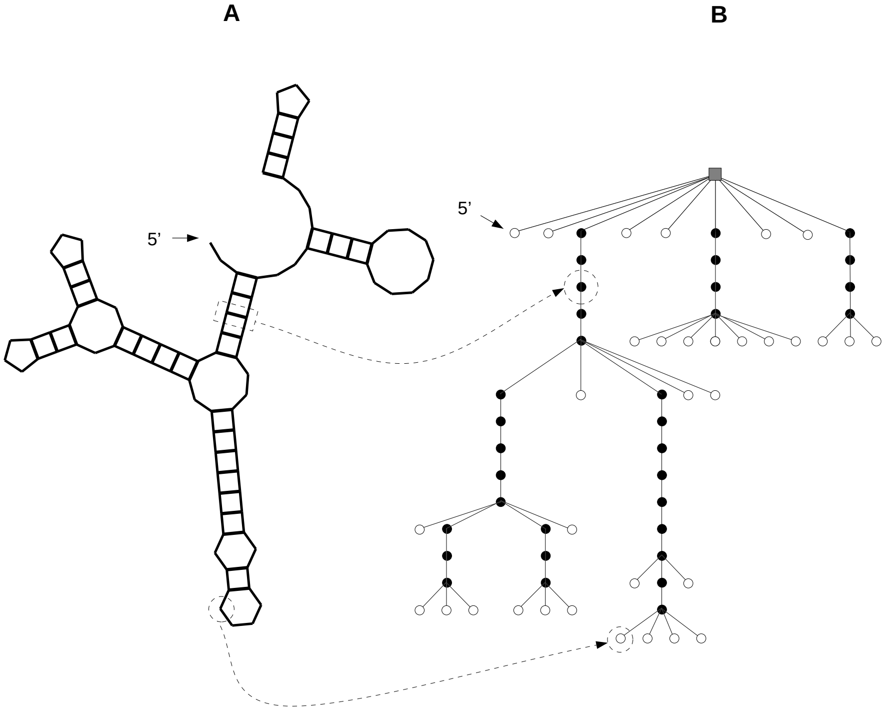


Figure 8: Interconversion of secondary structures and trees. A secondary structure graph (A) is equivalent to an ordered rooted tree (B). An internal node (black) of the tree corresponds to a base pair (two nucleotides), a leaf node (white) corresponds to one unpaired nucleotide, and the root node (black square) is a virtual parent to the external elements. Contiguous base pair stacks translate into "ropes" of internal nodes and loops appear as bushes of leaves. Recursively traversing a tree by first visiting the root, then visiting its subtrees in left to right order, finally visiting the root again, assigns numbers to the nodes in correspondence to the 5'-3' positions along the sequence (Internal nodes are assigned two numbers reflecting the paired positions).

tree by a series of editing operations with predefined costs (Tai 1979, Sankoff & Kruskal 1983). The distance between two trees $d(T_j, T_k)$ is the smallest sum of the costs along an editing path. The parameters used in our tree editing are summarized in appendix A. The editing operations preserve the relative traversal order of the tree nodes. Tree editing can therefore be viewed as a generalization of sequence alignment. In fact, for trees that consist solely of leaves, tree editing becomes the standard sequence alignment. A sample session computing the tree distance between two arbitrarily chosen structures is shown in figure 9.

An alternative graphical method for the comparison of RNA secondary structures (Hogeweg & Hesper 1984, Konings 1989, Konings & Hogeweg 1989) encodes secondary structures as linear strings with balanced parentheses representing the base pairs, and some other symbol coding for unpaired positions.

Tree representations in full resolution make it often difficult to focus on the major structural features of RNA molecules since they are often overloaded with irrelevant details. Coarse-grained tree representations were invented previously to solve this problem (Shapiro 1988).

```
tram> RNAdistance -DfhwcHWC -B

Input structure; @ to quit
.........1.........2.........3.........4.........5.........6.........7.........
((.(((((((.....))))))).))....((..((((.....)))).)).
.....((((..((((..........)))).)))).....(((....))).
f:  26
(-----(.(((--((((.....-----))))-))).))....((..((((.....)))).)).
-.....(-(((..((((..........)))).)))-)-....--.--(((....-)))-.---
h:  32
(----((U1)((U5-)P7)(U1)P2)(U4)((U2)((U5)P4)(U1)P2)(U1)R1)
((U5)((U2)((U10)P4)(U1)P4)(U5)-----((U4)P3)(U1)-------R1)
w:  34
(((((H5-)S7)I2)S2)(((H5)S4)I3)S2)E5-)R1)
(((((H10)S4)I3)S4)--((H4)S3)------E11)R1)
c:  3
((((H1)S1)I1)S1)(((H1)S1)I1)S1)R1)
((((H1)S1)I1)S1)--((H1)S1)------R1)
H: 31
(R1------(P2(U1U1)(P7(U5U5)P7)(U1U1)P2)(U4U4)(P2(U2U2)(P4(U5U5)P4)(U1U1)P2)(U1U1)R1)

(R1(U5U5)(P4(U2U2)(P4(U1U1)P4)(U1U1)P4)(U5U5)---------(P3(U4U4)P3)---------(U1U1)R1)

W: 32
(R1(E5-(S2(I2(S7(H5H5)S7)I2)S2)(S2(I3(S4(H5H5)S4)I3)S2)E5)-R1)
(R1(E11(S4(I3(S4(H1H1)S4)I3)S4)------(S3(H4H4)S3)------E11)R1)
C: 3
(R(S(I(S(HH)S)I)S)(S(I(S(HH)S)I)S)R)
(R(S(I(S(HH)S)I)S)----(S(HH)----S)R)
```

Figure 9: Interactive sample session of `RNAdistance`. For this example we have used two random sequences folded by RNAfold.

Base pairing probability matrices may also be compared by an alignment-type method (Bonhoeffer *et al.* 1993). Since a secondary structure is representable as a string (see figure 9), comparison of structures can be done by standard string alignment algorithms (e.g. Waterman 1984). This approach has been generalized to structure ensembles in (Bonhoeffer *et al.* 1993). We compute for each position i of the sequence the probability to be upstream paired, downstream paired or unpaired.

$$\begin{aligned} p_i^{(} &= \sum_{j>i} p_{ij} \\ p_i^{)} &= \sum_{j<i} p_{ij} \end{aligned} \tag{7}$$

The probability that the base at position i is unpaired is $p_i^\circ = 1 - p_i^{(} - p_j^{)}$.

A reasonable definition for the distance of two such vectors, $p(\tilde{\mathcal{S}}_1)$ and $p(\tilde{\mathcal{S}}_2)$, uses again an alignment procedure at the level of the vectors $p^{(}$, $p^{)}$ and p°. We then define the distance measure for an aligned position (i,j) by

$$\sigma(i,j) = 1 - \sqrt{p_i^{)}(\tilde{\mathcal{S}}_1) p_j^{)}(\tilde{\mathcal{S}}_2)} + \sqrt{p_i^{(}(\tilde{\mathcal{S}}_1) p_j^{(}(\tilde{\mathcal{S}}_2)} + \sqrt{p_i^\circ(\tilde{\mathcal{S}}_1) p_j^\circ(\tilde{\mathcal{S}}_2)}. \tag{8}$$

and $\delta(i) = 0$ for inserted or deleted positions. The distance of two structure ensembles is given by the minimum total edit costs as in an ordinary string alignment (The numerical value of this distance is twice the distance measure defined in Bonhoeffer *et al.* 1993).

6. Applications

6.1. Long RNA Molecules

16sRNA, yeast.

Chain length: 1542 nucleotides.

Performance: 42min on an `IBM-RS6000/550` with 64 Mbyte.

Table 4: Free energies of 16sRNA from yeast. The phylogenetic structure decomposes into six components.

Bases			MFE	phylogenetic structure	
				completed	"as is"
1	–	556	−130.63	−97.76	−77.14
557	–	883	−89.18	−70.64	−61.31
884	–	920	−6.55	−6.55	−3.76
921	–	1396	−111.66	−68.70	−37.68
1397	–	1497	−21.00	−18.15	−18.15
1498	–	1542	−15.11	−14.11	−13.91
		Σ	−374.13	−275.91	−211.95
1	–	1542	−379.22	−279.48	−211.95

Remark. The fourth component, bases 921-1396, shows the largest deviations. This region contains large multiloops. It seems that in general local interactions are predicted more reliably than long range base pairs.

Minimum free energy structure contains 259 (58%) of the 441 base pairs predicted by a phylogenic structure (5 uncommon base pairs, and the 5 pairs forming a pseudo-knot are not counted). The cummulated pairing probability of the bases in the phylogenetic structure is about 54%. The cummulated pairing probability of the bases in the minimum free energy structure is 73%. The phlogenetic structure "as is" amounts to a free energy of −212 kcal/mol as compared to −379 kcal/mol for the minimum free energy structure. The difference of more than 300 RT indicates that the phylogenetic structure cannot be complete. We did a minimum free energy folding of the sequence subject to the base pairs prescribed by the phylogenetic structure using `RNAfold -C`. One finds -279.5 kcal/mol. This is still far away from the minimum free energy structure. The omitted interactions due to the pseudoknot and the uncommon base pairs cannot account for more than some 20 kcal/mol in the worst case. We are aware of the following possible explanations for the gap of about 100 RT:

- The energy model for secondary structures is **totally** wrong.
- 16sRNA is selected for its biochemical function which is not preformed by the free RNA, but by its complex with protein. The phylogenetically determined "structure" may then be completely different from the minimum free energy structure.

Qβ RNA.

Chain length: 4220 nucleotides.

Performance: 9.5h on an `IBM-RS6000/560` with 256 Mbyte.

Folding of long RNA sequences can be extended up to chain lengths of about 5 000 nucleotides. The entire Qβ-genome (Figure 10) was folded as a test case of an example of an entire viral genome. We do not imply that the real secondary structure of Qβ RNA is identical with its minimum free energy structure. Kinetic effects are highly important for long sequences.

The secondary structure obtained appears to be partitioned into three parts: (0-861), (862-3003), and (3004-4220). The middle part contains a large loop and other gross features which are also revealed by electron microscopy (Jacobson 1991).

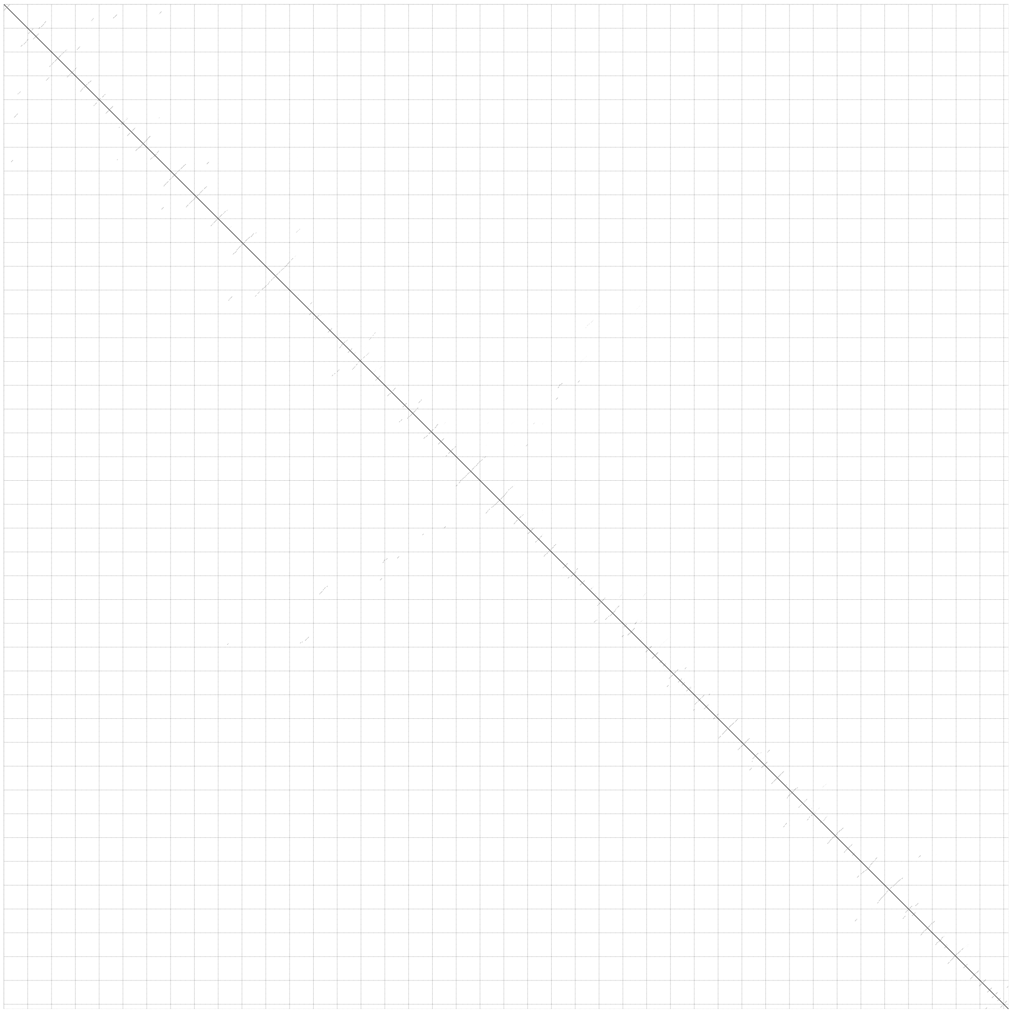


Figure 10: The base pairing probabilities for the entire Qβ genome (4420 nucleotides).

6.2. Heat Capacity of RNA Secondary Structures

The heat capacity can be obtained from the partition function using

$$C_p = -T\frac{\partial^2 G}{\partial T^2}, \qquad \text{and} \qquad \Delta G = -RT\ln Q \tag{9}$$

The numerical differentiation is performed in the following way: We fit the function $F(x)$ by the least square parabola $y = cx^2 + bx + a$ through the $2m+1$ equally spaced points $x_0 - mh$, $x_0 - (m-1)h$, …, x_0, …, $x_0 + mh$.

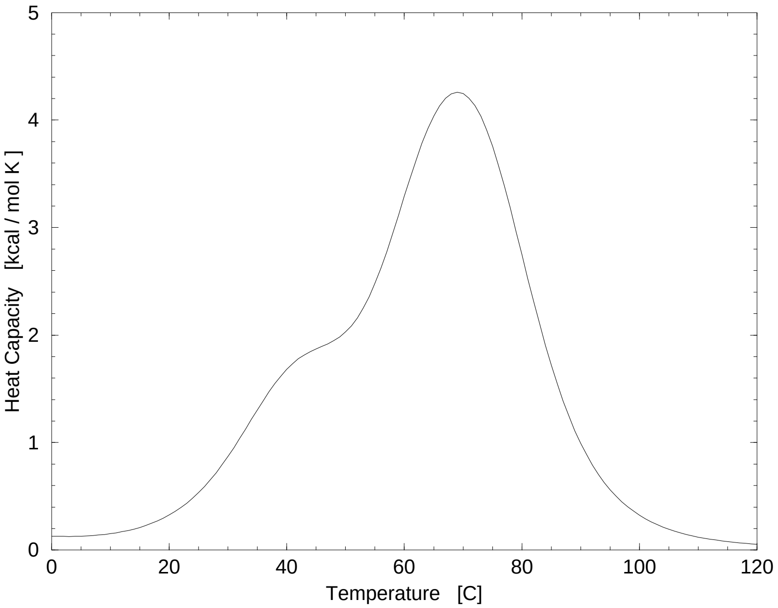


Figure 11: Heat Capacity of the tRNA-phe from yeast calculated using `RNAheat -Tmax 120 -h 0.1 -m5`.

The second derivative of F is then approximated by $F''(x_0) = 2c$. Explicitly, we obtain

$$F''(x_0) = \sum_{k=-m}^{m} \frac{30\left(3k^2 - m(m+1)\right)}{m(4m^2-1)(m+1)(2m+3)} F(x_0 + kh) \qquad (10)$$

As an example we show the heat capacity of tRNA$^t extphe$ from yeast (Figure 11).

6.3. Landscapes and combinatory maps

Statistical features of the relations between sequences, structures and other properties of RNA molecules were studied in several previous papers (Fontana *et al.* 1991, 1993a,b; Bonhoeffer *et al.* 1993, Tacker *et al.* 1993). We mention here free energy landscapes and structure maps as two representative examples. Relations between sequences **I** and derived objects $\mathcal{G}(\mathbf{I})$ (which may be numbers like free energies, or structures, or trees, or anything that might be relevant in studying RNA molecules) are considered as mappings from one metric space (the sequence space $\mathcal{X}$) into another metric space ($\mathcal{Y}$),

$$(\mathcal{X}; \mathbf{I}; d_h) \quad \Longrightarrow \quad (\mathcal{Y}; \mathbf{G}; d_g) \ . \tag{11}$$

The existence of a distance measure d_g inducing a metric on the set of objects $\mathcal{G}(\mathbf{I})$ is assumed. In the trivial case $\mathcal{Y}$ is $\mathbb{R}$. We are then dealing with a "landscape" which assigns a value to every sequence (a free energy, or an activation energy, for example). In general both spaces will be of combinatorial complexity and we coined the term "combinatory maps" for these mappings. In order to facilitate illustration and collection of data for statistical purposes a probability density surface is computed which expresses the conditional probability that two arbitrarily chosen sequences $\mathbf{I}_i$ $\mathbf{I}_j$ of Hamming distance $d_h(i,j) = h$ form two objects $\mathcal{G}_i$ and $\mathcal{G}_j$ at a distance $d_g(i,j) = \gamma$:

$$P_g(\gamma|h) \;=\; \text{Prob}\left(d_g(i,j) = \gamma \;\Big|\; d_h(i,j) = h\right) \ . \tag{12}$$

In the case of free energies we have $\gamma = |f_j - f_i|$, in case of the structure density surface (SDS) (Fontana *et al.* 1993b, Schuster *et al.* 1993) γ is the tree distance between the two structures obtained from the two sequences $\mathbf{I}_i$ and $\mathbf{I}_j$. An example of a typical probability density surface of RNA secondary structures is shown in figure 12. These structure density surfaces may be used to compute various quantities. Autocorrelation functions, for example, are obtained by a straightforward calculation (Fontana *et al.* 1993a,b).

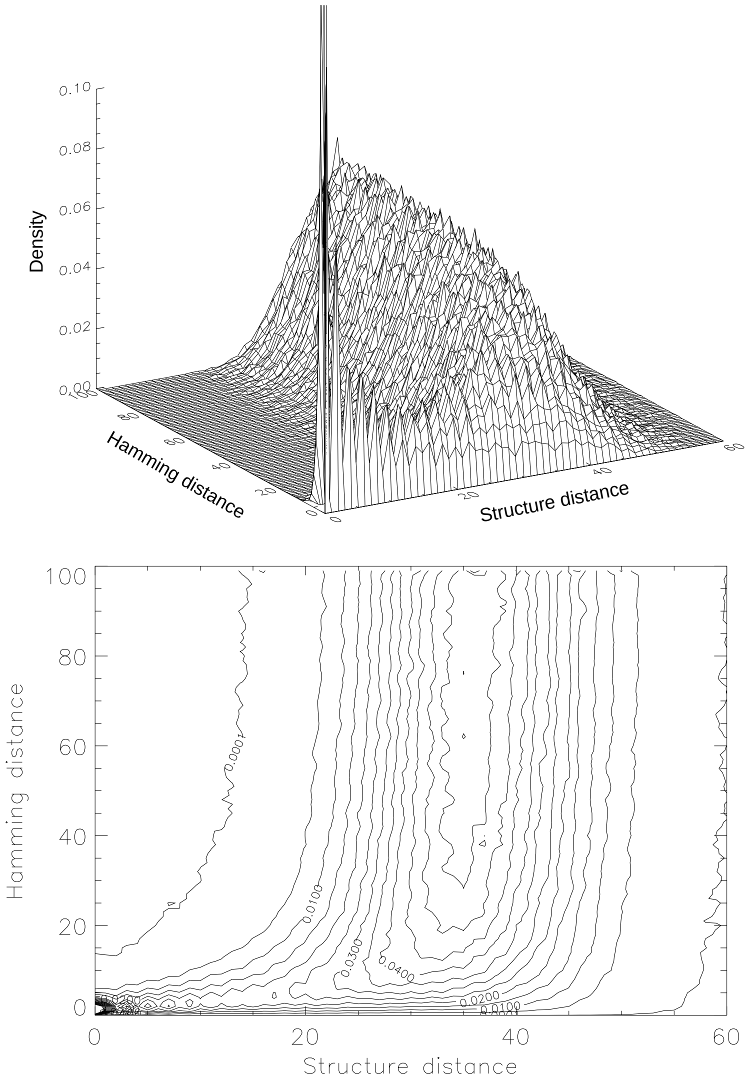


Figure 12: The structure density surface (SDS) for RNA sequences of length n=100 (upper). This surface was obtained as follows: (1) choose a random reference sequence and compute its structure, (2) sample randomly 10 different sequences in each distance class (Hamming distance 1 to 100) from the reference sequence, and bin the distances between their structures and the reference structure. This procedure was repeated for 1000 random reference sequences. Convergence is remarkably fast; no substantial changes were observed when doubling the number of reference sequences. This procedure conditions the density surface to sequences with base composition peaked at uniformity, and does, therefore, not yield information about strongly biased compositions.
Lower part: Contour plot of the SDS.

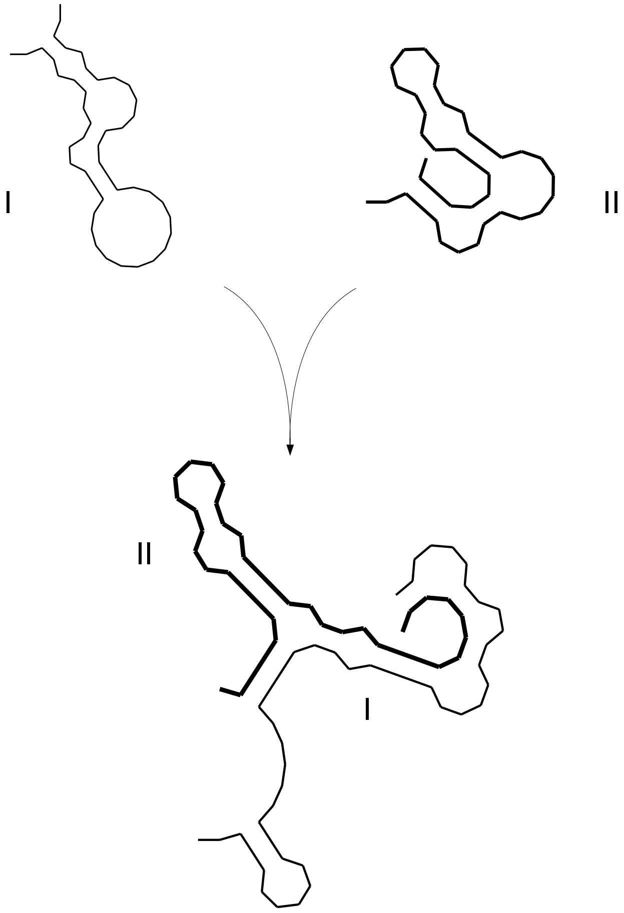


Figure 13: Schema of the folding of an RNA hybrid.

6.4. RNA Hybridisation

The folding algorithm can be generalized in a straightforward way to compute the structure resulting from the hybridisation of several RNA strands. This is done by concatenating the, say, N strands in all possible permutations. Any such permutation is folded almost as usual. The only difference concerns "loops" or a "stacked pair" which contain end-to-end junctions of individual strands, and which are treated as appropriate external elements. The structure with the least free energy among all optimal structures corresponding to the $N!$ sequence permutations is the optimal hybrid structure. In Figure 13 we show an example for $N = 2$.

6.5. Other applications

We mention investigations based on other criteria for RNA folding as, for example, maximum matching of bases or various kinetic folding algorithms which can be readily incorporated into the package. RNA melting kinetics has been studied as well by the statistical techniques mentioned here (Tacker *et al.* 1993).

In continuation of previous work (Fontana & Schuster 1987, Fontana *et al.* 1989) we started simulations experiments in molecular evolution which aim at a better understanding of evolutionary optimization processes on a molecular scale. Studies of this kind are becoming relevant for the design of efficient experiments in applied molecular evolution (Eigen & Gardiner 1984, Kauffman 1992).

Acknowledgements

The work reported here was supported financially in part by the Austrian *Fonds zur Förderung der wissenschaftlichen Forschung* (Project No. S 5305-PHY and Project No. P 8526-MOB). The Institut für Molekulare Biotechnologie e.V. is financed by the *Thüringer Ministerium für Wissenschaft und Kunst* and by the German *Bundesministerium für Forschung und Technologie.* The Santa Fe Institute is sponsored by core funding from the *U.S. Department of Energy* (ER-FG05-88 ER 25054) and the *National Science Foundation* (PHY-8714918). Computer facilities were supplied by the computer centers of the Technical University and the University of Vienna.

References

Bandelt, H.-J. & A.W.M.Dress. (1992) Canonical Decomposition Theory of Metrics on Finite Sets. *Adv.Math.* **92**, 47-105.

Beaudry, A.A. & G.F.Joyce. (1992) Directed evolution of an RNA enzyme. *Science* **257**, 635-641.

Bonhoeffer S., J.S.McCaskill, P.F.Stadler & P. Schuster. (1993) Temperature dependent RNA landscapes. A study based on partition functions. *Eur.Biophys.J.* **22**, 13-24.

Cech, T.R. & B.L.Bass. (1986) Biological catalysis by RNA. *Ann.Rev.Biochem.* **55**, 599-629.

Eigen, M. & W.Gardiner. (1984) Evolutionary molecular engineering based on RNA replication. *Pure & Appl.Chem.* **56**, 967-978.

Eigen, M., R.Winkler-Oswatitsch & A.Dress. (1988) Statistical geometry in sequence space: A method of quantitative comparative sequence analysis. *Proc.Natl.Acad.Sci.USA* **85**, 5913-5917.

Ellington, A.D. & J.W.Szostak. (1990) *In vitro* selection of RNA molecules that bind to specific ligands. *Nature* **346**, 818-822.

Fontana W. & P.Schuster. (1987) A computer model of evolutionary optimization. *Biophys.Chem.* **26**, 123-147.

Fontana W., W.Schnabl & P.Schuster. (1989) Physical aspects of evolutionary optimization and adaptation. *Phys.Rev.A* **40**, 3301-3321.

Fontana W., T.Griesmacher, W.Schnabl, P.F.Stadler & P.Schuster. (1991) Statistics of landscapes based on free energies, replication and degradation rate constants of RNA secondary structures. *Mh.Chem.* **122**, 795-819.

Fontana W., P.F.Stadler, E.Bornberg-Bauer, T.Griesmacher, I.L.Hofacker, M.Tacker, P.Tarazona, E.D.Weinberger & P.Schuster. (1993a) RNA folding and combinatory landscapes. *Phys.Rev.E* **47**, 2083-2099.

Fontana W., D.A.M.Konings, P.F.Stadler & P.Schuster. (1993b) *Statistics of RNA secondary structures. Biopolymers*, in press.

Freier, S.M., R.Kierzek, J.A.Jaeger, N.Sugimoto, M.H.Caruthers, T.Neilson & D.H.Turner. (1986) Improved free-energy parameters for predictions of RNA duplex stability. *Proc.Natl.Acad.Sci.USA* **83**, 9373-9377.

Hofacker I.L., P.Schuster & P.F.Stadler. (1993) *Combinatorics of RNA Secondary Structures.* Preprint.

Hogeweg, P. & B.Hesper. (1984) Energy Directed Folding of RNA Sequences. *Nucleic Acids Res.* **12**, 67-74.

Jacobson, A.B. (1991) Secondary structure of Coliphage Qβ RNA. Analysis by electron microscopy. *J.Mol.Biol.* **221**, 557-570.

Jaeger J.A., D.H.Turner & M.Zuker. (1989) Improved predictions of secondary structures for RNA. *Proc.Natl.Acad.Sci.USA* **86**, 7706-7710.

Kauffman, S.A. (1992) Applied molecular evolution. *J.Theor.Biol.* **157**, 1-7.

Konings, D.A.M. (1989) *Pattern analysis of RNA secondary structure.* Proefschrift, Rijksuniversiteit te Utrecht.

Konings, D.A.M. & P.Hogeweg. (1989) Pattern analysis of RNA secondary structure. Similarity and consensus of minimal-energy folding. *J.Mol.Biol.* **207**, 597-614.

Martinez, H.M. (1984) An RNA Folding Rule. *Nucleic Acids Res.* **12**, 323-334.

McCaskill, J.S. (1990) The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers* **29**, 1105-1119.

Nussinov, R., G.Pieczenik, J.R.Griggs & D.J.Kleitman. (1978) Algorithms for loop matching. *SIAM J.Appl.Math.* **35**, 68-82.

Nussinov, R. & A.B.Jacobson. (1980) Fast algorithm for predicting the secondary structure of single-stranded RNA. *Proc.Natl.Acad.Sci.USA* **77**, 6309-6313.

Sankoff, D. & J.B.Kruskal, eds. (1983) *Time warps, string edits and macromolecules: The theory and practice of of sequence comparison.* Addison Wesley, London.

Schuster, P., W.Fontana, P.F.Stadler & I.L.Hofacker. (1993) *The distribution of RNA structures over sequences.* Preprint.

Shapiro, B. (1988) An algorithm for comparing multiple RNA secondary structures. *CABIOS* **4**, 387-393.

Shapiro, B. & K.Zhang. (1990) Comparing multiple RNA secondary structures using tree comparisons. *CABIOS* **6**, 309-318.

Symons, R.H. (1992) Small catalytic RNAs. *Ann.Rev.Biochem.* **61**, 641-671.

Tacker, M., W.Fontana, P.F.Stadler & P.Schuster. (1993) *Statistics of RNA melting kinetics.* Preprint.

Tai, K. (1979) The Tree-to-tree Correction Probelem. *J.ACM* **26**, 422-433.

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

Waterman, M.S. (1984) General Methods of Sequence Comparison. *Bull.Math.Biol.* **46**, 473-500.

Waterman, M.S. & T.F.Smith. (1978) RNA secondary structure: A complete mathematical analysis. *Math.Biosc.* **42**, 257-266.

Zuker, M. & P.Stiegler. (1981) Optimal computer folding of large RNA sequences using thermodynamic and auxilary information. *Nucleic Acids Res.* **9**, 133-148.

Zuker, M. & D.Sankoff. (1984) RNA secondary structures and their prediction. *Bull.Math.Biol.* **46**, 591-621.

Zuker, M. (1989) On finding all suboptimal foldings of an RNA molecule. *Science* **244**, 48-52.

Appendix A: Edit Distances and Edit Costs

Let x, y, z be rooted ordered plane trees with labelled and weighted nodes. The labels are taken from some alphabet $\mathcal{A}$, the weights are non-negative real numbers. Denote the set of all such trees by $\mathcal{T}_{\mathcal{A}}$. (For the relation of trees and secondary structures see figure 5 and appendix B.)

We define *elementary* edit costs δ_a for insering or deleting a vertex with label $a \in \mathcal{A}$ and $\sigma_{ba} = \sigma_{ba}$ for replacing a vertex with label a by a vertex with label b. We require

$$\begin{array}{lllll} \delta_a & > & 0 \quad \text{and} \quad \sigma_{aa} = 0 & \forall & a \in \mathcal{A} \\ \sigma_{ab} & > & 0 & \forall & a \neq b \\ \sigma_{ab} & \leq & \sigma_{ac} + \sigma_{cb} & \forall & a, b, c \in \mathcal{A} \end{array} \tag{A.1}$$

The last line implies, that if substitution of a and b is allowed, than its cost is at most the cost for first deleting a and then inserting b.

Denote a vertex with label $a \in \mathcal{A}$ and weight $v \geq 0$ by $[a, v]$. We then define the *weighted* edit costs:

$$\begin{aligned} \Delta([a,v]) &= v\delta_a \\ \Sigma([a,v],[b,w]) &= \min(v,w)\sigma_{ab} + |v-w| \begin{cases} \delta_a & \text{if} \quad v \geq w \\ \delta_b & \text{if} \quad v \leq w \end{cases} \end{aligned} \tag{A.2}$$

for insertion/deletion and substitution, respectively. The tree edit distance $d_{\mathcal{T}}(x, y)$ of any two trees $x, y \in \mathcal{T}_{\mathcal{A}}$ is defined as the minimum total edit cost needed to transform x into y, such that the partial order of the tree remains untouched. For a description of tree editing in general, and of its implementation as a dynamic programming algorithm see e.g. Tai (1979) and Shapiro (1990). It is easy to see that $d_{\mathcal{T}}$ is a metric on $\mathcal{T}_{\mathcal{A}}$ for any alphabet $\mathcal{A}$.

Any tree $x \in \mathcal{T}_{\mathcal{A}}$ can be transformed into a linear parenthesized representation by appropriately traversing the tree. (A leaf is represented by both an open and closed parentehsis.) This representation is string-like in the sense that one may interpret each parenthesis as a generalized character j of a string $\tilde{x}$. Each character is defined by its label, its weight, and its "sign", that is, whether it is an open or a closed parenthesis, $\xi = [a, v, p]$ with $a \in A$, $v \geq 0$, and $p =$'(' or ')'. Its is easy to see that such a generalized string can be represented as a tree, provided it contains only matching parentheses and weigths and labels coincide for pairs of matching parentheses.

We define the generalized string edit distance $d_s(\tilde{x}, \tilde{y})$ of two generalized strings $\tilde{x}$ and $\tilde{y}$ as the minimum total edit cost needed to transform $\tilde{x}$ into $\tilde{y}$, where the edit costs for each step are given by

$$\begin{aligned} \tilde{\Delta}([a,v,p]) &= \Delta([a,v]) \\ \tilde{\Sigma}([a,v,p],[b,w,q]) &= \begin{cases} \Sigma([a,v],[b,w]) & \text{if} \quad p = q \\ \infty & \text{if} \quad p \neq q \end{cases} \end{aligned} \tag{A.3}$$

for insertion/deletion and substitution, respectively. This is to say that open parentheses may be matched with open parentheses only, and closed parentheses with closed parentheses only.

Corresponding to the above two distance measures we have two types of alignments: In case of the edit distance for generalized strings it is straight forward. For the tree edit distance we have to transform the aligned trees into their linear representation in order to print them. This implies that each inserted or deleted node is complemented by gap characters in *two* positions, corresponding to the left and right bracket, respectively, and that matched nodes give rise to two matches in the linear representation. (See figure 9 for examples). As a consequence, each alignment of two trees x and y can be written as a valid alignment of the corresponding generalized string $\tilde{x}$ and $\tilde{y}$ exhibiting the same edit cost. Hence

$$d_s(\tilde{x}, \tilde{y}) \leq d_{\mathcal{T}}(x, y) \qquad \text{for all } x, y \in \mathcal{T}_{\mathcal{A}} \tag{A.4}$$

In particular, let x be a subtree of y. Then $d_s(\tilde{x}, \tilde{y}) = d_{\mathcal{T}}(x, y)$. As a consequence there are arbitrarily large trees x and y such that the two distance measures coincide. The cost matrix we use is:

$$\left(\begin{array}{ccccccccccc}
\emptyset & U & P & H & B & I & M & S & E & R & \\
0 & 1 & 2 & 2 & 2 & 2 & 2 & 1 & 1 & \infty & \emptyset \\
1 & 0 & 1 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & U \\
2 & 1 & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & P \\
2 & \infty & \infty & 0 & 2 & 2 & 2 & \infty & \infty & \infty & H \\
2 & \infty & \infty & 2 & 0 & 1 & 2 & \infty & \infty & \infty & B \\
2 & \infty & \infty & 2 & 1 & 0 & 2 & \infty & \infty & \infty & I \\
2 & \infty & \infty & 2 & 2 & 2 & 0 & \infty & \infty & \infty & M \\
1 & \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty & \infty & S \\
1 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty & E \\
\infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & R
\end{array}\right)$$

Appendix B: Representation of Secondary Structures

The simplest way of representing a secondary structure is the "parenthesis format", where matching parentheses symbolize base pairs and unpaired bases are shown as dots. Alternatively, one may use two types of node labels, 'P' for paired and 'U' for unpaired; a dot is then replaced by '(U)', and each closed bracket is assigned an additional identifier 'P'. In (Fontana *et al.* 1993b) a condensed representation of the secondary structure is proposed, the so-called *homeomorphically irreducible tree* (HIT) representation. Here a stack is represented as a single pair of matching brackets labelled 'P' and weighted by the number of base pairs. Correspondingly, a contiguous strain of unpaired bases is shown as pair of matching bracket labelled 'U' and weighted by its length.

Bruce Shapiro (1988) proposed another representation, which, however, does not retain the full information of the secondary structure. He represents the different structure elements by single matching brackets and labels them as 'H' (hairpins loop), 'I' (interior loop), 'B' (bulge), 'M' (multi-loop), and 'S'

```
a) .((((..(((...)))..((..)))).)).
   (U)(((((U)(U)((((U)(U)(U)P)P)P)(U)(U)(((U)(U)P)P)P)P)(U)P)P)(U)
b) (U)(((U2)((U3)P3)(U2)((U2)P2)P2)(U)P2)(U)
c) (((H)(H)M)B)
   ((((((H)S)((H)S)M)S)B)S)
   (((((((H)S)((H)S)M)S)B)S)E)
d) (((((((H3)S3)((H2)S2)M4)S2)B1)S2)E2)
e) ((H)(H)M)

a) (UU)(P(P(P(P(UU)(UU)(P(P(P(UU)(UU)(UU)P)P)P)(UU)(UU)(P(P(UU)(U...
b) (UU)(P2(P2(U2U2)(P2(U3U3)P3)(U2U2)(P2(U2U2)P2)P2)(UU)P2)(UU)
c) (B(M(HH)(HH)M)B)
   (S(B(S(M(S(HH)S)(S(HH)S)M)S)B)S)
   (E(S(B(S(M(S(HH)S)(S(HH)S)M)S)B)S)E)
d) (E2(S2(B1(S2(M4(S3(H3)S3)((H2)S2)M4)S2)B1)S2)E2)
e) (M(HH)(HH)M)
```

Figure B.1: Linear representations of secondary structures used by the `Vienna RNA package`.

Above: Tree representations of secondary structures. a) Full structure: the first line shows the more convenient condensed notation which is used by our programs; the second line shows the rather clumsy expanded notation for completeness, b) HIT structure, c) different versions of coarse grained structures: the second line is exactly Shapiro's representation, the first line is obtained by neglecting the stems. Since each loop is closed by a unique stem, these two lines are equivalent. The third line is an extension taking into also the external digits. d) weighted coarse structure, e) branching structure.

Below: The corresponding tree in the notation used for the output of the string-type alignments.

Virtual roots are not shown here. The program RNAdistance accepts any of the above tree representations, the string-type representations occur on output only.

(stack). We extend his alphabet by an extra letter for external elements 'E'. Again the corresponding trees may be weighted by the number of unpaired bases or base pairs constituting them. An even more coarse grained representation considers the branching structure only. It is obtained by considering the hairpins and multiloops only (Hofacker *et al.* 1993). All tree representations (except for the condensed dot-bracket form) can be encapsulated into a virtual root (labeled 'R'), see also the example session in figure 6.

In order to discriminate the alignments produced by tree-editing from the alignment obtained by generalized string editing we put the label on both sides in the latter case.

Appendix C: List of Executables

```
RNAfold [-p[0]] [-T temp] [-4] [-noGU] [-e e_set] [-C]

RNAdistance [-D[fhwcFHWC]] [-X[p|m|f|c]] [-S] [-B [file]]

RNApdist [-Xpmfc] [-B [file]][-T temp] [-4] [-noGU] [-e e_set]

RNAheat [-Tmin t1] [-Tmax t2] [-h stepsize] [-m ipoints] [-4] [-noGU] [-e e_set]

RNAinverse [-F[mp]] [-a ALPHABET] [-R [repeats]] [-T temp] [-4] [-noGU] [-e e_set]

RNAeval [-T temp] [-4] [-e e_set]
```

Appendix D: Availability of Vienna RNA Package

Vienna RNA package is public domain software. It can be obtained by anonymous ftp from the server `ftp.itc.univie.ac.at`, directory `/pub/RNA` as `ViennaRNA-1.03.tar.Z`. The package is written in ANSI C and is known to compile on SUN SPARC Stations, IBM-RS/6000, HP-720, and Silicon Graphics Indigo. For comments and bug reports please send e-mail messages to `ivo@itc.univie.ac.at`.

The parallelized version of the minimum free energy folding is not part of the package. It can be obtained upon request from the authors.