RNA Secondary Structure Prediction with Word Concatenations

Danielle Dees
Georgia Institute of Technology
College of Computing
danielle@cc.gatech.edu

Laura Slaybaugh
Rose-Hulman Institute of Technology
Computer Science
slaybug@rose-hulman.edu

July 23, 2001

Abstract

There has been much study focused around RNA word designs, where a RNA word is defined as a short strand over the RNA alphabet A, U, C, and G. These RNA words are designed with specific properties in mind to reduce and negate the natural tendencies of this biological matter. One property that word designers must fight is the tendency of RNA to fold onto itself and form a secondary structure. In this paper, we study the secondary structure which occurs when RNA words are concatenated into strands with the ability to store larger quantities of information. Our goal was to find an effecient algorithm which will take in $2n$ pre-designed words, where a word is defined as a sequence of RNA neucleotides of length l, and it will return a "yes" or a "no" as to whether any of these 2^n combinations will fold into a secondary structure. This algorithm allows us to confirm or deny suspicions about which word design properties actually prevent secondary structure.

1 Introduction

RNA is a single strand of neucleodtides; therefore, secondary structure observation is well-defined and observable in nature. The RNA nucledotides bond to other neucleoties based on their Watson-Crick pairs (AU, UA, CG, GC) or wobble pairs (UG, GU). It is assumed that no psuedo-knots exist in any of the secondary structures and that each base can only bond to one other base. Each psuedo-knot free structure can be viewed as a collection of stacked pairs and loops with exterior base pairs closing each loop. The types of loops are hairpin, internal, and multibranched, with a bulge loop being a particular instance of an internal loop. These structures are illustrated in Figure 1 below.

RNA secondary structure prediction can be done by determining the most stable comfiguration of the given neucleotides. The optimal structure will have the lowest free energy. Each of the various loops and stacked pairs will contribute a certain amount of energy to the secondary structure configuration. Zuker develped an dynamic programming algorithm which runs in time $O((l * n)^4)$ which predicts the secondary structure of a single RNA strand based on this energy model. This algorithm is quite useful to learn about a single sequence of RNA neucleotides. We can determine the free energy over all 2^n structures by taking advantage of dynamic programming and modifying Zuker's recurance relations for single strand folding. Although similar results could be obtained by running each of the 2^n structures through Zuker's algorithm, our algorithm will solve this problem in polynomial time.

2 Basic Algorithm

A secondary structure formed from a strand $s \in S$ where S is the set of all strands that can be created, will have $1 \leq i \leq |s|$. This algorithm will take in words will be labeled as $w(1), \overline{w(1)}, w(2), \overline{w(2)}, \ldots w(n), \overline{w(n)}$, and the resultant strand will choose either $w(i)$ or $\overline{w(i)}$ for each of the n positions. This algorithm will

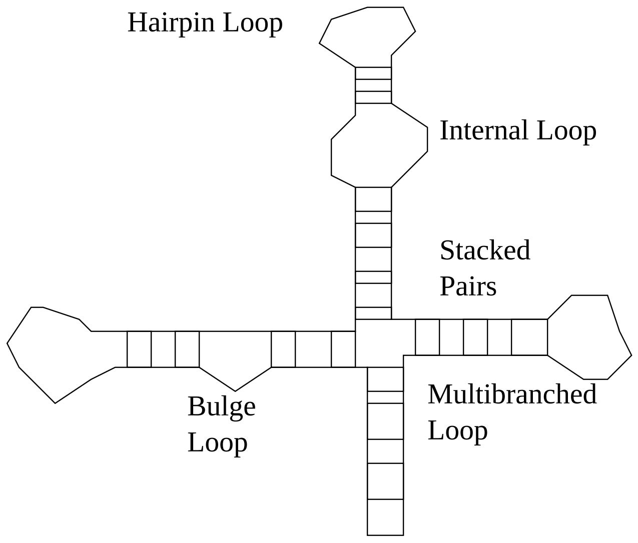

Figure 1: My beautiful picture

return a “yes” or a “no” as to whether any of the 2^n strands form a secondary structure. A “yes” is defined as the program returning a negative value. This negative value represents the energy released from the strand as it fold into a thermodynamically favorable structure. The word#(x) function determines the word number based on the word length l and parameter x. Our modifications to Zuker’s algorithm include added parameters into the equations. These parameters specify whether S_i, the base at position i where $1 \leq i \leq n * l$, is in $w(word\#(i))$ or $\overline{w(word\#(i))}$ based on the parameter $b_i \in \{T, B, E\}$. The value T represents $w(word\#(i))$, the value B represents $\overline{w(word\#(i))}$, and the value E represents the case where it does not matter to which word S_i belongs. The word specification is only T or B if the recurrance calls functions with a parameter that would be in the same word as a second parameter. Specifing the word verifies that the algorithm’s minimal energy is calculated over the concatenation of whole words, and not just over parts of words.

2.1 W'_S

The energy of the optimal structure from S_1 to S_j where j is the length of $\forall s \in S$ is:

$$W'_S(j) = W'_S(E, j) \tag{1}$$

$$W'_S(b_j, j) = \min \begin{cases} W'_S(x, j-1) \begin{array}{r} x = b_j \text{ if } j - 1 \bmod l \neq 0 \\ x = E \text{ if } j - 1 \text{mod} l = 0 \end{array} \\ \min\limits_{\substack{1 \leq i \leq j-1 \\ i-1 \text{mod} l \neq 0 \\ b_i \in \{B,T\}}} V'_S(b_i, b_j, i, j) + W'_S(b_i, i-1) \\ \min\limits_{\substack{1 \leq i \leq j-1 \\ i-1 \text{mod} l = 0}} V'_S(E, b_j, i, j) + W'_S(E, i-l) \end{cases} \tag{2}$$

2.2 V'_S

The energy of a loop which is closed with the base pair S_i, S_j is:

$$V'_S(b_i, b_j, i, j) = \min \begin{cases} eH'_S(b_i, b_j, i, j) \\ eS'_S(b_i, b_j, i, j) + V'_S(x, y, i+1, j-1) \\ VBI'_S(b_i, b_j, i, j) \\ VM'_S(b_i, b_j, i, j \end{cases} \tag{3}$$

where x which corresponds to b_{i+1} and y which corresponds to b_{j-1} are determined by the following psudocode:

```
x = E;
y = E;
if(word#(i) == word#(i+1)) then
        x = $b_i$;
if(word#(j) == word#(j-1)) then
        y = $b_j$;
if(word#(i+1) == word#(j-1)) then
        if(x == E and y == E) then
                x = $b_1$;
                y = $b_1$;
        else if(x == E) then
                x = y;
        else
                y = x;
```

2.3 VBI'_S

The energy of an internal loop which is closed by the pair S_i, S_j is:

$$VBI'_S(b_i, b_j, i, j) = \min \begin{cases} +\infty \text{ for} j < i+4 \\ \min\limits_{i<i'<j'<j} eL(b_i, b_j, x, y, i, j, i', j') + V'_S(x, y, i', j') \end{cases} \tag{4}$$

where x which corresponds to $b_{i'}$ and y which corresponds to $b_{j'}$ are determined by the following psudocode. The function $b()$ which takes in a number returns either $w(word\#(i))$ or $\overline{w(word\#(i))}$ when the important point is that two or more bases are in the same word and not which specific word.

```
x = E;
y = E;
if(word#(i) == word#(i')) then
        x = $b_i$;
if(word#(j) == word#(j')) then
        y = $b_j$;
if(word#(i') == word#(j')) then
        if(x == E and y == E) then
                x = b(1);
                y = b(1);
        else if(x == E) then
                x = y;
        else
                y = x;
```

2.4 VM'_S

The energy of a multibranched loop closed by the base pair S_i, S_j is:

$$VM'_S(b_i, b_j, i, j) = \min_{i+1<h\leq j-1} WM(w, y, i+1, h-1) + WM(x, z, h, j-1) + a \tag{5}$$

$$WM'_S(b_i, b_j, i, j) = \min \begin{cases} V'_S(b_i, b_j, i, j) + b \\ \min\limits_{i<h\leq j} WM'_S(b_i, y, i, h-1) + WM'_S(x, b_j, h, j) \end{cases} \tag{6}$$

where w which corresponds to b_{i+1},x which corresponds to b_h, y which correponds to b_{h-1}, and z which corresponds to $b_{j-1}$are determined by the following psudocode:

```
w = E;
x = E;
y = E;
z = E;
if(word#(i+1) == word#(i)) then
  w = $b_i$;
else if(word#(i+1) == word#(h-1)) then
  w = $b_1$;
if(word#(h-1) == word#(i+1)) then
  x = w;
else if(word#(h-1) == word#(h)) then
  x = $b(2)$;
if(word#(h) == word#(h-1)) then
  y = x;
else if(word#(h) == word#(j-1)) then
  y = b(3);
if(word#(j-1) == word#(h)) then
  z = y;
if(word#(j-1) == word#(j)) then
if(z == b(2)) then
  b(2) = $b_j$;
else if(z == b(3)) then
  b(3) = $b_j$;
else
  z = $b_j$;
```

3 Details of the calculations

This algorithm is dependent on eH'_S, eS'_S, and eL'_S which are the free energy equations for a hairpin loop, a stacked pair, and an internal loop respectively. Additionally, we have a table *stack* that returns the energy of stacking the pairs $i, i+1$ on $j, j-1$ and functions that return the penatly for various structures.

3.1 eS'_S

eS'_S originally takes 2 values, i and j. $eS'_S(i,j)$ in turn calls $stack(i, i+1, j-1, j)$. When we call eS'_S we actually want $\min\limits_{s \in S} eS(s,i,j)$. However, we do not need to test every possible $s \in S$ because the nucleotides relevant to eS'_S are only $i, i+1, j-1$, and j. Thus, what we actually do is create a set of strings R as defined below.

$R : \{r_1, \ldots, r_n | r_q[0] = s_i, r_q[1] = s_{i+1}, r_q[2] = s_{j-1}, r_q[3] = s_j | s \in S \text{ and } 1 \leq q \leq n\}$

Thus, we actually call $\min\limits_{r \in R} stack(r_0, r_1, r_2, r_3)$.

3.2 eH'_S

eH'_S originally takes i and j as well. This returns the value $stack(i, i+1, j-1, j) + hPenalty(j-i-1)$. When we call eH'_S we actually want $\min\limits_{s \in S} eH(s,i,j)$ Which simplifies to $\min\limits_{r \in R}\{stack(r_0, r_1, r_2, r_3)\} + hPenalty(j-i-1)$.

3.3 eL'_S

eL'_S originally takes i, i', j', and j, where i and j close exterior end of the bulge or loop and i' and j' close the interior end of the bulge or loop. eL'_S returns the value $stack(i, i+1, j-1, j) + stack(i', i'+1, j'-1, j')$. For our case then,we first define a new set

$RL : \{rl_1, \ldots, rl_n | rl_q[0] = s_i, rl_q[1] = s_{i+1}, rl_q[2] = s_{i'}, \ldots, rl_q[7] = s_j | s \in S \text{ and } 1 \leq q \leq n\}$

eL'_S will return $\min_{rl \in RL} \{stack(rl_0, rl_1, rl_6, rl_7) + stack(rl_2, rl_3, rl_4, rl_5)\}$.

4 Protein Encodings

While researching this question, we initiated contact with a pair of researchers at SUNY-Stony Brook, Barry Cohen and Steven Skiena. They had come across the same problem we are working on, but had answered it for a different reason. They were researching the coding of proteins with RNA. Each RNA strand uses codes of triplets that translate to one of twenty amino acids or a terminate signal. As there are 4^3 possible triplets, there is some redundancy in codes for amino acids. Some may have just one triplet code, others have as many as six. When a protein is built from these amino acids, this results in an exponential number of possible RNA strands that would code for the protein, or the diamond graph we saw earlier.

The code we recieved from them read two input files, one that indicated all the triplet codes and the amino acids they represent, and one that read a string of amino acids used to build the protein. We were able to rewrite these files to use a set of words and "amino acids" of our own choosing.

5 Testing Considerations

We used this program to run simulations with words designs described in papers by Braich, Frutos, and others. Each of the word design strategies keep certain characteristics in mind, such as hamming distance, reverse hamming distance, GC content, three or four letter alphabet, and unique subwords. In each of the tests, there indicate that no secondary structures would form. This prompted us to ask the following question: When designing DNA words, what considerations are most important and can we get a ballpark "range" on acceptable values for them?

5.1

Before answering this, we must first develop a testing procedure, and acceptable limits for our testing. Therefore we ask the following question: Given a set of words S is there an acceptable limit n, for which we can state that no word in S^* will have a secondary structure if no word in S^n has secondary structure?

5.2

Our hypothesis is yes, this limit exists, and we hope to prove it in the following way:

Any strand with secondary structure will have a substructure *str* with positive energy held together by a set of stacked pairs. The energy of *str* will either be greater than or less than the energy of the smallest possible hairpin creatable by deleting whole unwanted words from *str*.

5.2.1 Case 1: *str* has energy > smallest possible hairpin

In this case, we can replace *str* with the smallest possible hairpin, creating *str'*. $|str'|$ gives us a bounds n for testing.

5.2.2 Case 2: *str* has energy < smallest possible hairpin

In this case, *str* can be built from the a context free grammar L, where L is defined as the following grammar.

$$
\begin{array}{rcl}
W & \rightarrow & tW|V_0 \\
V_0 & \rightarrow & tV_1\bar{t}|tI_0\bar{t}|tM_0\bar{t} \\
V_1 & \rightarrow & tV_2\bar{t}|H_1|tI_1\bar{t}|tM_1\bar{t} \\
V_2 & \rightarrow & tV_3\bar{t}|H_2|tI_2\bar{t}|tM_2\bar{t} \\
V_3 & \rightarrow & H_3|tI_3\bar{t}|tM_3\bar{t} \\
H_3 & \rightarrow & tt_1t_2t_3\bar{t} \\
H_2 & \rightarrow & tt_1t_2\bar{t} \\
H_1 & \rightarrow & tt_1\bar{t} \\
I_1 & \rightarrow & tV_0|V_0t \\
I_2 & \rightarrow & tV_1|V_1t|tI_1|I_1t \\
I_3 & \rightarrow & tV_2|V_2t|tI_2|I_2t \\
M_0 & \rightarrow & WM_0WM_0 \\
M_1 & \rightarrow & WM_0WM_1|WM_1WM_0 \\
M_2 & \rightarrow & WM_1WM_1|WM_2WM_0|WM_0WM_2 \\
M_3 & \rightarrow & WM_0WM_3|WM_1WM_2|WM_2WM_1|WM_3WM_0 \\
WM_0 & \rightarrow & V_0WM_0|V_0 \\
WM_1 & \rightarrow & tWM_0|V_0WM_1|V_1WM_0|V_1|t \\
WM_2 & \rightarrow & tWM_1|V_0WM_2|V_1WM_1|V_2WM_0|V_2 \\
WM_3 & \rightarrow & tWM_2|V_0WM_3|V_1WM_2|V_2WM_1|V_3WM_0|V_3
\end{array}
$$

Thus to determine if any of these exist in S^* we run a context free grammar on LS^* and see if it is empty. If it is not empty, we can find the pumping length, p, of the context free grammar, to get a bounds for testing. The maximum bounds for testing will be $\max\{n, p\}$.

6 Future Work

The most practical direction for future work would be to preform further tests on RNA word sets to determine which elements of word design are most important. These tests would determine which of the more common constraints of word design, GC percent contraint, Hamming constraint, reverse-complement constraint, the reverse constraint, and the free energy constraint [1], are most relevant to word design. This task would include determining ways to hold all but one constraint constant, while varying the other over a range of values. This is a very difficult and time consuming task, as all the variables are dependent on one another. The program that Barry Cohen and Steven Skiena developed could also be extended in many ways to test word design problems involving words of different lengths. In many current word designs, spacers of a smaller length than the words are placed between the words to reduce the secondary structure occurances. A program which could test these spaces would be useful in many instances.

References

[1] A. Marathe, A. Condon, R. Corn. *On Combinatorial DNA word design.* DNA based Computers V, DIMACS Series, E. Winfree, D. Gifford Eds., AMS Press, 2000, 75-89.

[2] R. B. Lyngso, M. Zuker, and C. N. S. Pedersen. *An improved algorithm for RNA secondary structure prediction.* Tech. report BRICS-RS-99-15, Aarhus Univ., Datalogisk afdeling, May1999. http://www.daimi.aau.dk/ cstorm/papers/brics$_r$*na.ps*

[3] B. Cohen and S. Skiena. *Optimizing RNA Secondary Structure Over All Possible Encouding of a Given Protein.* Currents in Computational Molecular Biology, 2000, Universal Academy Press, Tokyo.