Zuker Algorithm Implementation and Testing

Michale Worrell, Mary Zhang

November 2021

1 Introduction

1.1 Background

RNA, or ribonucleic acids, is an molecule with many important functions in the biological system, including transporting amino acids for translation and acting as enzymes in base editing. Unlike DNA, RNA is often single-stranded and forms pairs with itself; it also uses the base U instead of T [1]. Base pairs in RNA form loops, which might stabilize or destabilize the structure. A lot of algorithms were developed to predict RNA secondary structures based on RNA sequence and hypotheses on the secondary structure forming process, and the Zuker algorithm is one of them. Similar to the sequence alignment algorithms that we have learned in class [2], the Zuker algorithm also relies on dynamic programming. The process involves filling out matrices that store information about the sequence, then performing traceback for an optimal case. Since different layers of information needed to be retrieved, multiple tables are needed, a little similar to the sequence alignment with affine gap algorithm.

1.2 Zuker Algorithm Assumption

Let Sq(i, j) denote the input RNA sequence starting at nucleotide i and ending at nucleotide j, P the set of structures that Sq(i, j) can possibly take. For potential pairs (S[a], S[b]), the bases S[a] and S[b] must be complementary to each other and satisfies $b > a$. Let eP denote the free energy of a specific structure. Here $\forall(S[i], S[j]) \notin P$ are unpaired.

1. Minimal Loop: $\forall(S[i], S[j]) \in P : j - i \geq m$. This assumption makes sense because loops that are too small are energetically unfavorable.

2. Free energy Minimization: Let E denotes the total free energy of a specific structure P'. Here, E(P') = $\sum_{(S[i],S[j]) \in P'} E_{(S[i],S[j])}$. The Zuker algorithm output P'$\in$ P: $E(P') \leq E(P \neq P')$, essentially the structure with minimized total free energy. To put this assumption in context of chemistry, the Zuker algorithm is really calculating the change in free energy caused by RNA folding, and the output would be the structure that reduces the

free energy the most. It can be inferred that a sequence with no folding would have a free energy change value of 0.

3. Noncrossing loops: $\forall(S[i], S[j]), S([i'], S[j']) \in P : i < j < i' < j'$ or $i' < j' < i < j$ or $i < i' < j' < j$ or $i' < i < j < j'$. This assumption determines that the Zuker algorithm would not predict a structure called pseudoknot, which is known to be extremely hard for dynamic programming algorithms to compute [3].

1.3 Loops

Defining different types of loops are important because they have different contributions to free energy and require different information in traceback.

1. Hairpin loop H(i, j): $(S[i], S[j]) \in P$ and $\forall(S[i'], S[j']), i < i' < j' < j : (S[i'], S[j']) \notin P$. Figuratively, the hairpin loop can be understood as sealing a section of unpaired bases.

2. Stacking loop S(i, j): $(S[i], S[j]) \in P$ and $(S[i+1], S[j-1]) \in P$. The stacking loop can be seen as pairs that extend from existing pairs.

3. Internal loop I(i, j):

   (a) $(S[i], S[j]) \in P$

   (b) $\exists(S[i'], S[j') \in P : i < i' < j' < j$.

   (c) (S[i], S[j]) $\neq$ S(i, j)

   (d) $\forall S[k], S[h] : i < k < i', j' < h < j$ unpaired

   This can be seen as structures that have partial openings between two pairs in a closed stacking loop.

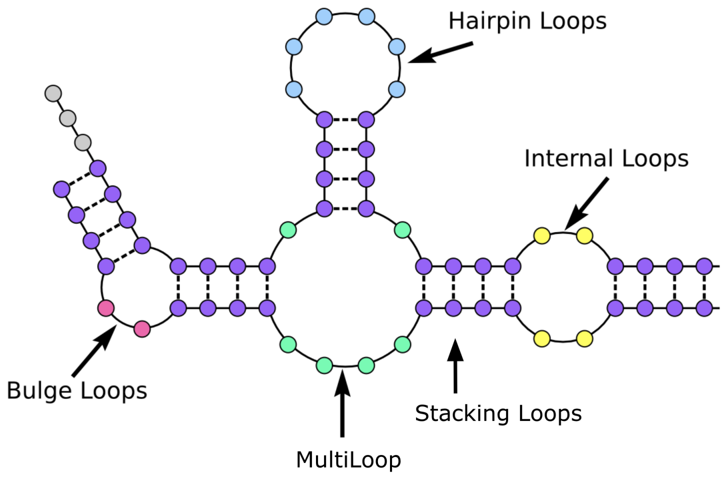
4. Multiloop M(i, j):

   (a) $(S[i], S[j]) \in P$

   (b) $\exists(S[i'], S[j') \in P : i < i' < j' < j$; specifically, more than 1 (S[i'], S[j']) that satisfies this requirement

   (c) $\forall S[k], S[h] : i < k < i', j' < h < j$ unpaired

   This can be seen as a loop that branches into two or more smaller loops.



2 Implementation

2.1 Tables and Initialization

Three tables were built for Zuker. W(i, j) stores the minimal eSq(i,j). V(i, j) stores the minimal eSq(i,j), assuming that (S[i], S[j]) $\in$ {H(i,j), S(i,j), I(i,j), M(i,j)}. WM(i, j) stores the minimal eSq(i,j), assuming that (S[i], S[j]) = M(i, j). Because of the minimal loop assumption, any entries (i,j) in W, V and WM where j - i < m are initialized to a default value that will not be chosen as optimal. In W, this default is 0, and in V and WM this default is positive infinity. This is because we are finding minimal energy of a certain substructure in V and WM, so the impossible case (a loop smaller than the minimal length) is set to positive infinity to be avoided. For W, according to the free energy minimization assumption, we assume that the free energy for sequences that do not form loops is 0.

2.2 Recursive Table Filling

In dynamic programming, recursively filling out tables usually takes the principle of computing values from all possible cases and chose the optimal value to fill in. Filling out tables in Zuker algorithm is no different.

- Filling out W: W(i,j) = min(W(i, j-1), $min_{i \leq k \leq j-m}$ {W(i, k-1)+ V(k, j)})
  - W computes the possible situations of S[j] either involve or does not involve in pairing and these two cases would be referred as case 1 and case 2 for table W respectively later. The optimal free energy would be recorded in the table.
- Filling out V: V(i, j) = min(eH(i,j), V(i+1, j-1)+ eS(i,j), $min_{i<i'<j'<j}$eI(i,j) +V(i',j'), $min_{i<k<j}$WM(i+1,k)+WM(k+1, j-1)+a)

  - Here there are four cases corresponding to the four loop types. Hairpin, stacking, internal and multiloops would be referred as case 1, 2, 3, 4 respectively for table V later.
  - For the multiloop case, the logic behind is that it is assumed that at least two loops are contained by the subsection (i,j). Since the multiloop is an especially complicated and a special table WM is devoted to multiloops, in V we only need to reduce the subsection smaller and let WM handle the rest. So here we attempt different ways of splitting the subsection (i+1,j-1) (we already know that (S[i], S[j]) is a pair) into two smaller multiloops where multiloop computations are performed. Here a is a constant as the energy contribution of the pair (S[i], S[j]) for sealing the multiloop.

- Filling out WM: WM(i, j) = min(WM(i,j-1)+c, WM(i+1, j)+c, V(i,j)+b, $min_{i<k<j}$WM(i,k)+WM(k+1,j))
  - According to the assumption of table WM, it is essentially implied that multiloops contain loops over a smaller subsection and these loops can be handled in V, which would divide the problems even smaller. There are different ways that these smaller loops can be reached.
  - 1) It could be that S[i] do not involve in such loops but S[j] does or vice versa. These would be referred as case 1 and 2 for table WM respectively later. Here c is another constant as the energy contribution for loops contained in multiloops.
  - 2) It could be that S[i] and S[j] pair with each other and the smaller loop problem is just V(i, j). This would be referred as case 3 for table WM later. Here a different constant is assigned to be the energy contribution.
  - 3) It could be that both S[i] and S[j] involve in pairing but in two separate loops. This would be referred as case 4 for table WM later. Here the subsection (i, j) is divided into smaller sections to locate the pairing partners of S[i] and S[j].

More information about the overall recursion relation can be found in our index.

2.3 Table Filling Order

Filling out the tables in order requires frequently switching between filling slices of entries in table V and filling slices of entries in table WM. This is similar to the strategy used in the affine-gap alignment algorithm, except our method of filling out tables in Zuker's algorithm alternates between filling out large subsections of V and filling out large subsections of WM, and only filling out table W once all of tables V and WM have been filled. In our implementaion process, we have found our that the energies of all subsequences of the same length are not interdependent. Figurtively on the table, these subsequences rest along the same diagonal slice of each table (see Index). Therefore, we first need to know the table entries each slice refers back to. For all subsequences of length n' that are part of slice n':

- Filling out their appropriate entries in table W requires knowing all slices x in W where $x < n'$ (for W case 1), as well as knowing all slices x in V where $x \leq n'$ (for W case 2).
- Filling out their appropriate entries in table V requires knowing all slices x in V where $x < n'$ (for V case 2, 3), as well as knowing all slices x in WM where $x < n'$ (for V case 4).
- Filling out their appropriate entries in table WM requires knowing all slices x in WM where $x < n'$ (for WM case 1,2 and 4), as well as knowing slices n' in V (for WM case 3).

Given such information, here is how we actually fill out the tables:

- While n' > m and n' < j-i+1:
    1. Fill out all entries in table V that are part of slice n'
    2. Fill all entries in table WM that are part of slice n'
    3. Move on to the next slice (increment n' by 1)
- Once tables V and WM have been filled, fill out table W, slice by slice.

2.4 Traceback

Different information are needed for tracebacking different loops. To tackle this problem, we created a traceback object for each entry in the three tables to store information about the case number as well as the index to visit next. For W and WM, the index takes one integer because the bigger problem is divided into two smaller problems by one separating index; for V case 3, two integers are needed for the index, which store the inner loop of the internal loop. The traceback process starts from the top right corner of table W because according to our slice-filling strategy, it is filled last. Then we traverse through the three tables indicated by the traceback objects encountered along the way, and this would give us the optimal RNA secondary structure computed.

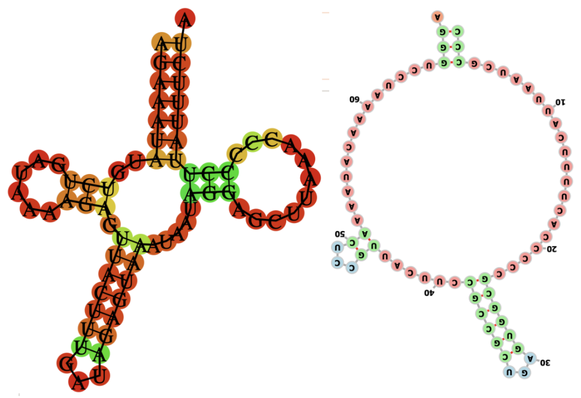
3 Results

3.1 Implementation Choice

In our implementation, we chose to output the RNA secondary structure in the dot-bracket format [4], which is widely-used and we can use it to visualize the structure with visualization tools online [5]. But we do encounter a problem in our implementation with energy contributions. This part is similar to the penalty table we have in sequence alignment algorithm. The Zuker algorithm has very complicated energy assignment for different types of loops that take into account of the length of unpaired potion and even some experimentally determined data. Here we modified the methods [6] used in RNAFold [7], the web app that utilizes Zuker algorithm to predict RNA secondary structure. This method takes into account the type of loops, the length of the loops, the base pairs that seal the loop and more [8]. More details can be found in our implementation.

3.2 Prediction Results Comparisons

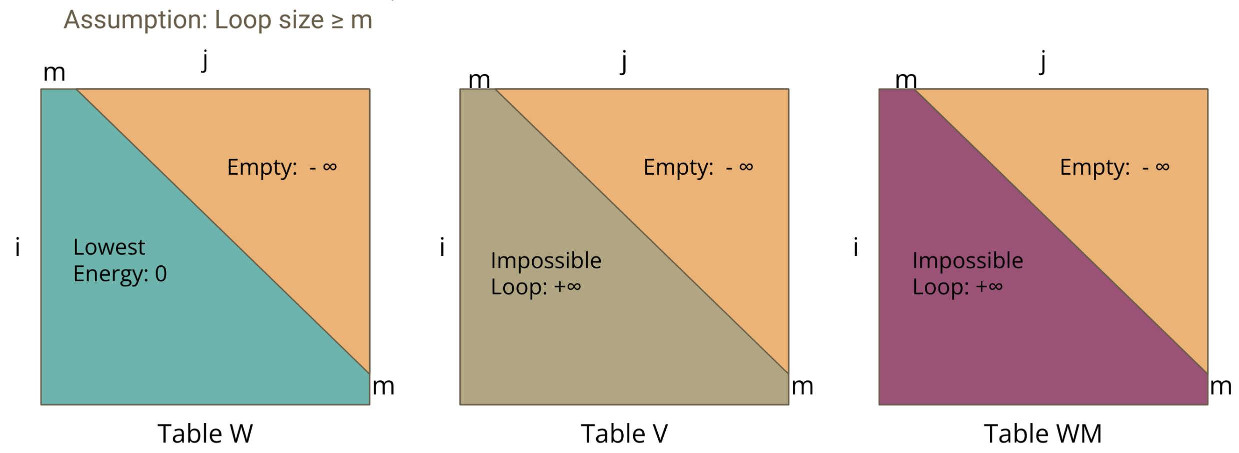
To test our implementaion, we chose to predict the structure of MT-TI, a human mitochondrial transfer RNA [9]. Transfer RNAs (tRNAs) are famous for their cloverleaf-shaped secondary structure [10]. tRNAs use one end to carry amino acids and the other end to match to mRNA sequences that are being translated. We chose MT-TI because though no experimentally determined MT-TI structures are unavailable, tRNA secondary structures are well-studied and would act as a good reference for evaluating our implementation. We retrieved the sequence of MT-TI from NCBI and the sequence is also included in our code package to be the test data. We used RNAFold to predict its structure and compared it with our results. In the figure below, the left panel shows the structure predicted by RNAFold, the right panel shows the structure predicted by our algorithm, which we had to flip to match the way structures are visualized by RNAFold.



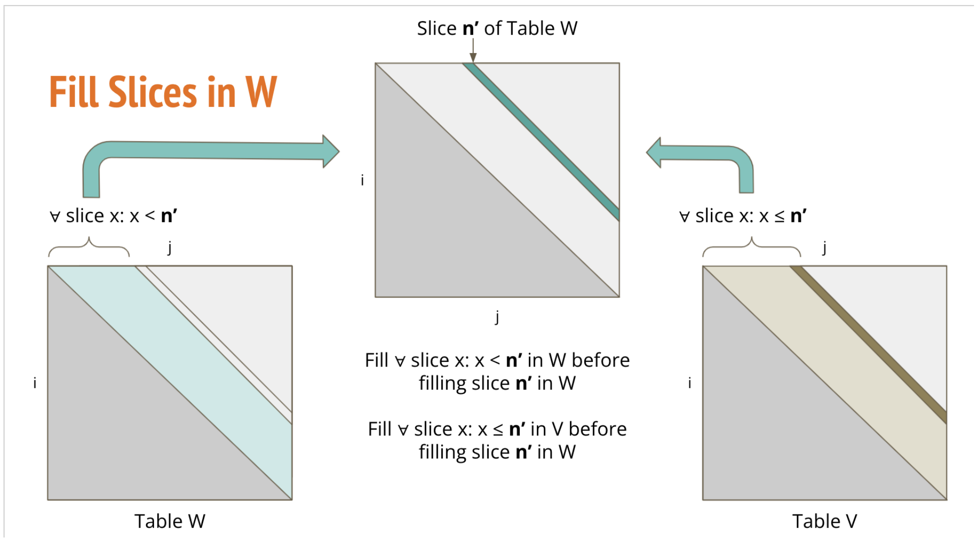
It can be seen that our predicted structure is very different from the RNAFold structure with more unpaired bases. The predicted free energy change is also very different: RNAFold predicted -7.5kcal/mol for the specific structure, our implementation predicted -19.6kcal/mol. From the fact that our structure had more unpaired bases but more energy minimization, we figured that the main reason behind the difference in prediction was the energy contribution functions. Within the limited time, we were only able to implement a rudimentary version of the energy contribution functions. It could be that our lack of consideration for special hairpin energies, internal bulge loops and use of different energy parameters contribute to the difference in prediction results.

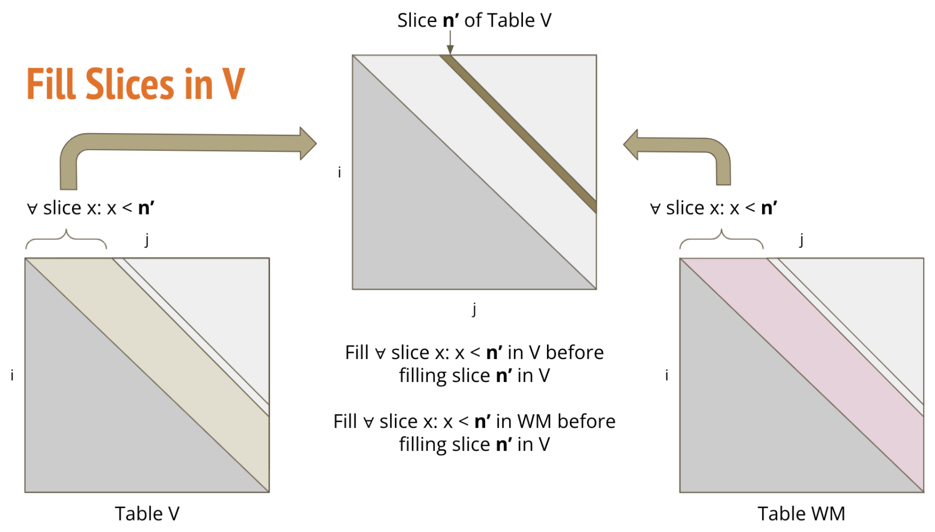
4 Index

A visual representation of the way the Zuker tables are initialized.

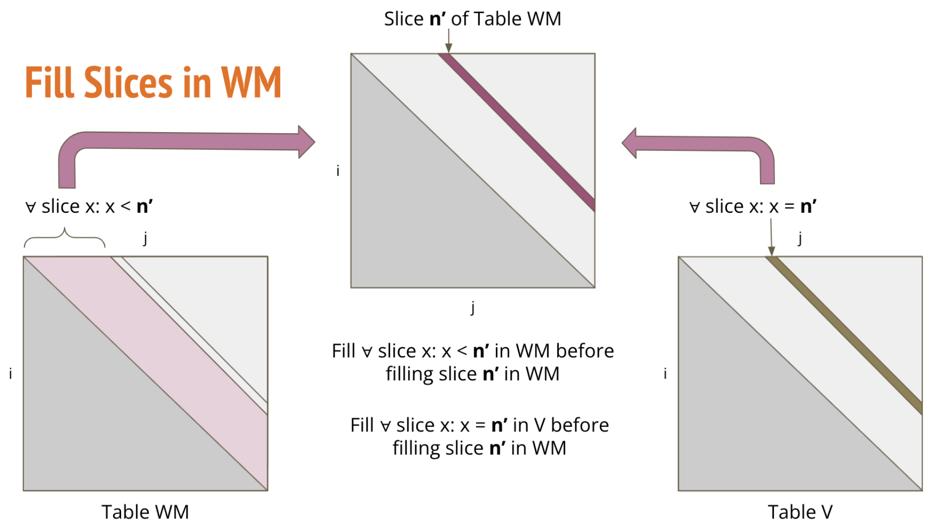


A visual representation of the order of filling out Zuker tables slice by slice.


Fill Slices in W
Slice n' of Table W
∀ slice x: x < n'
∀ slice x: x ≤ n'
i
j
Fill ∀ slice x: x < n' in W before filling slice n' in W
Fill ∀ slice x: x ≤ n' in V before filling slice n' in W
Table W
Table V



Fill Slices in V
Slice **n'** of Table V
∀ slice x: x < **n'**
∀ slice x: x < **n'**
i
j
i
j
i
j
Fill ∀ slice x: x < **n'** in V before filling slice **n'** in V
Fill ∀ slice x: x < **n'** in WM before filling slice **n'** in V
Table V
Table WM



Fill Slices in WM
Slice **n'** of Table WM
∀ slice x: x < **n'**
∀ slice x: x = **n'**
i
j
i
j
i
j
Fill ∀ slice x: x < **n'** in WM before filling slice **n'** in WM
Fill ∀ slice x: x = **n'** in V before filling slice **n'** in WM
Table WM
Table V

References

[1] "Ribonucleic acid (rna)." [Online]. Available: https://www.genome.gov/genetics-glossary/RNA-Ribonucleic-Acid

[2] "Zuker algorithm." [Online]. Available: https://math.mit.edu/classes/18.417/Slides/rna-prediction-zuker.pdf

[3] "Pseudoknot," Sep 2021. [Online]. Available: https://en.wikipedia.org/wiki/Pseudoknot

[4] "Rna secondary structure formats." [Online]. Available: https://software.broadinstitute.org/software/igv/RNAsecStructure

[5] S. Hammer and P. Kerpedjiev, "Viennarna web services." [Online]. Available: http://rna.tbi.univie.ac.at/forna/.

[6] "Turner 2004 nearest neighbors." [Online]. Available: https://rna.urmc.rochester.edu/NNDB/turner04/index.html

[7] "Rnafold." [Online]. Available: http://rna.tbi.univie.ac.at/cgi-bin/RNAWebSuite/RNAfold.cgi

[8] D. H. Mathews, M. D. Disney, J. L. Childs, S. J. Schroeder, M. Zuker, and D. H. Turner, "Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of rna secondary structure," *Proceedings of the National Academy of Sciences*, vol. 101, no. 19, p. 7287–7292, 2004.

[9] "Mt-ti," Sep 2021. [Online]. Available: https://en.wikipedia.org/wiki/MT-TI

[10] "Cloverleaf model of trna," Oct 2020. [Online]. Available: https://en.wikipedia.org/wiki/Cloverleaf$_{m}odel_{o}f_{t}RNA$