ALGORITHMS AND THERMODYNAMICS FOR RNA SECONDARY STRUCTURE PREDICTION: A PRACTICAL GUIDE.

A. M. ZUKER
Institute for Biomedical Computing
Washington University
St. Louis, MO 63110

AND

B. D.H. MATHEWS & C. D.H. TURNER
Department of Chemistry
University of Rochester
Rochester, NY 14627-0216

Abstract

This article is about the current status of the *mfold* package for RNA and DNA secondary structure prediction using nearest neighbor thermodynamic rules. The details of the free energy rules and of the latest version 3.0 software are described. Future plans are also discussed.

The *mfold* software now runs on a variety of Unix platforms; specifically SGI Irix, Sun Solaris, Dec alpha Ultrix and on Linux. While the older interactive programs of version 2 still exist, they are now run by a variety of scripts that make for much easier handling. There is both a command line interface for *mfold* and an HTML interface that runs in a Unix environment but can be accessed by anyone with a web browser.

The thermodynamic basis for the folding model is presented in detail, with references given for both specific free energy parameters and to overview articles that have summarized the state of these nearest neighbor parameters over the past dozen years. Both RNA and DNA rules are discussed, with some mention of parameters for RNA/DNA hybridization. Although the thermodynamic model has grown in complexity to accommodate new types of information, the folding algorithm has not yet incorporated some features, such as coaxial stacking of adjacent helices, and other features will probably remain too difficult or computationally expensive to implement. For this reason, a new energy calculation program has been introduced to recompute the free energies of predicted foldings to reflect the best of our knowledge.

The most significant improvements in the *mfold* software are:

1. Folding times have been greatly reduced in recent years, partly because of

faster computers and partly because of improvements in the algorithm.

2. Folding constraints have been expanded and are now implemented without the use of *bonus energies* that distort the results.

3. The output is significantly improved. Clear and enlargeable images of *dot plots* and of predicted foldings are now produced in PostScript and *gif* formats. Bases in structures may be annotated using different colors that reflect how *well-determined* they are in terms of their tendency to pair with other bases or to be single-stranded. Similarly, base pair probabilities from partition function calculations may be used for annotation. A detailed decomposition of each predicted folding into stacked pairs and loops with associated free energies is now provided.

The *mfold* software has a variety of parameters that may be adjusted to improve the predictions. Several examples are presented to illustrate how to interpret folding results and how to adjust these parameters to obtain better results.

1 Introduction

RNA is ubiquitous in the cell and is important for many processes. The activity of RNA is determined by its structure, the way it is folded back on itself. Secondary structure modeling of RNA predicts, or otherwise determines, the pattern of Watson-Crick (WC), wobble and other, non-canonical pairings that occur when the RNA is folded. For the purposes of this article, a non-canonical base pair is defined as non Watson-Crick (non W-C) and non-wobble (not GU or UG). There are many different kinds of RNA. Ribosomal RNA (rRNA) is a crucial part of the ribosome which is found in all living cells and in organelles such as mitochondria and chloroplasts. Small nuclear RNAs (snRNA) form a vital part of sliceosomes that process mRNAs in eukaryotes. These are 2 examples of *structural* RNAs.

Messenger RNAs (mRNA) do more than just carry information. Secondary structures can be used in part to explain translational controls in mRNA [1, 2], and replication controls in single-stranded RNA viruses [3]. Although the vast majority of known mRNAs code for proteins or structural RNAs, some do not [4, 5]; and it is likely that the secondary structures of these transcripts play an important role in their regulatory function in the cell. It is to be expected that many more such functional RNAs will be discovered in the future.

RNA is not just a passive structural element or a regulator. It is also an active component in many situations. Thus RNA acting alone is able to catalyze RNA processing [6, 7]. In a protein-RNA complex, the RNA component of ribonuclease P is an active component of tRNA processing [8].

The function of an RNA can only be understood in terms of its secondary or tertiary structure. For the understanding of catalytic activity, knowledge of secondary structure alone is insufficient. However, few large structures have been determined by crystallography [9, 10, 11] and the need for modeling is great. Secondary structure modeling can reasonably be viewed as a first step towards three dimensional modeling. For example, in small and large subunit rRNA, all tertiary

TABLE 1: *mfold* runs on a variety of Unix platforms. The ones shown have been tested by the first author.

Operating System	Hardware
Irix	SGI
Solaris	Sun SPARC
Ultrix	Dec Alpha
Solaris	Intel
Linux	Intel

interactions, including base triples, involve only 3% and 2% of the nucleotides, respectively [12]. In contrast, nucleotides in secondary structure comprise 60% and 58% of these rRNAs.

Secondary structure modeling is therefore a significant first step to the far more difficult process of three dimensional atomic resolution modeling. Knowledge of secondary structure, together with additional information on structural constraints or tertiary interactions, can be used to construct atomic resolution structural models [13, 14, 15].

2 Software platforms and environment

The *mfold* package [16, 17, 18, 19] consists of a group of programs written in Fortran or C, and a group of Bourne shell or Perl scripts. All of these programs and scripts run in a Unix environment. The *mfold* software is currently running on a number of different platforms, as indicated in Table 1.

In addition, some of the core programs of *mfold* have been translated into C++ and the resulting software is running on Windows 98/Windows NT (Intel hardware) with a point and click interface that uses proprietary Microsoft windowing tools. This program is called *RNAstructure*[20].

Up to version 2.2, the programs in *mfold* were interactive and had to be run individually. Versions 2.3 and 3.0 provide a simple command line interface, as described below, although it is still possible to run individual programs interactively or, for that matter, to write one's own scripts to link programs together in novel ways. The RNA folding web server (`www.ibc.wustl.edu/~zuker/rna/form1.cgi`) provides an HTML interface to *mfold* that uses a similar, but not identical script. This server will be described elsewhere. It's main advantage is that anyone with a web browser can use it.

An early version of *mfold* (2.0) was ported to run on Mac computers. It lacked the *energy dot plot* and could only fold about 350 bases. We no longer recommend its further use, especially since the energy parameters it uses have not been updated. The current *mfold* version 3.0 runs in command line mode under Windows, but this implementation requires a type of Unix emulation program to be run.

Two Unix "environment variables" must be defined for *mfold* to function. The first, MFOLDBIN, defines the directory where all the executable files for *mfold* are stored. The second, MFOLDLIB, defines the directory containing all the free energy and other data files. Dynamic memory allocation is not yet available for *mfold* , and the largest fragment size that can be folded is defined by the value of the MAXN parameter in the "Makefile" file.

3 Loops and Nearest neighbor rules

The *mfold* software uses what are called *nearest neighbor* energy rules. That is, free energies are assigned to loops rather than to base pairs. These have also been called loop dependent energy rules. In an effort to keep this article as self-contained as possible, we are including some well-known definitions that may be found elsewhere [21, 22, 23].

A secondary structure, $\mathbf{S}$, on an RNA sequence, $\mathbf{R} = r_1, r_2, r_3, \ldots, r_n$, is a set of *base pairs*. A base pair between nucleotides r_i and r_j $(i < j)$ is denoted by $i.j$. A few constraints are imposed.

- Two base pairs, $i.j$ and $i'.j'$ are either identical, or else $i \neq i'$ and $j \neq j'$. Thus base triples are deliberately excluded from the definition of secondary structure.
- Sharp U-turns are prohibited. A U-turn, called a hairpin loop, must contain at least 3 bases.
- Pseudoknots are prohibited. That is, if $i.j$ and $i'.j' \in \mathbf{S}$, then, assuming $i < i'$, either $i < i' < j' < j$ or $i < j < i' < j'$.

Pseudoknots [24, 25, 26, 27, 28, 29] and base triples are not excluded for frivolous reasons. When pseudoknots are included, the loop decomposition of a secondary structure breaks down and the energy rules break down. Although we can assign reasonable free energies to the helices in a pseudoknot, and even to possible coaxial stacking between them, it is not possible to estimate the effects of the new kinds of loops that are created. Base triples pose an even greater challenge, because the exact nature of the triple cannot be predicted in advance, and even if it could, we have no data for assigning free energies.

A base $r_{i'}$ or a base pair $i'.j'$ is called accessible from a base pair $i.j$ if $i < i'(< j') < j$ and if there is not other base pair, $k.l$ such that $i < k < i'(< j') < l < j$. The collection of bases and base pairs accessible from a given base pair, $i.j$, but **not** including that base pair, is called the loop closed by $i.j$. We denote it by $\mathbf{L}(i.j)$. The collection of bases and base pairs not accessible from any base pair is called the exterior (or external) loop, and will be denoted by $\mathbf{L}_e$ here. It is worth noting that if we imagine adding a 0^{th} and an $(n+1)^{st}$ base to the RNA, and a base pair $0.(n{+}1)$, then the exterior loop becomes the loop closed by this imaginary base pair. We call this the *universal closing base pair* of an RNA structure. If $\mathbf{S}$ is a secondary structure, then $\mathbf{S}'$ denotes the same secondary structure with the addition of the universal closing base pair. The exterior loop exists only in linear RNA. It is treated differently than other loops because we assume as a first approximation that there are no conformational constraints, and therefore no associated entropic costs.

Any secondary structure, **S**, decomposes an RNA uniquely into loops. We can write this as:

$$\mathbf{R} = \bigcup_{i.j \in \mathbf{S}'} \mathbf{L}(i.j)$$

Loops may contain 0, 1 or more base pairs. The term k-loop denotes a loop containing $k-1$ base pairs, making a total of k base pairs by including the closing base pair. We introduce the terms $l_s(\mathbf{L})$ and $l_d(\mathbf{L})$ to denote the number of single-stranded bases and base pairs in a loop, respectively. The size of a 1 or 2-loop is defined as $l_s(\mathbf{L})$.

A 1-loop is called a hairpin loop. Polymer theory predicts that the free energy increment, $\delta\delta G$, for such a loop is given by

$$\delta\delta G = 1.75 \times RT \times \ln(l_s), \qquad (1)$$

where T is absolute temperature and R is the universal gas constant (1.9872 cal/mol/K). The factor 1.75 would be 2 if the chain were not self-avoiding in space. In reality, we use tabulated values for $\delta\delta G$ for l_s from 3 to 30. These values are based on measurements and interpolations of measurements, and are stored in an file named loop.dg, or loop.TC, where TC is a temperature (integral) in °C. We use the latter only when departing from our temperature standard of 37 degrees. Thus loop.dg and loop.37 refer to the same file. The same convention holds for other files defined below. Equation 1 is used to extrapolate beyond size 30. Thus, for $l_s > 30$,

$$\delta\delta G = \delta\delta G_{30} + 1.75 \times RT \times \ln(l_s/30). \qquad (2)$$

Figure 1 shows the information stored in the loop file.

In addition, the effects of *terminal mismatched pairs* are taken into account for hairpin loops of size greater than 3. For loops of size 4 and greater closed by a base pair $i.j$, an extra $\delta\delta G$ is applied. This is referred to as the *terminal mismatch* free energy for hairpin loops. These parameters are stored in a file named tstackh.dg or tstackh.TC, as above. The data are arranged in 4×4 tables that each comprise 4 rows and columns. Figure 2 illustrates how the parameters are stored.

Both the loop and tstackh files treat hairpin loops in a generic way, and assume no special structure for the bases in the loop. We know that this is not true in general. For example, the anti-codon loop of tRNA is certainly not unstructured. For certain small hairpin loops, special rules apply. Hairpin loops of size 3 are called triloops and those of size 4 are called tetraloops. Files of *distinguished* triloops and tetraloops have been created to store the free energy bonus assigned to those loops. These parameters are stored in files triloop.dg and tloop.dg, respectively (or triloop.TC and tloop.TC for a specific temperature, TC). Some typical entries are given in Figure 3

Finally, there are some special hairpin loop rules derived from experiments that will be defined explicitly here. A hairpin loop closed by r_i and r_j $(i < j)$ called a "GGG" loop if $r_{i-2} = r_{i-1} = r_i = G$ and $r_j = U$. Such a loop receives a free energy bonus that is stored in the miscloop.dg or miscloop.TC file, which contains

```
DESTABILIZING ENERGIES BY SIZE OF LOOP (INTERPOLATE WHERE NEEDED)
hp3 ave calc no tmm;hp4 ave calc with tmm; ave all bulges
SIZE         INTERNAL          BULGE            HAIRPIN
-------------------------------------------------------
1               .               3.8               .
2               .               2.8               .
3               .               3.2              5.6
4              1.7              3.6              5.5
5              1.8              4.0              5.6
6              2.0              4.4              5.3
7              2.2              4.6              5.8
8              2.3              4.7              5.4
 ...
30             3.7              6.1              7.7
```

Figure 1: The loop.dg or loop.TC contains size based free energy increments for hairpin, bulge and interior loops up to size 30. Entries with '.' are undefined.

a variety of miscellaneous, or extra free energy parameters. Another special case is the "poly-C" hairpin loop, where all the single stranded bases are C. If the loop has size 3, it is given a free energy penalty of $c3$. Otherwise, the penalty is $c_2 + c_1 \times l_s$. The constants c_1, c_2 and c_3 are all stored in the miscloop file.

To summarize, we can write the free energy, $\delta\delta G_H$ of a hairpin loop as:

$$\delta\delta G_H = \delta\delta G_H^1 + \delta\delta G_H^2 + \delta\delta G_H^3 + \delta\delta G_H^4, \tag{3}$$

where

1. $\delta\delta G_H^1$ is the size dependent contribution from the loop file, or from equation 2 for sizes > 30,

```
          5' --> 3'                 5' --> 3'
             WX                        CX
             ZY                        GY
          3' <-- 5'                 3' <-- 5'
    Y: A    C    G    U         A     C     G     U
      -------------------       ------------------
X:A | aAA  aAC  aAG  aAU       -1.5 -1.5 -1.4 -1.8
  C | aCA  aCC  aCG  aCU       -1.0 -0.9 -2.9 -0.8
  G | aGA  aGC  aGG  aGU       -2.2 -2.0 -1.6 -1.2
  U | aUA  aUC  aUG  aUU       -1.7 -1.4 -1.9 -2.0
```

Figure 2: On the left, a typical 4×4 table. The pairs WX and YZ are covalently linked. WZ is assumed to be the closing base pair of a hairpin loop, and XY is the mismatched pair. 'X' refers to row , and 'Y' to column, in order A, C, G and U. Thus 'aGU' is the same as 'a34' and is the mismatch free energy for a GU mismatch (X=G and Y=U). On the right is a sample table for W=C and Z=G.

```
Seq     Energy
-------------
 GGGGAC -3.0
   ...
 CGAAGG -2.5
 CUACGG -2.5
   ...
 GUGAAC -1.5
 UGGAAA -1.5
```

Figure 3: Sample *distinguished* tetraloops together with the free energy bonuses, in kcal/mole, attached to them. These entries include the closing base pair of the loop. Triloops are not shown since they are not currently in use for RNA folding.

2. $\delta\delta G_H^2$ is the terminal mismatch stacking free energy, taken from the tstackh file (0 for hairpin loops of size 3),

3. $\delta\delta G_H^3$ is the bonus free energy for triloops or tetraloops listed in the TRILOOP or TLOOP files. This value is 0 for loops not listed in the TRILOOP or TLOOP files and for loop sizes > 4,

4. $\delta\delta G_H^4$ is the bonus or penalty free energy for special cases not covered by the above.

A 2-loop, $\mathbf{L}$, is closed by a base pair $i.j$ and contains a single base pair, $i'.j'$, satisfying $i < i' < j' < j$. In this case, the loop size, $l_s(\mathbf{L})$, can be written as:

$$l_s(\mathbf{L}) = l_s^1(\mathbf{L}) + l_s^2(\mathbf{L}),$$

where $l_s^1(\mathbf{L}) = i' - i - 1$ and $l_s^2(\mathbf{L}) = j - j' - 1$.

A 2-loop of size 0 is called a *stacked pair*. This refers to the stacking between the $i.j$ and immediately adjacent $i+1.j-1$ base pair contained in the loop. Free energies for these loops are stored in a file named stack.dg, or stack.TC, where TC is a temperature, as defined above. The layout is the same as for the tstackh file. A portion of such a file is given in Figure 4. A group of 2 or more consecutive base pairs is called a *helix*. The first and last are the closing base pairs of the helix. They may be written as $i.j$ and $i'.j'$, where $i < i' < j' < j$. Then $i.j$ is called the external closing base pair and $i'.j'$ is called the internal closing base pair. This nomenclature is used for circular RNA as well, even though it depends on the choice of origin.

Only Watson-Crick and wobble GU pairs are allowed as *bona fide* base pairs, even though the software is written to allow for any base pairs. The reason is that nearest neighbor rules break down for non-canonical, even GU base pairs, and that mismatches must instead be treated as small, symmetric interior loops.

```
       5' --> 3'
          CX
          GY
       3' <-- 5'
  Y:  A    C    G    U
 ----------------------
X:A |  .    .    .   -2.1
  C |  .    .   -3.3   .
  G |  .   -2.4   .   -1.4
  U | -2.1   .   -2.1   .
```

Figure 4: Sample free energies in kcal/mole for CG base pairs stacked over all possible base pairs, XY. X refers to row and Y refers to column, in the order A, C, G and U respectively. Entries denoted by an isolated period, '.', are undefined, and may be considered as $+\infty$.

Note that the stacks $\begin{array}{ccc} 5' & --> & 3' \\ & WX & \\ & ZY & \\ 3' & <-- & 5' \end{array}$ *and* $\begin{array}{ccc} 5' & --> & 3' \\ & YZ & \\ & XW & \\ 3' & <-- & 5' \end{array}$ are identical, and yet formally different for $W \neq Y$ and $X \neq Z$. These stacked pairs are stored twice in the file, and the *mfold* software checks for symmetry. This is an example of built in redundancy as a check on precision.

A 2-loop, $\mathbf{L}$, of size > 0 is called a *bulge loop* if $l_s^1(\mathbf{L}) = 0$ or $l_s^2(\mathbf{L}) = 0$, and an interior loop if **both** $l_s^1(\mathbf{L}) > 0$ and $l_s^2(\mathbf{L}) > 0$.

Bulge loops up to size 30 are assigned free energies from the loop file (See Figure 1). For larger bulge loops, equation 2 is used. When a bulge loop has size 1, the stacking free energy for base pairs $i.j$ and $i'.j'$ are used (from the stack file).

Interior loops have size ≥ 2. If $l_s^1(\mathbf{L}) = l_s^2(\mathbf{L})$, the loop is called *symmetric*; otherwise, it is *asymmetric*, or lopsided. The asymmetry of an interior loop, a($\mathbf{L}$) is defined by:

$$a(\mathbf{L}) = |l_s^1(\mathbf{L}) - l_s^2(\mathbf{L})|. \tag{4}$$

The free energy, $\delta\delta G_I$, of an interior loop is the sum of 4 components:

$$\delta\delta G_I = \delta\delta G_I^1 + \delta\delta G_I^2 + \delta\delta G_I^3 + \delta\delta G_I^4. \tag{5}$$

1. $\delta\delta G_I^1$ is the size dependent contribution from the loop file, or from equation 2 for sizes > 30.

2. $\delta\delta G_I^2$ and $\delta\delta G_I^3$ are terminal mismatch stacking free energies, taken from the tstacki file. The format of this file is identical to the format of the tstackh

```
            5' --> 3'                 5' --> 3'
               X                         X
              C A                       C  A
              G T                       G  U
               Y                         YA
            3' <-- 5'                 3' <-- 5'
   Y:   A    C    G    T      Y:   A    C    G    U
   ---------------------      ---------------------
X:A | 1.1  2.1  0.8  1.0   X:A | 3.2  3.0  2.4  4.8
  C | 1.7  1.8  1.0  1.4     C | 3.1  3.0  4.8  3.0
  G | 0.5  1.0  0.3  2.0     G | 2.5  4.8  1.6  4.8
  T | 1.0  1.4  2.0  0.6     U | 4.8  4.8  4.8  4.8
```

Figure 5: Left: Free energies for all 1×1 interior loops in DNA closed by a CG and an AT base pair. Right: Free energies for all 1×2 interior loops in RNA closed by a CG and an AU base pair, with a single stranded U 3' to the double stranded U. As in similar Figures, X refers to row and Y to column.

file. There are 2 terms because of the terminal stacking of both r_{i+1} and r_{j-1} on the $i.j$ base pair, and of both $r_{i'-1}$ and $r_{j'+1}$ on the $i'.j'$ base pair. This may be visualized as

$$\begin{array}{ccccc} 5'- & r_i & - & r_{i+1} & -3' \\ & \bullet & & \circ & \\ 3'- & r_j & - & r_{j-1} & -5' \end{array} \quad \text{and} \quad \begin{array}{ccccc} 5'- & r_{j'} & - & r_{j'+1} & -3' \\ & \bullet & & \circ & \\ 3'- & r_{i'} & - & r_{i'-1} & -5', \end{array}$$

where $\bullet$ denotes a base pair and $\circ$ denotes a mismatched pair.

3. $\delta\delta G_I^4$ is the asymmetry penalty, and is a function of a($\mathbf{L}$) defined in equation 4. The penalty is 0 for symmetric interior loops. The asymmetric penalty free energies come from the miscloop.dg or miscloop.TC file.

Equation 5 is now used only for loops of size > 4 or of asymmetry > 1. This means that special rules apply to 1×1, 1×2 and 2×2 interior loops. Free energies for these symmetric and almost symmetric interior loops are stored in files sint2.dg, asint1x2.dg and sint4.dg, respectively. As above, the suffix TC is used in place of dg when explicit attention is paid to temperature. These files list all possible values of the single stranded bases, and all possible Watson-Crick and GU base pair closings. The sint2 file comprises a 6×6 array of 4×4 tables. There is a table for all possible 6×6 closing base pairs. The free energy values for each choice of closing base pairs are arranged in 4×4 tables. The term "closing base pairs" refers to the closing base pair of the loop and the contained base pair of the loop, as in the strict definition of a loop. An example of such a table is given in Figure 5.

The asint1x2 file comprises a 24 row by 6 column array of 4×4 tables. There is a 4×4 table for all possible $6 \times 6 \times 4$ closing base pairs and choice of one of the

```
 5' ------> 3'
  C \/ \_/ A
  G /\  |  U
3' <------ 5'
```

Y:		A	A	A	C	C	C	G	G	G	U	U	U
		A	C	G	A	C	U	A	G	U	C	G	U
	AA	2.0	1.6	1.0	2.0	2.6	2.6	1.0	1.4	0.2	2.3	1.5	2.2
	AC	2.4	1.9	1.3	2.4	2.4	2.4	1.3	1.7	-0.4	2.1	0.8	1.5
	AG	0.9	0.4	-0.1	0.9	1.9	1.9	-0.1	0.2	-0.1	1.6	1.2	1.8
	CA	1.9	1.5	0.9	1.9	1.9	1.9	0.9	1.3	-0.9	1.6	0.4	1.1
	CC	2.8	1.8	2.2	2.2	2.2	2.2	2.2	2.2	0.4	1.9	1.7	1.4
X	CU	2.7	1.6	2.0	2.1	2.1	2.1	2.0	2.0	0.3	1.8	1.5	1.2
	GA	1.0	0.6	0.0	1.0	2.0	2.0	0.0	0.4	0.0	1.7	1.3	2.0
	GG	1.8	1.3	0.7	1.8	2.4	2.4	0.7	1.1	0.0	2.1	1.2	1.9
	GU	1.8	0.4	1.6	0.8	1.8	1.8	1.6	1.2	-2.0	1.5	-0.7	1.8
	UC	2.7	1.6	2.0	2.1	2.1	2.1	2.0	2.0	0.3	1.8	1.5	1.2
	UG	0.3	-1.1	0.1	0.7	0.3	0.3	0.1	0.3	-3.5	0.0	-2.2	0.3
	UU	2.2	0.7	1.9	1.2	1.2	1.2	1.9	1.5	0.2	0.9	1.5	0.3

Figure 6: Free energies for all interior loops in RNA closed by a CG and an AU base pair. Values of 'X' or 'Y' that correspond to bases that could form Watson-Crick pairs have been removed for brevity.

single stranded bases. The free energy values for each choice of closing base pairs and a single stranded base are arranged in 4×4 tables. An example of these tables is given in Figure 5.

Finally, the sint4 file contains 36 16×16 tables, 1 for each pair of closing base pairs. A 2×2 interior loop can have 4^4 combinations of single stranded bases. If, for example, the loop is closed by a GC base pair and an AU base pair, we can write it as:

```
 5' ------> 3'
  G \/ \_/ A
  C /\  |  U
3' <------ 5'
```

Both the large 'X' and large 'Y' refer to an unmatched pair of bases that are juxtaposed. They can each take on 16 different values, from 'AA','AC', ..., to 'UU', or 1 to 16, respectively. The number in row 'X' and column 'Y' of the table is the free energy of the 2×2 interior loop with the indicated single stranded bases. Figure 6 shows the full table for the CG and AU closing base pairs.

Some special rules apply to 2-loops. A stacked pair that occurs at the end of a helix has a different free energy than if it were in the middle of a helix. Because of the availablility and precision of data, we distinguish between GC closing and non-GC closing base pairs. In particular, a penalty (terminal AU penalty) is assigned to each non-GC closing base pair in a helix. The value of this penalty is stored in the MISCLOOP file.

Because free energies are assigned to loops, and not to helices, there is no *a priori* way of knowing whether or not a stacked pair will be terminal or not. For this reason, the terminal AU penalty is built into the TSTACKH and TSTACKI tables. For bulge, multi-branch and exterior loops, the penalty is applied explicitly. In all of these cases, the penalty is *formally* assigned to the adjacent loop, although it really belongs to the helix.

A "Grossly Asymmetric Interior Loop (GAIL)" is an interior loop that is $1 \times n$, where $n > 2$. The special "GAIL" rule that is used in this case substitutes AA mismatches next to both closing base pairs of the loop for use in assigning terminal stacking free energies from the TSTACKI file.

A k-loop, $\mathbf{L}$, where $k > 2$, is called a *multi-branch* loop. It contains $k-1$ base pairs, and is closed by a k^{th} base pair. Thus there are k stems radiating out from this loop. Because so little is known about the effects of multi-branch loops on RNA stability, we assign free energies in a way that makes the computations easy. This is the justification for the use of an *affine* free energy penalty for multi-branch loops. The free energy, $\delta\delta G(\mathbf{L})$, is given by:

$$\delta\delta G(\mathbf{L}) = a + b \times l_s(\mathbf{L}) + c \times l_d(\mathbf{L}) + \delta\delta G_{stack}, \qquad (6)$$

where a, b and c are constants that are stored in the miscloop file and $\delta\delta G_{stack}$ includes stacking interactions that will be explained below. This simple energy function allows the dynamic programming algorithm used by *mfold* to find optimal multi-branch loops in time proportional to n^3. It would take exponentially increasing time (with sequence length) to use a more appropriate energy function derived from Jacobson-Stockmeyer theory [30] that grows logarithmically with $l_s(\mathbf{L})$. In the *efn2* program that recalculates folding free energies using more realistic rules (defined below), equation 6 is replaced by:

$$\delta\delta G(\mathbf{L}) = a + 6b + 1.75 \times RT \times \ln(l_s(\mathbf{L})/6) + c \times l_d(\mathbf{L}) + \delta\delta G_{stack}. \qquad (7)$$

That is, the linear dependence on l_s changes to a logarithmic dependence for more than 6 single stranded bases in a multi-branch loop.

Stacking free energies, $\delta\delta G_{stack}$ are computed for multi-branch and exterior loops. In the folding algorithm these are single strand stacking free energies, also known as *dangling base* free energies, because they are applied to single stranded bases adjacent to a base pair that is either in the loop, or closes the loop. This single stranded base may "dangle" from the $5'$ or $3'$ end of the base pair. These parameters are stored in a file named dangle.dg or dangle.TC, as above.

Figure 7 shows some single strand stacking free energies.

If $i.j$ and $j+2.k$ are 2 base pairs, then r_{j+1} can interact with both of them. In this case, the stacking is assigned to only 1 of the 2 base pairs, whichever has a lower free energy (usually the $3'$ stack). If $k.l$ is a base pair and both r_{k-1} and r_{l+1} are single stranded, then both the $5'$ and $3'$ stacking are permitted. The value of $\delta\delta G_{stack}$ is then the sum of all the single base stacking free energies associated with the base pairs and closing base pair of the loop.

It has been evident for some time that to make the free energy rules more realistic for multi-branch and exterior loops, and to improve folding predictions, we would

```
          X                                 X
  ------------------                ------------------
   A    C    G    U                  A    C    G    U
  ------------------                ------------------
      5' --> 3'                         5' --> 3'
         CX                                C
         G                                 GX
      3' <-- 5'                         3' <-- 5'
  -1.7 -0.8 -1.7 -1.2               -0.2 -0.3  0.0  0.0
```

Figure 7: Free energies for all possible single stranded bases that are adjacent to a CG base pair. 'X' refers to column. Note that the $3'$ dangling free energies are larger in magnitude than the $5'$ dangling free energies.

be compelled to take into account the stacking interactions between adjacent helices. Two helices, $\mathbf{H_1}$ and $\mathbf{H_2}$ in a multi-branch or exterior loop are adjacent if there are 2 base pairs $i.j$ and $j+1.k$, $i.j$ and $i+1.k$ or $i.j$ and $k.j-1$ that close $\mathbf{H_1}$ and $\mathbf{H_2}$, respectively. The last 2 cases can only occur in a multi-branch loop. In addition, we define *almost adjacent* helices as 2 helices where the addition of a single base pair (usually non-canonical), results in an adjacent pair. The concept of adjacent helices is important, since they are often coaxial in 3 dimensions, with a stacking interaction between the adjacent closing base pairs. The concept of almost adjacent comes from tRNA where, in many cases, the addition of a GA base pair at the base of the anti-codon stem creates a helix that is adjacent to, and stacks on, the D-loop stem.

Mfold does not yet take into account coaxial stacking of adjacent or almost adjacent helices. The *efn2* program that re-evaluates folding energies based on our best estimates does take this into account. It is not a trivial matter to decide which combination of coaxial stacking and single base stacking gives the lowest free energy in a multi-branch or external loop, and a recursive algorithm is employed to find this optimal combination. For example, coaxial stacking excludes single base stacking adjacent to the stacked helices. Free energies for the stacking of adjacent helices are stored in a file called coaxial.dg. The format is the same as for stack.dg. When 2 helices are almost adjacent, then 2 files, named coaxstack.dg and tstackcoax.dg are used. The format is the same as for stacking free energies. The use of these 2 files is explained with the aid of Figure 8. Thus, in the *efn2* program, $\delta\delta G_{stack}$ is a combination of single base stacking and coaxial stacking, depending on the loop.

In the case of circular RNA, the choice of origin is arbitrary. However, once it is made, what would be the exterior loop in linear RNA becomes equivalent to a hairpin, bulge, interior or multi-branch loop, or a stacked pair.

The Turner parameters for RNA folding have been published and summarized a number of times. The most significant older publications are [31, 32, 33], and *mfold* was originally used these results alone. Version 1 of *mfold* had no tloop.dg file, and there was a single terminal stacking free energy file, tstack.dg. Tetraloop

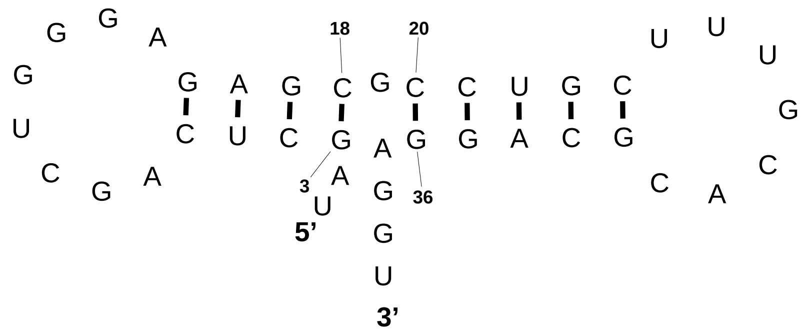


Figure 8: The helices closed by G^3-C^{18} and C^{20}-G^{36} are almost adjacent. Their stacking is mediated by a non-canonical G^{19}-A^{37} base pair. The free energy for the G^{19}-A^{37} to C^{20}-G^{36} comes from the tstackcoax.dg file. This is used where the phosphate backbone is unbroken, since there are 2 covalent links. The C^{18}-G^3 to G^{19}-A^{37} stacking free energy comes from the coaxstack.dg, which is used for stacking where the backbone is broken. In this case, G^3 and A^{37} are not linked.

bonus free energies were added in version 2.0. The tstack file was split into 2 files in version 2.2. Version 3.0 introduces triloop bonus energies, and both tetraloop and triloop bonus energies now depend on the closing base pair. Special rules for small interior loops are also new. For example, the 2×2 interior loop rules have evolved from [34]. Coaxial stacking was also introduced in version 3.0, although it's importance was realized eariler [35].

A complete set of DNA folding parameters have recently become available [36]. These are based on measuremments for stacking and mismatches, and on the literature for loop and other effects. Parameters are also available for predicting the formation of RNA/DNA duplexes [37].

4 Constrained folding

In addition to the free energy rules, specific constraints may be used to force or prohibit base pairs. A special file containing commands to constrain folding is used. The command syntax is rigid. The various commands and syntax are given below.

1. Forcing a string of consecutive bases to pair.
   Syntax: F i 0 k
   Ribonucleotides $r_i, r_{i+1}, \ldots, r_{i+k-1}$ are forced to be double stranded. The partners for these bases are chosen by the program. As an example, the command:
   F 23 0 5
   would cause bases 23, 24, 25, 26 and 27 to pair.

TABLE 2: Unsupported ambiguous codes for RNA/DNA. *mfold* does not currently support the convention for ambiguous codes. Unrecognized bases will not be allowed to pair.

Ambiguity	A,G	C,U/T	A,U/T	C,G	A,C	G,U/T
Code letter	R	Y	W	S	M	K
Ambiguity	C,G,U/T	A,G,U/T	A,C,U/T	A,C,G	A,C,G,U/T	
Code letter	B	D	H	V	N	

2. Forcing a string of consecutive base pairs.
   Syntax: `F i j k`
   Base pairs $r_i - r_j, r_{i+1} - r_{j-1}, r_{i+2} - r_{j-2}, \ldots, r_{i+k-1}.r_{j-k+1}$ are forced to occur. This is the same thing as forcing a helix to form. The helix is designated by its (external) closing base pair, $i.j$. As an example, the command:
   `F 2 110 3`
   would force base pairs 2.110, 3.109 and 4.108. Note that these base pairs **must be able to form!** Be aware also that *mfold* filters out isolated base pairs.

3. Prohibiting a string of consecutive bases from pairing.
   Syntax: `P i 0 k`
   Ribonucleotides $r_i, r_{i+1}, \ldots, r_{i+k-1}$ are prevented from pairing.

4. Prohibiting a string of consecutive base pairs
   Syntax: `P i j k`
   Base pairs $r_i - r_j, r_{i+1} - r_{j-1}, r_{i+2} - r_{j-2}, \ldots, r_{i+k-1}.r_{j-k+1}$ are not allowed to form. This is equivalent to prohibiting a helix.

5. Prohibiting 1 segment of a sequence from pairing with another
   Syntax: `P i-j k-l`
   where $i \leq j$ and $k \leq l$. In this case, no base pairs are allowed between $r_i, r_{i+1}, \ldots, r_j$ and $r_k, r_{k+1}, \ldots, r_l$. Note that the 2 segments need not be distinct. For example, the command:
   `P i-j i-j`
   will not allow $r_i, r_{i+1}, \ldots, r_j$ to pair with itself.

6. Annotated bases.

   (a) *mfold* recognizes A, C, G, U and T. In RNA folding, a 'T' will be treated as a 'U'; and *vice versa* for DNA folding. In addition, B, D, H and V are recognized as A, C, G and U/T, respectively. Bases marked in this way are regarded as susceptible to nuclease cleavage. They are allowed to pair only if their 3′ neighbor is unpaired. This is an old feature of *mfold* .

   (b) *mfold* also recognizes W, X, Y and Z as A, C, G and U/T, respectively. These bases are regarded as "modified" and are allowed to pair only at the ends of helices. At this time, the commonly used ambiguous codes shown in Table 2 are not supported by *mfold* .

5 Running the programs

The main program in the *mfold* package is named *mfold* . It is really a (Unix shell) script that calls a number of Fortran and C programs and puts together a reasonable output. Calling the *mfold* script without any command line parameters will cause the following output:

```
Usage is
mfold SEQ='file\_name' with optional parameters:
[ AUX='auxfile\_name' ] [ RUN\_TYPE=text (default) or html ]
[ NA=RNA (default) or DNA ] [ LC=sequence type (default = linear) ]
[ T=temperature (default = 37) ] [ P=percent (default = 5) ]
[ W=window parameter (default - set by sequence length) ]
[ MAXBP=max base pair distance (default - no limit) ]
[ MAX=maximum number of foldings to be computed (default 100) ]
[ ANN=structure annotation (default = none, p-num or ss-count) ]
[ START=5' base # (default = 1)] [ STOP=3' base # (default = end) ]
[ REUSE=NO/YES (default=NO) reuse existing .sav file ]
```

The meaning and use of these parameters are discussed below.

5.1 COMMAND LINE PARAMETERS

These command line parameters can only be fully understood when the meaning of the different type of output files, described in the next subsection, is known.

SEQ : The user must supply the name of a sequence file, called 'file_name' here. If 'file_name' ends with a suffix, that is, a period ('.') followed by other characters, then the suffix is removed and the result is called 'fold_name'. If no periods exist in 'file_name', then 'fold_name' = 'file_name'. For example, if the sequence is stored in 'trna.seq', then 'file_name' becomes 'trna'. If, on the other hand, the sequence file is named 'trna-file', then 'file_name' becomes 'trna-file'. The 'file_name', which may contain periods, becomes the 'prefix' for all the output files, such as 'file_name.out', 'file_name.det' and others.

Accepted sequence file formats are *GenBank*, *EMBL*, *FASTA* and *IntelliGenetics*. The sequence file may contain multiple sequences. At present, the *mfold* script will fold the first sequence by default. A new command line variable, NUM='#' may be added that directs the script to fold the '#'th sequence in the input file.

AUX : This is the name of an auxiliary input file of folding constraints. If this parameter is not used, *mfold* looks for a file named 'fold_name.aux'. If this file exists and is not empty, then it is interpreted as a constraint file. Thus constraints may be used without the use of this command line parameter.

RUN_TYPE : This parameter takes on 2 values; 'text', by default, and 'html' otherwise. The text option creates plain text files for the 'fold_name.out' and 'fold_name.det' files described below. The html option creates HTML versions of these files for display with a web browser.

TABLE 3: If 'W' is not specified, *mfold* will choose its value from this table based on sequence length. The user is encouraged to experiment with this parameter.

Sequence length	**Default window size**
1-29	0
30-49	1
50-119	2
120-199	3
200-299	5
300-399	7
400-499	8
500-599	10
600-699	11
700-799	12
800-1199	15
1200-1999	20
$\geq$ 2000	25

NA : This parameter takes on 2 values; 'RNA' by default, and 'DNA' otherwise. It tells *mfold* what type of nucleic acid is being folded.

LC : This parameter takes on 2 values; 'linear' by default, and 'circular' otherwise. It indicates to *mfold* whether a linear or circular nucleic acid is being folded.

T : This is the temperature, in °C. By default, it is 37°. Non-integral values will be rounded down to the nearest integer. Values should be in the range $0 \leq T \leq 100$.

P : This is the percent suboptimality for computing the *energy dot plot* and suboptimal foldings. The default value is 5%. This parameter controls the value of the free energy increment, $\Delta\Delta G$. $\Delta\Delta G$ is set to P% of ΔG, the computed minimum free energy. The *energy dot plot* shows only those base pairs that are in foldings with free energy $\leq \Delta G + \Delta\Delta G$. Similarly, the free energies of computed foldings are in the range from ΔG to $\Delta G + \Delta\Delta G$. No matter the value of P, *mfold* currently keeps $\Delta\Delta G$ in the range $1 \leq \Delta\Delta G \leq 12$ (kcal/mole).

W : This is the window parameter that controls the number of foldings that are automatically computed by *mfold* . 'W' may be thought of as a distance parameter. The distance between 2 base pairs, $i.j$ and $i'.j'$ may be defined as $\max\{|i-i'|, |j-j'|\}$. Then if $k-1$ foldings have already been predicted by *mfold* , the k^{th} folding will have at least W base pairs that are at least a distance W from any of the base pairs in the first $k-1$ foldings. As W increases, the number of predicted foldings decreases. If W is not specified, *mfold* selects a value by default based on sequence length, as displayed in Table 3.

MAXBP : A base pair $i.j$ will not be allowed to form (in linear RNA) if $j - i > MAXBP$. For circular RNA, a base pair $i.j$ cannot form if $\min\{j-i, n+i-j\} > MAXBP$. Thus small values of MAXBP ensure that only short range base pairs will be predicted. By default, $MAXBP = +\infty$, indicating no constraint.

MAX : This is the maximum number of foldings that *mfold* will compute (50 by default). It is better to limit the number of foldings by careful selection of the P and W parameters.

ANN : This parameter currently takes on 3 values. 1. 'none' : secondary structures are drawn without any special annotation. Letters or outline are in black, while base pairs are red lines or dots for GC pairs and blue lines or dots for AU and GU pairs. 2. 'p-num' : Colored dots, colored base characters or a combination are used to display in each folding how well-determined each base is according to the P-num values in the 'fold_name.ann' file. 3. 'ss-count' : Colored dots, colored base characters or a combination are used to display in each folding how likely a base is to be single-stranded according to sample statistics stored in the 'fold_name.ss-count' file. Both 2. and 3. were recently described [38].

START : A segment to be folded is selected from the entire sequence. START is the first base, and is 1 by default.

STOP : This is the last base in the folded segment. It is the entire sequence length by default.

REUSE : This parameter is either N (no, the default) or Y (yes). *mfold* creates a large save file, 'fold_name.sav' that contains all the input parameters and the arrays of minimum folding energies for all sub-fragments of the folded sequence. This file, especially for large sequences, is expensive to create. If REUSE is 'Y', then a file named 'fold_name.sav' should exist from a previous run. You must specify the sequence file. Any (new) constraint file will be ignored, since constraints from the initial folding will be used. Similarly, NA, LC, T, MAXBP, START and STOP are determined from the initial run. However, RUN_TYPE, P, W, MAX and ANN may be altered to give different numbers of foldings, different *energy dot plots*, and/or different types of structure annotation.

Additional command line parameters for future development are discussed in the section on "Future plans".

5.2 Output

As described above, a prefix name, called 'file_name', is derived from the name of the input sequence file. All output files begin with this prefix. *mfold* produces 2 kinds of output, the *energy dot plot* and a selection of foldings within a prescribed increment from the minimum folding energy.

5.2.1 The energy dot plot

A nucleic acid secondary structure dot plot is a triangular plot that depicts base pairs as dots or other symbols. We shall refer to these symbols as dots. A dot in column i and row j of a triangular array, $\{(i,j)|1 \le i \le j \le n\}$ represents the base pair $i.j$. The advantage of a dot plot is that it can display the base pairs in more than 1 folding simultaneously. It can be used to compare a few foldings, or the base pair distribution in many millions of foldings.

Mfold computes a number, $\Delta G(i,j)$ for every possible base pair, $i.j$. This is the minimum free energy of any folding that contains the $i.j$ base pair. As above, we let ΔG be the overall minimum folding free energy, and $\Delta\Delta G$ a user selected free energy increment. Clearly

$$\Delta G = \min_{1\leq i<j\leq n} \Delta G(i,j).$$

The energy increment is derived from ΔG and P. That is, $\Delta\Delta G = P \times \Delta G/100$. The current convention is to lower $\Delta\Delta G$ to 12 kcal/mole when it would otherwise be greater, and to raise it to 1 kcal/mole when it would otherwise be smaller. Then the *energy dot plot* is defined to be the collection of all base pairs $i.j$ satisfying:

$$\Delta G(i,j) \leq \Delta G + \Delta\Delta G.$$

This dot plot contains the **superposition of all possible foldings** whose folding energy is within $\Delta\Delta G$ of the minimum folding energy. Typically, $|\Delta\Delta G|$ is small compared to $|\Delta G|$, or P is a small percentage. In this case, the *energy dot plot* contains the superposition of all close to optimal foldings.

The *energy dot plot* gives an overall visual impression of how "well-defined" the folding is. A cluttered plot, or cluttered regions, indicate either structural plasticity (the lack of well-defined structure) or else the inability of the algorithm to predict a structure with confidence. A couple of crude measures of "well-definedness" have been introduced in *mfold* . The first is "P-num". $P\!-\!num(i)$ is a measure of the level of promiscuity of r_i in its pairing with other bases in foldings within $\Delta\Delta G$ of ΔG. It is the number of different base pairs, $i.j$, or $k.i$ that can form in this set of foldings, and is simply the number of dots in the i^{th} row and i^{th} column of the *energy dot plot* . If $\delta(expression)$ is defined to be 1 when "expression" is true, and 0 otherwise, then P-num may be defined as:

$$P\!-\!num(i) = \sum_{k<i} \delta(\Delta G(k,i) \leq \Delta G + \Delta\Delta G) + \sum_{i<j} \delta(\Delta G(i,j) \leq \Delta G + \Delta\Delta G).$$

P-num pertains to individual bases. H-num is "well-definedness" measure for a base pair $i.j$. It is the average value of the two P-num quantities, adjusted by removing the "desirable" $i.j$ base pair. That is:

$$H\!-\!num(i,j) = (P\!-\!num(i) + P\!-\!num(j) - 1)/2.$$

A helix, already defined as a collection of two or more consecutive base pairs, may be described as a triple i,j,k, where k is the number of base pairs, and the actual base pairs are $i.j, i{+}1.j{-}1, \ldots, i{+}k{-}1.j{-}k{+}1$. When $k = 1$, the helix becomes a single base pair. With some abuse of notation, we may also write $H\!-\!num(i,j,k)$ to be the H-num value of the helix, i,j,k. This is the average value of H-num over all the base pairs in the helix.

There are 5 files associated with the *energy dot plot* .

'FILE_NAME.PLOT' : This is a text file that contains all the base pairs on the *energy dot plot* , organized into helices for which $\Delta G(i,j)$ is constant. The

level	length	istart	jstart	energy
1	8	206	242	-972
1	7	319	434	-972
1	7	108	141	-972
1	7	53	185	-972
1	6	334	412	-972
1	6	308	444	-972
1	6	288	472	-972
1	6	247	279	-972
...				
2	4	8	23	-971
2	2	69	78	-971
2	4	1	24	-970
2	2	10	17	-970
2	3	345	400	-967
2	2	297	462	-967
...				

Figure 9: Selected records from a plot file. "level" refers to a free energy range that is to be plotted in the same color, where 1 is always optimal. The "level" parameter is obsolete in the newer plotting programs of *mfold* 3.0. "istart", "jstart" and "length" define a helix and refer to i, j, k, respectively. The "energy" is free energy expressed as an integer in 10^{th}s of a kcal/mole. Note that this is not the free energy of the helix, but the mimimum free energy of any folding that contains the helix.

first record is a header, and each subsequent record describes a single helix. The records are usually sorted by $\Delta G(i,j)$, and are often filtered so that short helices or isolated base pairs (helices of length 1) in suboptimal foldings are removed. Figure 9 shows a sample plot file.

'FILE_NAME.ANN' : This file contains P-num information for a particular $\Delta\Delta G$. The i^{th} record contains i and $P\!-\!num(i)$. This file is used for annotating plotted structures.

'FILE_NAME.H-NUM' : This file is the same as 'file_name.plot', except that the "energy" column is replaced by an "h-num" column. These files are usually sorted by h-num; lowest to highest, or best determined to worst determined. Often, only helices in optimal foldings are retained. Figure 10 shows part of a sorted and filtered h-num file corresponding to the plot file in Figure 9.

'FILE_NAME.PS' : This is a PostScript file of the *energy dot plot* .

'FILE_NAME.GIF' : This is an image of the *energy dot plot* in "gif" format, suitable for display on web pages.

5.2.2 Optimal and suboptimal foldings

Mfold predicts a number of optimal and suboptimal foldings. They are automatically predicted in order of increasing free energy, although this order may change

level	length	istart	jstart	h-num
1	4	38	194	6.8
1	4	215	232	7.3
1	5	31	201	8.4
1	7	53	185	8.4
1	2	47	189	11.0
1	8	206	242	11.9
1	6	61	176	13.7
1	4	89	163	13.8
1	3	255	271	14.0
1	3	104	145	15.0
1	1	68	79	16.0
1	4	121	131	17.0
1	6	288	472	17.3
...				
1	2	353	389	35.0
1	3	364	377	38.7
1	3	297	459	39.0

Figure 10: The beginning and end of an h-num file sorted by h-num and filtered to include only helices in optimal foldings. As with P-num, H-num values are relative to a particular sequence and free energy increment.

when the more exact *efn2* program is used to re-evaluate free energies. The number of computed foldings is limited directly by the MAX parameter, and in more subtle ways by the P and W parameters. It should be stated clearly here that while the *energy dot plot* rigorously displays all possible base pairs that can take part in all possible foldings within $\Delta\Delta G$ of ΔG, the computation of foldings is arbitrary. They do not represent a statistical sample of likely foldings, but rather a collection of foldings that show the variation that is possible within optimal and suboptimal foldings.

The collection of triples, $i, j, \Delta G(i, j)$, for all possible base pairs is sorted in order of increasing $\Delta G(i, j)$. The algorithm to construct foldings proceeds as follows:

1. The base pair at the top of the list is selected, and an optimal folding *containing the selected base pair* is computed.
2. All base pairs in the computed folding, as well as all those within a distance of W of base pairs in the computed folding, are crossed off the list.
3. The computed folding is retained if it contains at least W base pairs that were not found in previous foldings.

The first structure is always retained, even if it contains fewer than W base pairs. Steps 1 to 3 are repeated until either MAX structures have been computed and retained, or until there are no more base pairs on the list.

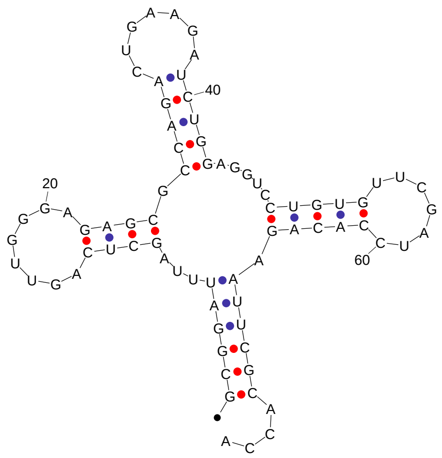
```
 FOLDING BASES    1 TO   76 OF tRNA
 Initial ENERGY  =      -22.3

             10
----      UUA       AGU
    GCGGAU     GCUC   U
    CGCUUA     CGAG   G
ACCA      --A       AGG
     70             20

               30
         G        CUG
            CCAGA
            GGUCU    A
         -        AGA
               40

                  50
            AGGUC  |  UUC
                 CUGUG
                 GACAC    G
            -----  ^  CUA
                      60
```

(a) Text



(b) Plot

Figure 11: The second and final folding of *S. cerevisiae* Phe-tRNA at 37°, with P=5% and W=3 (default values). (a) The selected base pair is G^{51}-C^{63}. The base numbers are placed so that the least significant digit, always a 0, is above or below the enumerated base. (b) The usual plotted representation. The *efn2* program has adjusted ΔG from -22.3 to -22.7 kcal/mole

Mfold creates a number of files associated with predicted structures. The files marked with an optional "html" are created only when RUN_TYPE is html. Files that contain an underscore, '_', in their names enumerate the individual foldings, so that 'file_name_i.ct' refers to the ct file for the i^{th} predicted structure.

'FILE_NAME.OUT(.HTML)' : This is a text file (html file) containing a plain text form of output for each of the predicted foldings. It is useful because it can always be displayed and is intelligible for foldings on short sequences. The selected base pairs for computing each structure are specially marked with a '|' above and a '^' below. A sample output is shown in Figure 11.

'FILE_NAME_i.CT' : The "ct" file (connect table) contains the sequence and base pair information, and is meant to be an input file for a structure drawing program. In addition to containing base pair information, it also lists the $5'$ and $3'$ neighbor of each base, allowing for the representation of circular RNA or multiple molecules. The ct file also lists the historical base numbering in the original sequence, as bases and base pairs are numbered according from 1 to the size of the folded segment. A portion of a ct file is displayed in Figure 12.

'FILE_NAME.DET(.HTML)' : This is a text file (html file) containing the detailed breakdown of each folding into loops, and the corresponding decomposition of the overall free energy, ΔG, into the free energy contributions, $\delta\delta G$, for each loop. A sample output is shown in Table 4.

TABLE 4: Free energy details for the second and final folding of *S. cerevisiae* Phe-tRNA at 37°, with default folding parameters. This layout mimics the html output.

Loop Free-Energy Decomposition

Structure 3

tRNA.seq Initial Free energy = -22.3

Structural element	$\delta\delta G$	**Information**
External loop:	-1.7	4 ss bases & 1 closing helices.
Stack:	-3.4	External closing pair is G^{1}-C^{72}
Stack:	-2.4	External closing pair is C^{2}-G^{71}
Stack:	-1.5	External closing pair is G^{3}-C^{70}
Stack:	-1.3	External closing pair is G^{4}-U^{69}
Stack:	-1.1	External closing pair is A^{5}-U^{68}
Helix	-9.7	6 base pairs.
Multi-loop:	1.0	External closing pair is U^{6}-A^{67} 10 ss bases & 4 closing helices.
Stack:	-2.1	External closing pair is C^{49}-G^{65}
Stack:	-2.1	External closing pair is U^{50}-A^{64}
Stack:	-2.2	External closing pair is G^{51}-C^{63}
Stack:	-2.1	External closing pair is U^{52}-A^{62}
Helix	-8.5	5 base pairs.
Hairpin loop:	4.8	Closing pair is G^{53}-C^{61}
Stack:	-3.3	External closing pair is C^{27}-G^{43}
Stack:	-2.1	External closing pair is C^{28}-G^{42}
Stack:	-2.1	External closing pair is A^{29}-U^{41}
Stack:	-2.4	External closing pair is G^{30}-C^{40}
Helix	-9.9	5 base pairs.
Hairpin loop:	5.7	Closing pair is A^{31}-U^{39}
Stack:	-3.4	External closing pair is G^{10}-C^{25}
Stack:	-2.1	External closing pair is C^{11}-G^{24}
Stack:	-2.4	External closing pair is U^{12}-A^{23}
Helix	-7.9	4 base pairs.
Hairpin loop:	3.9	Closing pair is C^{13}-G^{22}

```
76   ENERGY = -22.7  [initially  -22.3] yeast tRNA Phe
 1 G       0    2   72    1
 2 C       1    3   71    2
 3 G       2    4   70    3
 4 G       3    5   69    4
 5 A       4    6   68    5
 6 U       5    7    0    6
 7 U       6    8    0    7
 8 U       7    9    0    8
...
67 A      66   68    0   67
68 U      67   69    5   68
69 U      68   70    4   69
70 C      69   71    3   70
71 G      70   72    2   71
72 C      71   73    1   72
73 A      72   74    0   73
74 C      73   75    0   74
75 C      74   76    0   75
76 A      75    0    0   76
```

Figure 12: The ct file for the second and final folding of *S. cerevisiae* Phe-tRNA at 37°, with default parameters. The first record displays the fragment size (76), ΔG and sequence name. The i^{th} subsequent record contains, in order, i, r_i, the index of the $5'$-connecting base, the index of the $3'$-connecting base, the index of the paired base and the historical numbering of the i^{th} base in the original sequence. The $5'$, $3'$ and base pair indices are 0 when there is no connection or base pair.

'FILE_NAME.SS-COUNT' : If l foldings are predicted, then $ss-count(i)$ is the number of times that r_i is single stranded in these foldings. Thus $\frac{ss-count(i)}{l}$ is a *sample based probability* for single strandedness. The ss-count file contains the number of computed foldings in the first record. The i^{th} subsequent record contains i and $ss-count(i)$. This file may be used to predict which regions of an RNA are likely to be single stranded, and values of ss-count, averaged over a window of perhaps 5 to 25 base pairs, are often plotted. This file is also used for annotating plotted structures.

'FILE_NAME_i.PLT2' : This is an intermediate, device independent plot file. It is the output of *mfold*'s adaptation of the *naview* program for plotting secondary structures. This file is used as input to the *plt22ps* and *plt22gif* programs. It was originally intended to be used as input to the *plt2* plotting package [39], but this software is now old and not maintained.

'FILE_NAME_i.PS' : This is a PostScript file of a secondary structure. It is the output of the *plt22ps* program.

'FILE_NAME_i.GIF' : This is an image file (gif) of a secondary structure. It is the output of the *plt22gif* program.

The progression from ct file to images of secondary structures is:
'file_name_i.ct' → *naview* → 'file_name_i.plt2' → *plt22ps* → 'file_name_i.ps'
or
'file_name_i.ct' → naview → 'file_name_i.plt2' → *plt22gif* → 'file_name_i.gif'

'FILE_NAME.HTML' : This is a simple html file that links together some of the output files. It is an early version of a format originally used by the *mfold* web server.

'FILE_NAME.LOG' : This is a log file containing the standard output and standard error of the various programs and scripts that make up *mfold* . It can be useful for debugging.

'FILE_NAME.PNT' : This is a human readable file containing the entire input sequence. Every 10^{th} base is labeled. In addition, auxiliary information is incorporated, if there is any. Bases that are forced to be double stranded have the letter 'F' underneath. Those that are forced to be single stranded have the letter 'P' underneath. Pairs of rounded brackets '(' and ')' underline forced base pairs, and pairs of curly brackets '{' and '}' underline prohibited base pairs. If 2 disjoint segments are prohibited from pairing with one another, then these segments are highlighted by underlining the residues of the first with a common lowercase letter, and the residues of the second with the same letter in uppercase. Different letters are used for different prohibited pairs. 'F' and 'P' are not used in this case.

5.3 AUXILIARY AND INDIVIDUAL PROGRAMS

The *mfold* package contains a script, also named *mfold* , that performs a folding according to information entered on the command line. This script is itself composed of scripts and (Fortran or C) executable programs, many of which can be run separately. Some of these programs are now described.

1. *auxgen*: This program creates the 'FILE_NAME.PNT' file.
2. *boxplot97_ng*: This program creates an *energy dot plot* in PostScript or gif form from 'FILE_NAME.PLOT'. The file BOXPLOT97_NG.DOC contains instructions.
3. *ct_boxplot*: This program creates a dot plot containing only those base pairs found in a collection of structures (in "ct format") that are specified on the command line. These must all be foldings of the same sequence fragment. At present, the *mfold* script does not use this program.
4. *ct_compare*: This program compares 2 "ct files". The first contains a single reference structure. The second contains 1 or more foldings of the same sequence. The number of bases and helices from the first structure that are conserved in the other structures are computed and displayed. For the purposes of this program, a "helix" may contain bulge or interior loops of size 1 or 2 and must have at least 3 base pairs [17, 40].
5. *efn*: This program computes ΔG for all the foldings in a "ct file". The energy rules correspond exactly to what is used in *mfold* .

6. *efn2*: This program computes ΔG for all the foldings in a "ct file" using a more precise free energy computation that takes into account coaxial stacking and Jacobson-Stockmeyer theory for multi-branched loops. (See equation 7 and Figure 8, respectively.)
7. *h-num*: This program computes an "h-num" file from a "plot" file.
8. *nafold*: This is the principle folding program of the *mfold* package, and corresponds to the program "lrna" and "crna" in earlier versions of *mfold* . It has 2 command line arguments. The first takes on the values 'l' or 'c', for linear (default) or circular folding, respectively. The second has values "text" (default) or "html" for plain text output, and for some html output, as described above. It is run twice by the *mfold* script. It may be run alone, as it prompts the user for input on an interactive basis. In this mode, the user may fold all the sequences contained in a single file, an option not available with the *mfold* script. The instructions for running this program have been described in [19] and also on the WWW at `http://www.ibc.wustl.edu/~zuker/mfold-2.0`. The interactive *energy dot plot* is not functional in version 3.0.
9. *naview*: This is a modified version of the *naview* program described in [41]. Actually, in *mfold* , *naview* is a script that runs a binary called *naview.exe*. Both *naview* and *naview.exe* may be run alone in interactive form. The *mfold* script uses the files "bases.nav" and "lines.nav", stored in MFOLDLIB to direct "naview". The first produces an output that displays individual residues as letters, while the second gives a structure outline only.
10. *newtemp*: The *newtemp* program creates free energy files for folding RNA or DNA at different temperatures. The latest RNA parameters consist only of free energies measured at 37°. Since there are no corresponding enthalpy files, it is not possible to fold RNA at temperatures other than 37°. If the user wishes to fold RNA at different temperatures, then the "dg" and "dh" files from *mfold* version 2.3 should be copied into MFOLDLIB. These older free energy parameters will be supplied with *mfold* version 3.0. DNA folding may be done at arbitrary temperatures between 0 and 100 degrees. However, it should be remembered that the DNA loop parameters are all estimated from values published in the literature or by comparison with RNA.
11. *plt22ps*: This program takes a "plt2" file from *naview* and creates a PostScript file of a plotted structure. It can use an "ann" file or an "ss-count" file to annotate with P-num or ss-count values, respectively.
12. *plt22gif*: This program is the same as *plt22ps*, except that the structure output file is in gif format.
13. *sav2p-num*: This interactive program uses an existing 'fold_name.sav' file to create a P-num file.
14. *sav2plot*: This interactive program uses an existing 'fold_name.sav' file to create a "plot" file. The resulting "plot" file may be sorted and filtered using the command: FILTER-SORT NAME.PLOT N, where name.plot is a "plot" file, and N is the minimum helix size that is desired. Helices shorter than N and not removed in optimal foldings.

15. *scorer*: This program is similar to *ct_compare*. It compares foldings helix by helix, displaying a more detailed output.
16. *split_ct.awk*: This is a Unix awk script that splits a ct file into multiple files, containing 1 folding in each. These individual files may be processed by *naview*, *plt22ps* and *plt22gif*.
17. *ss-count*: This program computes single stranded sample statistics from a collection of foldings stored in a single ct file.

6 Sample foldings

This section illustrates the appearance of *mfold* PostScript output files and is meant also to give the reader some insight into the use of of these programs.

6.1 Example 1

The *energy dot plot* is an integral part of the folding prediction. Consider the folding of a short RNA sequence:
`ACCCCCUCCU UCCUUGGAUC AAGGGGCUCA A`,
using default parameters. $\Delta G = -9.8$ kcal/mole at 37°, so $\Delta\Delta G = 1.0$ rather than 5% of ΔG. A single, optimal folding is computed. A glance of the *energy dot plot* , shown in Figure 13, reveals the optimal folding in black dots (symbols), but another set of yellow dots, indicating base pairs in at least 1 other suboptimal folding. The default value of 'W' (2, from Table 3) is too large for this other folding to be predicted, but a glance at the dot plot shows that something else is there. When the sequence is refolded with 'W'=0, a second, totally different folding is predicted. Figure 14 displays these foldings with individual bases drawn.

6.2 Example 2

Important alternative foldings might not appear in the *energy dot plot* if $\Delta\Delta G$ is too small. This is especially true in the folding of short sequences. When the short sequence:
`AAGGGGUUGG UCGCCUCGAC UAAGCGGCUU GGAAUUCC`,
is folded, also with default parameters, a single optimal folding is computed. However, the *energy dot plot* contains only the optimal, black dots from Figure 15. Changing the window size would not reveal anything new. When the value of P is increased to 25 (25%), the *energy dot plot* now reveals a very distinct alternate folding as shown in Figure 15. The *mfold* program now computes 2 foldings, plotted in Figure 16, using the default value of W.

6.3 Example 3

Here we present some results from the folding of an RNA that is related to a Human adenovirus pre-terminal protein (U52533). This RNA exhibits both "well-

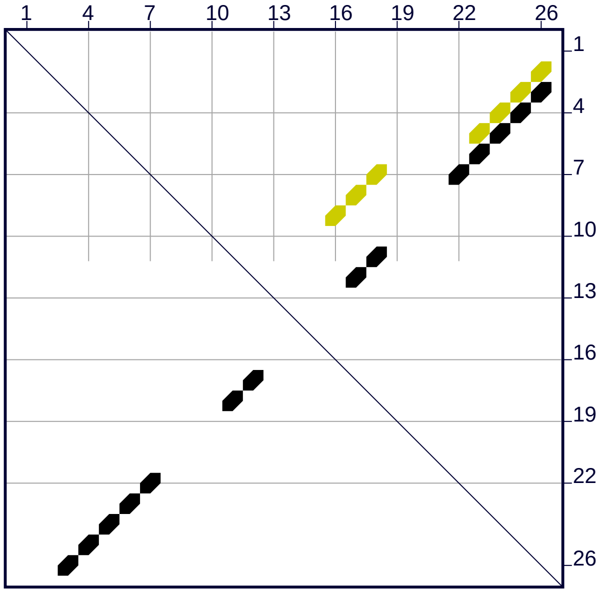
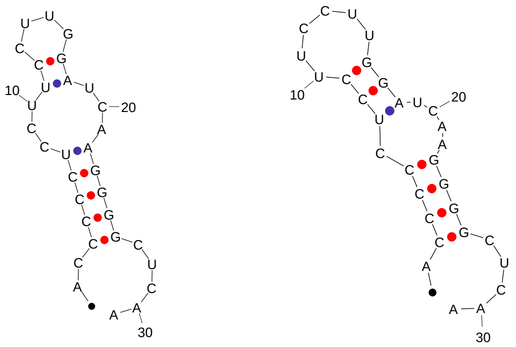


Figure 13: The *energy dot plot* for the "Example 1" sequence. Surrounding annotation, which would not be legible at this scale, has been removed. The yellow dots indicate base pairs in foldings within 0.3 kcal/mole of the optimal folding free energy of -9.8 kcal/mole.



(a) (b)

Figure 14: The 2 predicted foldings for the "Example 1" sequence. (a) The optimal folding with $\Delta G = -9.8$ kcal/mole. (b) The suboptimal fold ($\Delta G = -9.5$ kcal/mole) found after refolding with 'W'=0.

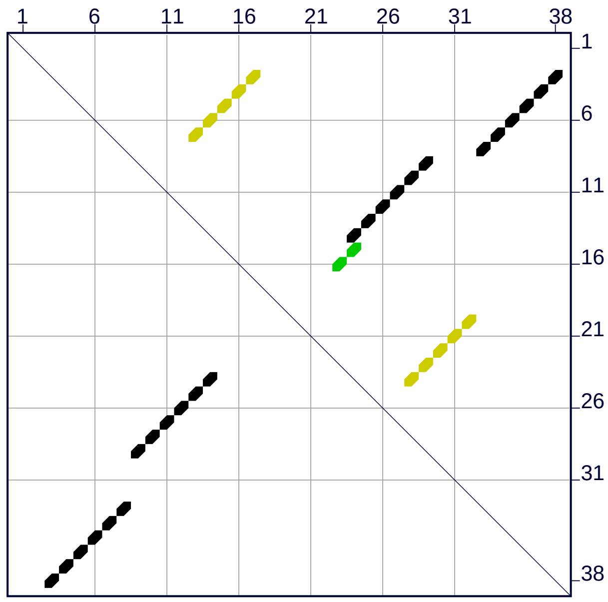


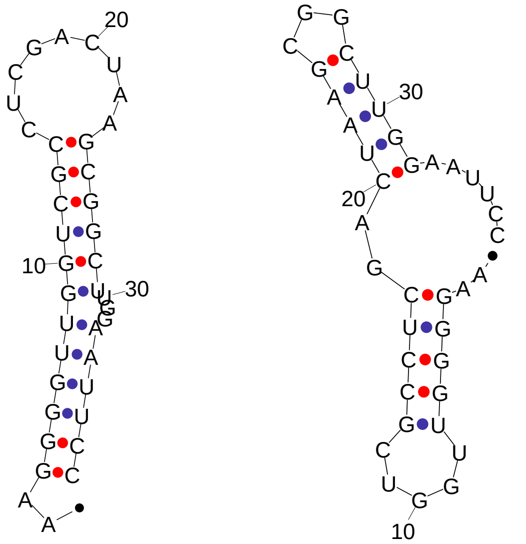
Figure 15: The *energy dot plot* for "Example 2" sequence with $\Delta\Delta G$ increased to 25% of 10.1, or 2.5 kcal/mole. The value of $\Delta\Delta G$ in the plot may be less than this maximum value, since there may be no base pairs in foldings that are *exactly* $\Delta\Delta G$ from the minimum free energy. The 2 green dots represent base pairs that can be in foldings with ΔG between -9.4 and -8.6 kcal/mole. These numbers are -8.6 and -7.9 for the yellow dots. In this case, the black dots comprise the optimal folding, and the yellow dots comprise the single suboptimal folding that is computed. The green dots would only be found in a folding if the value of W were lowered sufficiently.

defined" and "poorly-defined" folding regions, as shown in Figure 17. A total of 7 foldings were computed using the default parameters.

7 Future plans

The Unix version of *mfold* will remain. The newest programs, such as *plt22ps* and *plt22gif* are written in command line mode. The older programs have shell scripts or Perl "wrappers" around them to make them appear as single binaries that operate in command line mode. The trend will be to replace older code as necessary with non-interactive programs. This makes it easier to piece together different programs to create new forms of output.

When *mfold* was first created, the limitations of personal computers did not make a PC version practical. This has changed radically in the past 10 years, and an Intel/Windows PC is now a fine environment for running *mfold* . The basic Fortran and C programs have already been ported and will run under Widows, but a Unix-like shell is necessary. The *RNAstructure* program is now a faithful recreation of *mfold* in Windows, with a convenient user interface. The major problem with the



(a) (b)

Figure 16: The 2 predicted foldings for the "Example 2" sequence. (a) The optimal folding with $\Delta G = -10.1$ kcal/mole. (b) The suboptimal fold ($\Delta G = -7.9$ kcal/mole) found after refolding with 'P'=25.

existing setup is that the Unix and Windows versions are totally different and will have to be updated in parallel to keep them equivalent.

The *mfold* programs running on SGI/Irix, SPARC/Solaris and Intel/Solaris have been incorporated into a world wide web (WWW) server that allows users from around the world to submit sequences for folding. This server has some extra features not available in the *mfold* package, and goes well beyond the very simple HTML output of the current *mfold* software. However, this software offers nothing new in terms of predictions, and it will be described elsewhere. Rapid developments in web browsers and languages such as Java-script and Java may make an HTML (or similar) interface to *mfold* better than others. As things stand now, the *mfold* server can run on a (local) Unix computer and be accessed by web browsers running on personal computers.

Additional parameters will be added to the command line version of *mfold* in the future. These will be described when the *mfold* command is given without parameters and the documentation will be altered accordingly. In the near future, a base labeling frequency will be added so that the user can specify the frequency of base enumeration. Other controls on secondary structure, such as zooming on images about specified coordinates, could be added, but these are already available

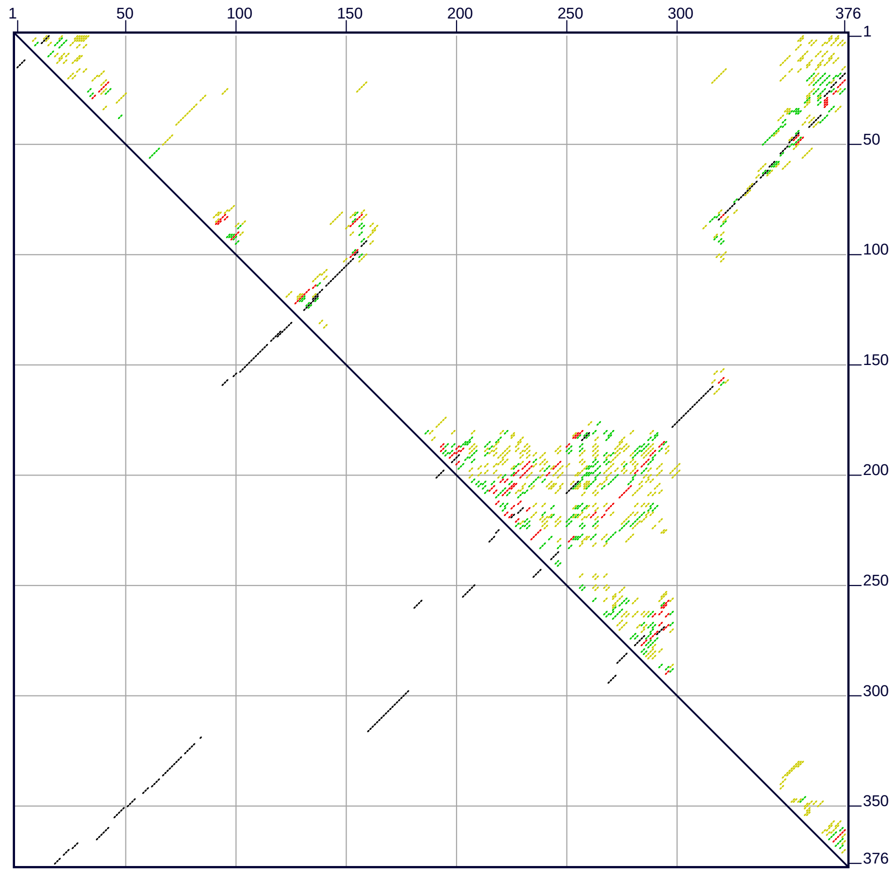


Figure 17: The *energy dot plot* for the "Example 3" sequence, with ΔG = -196.3 kcal/mole and $\Delta\Delta G$ = 9.8 kcal.mole. The region from bases 180 to 295 shows a great deal of uncertainty in its folding. This may be interpreted as a large ensemble of different foldings, or simply an "unstructured" region. In contrast, the long stem pairing bases 160-178 with 316-298, respectively, is extremely well determined. Also well-determined is a stem loop region that stretches from 37-81 and 322-365.

TABLE 5: Additional command line variables that could be added to *mfold* .

Parameter	**Description**	**Default value**
ddSTACK	extra $\delta\delta G$ per stack	0
ddBULGE	extra $\delta\delta G$ per bulge loop	0
ddILOOP	extra $\delta\delta G$ per interior loop	0
ddHLOOP	extra $\delta\delta G$ per hairpin loop	0
MAXILOOP	maximum size of an internal loop	30
MAXLOP	maximum asymmetry of an internal loop	30

through the use of the *plt22ps* and *plt22gif* programs.

The "energy" parameters from the older, interactive versions of *mfold* could easily be introduced in command line form. This would make it unnecessary to run the *nafold* program directly to alter them. Table 5 lists these parameters that are not defined in the MISCLOOP file.

Current plans call for the addition of coaxial stacking to the folding algorithm and possibly the creation of a special version that uses Jacobson-Stockmeyer theory to assign more realistic free energies to multi-branch loops, as in equation 7. In addition, a practical approach to pseudoknots will be attempted, where pairs of mutually exclusive helices that create pseudoknots are identified in the *energy dot plot* . In these case, the bases involved in 1 or perhaps 2 of these pseudoknots can be constrained to be single stranded, and foldings predicted to fill in the rest of the secondary structure.

Another future development will be the introduction of a 2 molecule folding system. This immediately complicates the problem, since concentration now becomes important. In addition, the folding of certain very simple bi-molecular systems is at least as hard as predicting pseudoknots in the folding of a single sequence.

Acknowledgment

We thank Darrin Stewart for creating the new *energy dot plot* and structure display programs. The plt22gif and boxplot97 programs use the *tgd* program of Bradley K. Sherman, and this uses the *gd 1.2* graphics library developed by Thomas Boutell, 1994, 1995, Quest Protein Database Center, Cold Spring Harbor Laboratory. We thank Ann Jacobson for her suggested improvements in the software. MZ is supported by NIGMS grant GM54250. DHM and DHT are supported, in part, by NIGMS grant GM22939.

References

[1] P.M. MacDonald. (1990) *Bicoid* mRNA localization **signal**, phylogenetic conservation of function and RNA secondary structure. *Development*, **110**, 161–171.

[2] M.H. de Smit and J. van Duin. (1990) Control of prokaryotic translation initiation by mRNA secondary structure. *Progress in Nucleic Acid Research in Molecular Biology*, **38**, 1–35.

[3] D.R. Mills, C. Priano, P.A. Merz, and B.D. Binderow. (1990) Qβ RNA **bacteriophage**, mapping cis-acting elements within an RNA genome. *J. Virol.*, **64**, 3872–3881.

[4] C.I. Brannan, E.C. Dees, R.S. Ingram, and S.M. Tilghman. (1990) The product of the h19 gene may function as an RNA. *Mol. Cell. Biol.*, **10**, 28–36.

[5] C.J. Brown, A. Ballabio, J.L. Rupert, R.G. Lafreniere, M. Grompe, R. Tonlorenzi, and H.F. Willard. (1991) A gene from the region of the human X inactivation centre is expressed exclusively from the inactive X chromosome. *Nature*, **349**, 38–44.

[6] T.R. Cech and B.L. Bass. (1986) Biological catalysis by RNA. *Ann. Rev. Biochem.*, **55**, 599–629.

[7] T.R. Cech. (1990) Self-splicing of group I introns. *Ann. Rev. Biochem.*, **59**, 543–568.

[8] S.C. Darr, J.W. Brown, and N.R. Pace. (1992) The varieties of Ribonuclease P. *Trends Biochem. Sci.*, **17**, 178–182.

[9] S.H. Kim, F.L. Suddath, G.J. Quigley, A. McPherson, and J.L. Sussman. (1974) Three dimensional tertiary structure of yeast phenylalanine transfer RNA. *Science*, **185**, 435–440.

[10] J.D. Robertus, J.E. Ladner, J.T. Finch, D. Rhodes, and R.S. Brown. (1974) Structure of yeast phenylalanine tRNA at 3 Å resolution. *Nature*, **250**, 546–551.

[11] H.W. Pley, K.M Flaherty, and D.B. McKay. (1994) Three-dimensional structure of a hammerhead ribozyme. *Nature*, **372**, 68–74.

[12] R.R. Gutell, (1995) personal communication.

[13] F. Michel and E. Westhof. (1990) Modelling of the three-dimensional architecture of group I catalytic introns based on comparative sequence analysis. *J. Mol. Biol.*, **216**, 585–610.

[14] F. Major, M. Turcotte, D. Gautheret, G. Lapalme, E. Fillion, and R.J. Cedergren. (1991) The combination of symbolic and numerical computation for three- dimensional modeling of RNA. *Science*, **253**, 1255–1260.

[15] F. Major, D. Gautheret, and R. Cedergren. (1993) Reproducing the three-dimensional structure of a tRNA molecule from structural constraints. *Proc. Natl. Acad. Sci. USA*, **90**, 9408–9412.

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

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

[18] J.A. Jaeger, D.H. Turner, and M. Zuker. (1990) Predicting optimal and suboptimal secondary structure for RNA. *Meth. Enzymol.*, **183**, 281–306.

[19] M. Zuker. (1994) *Prediction of RNA Secondary Strcture by Energy Minimization.*, volume 25 of *Computer Analysis of Sequence Data, Part II, A.M. Griffin & H.G Griffin, Eds.*, chapter 23, pages 267–294. CRC Press, Inc., Totowa, NJ.

[20] D.H. Mathews, T.C. Andre, J. Kim, D.H. Turner, and M. Zuker. (1998) *An Updated Recursive Algorithm for RNA Secondary Structure Prediction with Improved Free Energy Parameters.*, chapter 15, pages 246–257. American Chemical Society Symposium Series 682. American Chemical Society, Washington, DC.

[21] D. Sankoff, J.B. Kruskal, S. Mainville, and R.J. Cedergren. (1983) *Fast algorithms to determine RNA secondary structures containing multiple loops.*, chapter 3, pages 93–120. Time warps, string edits, and macromolecules: the theory and practice of sequence comparison, Sankoff D., Kruskal J.B., Eds. Addison-Wesley, Reading, MA.

[22] M. Zuker and D. Sankoff. (1984) RNA secondary structures and their prediction. *Bull. Math. Biol.*, **46**, 591–621.

[23] M. Zuker. (1986) RNA folding prediction: The continued need for interaction between biologists and mathematicians. *Lectures on Mathematics in the Life Sciences*, **17**, 86–123.

[24] C.W. Pleij and L. Bosch. (1989) RNA pseudoknots: structure, detection, and prediction. *Meth. Enzymol.*, **180**, 289–303.

[25] J.P.Abrahams, M. van den Berg, E. van Batenburg, and C.W. Pleij. (1990) Prediction of RNA secondary structure, including pseudoknotting, by computer simulation. *Nucleic Acids Res.*, **18**, 3035–3044.

[26] R.R. Gutell and C.R. Woese. (1990) Higher order structural elements in ribosomal RNAs: Pseudo-knots and the use of noncanonical pairs. *Proc. Natl. Acad. Sci. USA*, **87**, 663–667.

[27] E. Dam, K. Pleij, and D. Draper. (1992) Structural and functional aspects of RNA pseudoknots. *Biochemistry*, **31**, 11665–11676.

[28] C.W. Pleij. (1994) RNA pseudoknots. *Curr. Opin. Struct. Biol.*, **4**, 337–344.

[29] Z. Du, D.P. Giedroc, and D.W. Hoffman. (1996) Structure of the autoregulatory pseudoknot within the gene 32 messenger RNA of bacteriophages T2 and T6: A model for a possible family of structurally related RNA pseudoknots. *Biochemistry*, 35 **(13)**, 4187–4198.

[30] H. Jacobson and W.H. Stockmayer. (1950) Intramolecular reaction in polycondensations. I. The theory of linear systems. *J. Chem. Phys.*, **18**, 1600–1606.

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

[32] D.H. Turner, N. Sugimoto, J.A. Jaeger, C.E. Longfellow, S.M. Freier, and R. Kierzek. (1987) Improved parameters for prediction of RNA structure. *Cold Spring Harb. Symp. Quant. Biol.*, **52**, 123–133.

[33] D.H. Turner. N. Sugimoto, and S.M. Freier. (1988) RNA structure prediction. *Annu. Rev. Biophys. Biophys. Chem.*, **17**, 167–192.

[34] M. Wu, J.A. McDowell, and D.H. Turner. (1995) A periodic table of symmetric tandem mismatches in RNA. *Biochemistry*, **34**, 3204–3211.

[35] A.E. Walter, D.H. Turner, J. Kim, M.H. Lyttle, P. Muller, D.H. Mathews, and M. Zuker. (1994) Coaxial stacking of helixes enhances binding of oligoribonucleotides and improves predictions of RNA folding. *Proc Natl Acad Sci USA*, **91**, 9218–9222.

[36] J.Jr. SantaLucia. (1998) A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics. *Proc. Natl. Acad. Sci. USA*, **95**, 1460–1465.

[37] N. Sugimoto, S. Nakano, M. Katoh, A. Matsumura, H. Nakamuta, T. Ohmichi, M. Yoneyama, and M. Sasaki. (1995) Thermodynamic parameters to predict stability of RNA/DNA hybrid duplexes. *Biochemistry*, **34**, 11211–11216.

[38] M. Zuker and A.B. Jacobson. (1998) Using Reliability Information to Annotate RNA Secondary Structures. *RNA*, **4**, 669–679.

[39] R.C. Beach. (1981) *The Unified Graphics System for Fortran 77 Programming Manual.* Stanford Linear Accelerator Center Computational Research Group, Stanford, CA, Technical Memo 203.

[40] M. Zuker, J.A. Jaeger, and D.H. Turner. (1991) A comparison of optimal and suboptimal RNA secondary structures predicted by free energy minimization with structures determined by phylogenetic comparison. *Nucleic Acids Res.*, **19**, 2707–2714.

[41] R.E. Bruccoleri and G. Heinrich. (1988) An improved algorithm for nucleic acid secondary structure display. *Comput. Appl. Biosci.*, **4**, 167–173.