RNA Folding Algorithms

By

Mehak

A dissertation submitted for the partial fulfilment of MSc.

degree in Science

Indian Institute of Science Education and Research

Mohali

April 2021

Certificate of Examination

This is to certify that the dissertation titled "**RNA Folding Algorithms**" submitted by **Mehak** (Reg.No. MP18008) for the partial fulfilment of MSc programme of the Institute, has been examined by the thesis committee duly appointed by the Institute. The committee finds the work done by the candidate satisfactory and recommends that the report be accepted.

Dr. Abhishek Chaudhari	Dr. Shashi Bhushan Pandit	Dr. Monika Sharma
(Committee member)	(Committee member)	(Supervisor)

Dated: April 30, 2021

Declaration

The work presented in this dissertation has been carried out by me under the guidance of Dr. Monika Sharma at the Indian Institute of Science Education and Research Mohali.

This work has not been submitted in part or in full for a degree, a diploma, or a fellowship to any other university or institute. Whenever contributions of others are involved, every effort is made to indicate this clearly, with due acknowledgement of collaborative research and discussions. This thesis is a bonafide record of original work done by me and all sources listed within have been detailed in the bibliography.

Mehak.
Mehak

(Candidate)

Dated: April 30, 2021

In my capacity as the supervisor of the candidate's project work, I certify that the above statements by the candidate are true to the best of my knowledge.

Monika Sharma
Dr. Monika Sharma

(Supervisor)

Acknowledgements

I would like to thank my supervisor Dr. Monika Sharma, for assigning me this project and giving me the opportunity to explore this field under her guidance and constantly helping me throughout the writing of this dissertation.

Deep gratitude to Dr. Abhishek Chaudhari, who enlightened me at every step of my journey in IISER Mohali and inculcated my interest in interdisciplinary sciences.

I would like to thank my friends Saurabh Annadale and Abhijeet Singh for their assistance and cooperation in learning the python programming language and writing codes.

My heartfelt appreciation extends to IISER Mohali Library for the kind of support they have been providing us throughout this pandemic by keeping the library open and ensuring that we can access every kind of resource (books, journals, research papers) for our research.

This acknowledgement would never really be justified without thanking my friends, especially Lakshita, Abhijeet, Sasank and Saurav, who had always encouraged me to give my best efforts. They were always there for me through my good and bad times. I also want to thank IISER Mohali for allowing me to be a part of this beautiful three-year journey in the company of some amazing friends.

Finally, I would like to acknowledge with gratitude the support and love of my family. They all kept me going, no matter how difficult the situation was, and this thesis work would not have been possible without them.

Abstract

Prediction of RNA secondary structures is a problem of considerable importance to biologists. A sequence of RNA folds onto itself to attain the most stable thermodynamic structure. The prediction of secondary structure is beneficial in predicting the tertiary structure and the biological functions RNA performs in living beings. Research in dynamic programming algorithms has led to the prediction of the most stable RNA secondary structures. Nussinov's and Zuker's algorithm predict RNA secondary structure without pseudoknots.

This thesis deals with a review of these two RNA folding algorithms which do not involve pseudoknots. I have implemented these algorithms in python and tried to visualize the obtained structures. I have tried some variations in the code resulting in different structures. I have drawn comparisons in these algorithms based on the results obtained. Nussinov's algorithm deals with maximization of the base pairs for the given sequence whereas Zuker's algorithm incorporates information regarding the neighbouring loops. The results suggest that a better knowledge of the chemical and biological aspects of RNA needs to be incorporated in these algorithms to attain the most stable structures.

Contents

List of Figures

Chapter 1

RNA Secondary Structure Prediction

1.1 RNA Structure

In an RNA molecule, ribonucleotides are connected together by covalent chemical bonds and every ribonuceotide comprises of the four bases: adenine (A), cytosine (C), guanine (G), uracil (U). The RNA structure is hierarchical in the sense that:

1. the first level of organization is the primary structure which is the sequence of the bases

2. the second level of organization is the set of the base pairs which is known as the secondary structure of the RNA molecule

3. the third level of organization is the three dimensional orientation of the various atoms which is called the tertiary structure

The complementary bases (A-U, G-C) which lie in close to each other form hydrogen bonds. Other base pairs such as G-U (wobble pair having 1 hydrogen bond) are also possible but they have low probability of pairing since they are energetically less favourable. The RNA strand comprises of alternating phosphates and ribose sugars. A nucleotide is a combination of single phosphate, ribose and base. The phosphate group gets attached to the 5' end of the ribose sugar and the base is attached to the 1' carbon atom. Chain

formation occurs when the OH group attached to the 3' end and the OH group attached to the neighbouring nucleotide form a bond. When a large number of nucleotides connect in this way, an RNA polymer is formed. The RNA sequence is numbered from the 5' end (carbon with the phosphate group) to the 3' end (carbon atom attached to the ribose). Since adenine and guanine are the derivatives of purine, hence they are called purine bases. Cytosine and uracil are the derivatives of pyrimidine, therefore they are called pyrimidine bases. A purine base pairs up with a pyrimidine base. This is because pyrimidines are smaller compared to purines, therefore combination of two pyrimidines is energetically unfavourable as the bonding atoms would end up being far from each other. Similarly two purines would end up being too close to each other and would repel.

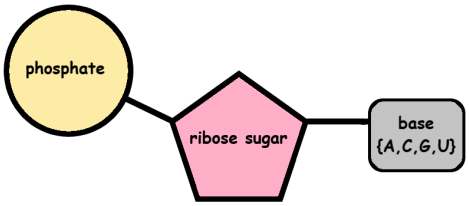

Figure 1.1.1: Basic RNA structure

1.2 Importance of structure prediction

Knowledge of the secondary structure helps in the determining the tertiary structure which thereby permits the biological function of the RNA molecule. Since RNAs are involved in many biological processes such as acting like catalytic molecules, protein synthesis etc the problem of RNA secondary structure prediction is of huge biological importance. The experimental techniques used to determine the tertiary structure are extremely expensive and time consuming and are usually unable to determine the structure

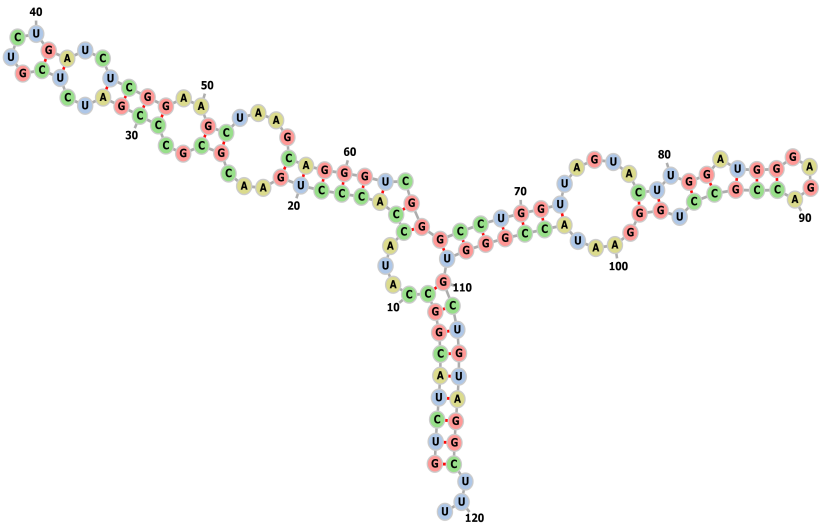

Figure 1.1.2: Example of folded structure of RNA5S1, drawn in Forna

completely. Therefore to understand the roles of RNA, it is important to investigate RNA structures with the use of computation. Apart from this, structures of many sequences have not yet been successfully determined by experimental techniques, hence it is essential to predict the secondary structures. It has been observed that the secondary structure contacts are stronger than tertiary structure contacts and the bond formation is faster in case of secondary structures.

1.3 Dynamic Programming

Computational methods help in searching secondary structures for RNA sequences to get the minimum free energy structures. For this purpose many dynamic algorithms have been designed and implemented to obtain the possible secondary structures. The simplest way of finding the minimum free energy structure is to generate all the possible structures and from them, find the most optimal structure. It is a cumbersome job as it has been found that the total possible structures for a given sequence of length, N grows exponentially as 1.8^N. Even for small sequences, the possible structures would be large

and it would not be possible for present day computers to handle such a computation. Dynamic programming algorithms search all possible structures without actually generating them. I will be discussing two such algorithms, Nussinov's algorithm and Zuker's algorithm in great detail. The procedure is basically a two steps process. The first step also known as the fill stage involves finding minimum free energy for subsequences of shorter lengths and then moving towards longer fragments. The first step ends when the minimum free energy of the complete sequence is obtained recursively. At this stage the lowest conformational energy is known but the structure remains unknown. The next step leads to identification of the structure, known as the trace back stage. The trace back loop runs to find the exact structure corresponding to the energy obtained in the first step. The execution time corresponding to these algorithms scales as $O(N^3)$. These algorithms are unable to predict structures with pseudoknots.

Chapter 2

Nussinov's Algorithm

2.1 Introduction

The algorithm[11] is based on a simple method of estimation of free energy of loops present in the single stranded structure of RNA sequence.This method helps in finding the the most stable structure by comparing the stability of all the possible structures for a given sequence. This approach worked quite well for small nucleotide sequences but for longer sequences it was cumbersome to find all the possible structures and compare all of them to get the most probable structure.

All the earlier suggested algorithms followed a similar basic approach to tackle the problem.

1. Identification of perfectly matched helices in the given sequence.
2. Creation of an assembly of consistent sets of these helices.
3. Calculation of the overall free energy of each assembled structure.

This approach was not feasible for long chains of nucleotide since the time required to calculate the energies of each structure was extremely long.

Nussinov's algorithm tackles the problem by constructing two-half matrices by following a procedure that considers the energy of all the individual base pairs. The most stable

loop structure having the lowest free energy is found from the second matrix. The algorithm was initially devised to maximize the base-pairs along a polynucleotide chain. It was considered that the formation of base-pairs reduces the free energy of the structure, hence maximizing the base-pairs leads to most stable free energy structure.

2.2 Basic principle behind the algorithm

The algorithm evaluates the contribution of the individual base pairs to the secondary structure of the RNA chain. The basic principle behind the algorithm can be understood by considering a chain of n nucleotides lying on the circumference of a circle. To visualize the base pairs, non-interrupting arcs can be drawn linking the specific bases. This kind of linking leads to planar secondary structures which do not support knots/ pseudo-knots.

To find the most optimal structure of n nucleotides long RNA chain, the following criteria is considered:

Whether the base-pair A_pA_q (A_p and A_q are the p^{th} and q^{th} bases in the sequence) is present or absent from the structure.

The bond formation not just affects the free energy of the local region where it is present but has a way more important contribution to the free energy of the overall structure. Since knots are not allowed in the secondary structure, therefore A_pA_q bond divides the the primary sequence into two sections. Further base-pairs are allowed either within this bond or outside the bond. The various bonds so formed contribute to the total free energy of the folded structure. The best folded structure is obtained by minimizing the free energy of three parameters:

- Energy of the inner section.
- Energy of the outer section.

- The local contribution of A_pA_q bond.

2.3 Maximizing the number of base-pairs

There are certain rules following which the most stable folded structure of the polynucleotide chain is obtained:

- The bond stability/strength of C-G and A-U pairs are given same preference.
- Stacking contributions are not considered.
- Destabilizing consequences of single stranded loops are ignored.

The algorithm tackles the problem by dividing it into sub-problems. It begins with a subsection of the given nucleotide sequence and proceeds to sections of increasing lengths. Optimal folding of the entire sequence is obtained by finding the optimal folding of the sub-sequences.

Algorithm can be understood by considering a sequence of length n from A_1 to A_n. Let's assume that the sequence contains a sub-sequence of length l from A_p to A_q.

Let $u \in [p, q-1]$. Now the algorithm checks $\forall u$ whether A_u can base-pair with A_q. To determine the number of base-pairs in the section $[p, q]$, the algorithm finds the total number of base-pairs in the sub-sections $[p, u-1]$ and $[u+1, q-1]$.

A matrix M is created. The purpose of the matrix is to store the best value after all k positions have been tested. If none of the A_k's pair with A_q, then:

$$M[p,q] = M[p,q-1] \tag{2.1}$$

This process repeats for other sections of length l by incrementing the values of p and q in succession. The search for maximum number of base pairs is repeated for sections of increasing lengths by successively incrementing the value of l. The values of $M[p,q]$

with the sequence are given implementing the following conditions:

$$M[p,q] = \max \begin{cases} M[p,u-1] + M[u+1,q-1] + 1 \\ M[p,q-1],\ p \leq u < q = p + l \end{cases} \tag{2.2}$$

The 1 in the equation (2) refers to the base pair $A_u A_q$. The matrix M gets filled up subsequently by following the above mentioned process and the last value in the matrix is obtained which is given as $M[1,n]$. This value signifies the maximum number of base pairs possible for the given sequence of length n.

2.4 Trace back stage

The structure in which the given polynucleotide folds is obtained by the identification of the individual base pairs in the sequence. The process of identification of all the base pairs contributing to the maximum score is called back-tracing. This is the final step of the algorithm to determine the secondary structure in which the given polynucleotide chain folds. The algorithm constructs another $n \times n$ matrix $U[p,q]$ containing the position of the base A_u allowing maximum base pairing within the section $[A_p, A_q]$. $U[1,n]$ stores the position of u which on pairing with A_n gives best folding of the sequence. This pair gives rise to two sub-sections of the sequence. In the same manner, optimal folding is obtained by reading the values $U[p,q]$ for every subsection. This process repeats in the order of decreasing lengths of the sub-sections. All the subsections formed in this manner are a result of the chosen base pairs generated to obtain the structure with maximal number of base pairs. The matrices M and U are half filled.

2.5 Minimal free energy structure

The previous algorithm finds the most optimal secondary structure of the given sequence by maximizing the number of base pairs but does not take into account any energy considerations. Another algorithm which follows similar search mechanism but obtains the structure having minimum free energy. The above algorithm can modified to get an estimate of free energy of the loop structures based on sequence data. The energy contributions from helices, single-stranded loops and bulges have been considered. Just like the previous algorithm, now the energy of each section of the sequence is given in the following manner:

$$E[p,q] = \min \begin{cases} E[p,u-1] + E[u+1,q-1] + E_{uq} \\ E[p,q-1],\ p \leq u < q - l_{min} \end{cases} \tag{2.3}$$

For every sub-section (from A_p to A_q), energy is estimated by examination of each possible base pair ($A_u A_q$) lying in that sub-section and the lowest energy obtained by this method is stored in matrix element $M[p,q]$. Position of A_u is stored in $U[p,q]$. Here l_{min} refers to the minimal allowed loop length. Different values of l_{min} leads to different optimal structures. The energy of the substructures depend on the kind of structure formed by the pairs, i.e., adjacent base pairs, single stranded bulge, internal loops or branched structures. Similar back tracing is followed here as well resulting in the most optimal folded structure from the U matrix. The structure adjacent to the base pair $A_u A_q$ is obtained in the following way by assuming the values of u_1 as:

- $u_1 = u + 1 \implies$ the base pair $A_u A_q$ lies adjacent to the pair $A_{u+1} A_{q-1}$
- $u_1 > u + 1 \implies$ a single stranded region (bulge or internal loop) occurs on the left of $A_u A_q$, e.g. $u_1 = 3$, $A_u A_q$ lies next to a bulge comprising of two bases on its left as A_{u+3} pairs with A_{q-1}.

- Base at position u_1 pairing with a base at position $q < q-1$ results in a bulge at the right of $A_u A_q$, e.g. $u_1 = 1$ and A_{u+1} pairs with A_{q-3} resulting in $A_u A_q$ lying next to a bulge consisting of two bases on the right.
- base at the position $u_1 > u+1$ pairing with base at $q < q-1$ gives rise to several independent hairpin loops lying within $A_u A_q$

2.6 Summary

An RNA sequence comprises of a string of bases $\in$ { A, U, G, C } of length l. A secondary structure is obtained as a list of ordered base pairs { A, U }, {G, C } and {U, G }. No pseudoknots are allowed and the bases are paired such that no two bases are paired lying next to each other. The most stable secondary structure is obtained via maximal base pairing. This is done as a scoring system. A square matrix of $l \times l$ is initialised such that the first two diagonals are set to zero:

- $M[i,i] = 0$ for $i \in [1,l]$
- $M[i,i-1] = 0$ for $i \in [2,l]$
- every base pair is assigned a score of 1
- every base which is left unpaired is assigned a score of 0

To maximize the score (maximum base pairs), algorithm states the following 4 rules to fill the matrix M thereby getting the best possible structure:

- If the best structure for sub sequence $\{p+1,q\}$ is obtained, an unpaired base at position p can be added to its left
- After having obtained the best structure for the sub sequence $\{p+1,q\}$, an unpaired base $q+1$ can be added to the right of it

- A base pair $p-1, q+1$ can be added to the best structure obtained for the sub sequence $\{p, q\}$
- Structures for two sub sequences can be combined, also known as bifurcation ($\{p, u\}$ combined with $\{u+1, q\}$), P and q are paired but not to each other

The final score is assigned by examining the maximum of the four rules:

$$M[p,q] = \max \begin{cases} M[p+1,q] \\ M[p,q-1] \\ M[p+1,q-1]+1 \\ \max \left\{ M[p,u] + M[u+1,q] \; u \in [p,q] \right. \end{cases} \tag{2.4}$$

Pairs which lead to the maximum score for the given sequence form the most optimal structure. Once the matrix M gets completely filled, the trace-back loop begins to find the contributing base pairs. The trace back algorithm is as follows:

- $x =$ length of the RNA sequence
- if $p < x$:
  - if:

    $M[p,x] = M[p+1,x] \implies$ traceback to pair the $(p+1, x)$
  - else if:

    $M[p,x] = M[p,x-1] \implies$ traceback to pair the $(p, x-1)$
  - else if:

    $M[p,x] = M[p+1,x] + M[p+1,x-1] \implies$ traceback to pair the $(p+1, x-1)$
  - else:

    for $u \in [p+1, x-1]$

if $M[p,x] = M[p,u] + M[u+1,x] \implies$ traceback to the pair $(p,u) \implies$ trace back to the pair $(u+1,x)$

2.7 Implementation of the algorithm

I have implemented the Nussinov's algorithm in python for different RNA sequences and tried to visualize their secondary structures. I have varied certain parameters to obtain different structures for the same sequence. This is done by varying the minimum allowed loop length and allowing non canonical (U, G) base pairs to couple.

1. I have tried to find the optimal structure of an RNA sequence = 'CUCCGGUUG-CAAUGUC' of 16 nucleotides. The minimal loop length is kept as 2 and only canonical pairs are allowed. This resulted in a total of 5 base-pairs. The dot bracket notation for the structure is obtained as (.(.))..(((.)).) and the time taken for the code to run is 0.03889 seconds. Figure 2.7.1 refers to the structure. Now the minimal loop length is set as 3 and the structure is obtained as (.(..)((..)).).. with 4 possible base-pairs. Figure 2.7.2 represents the structure for the same sequence with minimal loop length set as 3. On setting the minimal loop length as 4, Figure 2.7.3 is obtained as the optimal structure with 3 possible base pairs. Minimal loop length = 0 is avoided since structures with sharp turns do not exist. Similar structures can be obtained by allowing non canonical base pairs. For minimum loop length = 2 and allowing (U, G) to pair, Figure 2.7.4 is obtained as the optimal structure with a total of 5 base pairs. Keeping the minimal loop length = 3 and allowing non canonical base pairs yields Figure 2.7.5 as the optimal structure with a maximum of 4 base pairs and with minimal loop length = 4 yields Figure 2.7.6 as the most stable structure with 4 base pairs. Similar structures can be obtained by changing these parameters.

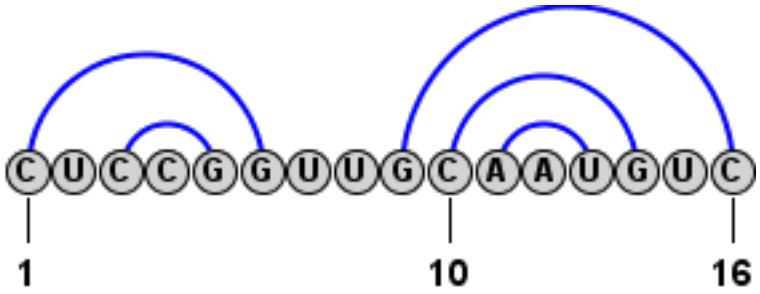

Figure 2.7.1: Optimal structure for 'CUCCGGUUGCAAUGUC' with minimal loop length = 2

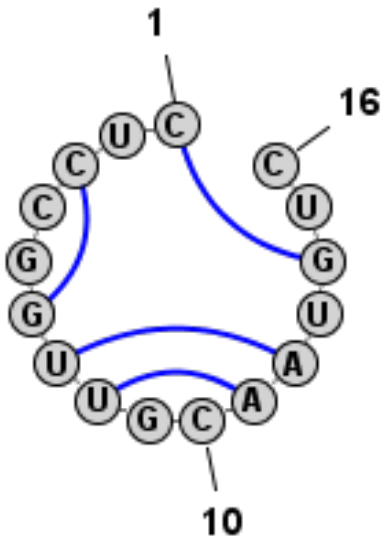

Figure 2.7.2: Optimal structure for 'CUCCGGUUGCAAUGUC' with minimal loop length = 3

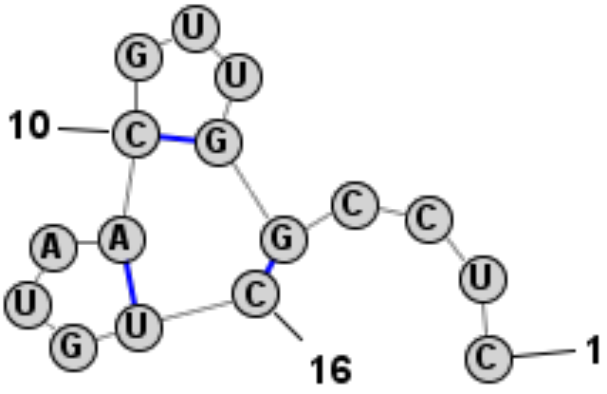

Figure 2.7.3: Optimal structure for 'CUCCGGUUGCAAUGUC' with minimal loop length = 4

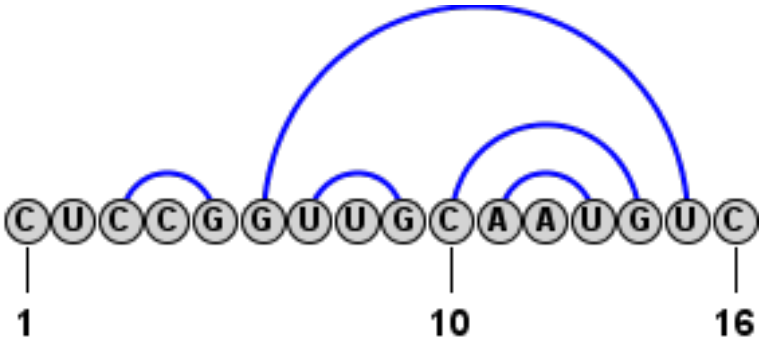

Figure 2.7.4: Optimal structure for 'CUCCGGUUGCAAUGUC' with minimal loop length = 2 and non canonical base pairs allowed

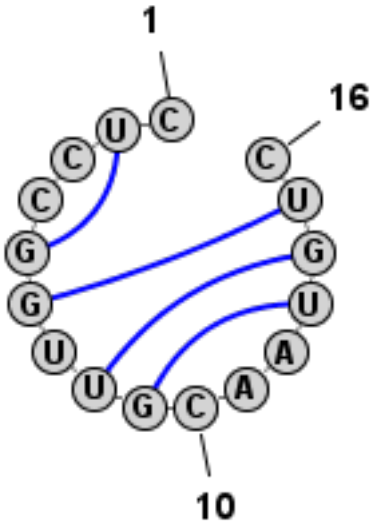

Figure 2.7.5: Optimal structure for 'CUCCGGUUGCAAUGUC' with minimal loop length = 3 and non canonical base pairs allowed

2. To find out the dependence of time taken by the algorithm to render the most optimal structure on the length of the RNA sequence, I executed the code for different lengths of RNA sequences and plotted it with the time elapsed. The minimum loop length for this analysis was kept as 5 and only the canonical base pairs were allowed. Figure 2.7.7 shows execution time v/s length of RNA sequence. I tried to fit the data and obtained 2.74 as the exponential dependence. Logarithmic fitting resulted in 2.64 power law dependence.

3. I have also tried to see how the number of base pairs corresponding to the optimal structure varies with length of the RNA sequence for Nussinov's algorithm. For

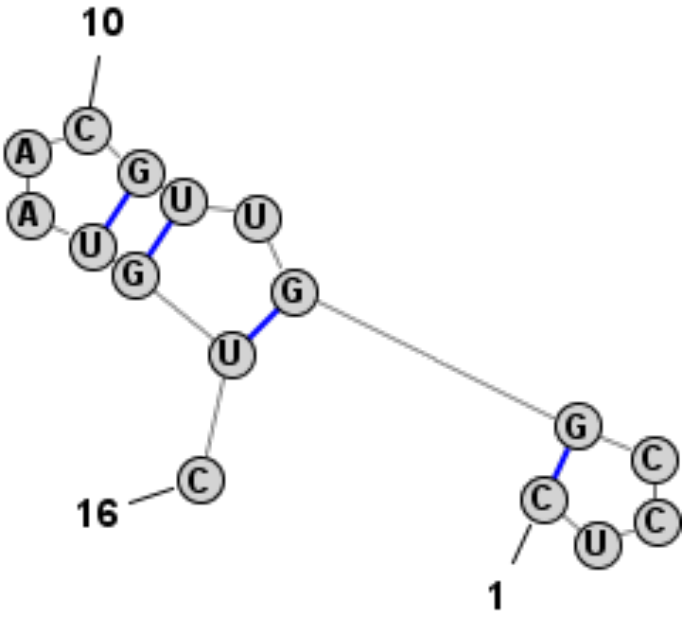

Figure 2.7.6: Optimal structure for 'CUCCGGUUGCAAUGUC' with minimal loop length = 4 and non canonical base pairs allowed

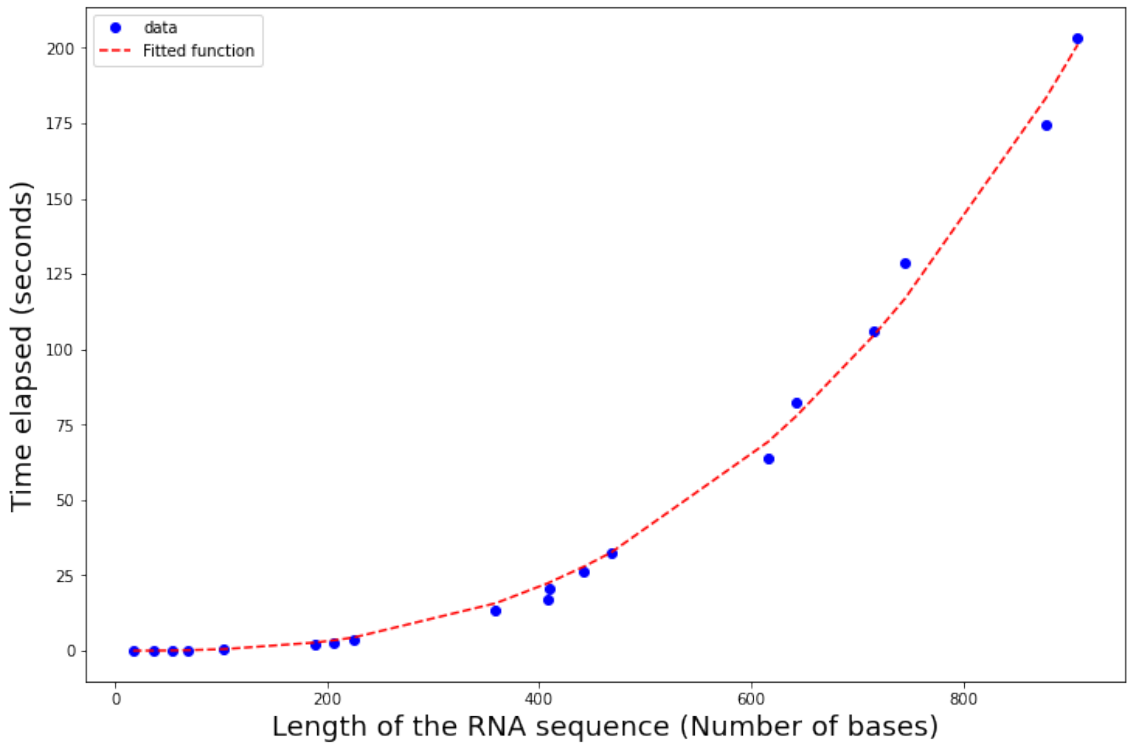

Figure 2.7.7: Dependence of run time on the length of the sequence

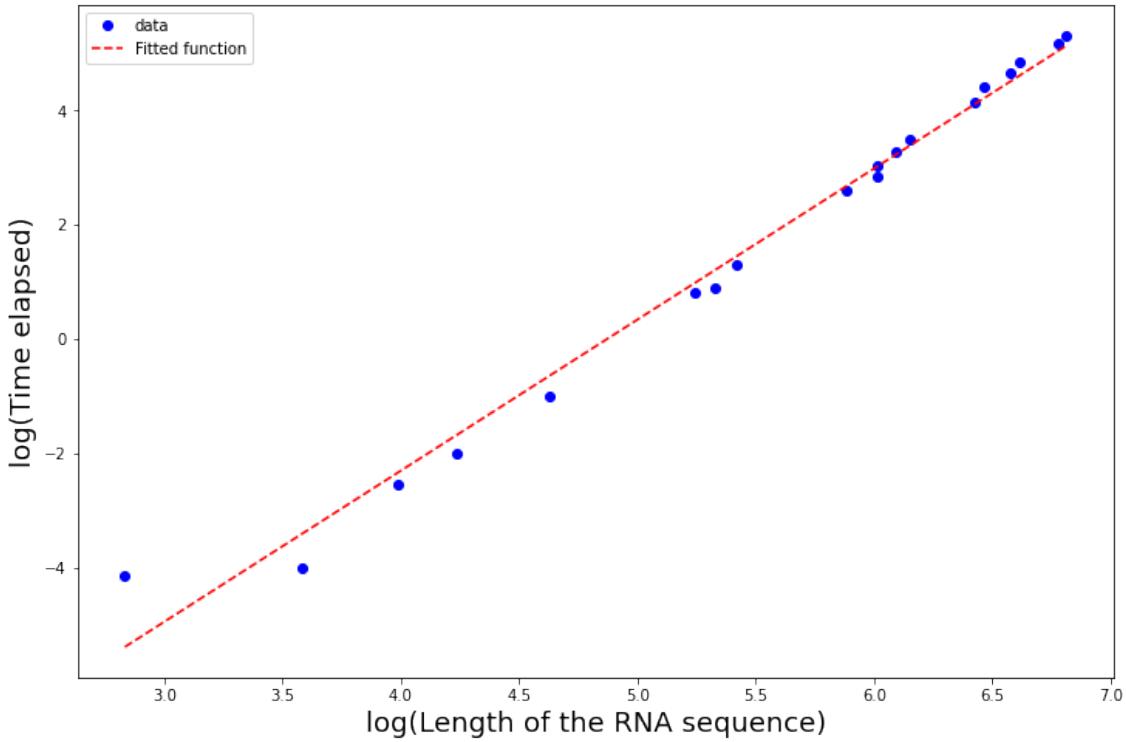

Figure 2.7.8: Dependence of run time on the length of the sequence

this analysis as well, the minimal loop length is kept as 5 and no non canonical base pairs are allowed. Figure 2.7.9 depicts the variation of total base pairs with the length of the sequence.

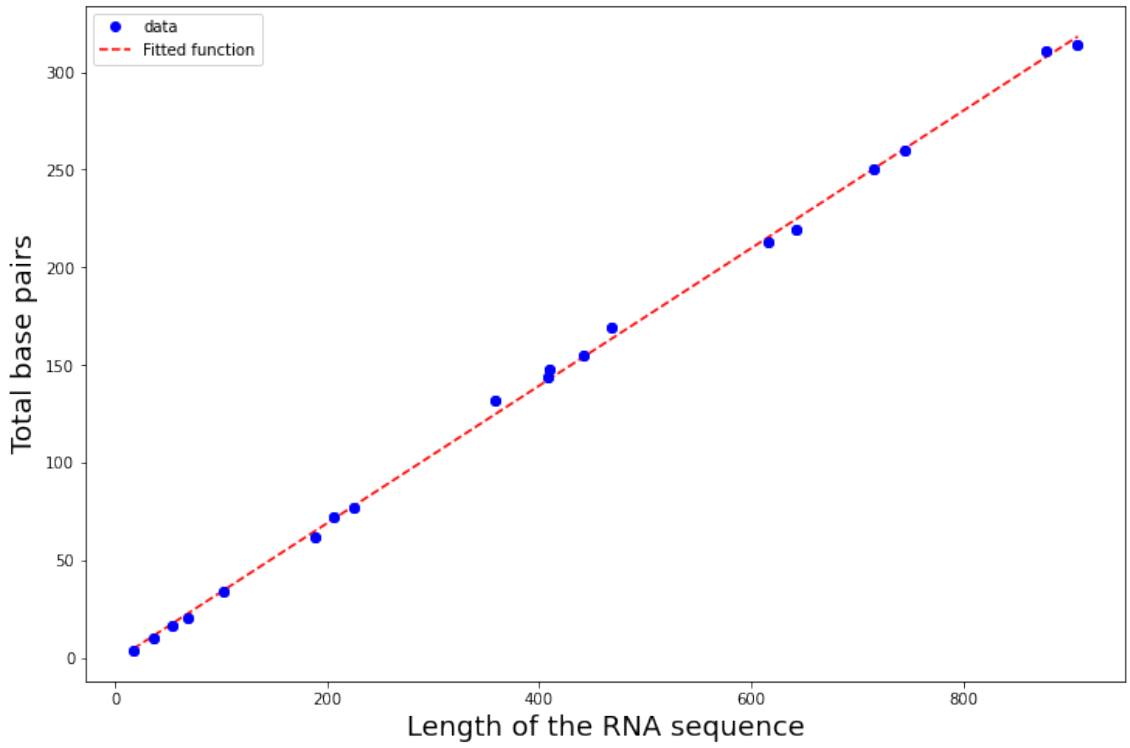

Figure 2.7.9: Dependence of total base pairs on the length of the sequence

Chapter 3

Zuker's Algorithm

3.1 Introduction

Various algorithms had been suggested earlier to determine the secondary structures of RNA molecules to attain a sketch of their organization in three dimensions. Earlier structure prediction was based on topological and thermodynamic rules for searching energetically most favourable structures for RNA sequences. Earlier algorithms were able to predict secondary structures of RNA sequences of limited lengths. Structures predicted by Nussinov's algorithm does not incorporate stacking and destabilizing energies.

Zuker's algorithm[16] follows dynamic programming and incorporates additional information such as reactivity of certain nucleotides to chemical modification to predict optimal structures of RNA sequences. It is based on a model proposed by Roger's *et al*, to calculate the stability of folded RNA molecules by the addition of independent contributions from base-pair stacking and loop destabilizing terms from the secondary structure. The algorithm determines all the base pairs that can participate in structures with a free energy within a specified range from the optimal.

The folding rules given by the Zuker's algorithm are similar to the rules given by Studnicka *et al.* allowing some impossible structures as well. These invalid structures are discarded by assigning high energy values to them.

3.2 Notation and Definitions

The RNA sequence is denoted with S and the i^{th} nucleotide in the sequence is denoted as S_i and S_{ij} represents the sub sequence from i^{th} nucleotide to j^{th} nucleotide. The numbering of the bases begins from the $5^{'}$ end of the RNA molecule by convention. The admissible structures include the base pairs: G-C, A-U or U-G. Each nucleotide are allowed to base pair with at most one other nucleotide. No pseudoknots are allowed in the final structures.

The free energy associated with a structure is related to the regions between the bonds, i.e., it depends on the various internal loops in the structure. According to the graphical representation of the structure, the energy depends on the faces of the graph. A face is defined as a planar region bounded by edges on all the sides. The different types of faces are:

1. Hairpin loop- face with a single interior edge.
2. Faces having two inner edges are classified into three groups:
    - Stacking region- face whose interior edges are separated by single interior edges on both sides.
    - Bulge loop- face whose interior edges are separated by a single exterior edge on one side and more than one exterior edges on the other side.
    - Interior loop- faces other than stacking region and bulge loop.
3. Bifurcation loop- face having three or more than three interior edges.

Hairpin loop is a consequence of consecutive stacking regions, bulge loops, internal loops. Different energies are associated with the above mentioned substructures. The energy of the complete structure is a sum of the energies of its substructures. The energy function can be modified in such a manner to eliminate the undesired structures, e.g.,

a hairpin loop comprising of less than four edges can assigned infinite energy thereby discarding stearically impossible structures. Similar modifications can be done in the energy function to favour certain types of structures. There are two versions of the algorithm:

1. Bifurcation loops are assigned zero energy.

2. Bifurcation loops are treated as interior loops.

3.3 Algorithm

The algorithm finds the most optimal secondary structure with minimum free energy from a large number of possible structures even for a molecule that is as small as the 5S ribosomal RNA.

The algorithm computes two different energies for each subsequence of the given RNA sequence. These energies are stored in two matrices W and V. For a subsequence S_{ij}, $W(i,j)$ is considered as the minimum free energy of all the admissible structures which can be formed from S_{ij} and $V(i,j)$ is considered as the minimum free energy of all the admissible structures which can be formed from S_{ij} when S_i and S_j pair with one another. $V(i,j)$ is given value infinity if S_i and S_j do not base pair with each other. Both these energies are computed in a recursive manner for the subsequent larger subsequences of S.

If $j - i = 4$:

- $W(i,j) = 0$
- For a three nucleotide hairpin loop comprising of A-U or G-C pairs, $V(i,j) = +8.4$ or $+8.0$ Kcal/mol

If $j - i = q > 4$:

- $V(i',j')$ and $W(i',j')$ for $j^{'} - i^{'} < q$ are already computed and are used to compute $V(i,j)$ and $W(i,j)$

Let's consider an admissible structure S_{ij} having energy $V(i,j)$ in which S_i and S_j base pair. $V(i,j)$ is computed as the minimum of the three possible energies. These energies arise from three possible structures.

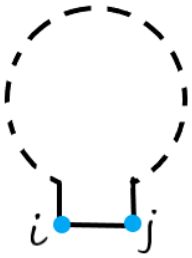

Figure 3.3.1: Hairpin loop

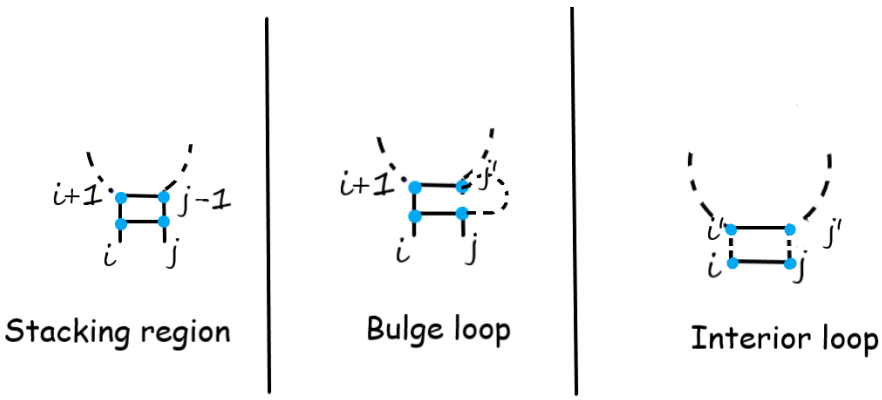

Figure 3.3.2: Stacking region, bulge loop and the interior loop

1. Figure 3.3.1 represents the hairpin loop structure which contains the interior edge between S_i and S_j. For this case $V(i,j) = E(FH(i,j))$and let it be called E_1.

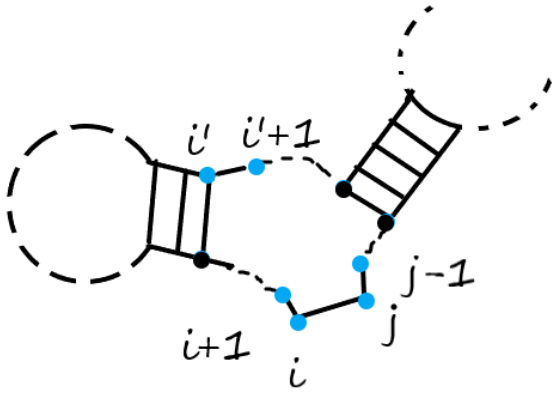

Figure 3.3.3: Bifurcation loop

2. Figure 3.3.2 represents the three possible types of the face adjacent to the edge between S_i and S_j. For this case $V(i,j) = E(FL(i,j,i',j')) + V(i',j')$ and let it be called E_2 such that $i < i' < j' < j$.

3. Figure 3.3.3 represents the bifurcation of two substructures. The energy is the addition of the energies of the two substructures whose mathematical representation is, $V(i,j) = W(i+1,i') + W(i+1,j-1)$ where $i+1 < i' < j-2$ and let this be called E_3.

$$V(i,j) = \min \begin{cases} E_1 \\ E_2 = \min\limits_{i<i'<j'<j} \left\{ E(FL(i,j,i',j')) + V(i',j') \right. \\ E_3 = \min\limits_{i+1<i'<j-2} \left\{ W(i+1,i') + W(i+1,j-1) \right. \end{cases} \tag{3.1}$$

Another structure S_{ij} can be considered with energy $W(i,j)$. This structure also comprises of three possibilities.

1. Figure 3.3.4 represents the possible structures when S_i and S_j do not participate in base pairing. This leads to at least one dangling end. For this case, $W(i,j) = W(i+1,j)$ or $W(i,j) = W(i,j-1)$

2. Figure 3.3.5 represents the trivial structure in which S_i and S_j base pair with each other. For this case, $W(i,j) = V(i,j)$ (previously computed).

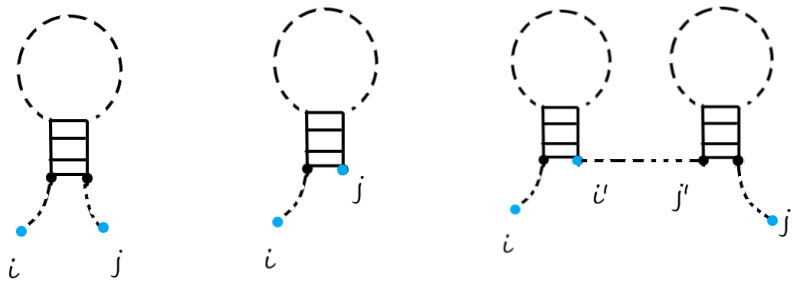

Figure 3.3.4: Structures corresponding to W(i,j) when S_i and S_j do not participate in base pairing

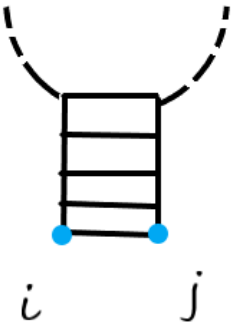

Figure 3.3.5: Structure corresponding to W(i,j) when S_i and S_j base pair with each other

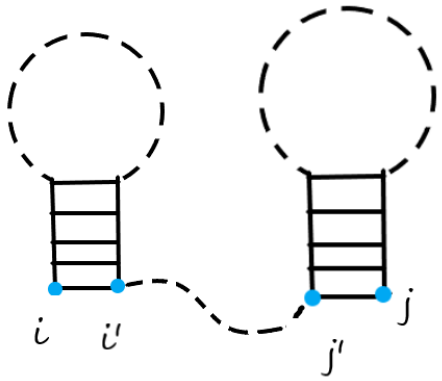

Figure 3.3.6: Structure corresponding to W(i,j) when S_i and S_j participate in base pairing but not with each other

3. Figure 3.3.6 refers to the bifurcation structure where S_i and S_j base pair with some other bases. For this case $W(i,j) = W(i,i') + W(i'+1,j) = W(i,j'-1) + W(j',j)$ where $i < i' < j' < j$

The minimum energy $W(i,j)$ is computed as:

$$W(i,j) = \min \begin{cases} W(i+1,j) \\ W(i,j-1) \\ V(i,j) \\ E_4 = \min\limits_{i<i'<j-1} \{ W(i,i') + W(i'+1,j) \end{cases} \tag{3.2}$$

The algorithm adds one base at a time to the sequence and observes the optimal structure at every step and finally computes the value of $W(1,n)$ which corresponds to the minimum free energy structure obtained for the given sequence. To obtain meaningful structures for a given sequence, the energy function can be tailored accordingly by supplying additional information. This can be achieved by getting data on the reactivity of some bases to a specific chemical reagent and thereby preventing these nucleotides from participating in based pairing.

3.4 Implementation

A simplified version of the Zuker's algorithm has been implemented in python which gives a scoring scheme for bonds and hairpins. I have tried to find the secondary structure of RNA5S1 RNA, 5S ribosomal 1 [*Homo sapiens* (human)], also known as RN5S1. It is a 121 nucleotides long sequence. The sequence has folded in a total of 37 base pairs. The time taken to execute the code is 15.458 seconds. The same sequence results in Figure 3.4.2 when implemented using Nussinov's algorithm (minimal loop length = 5). This resulted in a total of 40 base pairs and time taken to execute the code was

0.9616 seconds. Another RNA sequence 206 nucleotides long has been folded following both the algorithms. Figure 3.4.3 shows the folded structure using Zuker's algorithm with a total of 65 base pairs and time taken to execute the code was 112.50 seconds whereas Nussinov's algorithm results in Figure 3.4.4 with 72 base pairs and time taken for execution was 2.43 seconds. Hence, it has been observed that Nussinov's algorithm favours larger number of base pairs.

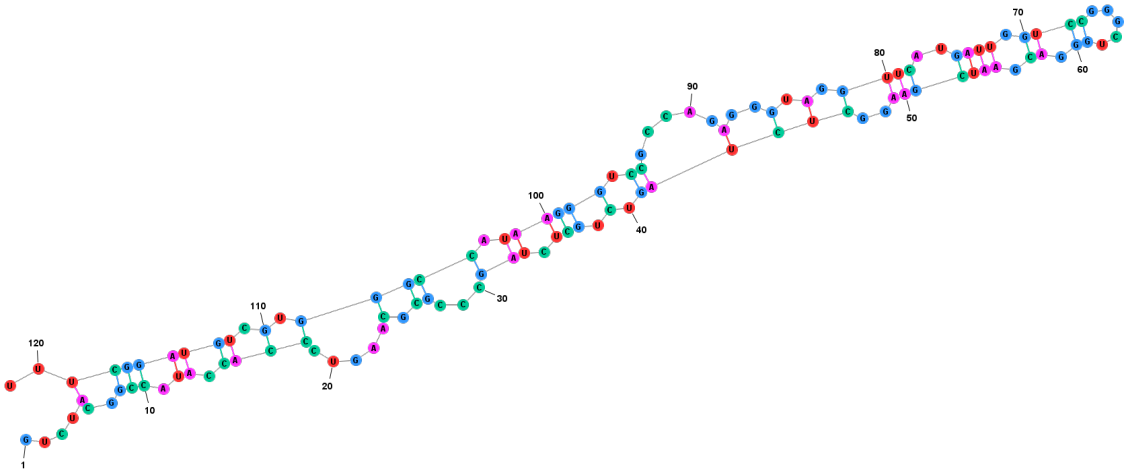

Figure 3.4.1: Folded structure for RNA5S1 using Zuker's Algorithm

To understand the dependence of execution time on the length of the RNA sequence, I implemented the code for sequences of different length and plotted the graph between the two. Execution time is observed to depend on the length of the RNA sequence as $O(L^{3.86})$. Logarithmic fitting results in $O(L^{3.56})$ dependence.

3.4.1 Comparison of structures

Figure 3.4.7 represents the predicted the structure of Bordetella bronchiseptica signal recognition particle RNA. The RNA sequence is 111 nucleotides long. The predicted structure has 34 base pairs and the execution time was 11.01 seconds. Figure 3.4.8 represents the secondary structure predicted using Nussinov's algorithm. For this case, the minimum loop length was set as 4 and only canonical base pairs were allowed to pair.

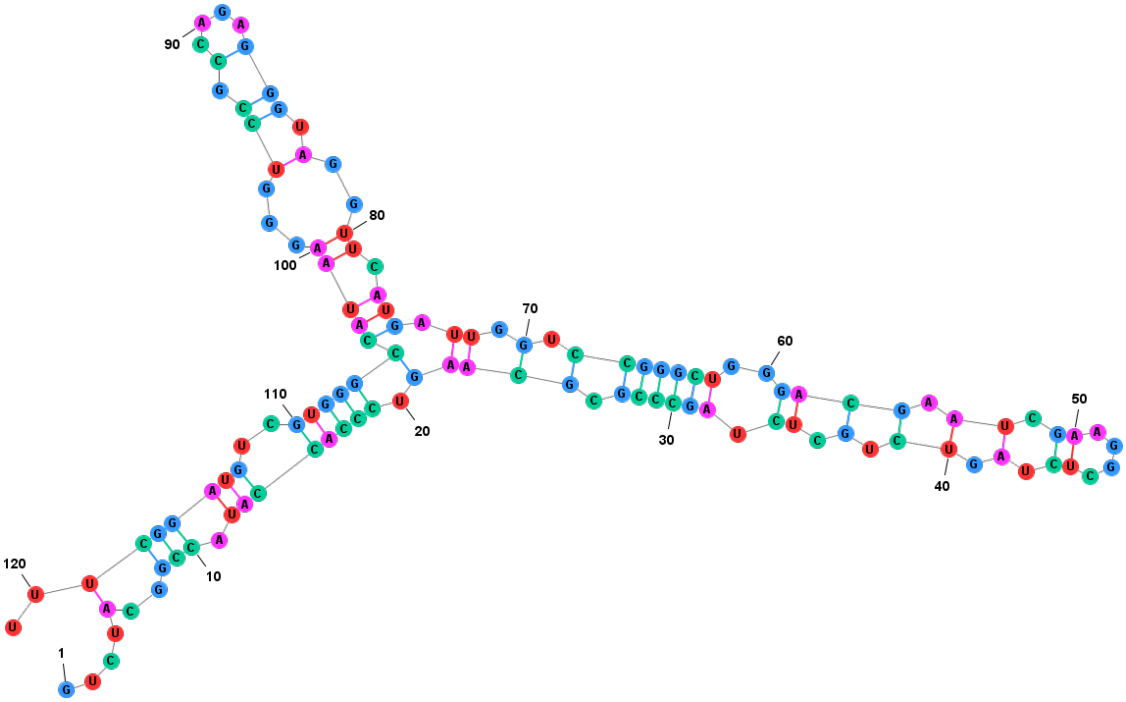

Figure 3.4.2: Folded structure for RNA5S1 using Nussinov's Algorithm

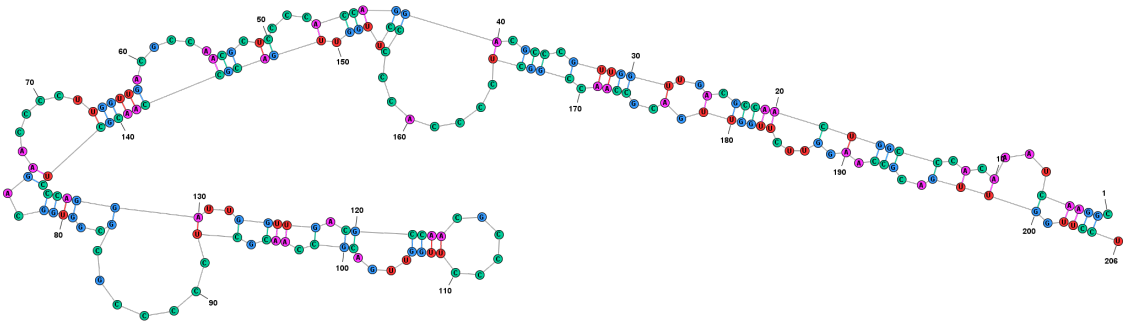

Figure 3.4.3: Folded structure for RNA sequence of length 206 using Zuker's algorithm

The structure resulted in a total of 37 base pairs and the execution time was 0.25 seconds. Figure 3.4.9 is the structure predicted by the RNAfold WebServer. The minimum free energy of the predicted structure is -40.80 kcal/mol.

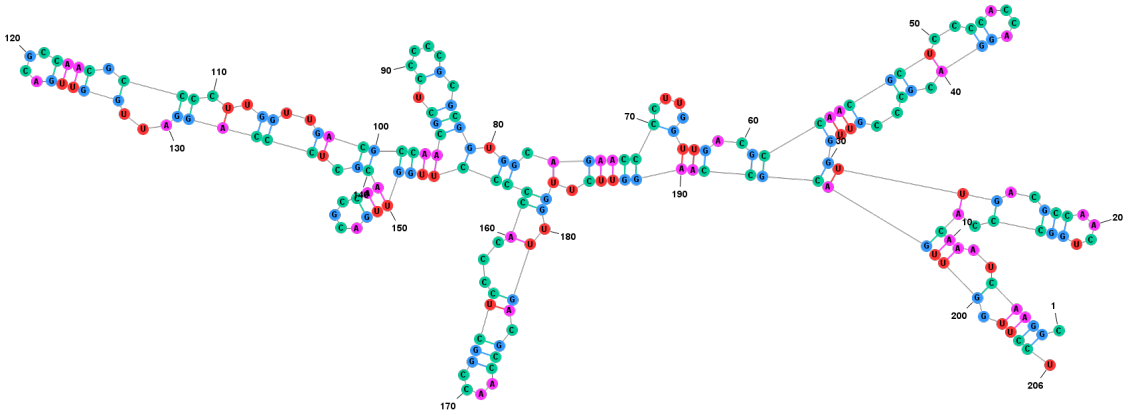

Figure 3.4.4: Folded structure for RNA sequence of length 206 using Nussinov's algorithm

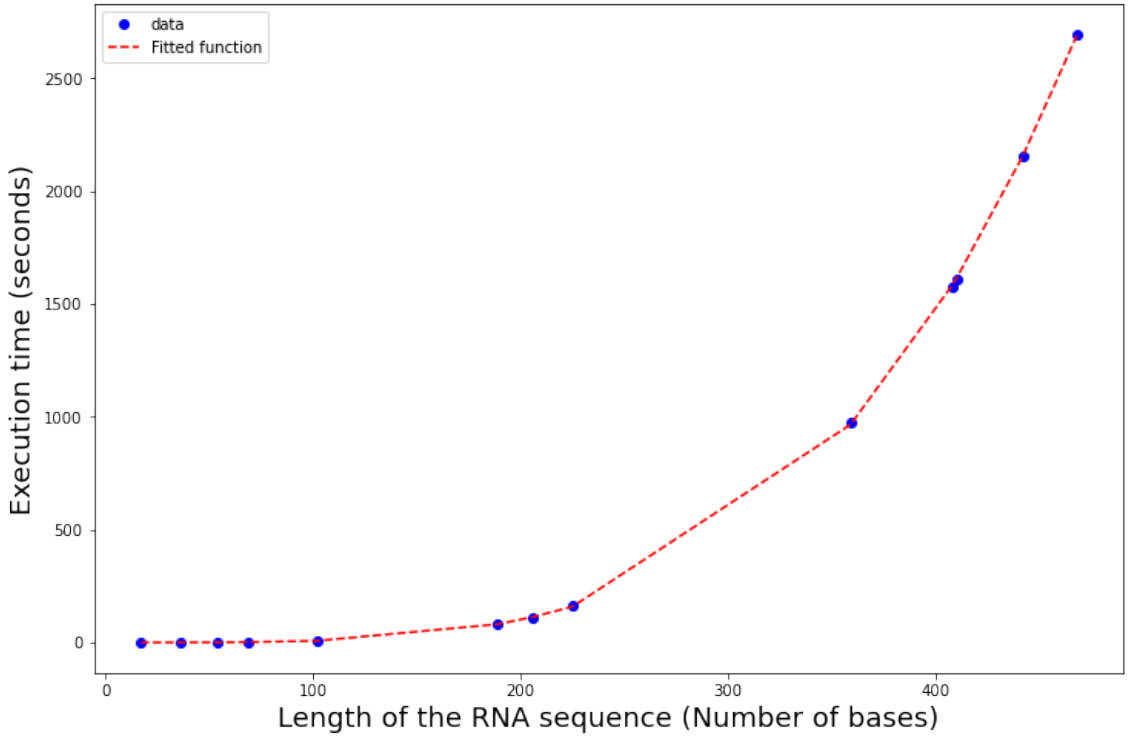

Figure 3.4.5: Dependence of execution time on the length of the sequence for Zuker's algorithm

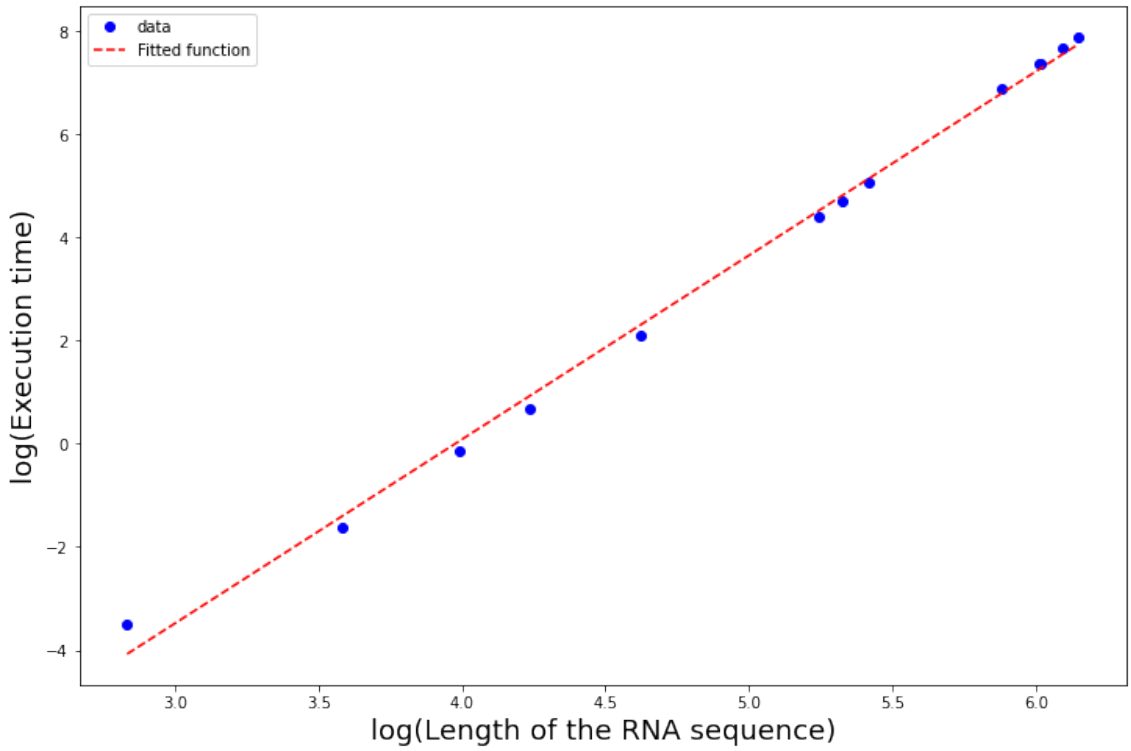

Figure 3.4.6: Dependence of execution time on the length of the sequence for Zuker's algorithm

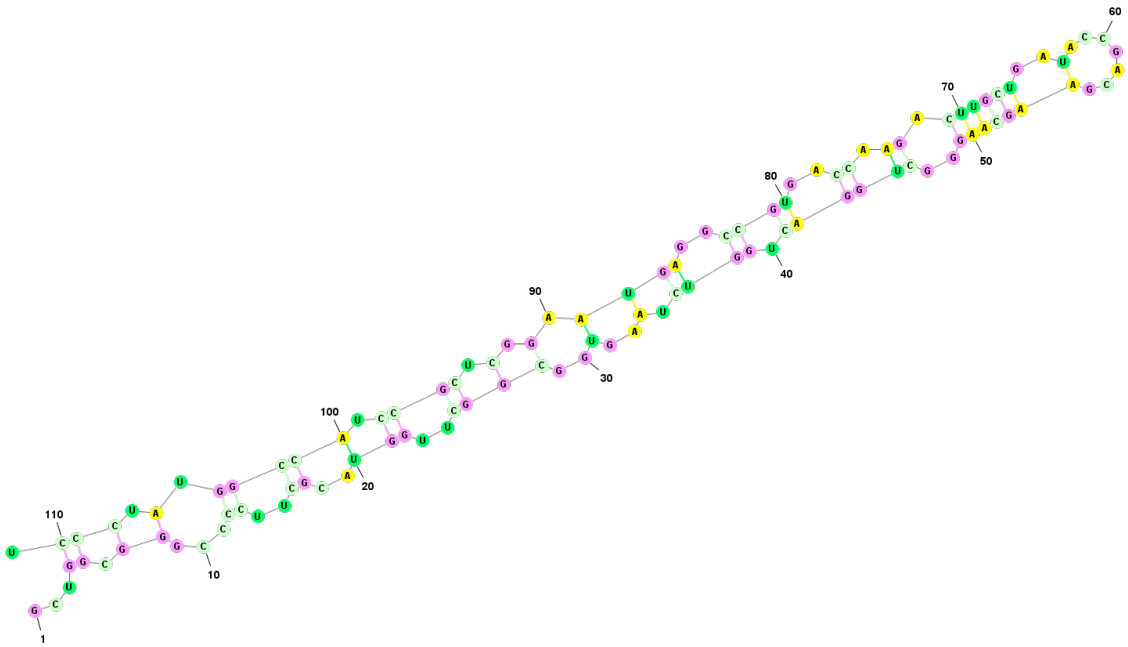

Figure 3.4.7: Secondary structure of Bordetella bronchiseptica signal recognition particle RNA predicted using Zuker's algorithm

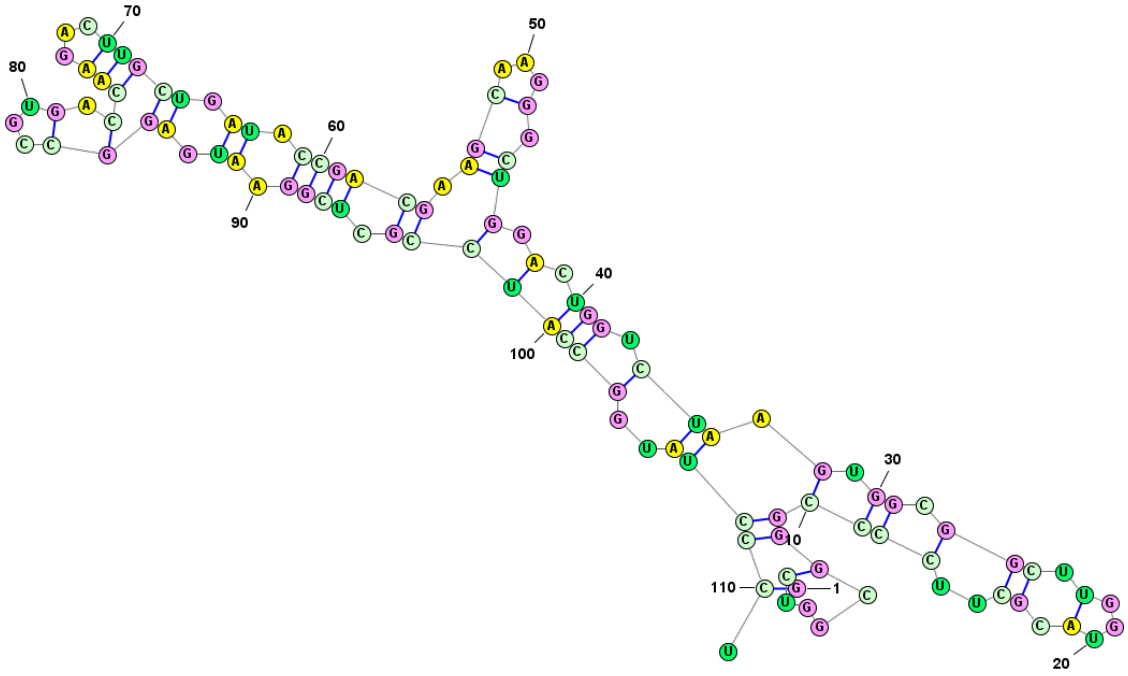

Figure 3.4.8: Secondary structure of Bordetella bronchiseptica signal recognition particle RNA predicted using Nussinov's algorithm

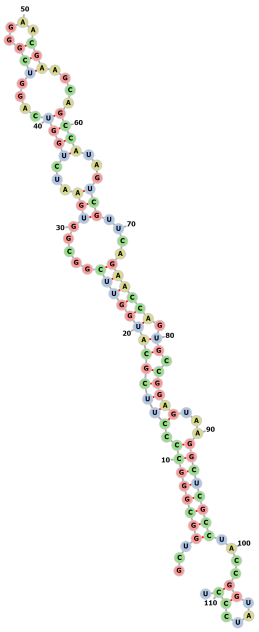

Figure 3.4.9: Secondary structure of Bordetella bronchiseptica signal recognition particle RNA predicted using RNAfold WebServer

Chapter 4

Additional Methods for RNA Structure Prediction

4.1 McCaskill's Approach

In 1990 J.S. McCaskill introduced an algorithm[9] that made use of partition function to determine the RNA secondary structure. The probability of occurrence of a proposed secondary structure can be calculated using the partition function. The complete structure partition function is given as:

$$Q = \sum_{\text{structures}} e^{-\Delta G(\text{structure})/RT} \tag{4.1}$$

The partition function is obtained by summing over all the possible secondary structures for a given RNA sequence. The corresponding probability for a probable secondary structure is given as:

$$P = \frac{e^{-\Delta G(\text{structure})/RT}}{\sum_{\text{structures}} e^{-\Delta G(\text{structure})/RT}} \tag{4.2}$$

Although the probability of a secondary structure is not very useful as there are a large number of possible low free energy structures within the range RT. Hence the probability of any structure is very small. The algorithm makes use of dynamic programming for implicitly considering all the possible structures. The partition function is calculated

starting from the subsequence of shortest length and subsequently moving to subsequences of larger lengths and thereby determining for all the subsequences. Calculation of partition function is attributed to the fill stage of the algorithm. Every subsequence contributing to the partition function is counted once only. The dynamic programming trace back stage can be used to determine the probability of the possible base pairs. The base pairing probability is extremely large compared to the probability of free energy structures.

This algorithm is similar to previous algorithms as it follows similar dynamic programming approach (fill stage and trace back stage). It is useful in determining the statistics of the secondary structures for a given RNA sequence.The program execution time based on the algorithm which do not allow pseudo-knots scales as $O(N^3)$.

4.2 Statistical Sampling Algorithm

This is also an example of dynamic programming algorithms which was introduced by Ding and Lawrence[2] that uses sampling of sub-optimal secondary structures for a given RNA sequence from Boltzmann ensemble of structures. This algorithm also follows the fill stage process followed by the trace back stage to determine the most optimal secondary structure. The fill stage follows similar process as in the partition function algorithm. The trace back stage comprises of probabilistic generation of base pairs according to the partition functions for all possible sub sequences. The sampling probability of any given substructure is equal to its probability of occurrence in the thermodynamic ensemble. The precision of calculation of occurrence probabilities depends on the size of the sample. The study [3] has shown that the sampling statistics are reproducible from sample to sample. The predicted base pair probabilities for two Boltzmann samples of an RNA of given length are nearly indistinguishable. Even if the number of possible secondary secondary structures for a given RNA sequence is large, the folding statistics

can be completely determined by the small sample size e.g., 1000 structures for a 1187 nucleotides long mRNA with total possible secondary structures being 10^{303}.

4.3 Kinetic Folding Algorithms

The dynamic programming algorithms are based on the equilibrium thermodynamics i.e., the RNAs are considered to be in equilibrium and the search algorithm finds the secondary structure with the minimum free energy. The kinetic folding algorithms on the other hand take into consideration the kinetics involved in the folding of the molecule. These algorithms are based on the assumption that the real molecules would rather fold in structures which are easy to obtain than folding in structures with minimum free energies. These algorithms comprise of adding RNA helices to the obtained structure thereby lowering the free energy of the structure at each step. Addition of any further helix stops when energy cannot be lowered any more. This process works in such a way that next helix is added to the structure such that it reduces the free energy by the largest amount. Studies have been done which make use of Monte Carlo simulations to simulate the real time formation and breakage of helices.

Pair Kinetics program assigns rate of formation to every base pair that can be formed with regard to the existing secondary structure along with the rate of removal. The probability of a reaction is proportional to its rate therefore the reactions with higher rate have higher probability of occurrence. The folded structure gets updated with the addition or removal of the base pair. The free energies are calculated using similar dynamic programming techniques.

Similar to this is the Helix Kinetics program which deals with the addition/removal of complete helix in place of a single base pair. A partially formed helix acts as an intermediate that provides activation energy for the formation of helix. The estimation of the reaction rates can be done using Monte Carlo methods and differential equations can

be solved numerically to obtain different structural probabilities for the given molecule. But this method would involve a pre-selection of the desired configurations from an exponentially large number of possible structures.

4.4 Comparative Methods

Comparative sequence analysis is a technique which is used find the secondary when sequences are known for a molecule for a number of different species. The basic principle behind this technique is that molecules which perform same functions in different species should have same structure, therefore it is crucial to find a structure that gives a base-pairing pattern for every sequence. Multiple alignment for all the sequences is implemented. The method searches for sites having changes which are correlated with changes on some other sites. Compensatory mutations often occur in RNAs. These mutations are such that a change which has occurred on one side of the helix is accompanied by a mutation on the other side. This is a good method for structure prediction if large number of sequences are available. This method has been instrumental in obtaining the structures of small and large sub-unit rRNAs. This method of finding RNA secondary structures has been considered more reliable than the existing thermodynamic methods. There are certain limitations associated with this method such as:

- Unable to work on a single sequence
- Unable to provide information about alternative structures of a sequence
- Method does not provide the folding pathway and thermodynamic stability

4.5 Genetic Algorithms

Genetic algorithms have been useful in solving complex optimization problems which imitate biological evolution. These algorithm are much useful in kinetic folding processes. These algorithms work by storing large number of alternative structures for a given sequence. The series of structures which form during a genetic algorithm simulation are used to predict the folding pathways. Thermodynamics is incorporated in the selection stage of the algorithm in a qualitative manner but there is no assurance that the population of alternative structures will end up in any meaningful equilibrium state. Fitness is related to the free energies of the structures such as structures with low free energy tend to have higher fitness and reproduce with a larger frequency compared to structures with high free energy. Genetic algorithms incorporate mutations and crossovers in determining fitness of any given structure. These algorithms have proven to be useful in analysing the folding pathways in combination with biological aspects of the molecule in the problem.

Appendix A

Computational Code for Zuker's Algorithm

```
import numpy as np
import math
import time
def initInfiniteArray(n):
  array = np.zeros((n,n))
  rows,cols = array.shape
  for i in range(rows):
    for j in range(cols):
      if j+const_dist > i:
        array[i,j] = np.inf
      else:
        array[i,j] = np.NAN
  return array
seq = 'CGGAAGAACUGGCAAAAAAGGACGGUGGGGAGUGCCCAGCU'
seq = seq.replace("T", "U")
print(seq)
const_dist = 3
n = len(seq)
begin = time.time()
```

```
W = initInfiniteArray(n)
V = initInfiniteArray(n)
Traceback = np.array([str('') for x in range(n*n*4)])
.reshape(n,n,4)
#print(W)
def compute_W(i,j):
  left = W[i-1,j]
  bottom = W[i,j+1]
  v_type = compute_V(i,j)
  min_k_val = np.inf
  opt_k = -1
  for k in range(j+2,i):
    curr = W[i,k] + W[k-1,j]
    if  curr < min_k_val:
      min_k_val = curr
      opt_k = k
 W[i,j] =  min(left,bottom,V[i,j],min_k_val)
  c = 0
  if W[i,j] == bottom:
    Traceback[i,j,c] = '\downarrow'
    c+=1
  if W[i,j] == left:
    Traceback[i,j,c] = '\leftarrow'
    c+=1
  if W[i,j] == V[i,j]:
    Traceback[i,j,c] = v_type
    c+=1
  if W[i,j] == min_k_val:
    Traceback[i,j,c] = str(opt_k)
def has_bond(base1,base2,bond):
  if base1+base2 == bond or base2+base1 == bond:
    return True
```

```
  else:
    return False
def s(i,j):
  base1 = seq[i]
  base2 = seq[j]
  WatsonCrick = ["GC","AU"]
  for bond in WatsonCrick:
    if has_bond(base1,base2,bond):
      return -4
  if has_bond(base1,base2,"GU"):
    return 0
  else:
    return 4
def h(i,j):
  if i > j:
    return i-j+3
  else:
    return 0
def compute_V(i,j):
  hairpin = s(i,j) + h(i-1,j+1)
  match = s(i,j) + W[i-1,j+1]
  V[i,j] = min(hairpin,match)
  if V[i,j] == hairpin:
    return 'H'
  else:
    return '\swarrow'
while(math.isnan(W[n-1,0])) :
  for i in range(n):
    for j in range(n):
      if j + const_dist <= i:
        down = W[i,j+1]
        left = W[i-1,j]
```

```
            if math.isnan(down) or math.isnan(left):
                continue
            compute_W(i,j)
def backtrack(i,j,matches):
    for x in range(0,4):
        index = Traceback[i,j,x]
        if index == '':
            continue
        elif index == '\leftarrow':
            backtrack(i,j-1,matches.copy())
        elif index  == '\downarrow':
            backtrack(i+1,j,matches.copy())
        elif index == 'H':
            print('\nFolding␣Found!')
            print('Hairpin',(i,j))
            print_folding(matches)
        else:
            matches.append((i,j))
            backtrack(i+1,j-1,matches.copy())
        return matches
def print_folding(matches):
    up = [seq[x[0]] for x in matches]
    down = [seq[x[1]] for x in matches]
    up_str = ''
    vert_str = ''
    for x in up:
        up_str += ('␣'+str(x) + '␣')
        vert_str += '␣|␣'
    down_str = ''
    for x in down:
        down_str += ('␣'+str(x) + '␣')
    print('---␣Matches␣---')
```

```
    print(matches)
    print(up_str)
    print(vert_str)
    print(down_str)
    dot = ["." for i in range(len(seq))]
    for s in matches:
      #print(min(s), max(s))
      dot[min(s)] = "("
      dot[max(s)] = ")"
    structure = "".join(dot)
    print("\n\n--------Structure------:\n\n",structure)
    print('\n\n total pairs', len(matches) )
print('\n========== \n========== W =====')
print(W.transpose(1,0))
print('')
print('\n========= \n======== V ======')
print(V.transpose(1,0))
print('')
print('\n====== \n=======TRACEBACK=======')
Traceback = Traceback.transpose(1,0,2)
backtrack(0,n-1,[].copy())
print('\n\n length of RNA', len(seq) )
end = time.time()
print('\\n Execution time', end-begin)
```

Appendix B

Computational Code for Nussinov's Algorithm

```
import numpy as np
import time
#Assigning allowed base pairs
def couple(pair):
        pairs = {"A": "U", "U": "A", "G": "C", "C": "G"
        ,"U": "G", "G": "U"}
#wobble base pairs can be removed
     if pair in pairs.items():
                return True
        return False
#Initializing the Nussinov matrix with diagonals=0
def init_matrix(rna):
        M = len(rna)
        nm = np.empty([M, M])
        nm[:] = np.NAN
        nm[range(M), range(M)] = 0
        nm[range(1, len(rna)), range(len(rna) - 1)] = 0
    return nm
#First stage of the algorithm, the fill stage following the 4 rules
def fill(nm, rna):
```

```
        #allowed minimum length of the loop
  minimal_loop_length = 5    (0,1,2,3,...)
    for k in range(1, len(rna)):
        for i in range(len(rna) - k):
                j = i + k
        if j - i >= minimal_loop_length:
                    # 1st rule
                    down = nm[i + 1][j]
                        # 2nd rule
                        left = nm[i][j - 1]
                        # 3rd rule
                        diag = nm[i + 1][j - 1] +
                        couple((rna[i], rna[j]))
            # 4th rule
                        rc = max([nm[i][t] + nm[t + 1][j]
                        for t in range(i, j)])
#max of all        nm[i][j] = max(down, left, diag, rc)
                else:
                        nm[i][j] = 0
        return nm
#Second stage of the DPA,
trace back stage to identify the most optimal structure
def traceback(nm, rna, fold, i, L):
         j = L
   if i < j:
         if nm[i][j] == nm[i + 1][j]: # 1st rule
                traceback(nm, rna, fold, i + 1, j)
         elif nm[i][j] == nm[i][j - 1]: # 2nd rule
                traceback(nm, rna, fold, i, j - 1)
         elif nm[i][j] == nm[i + 1][j - 1] + couple((rna[i],
                rna[j])): # 3rd rule
                fold.append((i, j))
```

```
                traceback(nm, rna, fold, i + 1, j - 1)
        else:
                for k in range(i + 1, j - 1):
            #4th rule
                    if nm[i][j] == nm[i, k] + nm[k + 1][j]:
                        traceback(nm, rna, fold, i, k)
                        traceback(nm, rna, fold, k + 1,j)
                        break
    return fold
#Function to obtain the dot bracket notation for the given sequence
def dot_write(rna, fold):
    dot = ["." for i in range(len(rna))]
    for s in fold:
            #print(min(s), max(s))
            dot[min(s)] = "("
            dot[max(s)] = ")"
            return "".join(dot)
#Starting the clock to note the execution time
begin = time.time()
#list containing the total base pairs corresponding
to the most optimal structure obtained
fold = []
#input
rna='CGGAACTAAACTCGTGGTTCCTGTGGTTCACACCTGA'
#output
nm = init_matrix(rna)
nm=fill(nm,rna)
secondary = traceback(nm, rna, fold, 0, len(rna) - 1)
res = dot_write(rna, fold)
print('initial␣matrix␣\n\n',init_matrix(rna))
print('\nNussinov␣matrix␣\n\n', nm)
print('\npairs\n\n␣', fold)
```

```
print('Total pairs = ', len(fold))
print('\nRNA structure',res)
print('RNA=', rna)
print('length of the rna = ', len(rna))
end = time.time() # stopping the clock
print("Time elapsed: ", end - begin)
```

Bibliography

[1] Peter Clote et al. “Asymptotic expected number of base pairs in optimal secondary structure for random RNA using the Nussinov–Jacobson energy model”. In: *Discrete applied mathematics* 155.6-7 (2007), pp. 759–787.

[2] Ye Ding and Charles E Lawrence. “A statistical sampling algorithm for RNA secondary structure prediction”. In: *Nucleic acids research* 31.24 (2003), pp. 7280–7301.

[3] Sean R Eddy. “How do RNA folding algorithms work?” In: *Nature biotechnology* 22.11 (2004), pp. 1457–1458.

[4] Jörg Fallmann et al. “Recent advances in RNA folding”. In: *Journal of biotechnology* 261 (2017), pp. 97–104.

[5] Paul G Higgs. “RNA secondary structure: physical and computational aspects”. In: *Quarterly reviews of biophysics* 33.3 (2000), pp. 199–253.

[6] Ran Libeskind-Hadas and Eliot Bush. *Computing for biologists: Python programming and principles*. Cambridge University Press, 2014.

[7] m. *Nussinov algorithm to predict secondary RNA fold structures*. Dec. 2020. URL: https://bayesianneuron.com/2019/02/nussinov-predict-2nd-rna-fold-structure-algorithm/.

[8] David H Mathews. “Revolutions in RNA secondary structure prediction”. In: *Journal of molecular biology* 359.3 (2006), pp. 526–532.

[9] John S McCaskill. “The equilibrium partition function and base pair binding probabilities for RNA secondary structure”. In: *Biopolymers: Original Research on Biomolecules* 29.6-7 (1990), pp. 1105–1119.

[10] NikolasGialitsis. *NikolasGialitsis/RNA$_2D_s$truct*. URL: https://github.com/NikolasGialitsis/RNA_2D_struct#rna_2d_struct.

[11] Ruth Nussinov and Ann B Jacobson. “Fast algorithm for predicting the secondary structure of single-stranded RNA”. In: *Proceedings of the National Academy of Sciences* 77.11 (1980), pp. 6309–6313.

[12] Ian Kings Oluoch et al. “A review on RNA secondary structure prediction algorithms”. In: *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*. IEEE. 2018, pp. 18–23.

[13] Erwin Aligam - styleshout.com. *Mathews lab*. URL: https://rna.urmc.rochester.edu/RNAstructure.html.

[14] Stefan Washietl. “6.047/6.878 Lecture 08: RNA Folding”. In: (2012).

[15] Michael Zuker and David Sankoff. “RNA secondary structures and their prediction”. In: *Bulletin of mathematical biology* 46.4 (1984), pp. 591–621.

[16] Michael Zuker and Patrick Stiegler. “Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information”. In: *Nucleic acids research* 9.1 (1981), pp. 133–148.