*Bioinformatics*

# An algorithmic approach to predicting RNA secondary structures.

Sebastian Dunn

Received on 16th October 2013


## ABSTRACT

This applications note seeks to introduce the reasoning behind RNA secondary structure predicting algorithms. It will explore the concept using a simple base-pair maximisation algorithm, and then discuss the optimisations employed by more complex algorithms to increase the accuracy of their predictions.


## 1 INTRODUCTION

In addition to transferring information, RNA plays a large role in the cell as a 3D-structured molecule. Ribosomal RNA makes up 60% of the ribosome used in protein synthesis, and the shape of transfer RNA is essential to its function. Hence there is a lot of interest in predicting the secondary structure of RNA from its base sequence.

RNA secondary structure prediction can be thought of as similar to finding palindromes in text. Loops of RNA fold back in on themselves and pair with their complementary bases to hold their structure. However, these loops may be nested inside one another, and the pairing regions that hold the loops may be close or far away from each other, so simply looking for palindromes isn't enough. We need to be able to look at the whole sequence at the same time and predict the optimal folding pattern, taking each potential conformer into account.

Dynamic programming is used to build this larger picture from smaller local optimals in the RNA sequence. The difficulty of most of these algorithms comes from very complex scoring structures – I will strip a lot of this away and focus on the core algorithmic technique.

## 2 ALGORITHM: BASE PAIR MAXIMISATION

To demonstrate the concepts of RNA folding algorithms I have implemented a base pair maximisation program. This uses a simpler scoring system of +1 for a base pairing, 0 for anything else, thereby maximising the total number of base pairs in the final solution.

Consider a subsequence of RNA, beginning at i and ending at j. The key to this algorithm is realising that the optimal score of this subsequence, s(i,j), is defined by the optimal score of a smaller subsequence, with only four possible cases that need to be considered (figure 1). Hence the optimal score for any subsequence can be defined recursively from its smaller subsequences.

The four cases as shown in figure 1:

(1) Bases i and j are paired, added on to the substructure for i+1…j-1.

(2) Base i is unpaired, added on to the structure for i+1…j.

(3) Base j is unpaired, added on to the structure for i…j-1.

(4) Both i and j are paired, but not each other. This is called a bifurcation, where the the total score for i…j is the addition of the scores of two structures i…k and k+1…j for some index i < k < j.

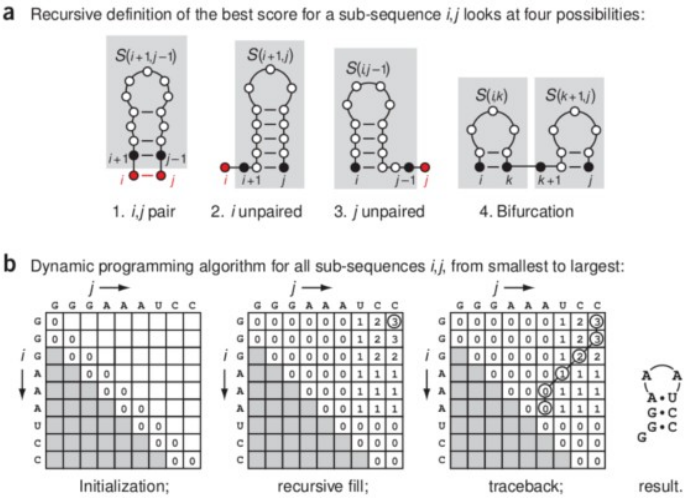


Figure 1: a) The four possible cases considered by an RNA folding recursion relationship. b) The three stages in the progression of the algorithm.

Now that we understand the cases, we need to decide how to score them. We have defined a score of +1 for each base pairing, 0 for any other case. Consider case 1: since we are using this relatively simple model we can assert that the score s(i, j) is independent of the structure of i+1…j-1. Therefore the score for the optimal conformation of i…j is equal to the score s(i+1, j-1) + 1: it is the same conformation with one additional base pair. Similarly we can argue that in case 2, the score s(i, j) is simply equal to s(i+1, j): there are no additional base pairs. The score of s(i, j) in case 3 is equal to s(i, j-1). In case 4 we can assert that the score for s(i, k) is

independent of the structure for k+1…j, and vice versa. Therefore s(i, j) is equal to s(i, k) + s(k+1, j). We have now defined our recursive relationship:

s(i, j) = max:

- s(i+1, j-1) + 1, if i and j form a base pair
- s(i+1, j)
- s(i, j-l)
- $\max_{i<k<j}$ s(i, k) + s(k+1, j)

As with any dynamic algorithm, we need to ensure that every smaller case is scrutinised before that value is accessed by a higher level case. We declare an N by N array where each element (i, j) represents the optimal score of the subsequence bounded by i and j (figure 1.b). We initialise sequences of length 1 or 0 to 0 – each element (i, i) and (i, i-1). Then we fill the array outwards, assessing larger and larger sequences with our recursive relationship until we reach the top right hand corner, (1, N), representing the whole RNA sequence. The score in this element will be the largest in the array and represent the number of base pairs in the whole sequence.

After we have the optimals grid filled in, it is relatively simple to trace back from the maximum to work out which bases are paired. In my implementation I kept a separate N by N by 2 array, which held a flag for each value in my optimals array denoting how that value was decided. This array was three dimensional because if the score came from a bifurcation, the k value of that bifurcation also needed to be stored.

# 3 COMPLEXITY

## 3.1 Space Complexity

Four matrices are stored in my implementation of this algorithm:

(1) An arraylist of length N, storing the characters of the bases at each index.

(2) An array of length N storing the paired base for every index (or -1 if unpaired).

(3) An N x N matrix storing the optimal score of any sub sequence (i, j).

(4) An N x N x 2 backtrack matrix storing the relationship that lead to the corresponding score in the optimals matrix – 1, 2, 3 or 4, in line with those cases illustrated in Figure 1.a. The third dimension is so that the k value of any bifurcation can also be stored.

Overall the space complexity is therefore $O(N^2)$. Even though there is a three dimensional backtrack array, the third dimension is always constant at 2.

## 3.2 Time Complexity

As with sequence alignment dynamic programming algorithms, this one fills in a triangle matrix of order $O(N^2)$. But unlike sequence alignment, the actions it takes for each cell are not constant. To calculate the optimal score, each loop must check every possible bifurcation around every possible value of k, leading to another layer of complexity of magnitude N. This means that overall, calculating the optimal values is $O(N^3)$ in regards to the number of bases in the RNA sample.

My implementation also uses a recursive function to trace backwards through the optimals and backtrack arrays in order to calculate the exact pairings that lead to the optimal score. This function makes one recursive call for each of the basic cases encountered in the backtrack array, and two for a bifurcation. However, it still only saves each pairing once, so the complexity of this function is O(N).

The complexity of the recursive function is dwarfed by the central algorithm, resulting in an overall complexity of $O(N^3)$.

# 4 OPTIMISATIONS

My implementation uses a comparatively simple base pair maximisation scoring system, with the only constraint being that each loop must have a minimum of 3 unpaired bases within it. Most implementations score the potential conformers on the overall free energy of their structure. RNA is much more likely to form a secondary structure with minimised global energy than one just with the most base pairs.

These energy scores are devised from combinations of elements of the secondary structure: different types and sizes of loops. and different stacking interactions of adjacent base pairings. The thermodynamic model has evolved as these algorithms have gotten more complex, but still only seems to be accurate to within about 5-10% (Eddy 2004), and a lot of variation in structure can happen within a 5-10% margin of global energy minimums. Accordingly, current leading software in this field still only score between 65-75% accuracy of structure prediction (Hajiaghayi, Condon and Hoos 2012). However, this is still much better that my implementation would manage by maximising the number of base pairs.

# REFERENCES

Eddy S.R. (2004) How do RNA folding algorithms work? *Nature Biotechnology.* **22,** 13, pp. 1457-1458

Hajiaghayi M., Condon A. & Hoos H.H. (2012) Analysis of energy-based algorithms for RNA secondary structure prediction. *BMC Bioinformatics.* **13,** 22