A STUDY ON RNA PSEUDOKNOT PREDICTIONS

By

LEO YEAN LING

A thesis submitted to the Department of Mathematical and Actuarial Sciences,

Faculty of Engineering and Science,

Universiti Tunku Abdul Rahman,

in partial fulfillment of the requirements for the degree of

Master of Mathematical Sciences

March 2013

ABSTRACT

A STUDY ON RNA PSEUDOKNOT PREDICTIONS

LEO YEAN LING

This research proposed an RNA pseudoknot prediction algorithm based on stem weight maximization. The proposed algorithm consists of three stem searching functions looking for stems in different searching regions. “Section search” looks for base pairing in three different regions of the earlier stem found. Then, “cross search” will identify regions for cross pairing and search for possible base pairing. Finally, “knot search” which look for H-type pseudoknots and kissing hairpins that is formed by base pairing between individual hairpin and other unpaired regions.

The resulting secondary structure is represented in a dot-bracket representation and could be visualized by VARNA. A total of 232 RNA structures have been downloaded from three databases (FRABASE, RNA STRAND and CompaRNA). Performance of the proposed algorithm is

evaluated by calculating the specificity and sensitivity between the predicted structures to the experimental structures obtained from database. In addition, execution time of algorithm proposed is recorded as well. Our results show that the proposed algorithm can produce reasonably accurate structure in practical time frame.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere thanks and appreciation to my supervisor, Dr. Goh Yong Kheng for being an excellent mentor. He has given me a lot of invaluable guidance and advice till the completion of the research. Besides, his encouragement contributes a lot for the achievement of this research.

Furthermore, I would like to express my heartfelt gratitude to Dr. Liew How Hui as he has provided lots of suggestions for a better programming approach. Also, a special note of thanks to my friends who had given me lots of precious ideas and motivation while finishing the research. Finally, to my parents, I will always owe them my deepest love; their unceasing support and encouragement have seen me through my studies in Universiti Tunku Abdul Rahman.

LEO YEAN LING

FACULTY OF ENGINEERING AND SCIENCE

UNIVERSITI TUNKU ABDUL RAHMAN

Date: _____

SUBMISSION OF THESIS

It is hereby certified that **LEO YEAN LING** (ID No: **10UEM01828**) has completed this thesis entitled "A STUDY ON RNA PSEUDOKNOT PREDICTIONS" under the supervision of Dr. GOH YONG KHENG (Supervisor) from the Department of Mathematical and Actuarial Sciences, Faculty of Engineering and Science, and Dr. LIEW HOW HUI (Co-Supervisor) from the Department of Mathematical and Actuarial Sciences, Faculty of Engineering and Science.

I understand that the University will upload softcopy of my thesis in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

_____

(LEO YEAN LING)

APPROVAL SHEET

This thesis entitled "**A STUDY ON RNA PSEUDOKNOT PREDICTIONS**" was prepared by LEO YEAN LING and submitted as partial fulfillment of the requirements for the degree of Master of Mathematical Sciences at Universiti Tunku Abdul Rahman.

Approved by:

_____

(Dr. GOH YONG KHENG) Date : ……..
Assistant Professor/Supervisor
Department of Mathematical and Actuarial Sciences
Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

_____

(Dr. LIEW HOW HUI) Date : ……..
Assistant Professor/Co-supervisor
Department of Mathematical and Actuarial Sciences
Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

DECLARATION

I, LEO YEAN LING hereby declare that the thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTAR or other institutions.

_____

(LEO YEAN LING)

Date:

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

DWM	Dynamic Weighted Matching
FRABASE	RNA Fragments Search Engine and Database
MMC	Matthew's Correlation Coefficient
NMR	Nuclear Magnetic Resonance
RNA	Ribonucleic Acid
SCFG	Stochastic Context-Free Grammar
SMCFG	Stochastic Multiple Context-Free Grammar
tRNA	Transfer RNA
VARNA	Visualization Applet for RNA

CHAPTER 1

INTRODUCTION

1.1 Objectives

In this research, the author would like to

i. Predict and identify pseudoknots on various genome.
ii. Develop new algorithm or optimize existing pseudoknot prediction algorithm.
iii. Compare between structures obtained by pseudoknot prediction algorithm and experimentally determined structure.

RNA forms the secondary structure by base pairing among the complementary base pairs, G-C, A-U and G-U. RNA secondary structures include loops, stems, single strand regions and pseudoknot. Pseudoknots are significant in some biological processes (Staple & Butcher 2005). For example, researchers found that only those telomerase ribonucleoprotein complexes which contain a properly folded pseudoknot are catalytically active (Mihalusova, Wu & Zhuang 2011).

This research aims to develop an RNA pseudoknot prediction algorithm which is able to recognize two types of pseudoknot structure, namely H-type pseudoknot and kissing hairpins. Thus far, majority of RNA secondary structure algorithms are restricted to predict only the H-type pseudoknot. This might due to the high computational requirement of predicting kissing hairpins which is formed by the base pairing between two loop regions.

Then, the structure predicted will be compared to experimentally determined structure. These structures are obtained from the FRABASE (Popenda et al. 2010). This database contains only RNA structures determined through experimental method like NMR, X-ray diffraction and electron microscopy. By comparing RNA structures predicted by the proposed algorithm to FRABASE structures, this may provide an indication on how good is the performance of the proposed algorithm.

1.2 Research background on RNA secondary structure prediction

Traditionally, RNA is merely known as a helper in translation. This view has changed ever since it is known to be vital in regulation of genes (Valencia-Sanchez et al. 2006) and as a catalyst in various cellular processes (Vaish,

Kore & Eckstein 1998). Therefore, researchers began their study on RNA sequences.

RNA secondary structures are formed by base pairing among complementary bases. Then, the interaction among these structures would form the tertiary structure of RNA. Since determining tertiary structure by experimental method is expensive and time consuming, computational methods have been developed for secondary structure prediction because it serves as a foundation for the tertiary structure prediction (Mathews & Turner 2006).

Dynamic programming approach has been implemented for RNA pseudoknot prediction but this approach faced the problems of high time and space complexities. The pseudoknot prediction algorithm developed by Elena Rivas and Sean R. Eddy has a worst case of $O(N^6)$ in time complexity and $O(N^4)$ in storage complexity (Rivas & Eddy 1999). The algorithm proposed by Uemura et al. has a complexity of $O(N^5)$ in time and $O(N^4)$ in space (Uemera et al. 1999). Thus, these algorithms can only predict RNA structures for short sequences and will fail for long sequences due to insufficient memory or lengthy execution time.

Besides, various heuristic approaches have been proposed for RNA secondary structure prediction including pseudoknots, such as maximum weighted matching (Tabaska et al. 1998), iterated loop matching (Tahi, Engelen & Rgnier 2003), dynamic weighted matching (Liu et al. 2006), HotKnots (Ren et al. 2005), DotKnot (Sperschneider, Datta & Wise 2011), etc. Heuristic algorithms usually restrict on predicting specific type of pseudoknot. In addition, structure generated by algorithms which involved free energy calculation strongly depends on the energy model applied. Also, the amount of known pseudoknots is limited. Consequently, lack of sequences available for pseudoknot prediction by comparative approach and testing on accuracy of algorithms developed.

1.3 Problems of RNA secondary structure prediction

Pseudoknots are important RNA secondary structure. Numerous approaches have been implemented to predict RNA secondary structures including pseudoknots. Although various pseudoknot prediction algorithm has been proposed but each has its own restrictions or limitations. Below are some of the main problems encountered by RNA secondary structure prediction algorithm:

(i) Structure predicted from thermodynamic model is not the native structure (Reeder et al. 2006).

(ii) Structure predicted from heuristics algorithms is not the optimal structure (Liu, Ye & Zhang 2006, Van Batenburg, Gultyaev & Pleij 1995).

(iii) Lack of well-aligned sequences to perform sequence alignment (Wilm, Higgins & Notredame 2008).

(iv) Structure prediction algorithm excluding pseudoknots due to its high complexity (Akutsu 2000, Lyngso & Pederson 2000).

This research will focus on the fourth problem stated above.

Due to the difficulties in predicting pseudoknots, it has been excluded from most of the RNA secondary structure prediction algorithms developed. For those algorithms which implemented dynamic programming to predict pseudoknot structures, they faced the problem of high time and space complexities. Therefore, pseudoknot prediction algorithm by heuristic approach has been proposed. The time and space complexities of heuristic algorithms developed are much reduced while compared to dynamic programming approach. Hence, this research attempt to propose an algorithm which can predict reasonably accurate structure in a practical time frame.

1.4 Outline and contributions

This research is aimed to develop a pseudoknot prediction algorithm. Existing pseudoknot prediction algorithms by dynamic programming approach are facing the problems of high time and space complexities. Therefore, these pseudoknot prediction algorithms are limited to identify the most common type of pseudoknot, H-type pseudoknot. In addition, restriction on sequence length is imposed as well. Most of the developed pseudoknot prediction algorithms are able to handle only short RNA sequences.

In order to develop a pseudoknot prediction algorithm, reviews on developed RNA secondary structure prediction algorithms are done. In Chapter 2, eleven RNA secondary structure prediction algorithms are discussed. The earliest algorithm proposed for RNA structure prediction is based on the base pair maximization method (Nussinov et el. 1978). It looks for RNA structure with the maximum number of base pairs.

Then, RNA structure prediction is further enhanced by taking into consideration the free energy contribution of different structures (Zuker & Stiegler 1981, Zuker 1989). Besides, stochastic context-free grammar has been implemented for RNA structure prediction as well (Eddy 2005). This method predicts the RNA secondary structure according to some production rules.

Alternatively, RNA structure prediction by comparative sequence analysis method is proposed as well (Meyer & Miklós 2007). It predicts RNA structure by looking for conserved regions among RNA sequences. This method produces good result but it requires several homologous sequences for structure prediction.

Former methods discussed are proposed for RNA secondary structure prediction excluding pseudoknot. Here, several approaches proposed for pseudoknot prediction are briefly discussed. Tatsuya Akutsu has proposed a pseudoknot prediction algorithm by implementing dynamic programming technique (Akutsu 2000). This method has high time complexity and it is impractical for predicting long RNA sequences. Therefore, Jens Reeder and Giegerich proposed the implementation of canonization in predicting pseudoknot structures (Reeder, Steffen & Giegerich 2007).

Besides dynamic programming approach, structure prediction based on the idea of iteratively forming stable stems is proposed by Ren et al. (Ren et al. 2005). Another approach proposed for RNA structure prediction is the dynamic weighted matching algorithm (DWM) (Liu et al. 2006). It searches for stems with maximum compound weight value recursively. Stochastic context-free grammar has been modified for pseudoknot prediction as well (Mizoguchi, Kato & Seki 2011). This method generates RNA secondary

structure by enhanced production rules. Apart from the approaches aforementioned, a recent development for pseudoknot prediction including kissing hairpins is proposed by Sperschneider J et al. (Sperschneider, Datta & Wise 2011). This is a pseudoknot detection algorithm which output several near-optimal pseudoknot structures.

In this research, a pseudoknot structure prediction algorithm has been developed. The proposed algorithm is developed by extending the DWM algorithm (Liu et al. 2006). There are two stages involved in the proposed algorithm that is construction of weight matrix and stem searching. Stem searching is the core step in structure prediction. Three types of stem searching are implemented where each process is looking for stems in different searching regions. Details for each stage are presented in Chapter 3.

After structure prediction, analysis on structures predicted by the proposed algorithm is presented in Chapter 4. Performance of the proposed algorithm is evaluated by structure comparison between predicted structure and reference structure. Reference structures are obtained from three online databases, FRABASE (Popenda et al. 2010), RNA STRAND (Andronescu et al. 2008) and CompaRNA (Puton et al. n.d.). FRABASE is a database which collects all RNA structure determined through experimental method. Therefore, structures obtained from FRABASE are reliable and appropriate for

structure comparison. For RNA STRAND and CompaRNA, majority of the RNA structures downloaded are predicted using comparative sequence analysis method.

The last chapter concludes this thesis with a discussion about problems encountered, contributions and future work. In the proposed algorithm, all stem found during the stem searching process will be filtered before listing as the potential stems. The characteristics of potential stem can be found in Section 3.2.1. Besides, Chapter 5 discusses about some problems arouse while developing the proposed algorithm. The main problem of the proposed algorithm would be having additional base pairs in the predicted structure. Discussion for the problem is presented in Section 5.1 with relevant examples.

Subsequently, contributions and some suggestions for future work are presented. This research has developed a pseudoknot prediction algorithm which can predict two types of pseudoknot, which are H-type pseudoknot and kissing hairpins. In addition, results of structure comparison show that the proposed algorithm yields reasonably accurate structure. Among three databases, the proposed algorithm achieved highest average SP and SN values for the prediction of FRABASE structures (SP-95.60, SN-98.18). Thus, structures predicted by the proposed algorithm have high similarity to experimentally determined structures.

Moreover, the proposed algorithm can handle long RNA sequences. Thus far, the proposed algorithm has been tested on RNA sequence with the maximum length of 3174nt. For short RNA sequences (<400nt), the proposed algorithm can generate the structure of input sequence in a very short time frame (0.16s in average). This shows that the proposed algorithm can handle long RNA sequences and perform structure prediction in a short duration.

Finally, some suggestions are provided as the future work of the research. Since the proposed algorithm faced the problem of having extra base pairing in the predicted structure, a post processing might be included so as to eliminate these superfluous base pair. Then, the proposed algorithm can be further modified for the prediction of triple helix interaction which is a complex pseudoknot structure. Besides, free energy calculation can be added as the criteria of stem filtration. Consequently, a more stable RNA structure is produced.

CHAPTER 2

REVIEW ON RNA SECONDARY STRUCTURE PREDICTION ALGORITHM

2.1 Ribonucleic acids (RNA)

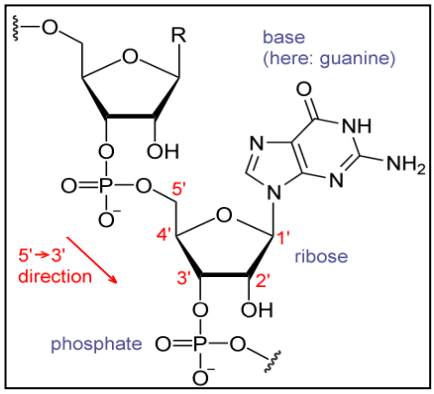


Figure 2.1: Chemical structure of RNA.

RNA is a nucleic acid consists of ribose sugar, nitrogenous base and phosphate groups. It plays a central role in various biological functions within cells. RNA is traditionally known to be involved in translation of protein. However, researchers have found that it is also important in performing other functions within cell. They might act as a catalyst of chemical reaction (Doudna & Sech 2002, Brown 1999). The Ribonuclease P RNA is found to be a ribozyme which cleaves the RNA sequences. Besides, RNA also help in the regulation of transcription and translation (Storz 2002), modulates protein across expression (Meister & Tuschi 2004) and act as a information carrier as well.

The primary structure of RNA is a sequence of nucleotides, namely A (Adenine), U (Uracil), C (Cytosine) and G (Guanine). Secondary structure is formed when RNA single strand fold onto itself by base pairing among complementary nucleotides (Refer to Figure 2.2 for the illustration of RNA folding.). Essentially, there are two types of base pairing, Watson-Crick (G-C and A-U) and Wobble (G-U). Among these three types of base pair, G-C is the most stable base pair, then follow by A-U and the least stable G-U pair. G-C contains three hydrogen bonds while A-U and G-U contains two hydrogen bonds.

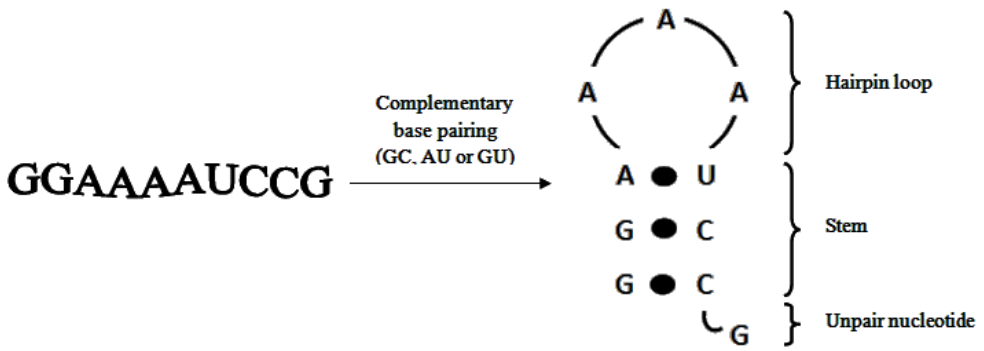


Figure 2.2: Folding of RNA sequence.

According to a survey conducted by Roy et al. (Roy et al. 2008), occurring percentage of G-C, A-U and G-U pairs in 145 RNA crystal structures are 54.04, 17.17 and 6.88 respectively. These three pairs are having higher occurrence frequency while compared to other noncanonical base pairs. This may due to the stabilization effect of having polar hydrogen bonding between bases of Watson-Crick and Wobble base pairs.

2.2 Pseudoknot

Base pairing interactions of RNA sequence form different types of secondary structures. These include stems, single stranded regions, bulge loops, interior loops, hairpin loops, multiloops and pseudoknots. Figure 2.3 shows different types of RNA secondary structures.

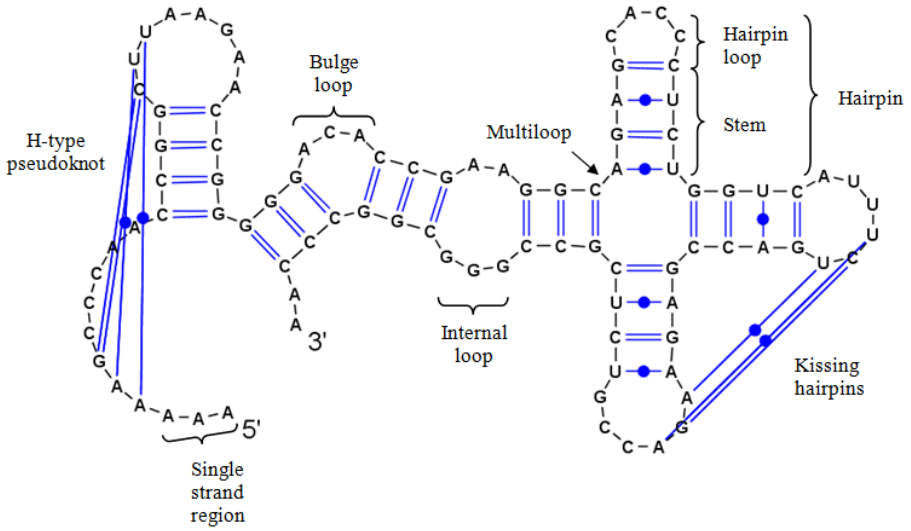
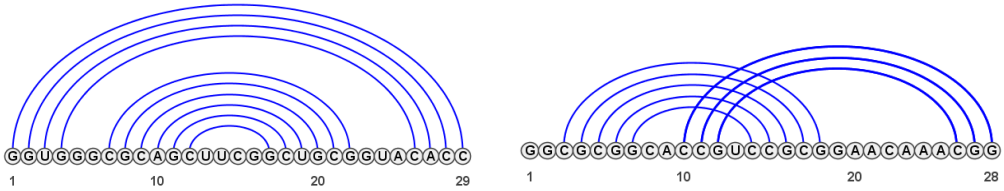


Figure 2.3: RNA secondary structure.

Pseudoknots are functionally important in several known RNAs. It plays functional roles in cases such as ribosomal frameshifting (Giedroc, Theimer & Nixon 2000), regulation of translation and splicing (Draper, Gluick & Schlax 1998), selinocystein biosynthesis, etc. A pseudoknot is an RNA structure that is formed when bases within a loop pair with complementary bases in another unpair region to form a stem (Refer to Figure 2.3 or Figure 2.5

for H-type pseudoknot and kissing hairpins.). Figure 2.4 shows the arc diagram of pseudoknot structure.



(a) Structure without pseudoknot. (b) Structure with pseudoknot.

Figure 2.4: Arc diagram for RNA structure.

Among several distinct types of pseudoknots, H-type pseudoknot is the simplest and classical pseudoknot (Chen & Chen 2009). It is formed by the base pairing between loop and unpair region. For kissing hairpins, it is formed by the base pairing between two loop regions. Figure below shows the structure of H-type pseudoknot and kissing hairpins.

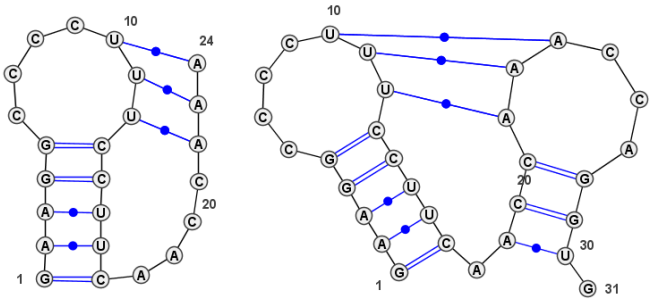
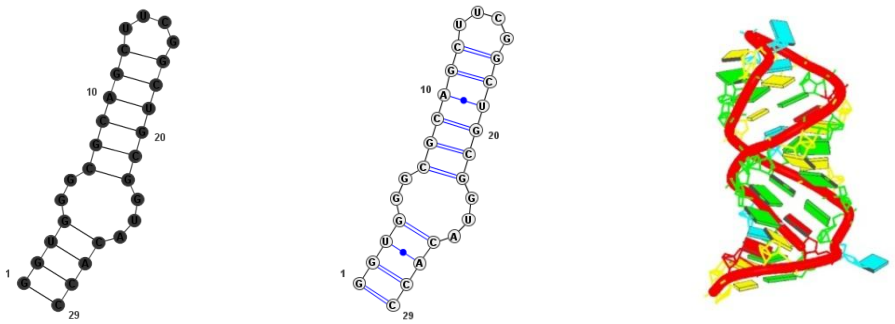


Figure 2.5: H-type pseudoknot (left) and kissing hairpins (right).

2.3 RNA structure representation

RNA structure predicted can be illustrated in various ways. Usually base pairing in RNA structure is represented by drawing a line between the corresponding bases as shown in Figure 2.6(a). For this type of RNA structure representation, every nitrogenous base is represented using a dot. Sometimes, the alphabet (A, C, G, U) representing each base is shown instead of a dot. Also, line connecting each type of bases can be different too. G-C pair is represented by double line; A-U pair is represented by single line with a dot in the middle; G-U pair is represented by single line. Figure 2.6 shows these two types of RNA structure representation and the corresponding 3D structure as well.



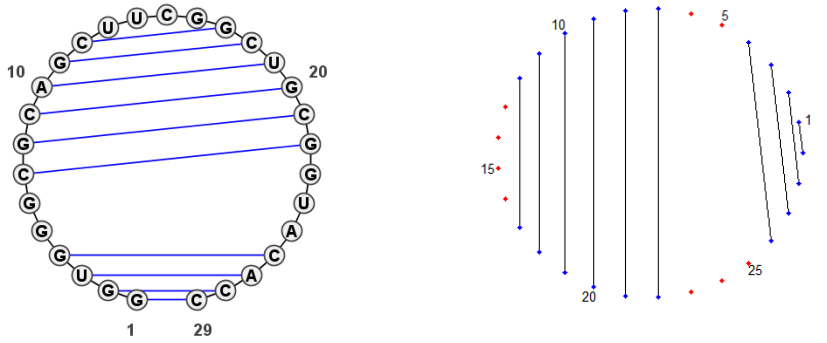
(a) Each base represented by a dot.

(b) Each base represented by its corresponding alphabet.

(c) 3D structure

Figure 2.6: RNA structure.

Besides the RNA representations shown in Figure 2.4 and Figure 2.6, RNA structure can be illustrated using circular representation as well. For this type of representation, nitrogenous bases are arranged in a circular manner. Then, base pairing is represented by drawing a line in between the corresponding bases. However, circular representation is seldom used.

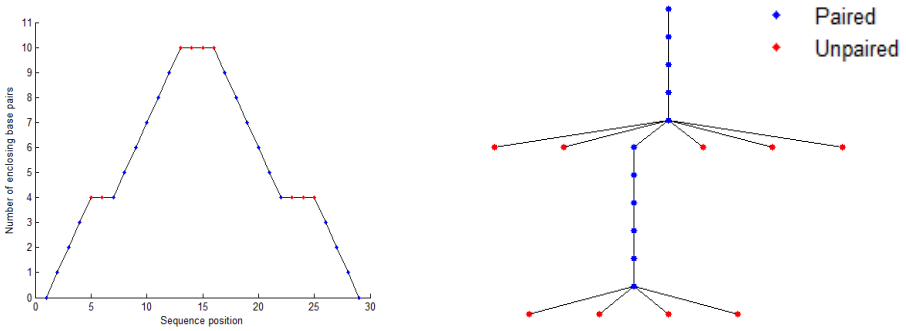


(a) Structure generated using VARNA.

(b) Structure generated using Matlab.

Figure 2.7: Circular representation of RNA structure.

Another type of RNA structure representation is the mountain diagram. In mountain diagram, unpair bases will form a horizontal line while continuous base pairing (stem structure) will form a slope. Another type of similar RNA structure representation is the tree representation. For tree diagram, unpair bases is represented by single dot while base pairing is represented by dots which connect with a straight line going downwards.



(a) Mountain diagram (b) Tree diagram

Figure 2.8: Mountain diagram and tree diagram of RNA structure generated using Matlab.

Lastly, the simplest type of RNA structure representation that is the dot-bracket representation. This representation is used throughout this research because it is simple and easy for storing purpose. In dot-bracket representation, base pairings are represented by round brackets (‘(’ and ‘)’) while unpair bases are represented by dots (‘.’). Pseudoknot which include cross pairing in between stem regions will be represented by square brackets (‘[’ and ‘]’). Figure 2.9 illustrates how an RNA structure is represented in dot-bracket format and Figure 2.10 shows the dot-bracket representation of structure shown in Figure 2.4.

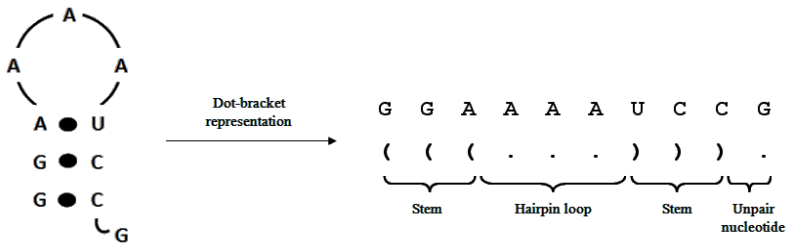


Figure 2.9: Representation of RNA secondary structure in dot-bracket format.

((((. . (((((. . . .)))))) . . .))))

(a) Structure without pseudoknot.

. . (((((. . [[[.))))) . . . . . .]]]

(b) Structure with pseudoknot.

Figure 2.10: Dot-bracket representation of RNA structure.

2.4 RNA secondary structure prediction excluding pseudoknot

This section discusses several methods for RNA secondary structure prediction excluding pseudoknot. These methods show how RNA structure is derived from its primary sequence. Firstly, base pair maximization is presented because this is the earliest method proposed for RNA secondary structure prediction. Then, three commonly used methods for RNA structure prediction, free energy minimization, stochastic context-free grammar and comparative sequence analysis are presented.

2.4.1 Base pair maximization

In 1978, RNA folding problem is formulated as a matching problem by Prof. Ruth Nussinov et al. (Nussinov et el. 1978). She implemented the algorithm using dynamic programming technique. Her algorithm aims to maximize the

base pairing for a given sequence which obeys planarity conditions as stated below:

(i) No crossing between any two paired nucleotides.

(ii) No two adjacent nucleotides may be paired.

This algorithm makes sure that every base-pairing must be adjacent to another base pair and this forces the formation of two or more parallel stems. Therefore, it focuses on "base-stacking" effects of RNA sequence. However, this algorithm doesn't consider the stabilizing and destabilizing effect of stem and loop respectively.

2.4.2 Free energy minimization

Optimal computer folding of RNA sequences by using thermodynamics is proposed by Zuker M and Stiegler P (Zuker & Stiegler 1981). Thermodynamic principles indicate that the structure with lowest free energy should be the most stable structure. Hence, Zuker's algorithm computes the structure with minimum free energy by dynamic programming. Free energy is computed by summing up the energy contributions of all types of RNA secondary structures. The recurrence relation for Zuker's algorithm is shown on next page.

$$W(i,j) = \min \begin{cases} W(i+1, j) \\ W(i, j-1) \\ V(i, j) \\ \min_{i<k<j} \{W(i,k) + W(k+1, j)\} \end{cases}$$

$$V(i,j) = \min \begin{cases} eh(i, j) \\ es(i, j) + V(i+1, j-1) \\ VBI(i, j) \\ VM(i, j) \end{cases}$$

$$VBI(i, j) = \min_{\substack{i<i'<j'<j \\ i'-i+j-j'>2}} \{ebi(i, j, i', j') + V(i', j')\}$$

$$VM(i, j) = \min_{i<k<j-1} \{W(i+1, k) + W(k+1, j-1)\} + a$$

where

$W(i, j)$	:	Minimum free energy from i to j.
$V(i, j)$	:	Minimum free energy from i to j where i is pair to j.
$eh(i, j)$	:	Energy of the hairpin loop closed by the base pair (i, j).
$es(i, j)$	:	Energy of the stacked pair (i, j) and $(i+1, j-1)$.
$ebi(i, j, i', j')$	:	Energy of the bulge or interior loop that is closed by (i, j).
$VBI(i, j)$	:	Energy of a bulge or interior loop that involves a base pair (i', j') and is closed by (i, j).
$VM(i, j)$	:	Energy of multiloop from two smaller structures.

The dynamic programming implementation involves 2 steps, that is “fill” and “traceback”. First, diagonal of matrix is initialized to zero. Then, the “fill” step computes and stores minimum folding energy for all fragments of the sequence. It incessantly builds up larger segments in a recursive manner by iteratively minimize the free energy. This process stops when it reaches the (1,*N*) position in the matrix (which is the upper right most corner). Then, from here it obtains the optimal structure by “tracing back” the optimal path which

leads to that particular value of minimum free energy. This process is shown in Figure 2.11 with a simpler recurrence relation shown below (Eddy 2004).

$$E(i,j) = \min \begin{cases} E(i+1, j-1) + e(x_i, x_j) \\ E(i+1, j) \\ E(i, j-1) \\ \min_{i<j<k} E(i,k) + E(k+1, j) \end{cases}$$

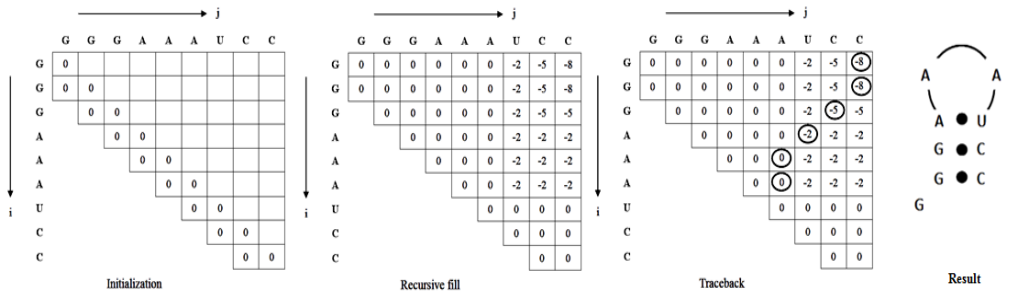


Figure 2.11: Dynamic programming algorithm for RNA secondary structure prediction.

However, Zuker's method may produce a minimum free energy structure which is not necessarily the native structure. Occasionally, there might be more than one structure with equivalent minimum free energy value but this method only returns one structure for every input sequence. Therefore, he further improved his algorithm by predicting suboptimal structures (Zuker 1989). He suggests that structures with free energy value fall within a specified range (usually 5-10%) from the minimum value should also be considered as the potential RNA structure. Although these structures possess free energy

value which is slightly higher than optimal structure, they can be topologically different from each other.

2.4.3 Stochastic context-free grammar

Sean R. Eddy has proposed stochastic context-free grammar (SCFG) for RNA structure prediction as an alternative to dynamic programming implementation (Eddy 2005). SCFG can model nested and long-distance pairwise correlations in strings of symbols. Long distance pairwise correlation is one reason why pseudoknot is difficult to predict. On the other hand, it generates parse tree which is the RNA secondary structure analog of a sequence alignment. Parse tree is generated based on the five production rules as listed below.

$S \rightarrow aS$	:	Leftwise
$S \rightarrow Sa$	:	Rightwise
$S \rightarrow aSb$	:	Pairwise
$S \rightarrow SS$	:	Bifurcation
$S \rightarrow e$	:	End

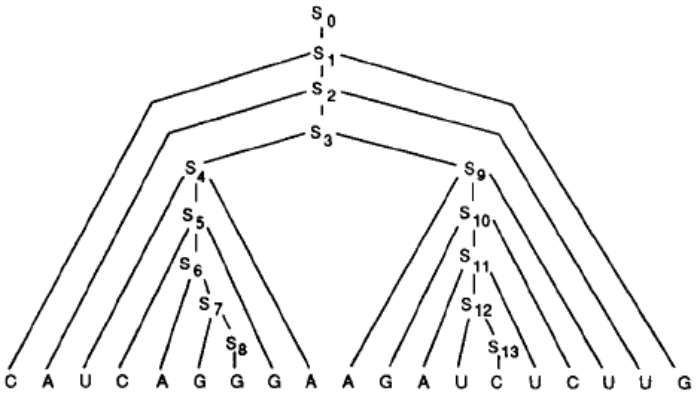
Every production rules contain non-terminal (‘S’) and terminal symbols (‘a’, ‘b’). For RNA structure prediction, terminal symbols represent A, U, C or G. Leftwise will generate a terminal symbol on the left while rightwise will generate a terminal symbol on the right. Pairwise is generating a base pair and bifurcation is generating a branch. The following example demonstrates how SCFG works in the recognition of CAUCAGGGAAGAUCUCUUG.

Example 1 : Structure prediction by implementing SCFG.

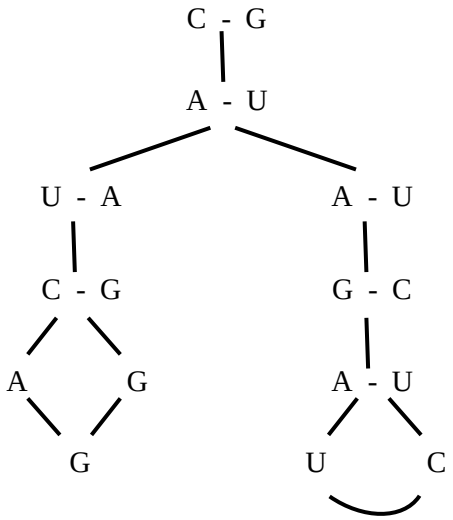
Sequence : CAUCAGGGAAGAUCUCUUG

S_1	Pairwise
$\boldsymbol{C}\,S_2\,\boldsymbol{G}$	Pairwise
C$\boldsymbol{A}\,S_3\,\boldsymbol{U}$G	Bifurcation
CA S_4 S_9 UG	Pairwise
CA$\boldsymbol{U}\,S_5\boldsymbol{A}\,S_9$ UG	Pairwise
CAU$\boldsymbol{C}\,S_6\,\boldsymbol{G}$A S_9 UG	Leftwise
CAUC$\boldsymbol{A}\,S_7$ GA S_9 UG	Leftwise
CAUCA$\boldsymbol{G}\,S_8$ GA S_9 UG	Leftwise
CAUCAG$\boldsymbol{G}$GA S_9 UG	Pairwise
CAUCAGGGA$\boldsymbol{A}\,S_{10}\,\boldsymbol{U}$UG	Pairwise
CAUCAGGGAA$\boldsymbol{G}\,S_{11}\,\boldsymbol{C}$UUG	Pairwise
CAUCAGGGAAG$\boldsymbol{A}\,S_{12}\,\boldsymbol{U}$CUUG	Leftwise
CAUCAGGGAAGA$\boldsymbol{U}\,S_{13}$UCUUG	Leftwise
CAUCAGGGAAGAU$\boldsymbol{C}$UCUUG	End

Parse tree :



Structure :



However, there are some limitations for structure prediction by SCFG method. SCFG algorithm would require time and memory proportional to at least L^3 (L is the length of sequence.) [23]. Thus, this method is only suitable for short sequence RNA structure prediction.

2.4.4 Comparative sequence analysis

RNA secondary structure prediction by comparative sequence analysis requires several sequences for evaluating similarity among them. Usually the input sequence for structure prediction is known as target sequence while sequence used to compare with target sequence will be known as reference sequence. Besides, aligning sequences may provide the evolutionary history and information of how closely related the sequences are.

Simple alignment of two sequences is just comparing the bases one at a time. If they are identical, then a match score is assigned according to the scoring system (Krane & Raymer 2003). Also, there would be a penalty for non-identical bases. Besides checking whether the bases are identical, gaps are also allowed while aligning sequences. By referring to the example scoring system (Refer to the following page), the score of aligning two sequences with gaps and without gaps is shown.

$$score = \sum_{i=1}^{N} S_i\,, \quad \text{where } S_i = \begin{cases} +1 & \text{match} \\ 0 & \text{mismatch} \\ -1 & \text{gap penalty} \end{cases}$$

Alignment without gap penalty :

Alignment	GGCCAUG	GGCCAUG	GGCCAUG
	GCAUG	GCAUG	GCAUG
Score	+1	+2	+4

Alignment with gap penalty (only three examples are shown here) :

Alignment	GGCCAUG	GGCCAUG	GGCCAUG
	GC--AUG	G-C-AUG	G--CAUG
Score	+2	+3	+3

Since alignment involve gap penalty will generate many possible alignments, scoring matrix is used to obtain the alignment with optimum score. This can be implemented using dynamic programming as in the case free energy minimization method. The recurrence relation for aligning two

sequences is shown on next page while scoring matrix and result are shown in Figure 2.12.

$$S(i,j) = \max \begin{cases} S(i-1,j-1)+1 & \text{match} \\ S(i-1,j)-1 & \text{gap penalty (left)} \\ S(i,j-1)-1 & \text{gap penalty (top)} \end{cases}$$

where $S(i, j)$ is the optimal score at position (i, j).

Score matrix :

		G	**C**	**A**	**U**	**G**
	0	-1	-2	-3	-4	-5
G	-1	1	0	-1	-2	-3
G	-2	0	1	0	-1	-1
C	-3	-1	1	1	0	-1
C	-4	-2	0	1	1	0
A	-5	-3	-1	1	1	1
U	-6	-4	-2	0	2	1
G	-7	-5	-3	-1	1	(3)

Optimal score : 3

Result :

```
-G-CAUG
GGCCAUG
```

Figure 2.12: Aligning two sequences using dynamic programming.

Simple alignment can be modified for multiple sequence alignment. Multiple sequence alignment will identified the conserve regions among sequences and produce the structure with highest similarity while compared to reference structures. Therefore, similar sequences are usually used for comparing with target sequence.

There are various implementation of comparative sequence analysis such as SimulFold (Meyer & Miklós 2007) and hxmatch (Witwer, Hofacker & Stadler 2004). Although results obtained by these two methods are comparable to other algorithms, they have some limitations too. Structure prediction by this approach requires several sequences for aligning the input sequence. Therefore, this method is generally used in prediction of RNA sequences belong to certain specific type of RNA in which their common structure is known. For example, tRNA always fold into a cloverleaf structure.

2.5 RNA secondary structure prediction with pseudoknot

In this section, some methods of RNA secondary structure prediction including pseudoknots will be discussed.

2.5.1 Dynamic programming

Prediction using dynamic programming is proposed by Akutsu T (Akutsu 2000). This method only deals with simple pseudoknot (H-type pseudoknot in Figure 2.5) using the recurrence relation shown on the following page.

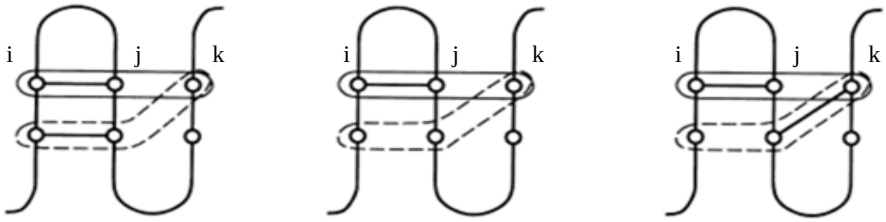
$$S_L(i,j,k) = v(a_i, a_j) + max\begin{Bmatrix} S_L(i-1,j+1,k), \\ S_M(i-1,j+1,k), \\ S_R(i-1,j+1,k) \end{Bmatrix}$$

$$S_R(i,j,k) = v(a_j, a_k) + max\begin{Bmatrix} S_L(i,j+1,k-1), \\ S_M(i,j+1,k-1), \\ S_R(i,j+1,k-1) \end{Bmatrix}$$

$$S_M(i,j,k) = max\begin{Bmatrix} S_M(i-1,j,k), S_M(i,j+1,k), S_M(i,j,k-1), \\ S_L(i-1,j,k), S_L(i,j+1,k), \\ S_R(i,j+1,k), S_R(i,j,k-1) \end{Bmatrix}$$

where $v(a_i, a_j) = 1$ if (a_i, a_j) is a base pair, otherwise $v(a_i, a_j) = -\infty$.

$S_L(i,j,k)$ corresponds to the case where *i*th and *j*th nucleotides make a base pair. This is illustrated in the Figure 2.13 and the similarly for $S_R(i,j,k)$ and $S_M(i,j,k)$.



$S_L(i,j,k)= v(a_i,a_j)$ $S_L(i,j,k)= v(a_i,a_j)$ $S_L(i,j,k)= v(a_i,a_j)$
$+\boldsymbol{S_L(i-1,j+1,k)}$ $+\boldsymbol{S_M(i-1,j+1,k)}$ $+\boldsymbol{S_R(i-1,j+1,k)}$

Figure 2.13: Illustration of the recurrence used in dynamic programming.

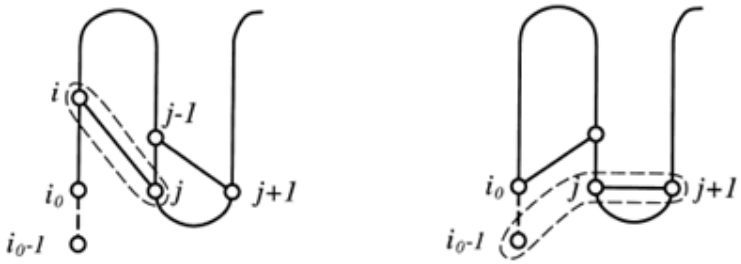
Besides, initialization is performed as well. Below is the initialization procedure and followed by figure illustrating this process.

$$S_L(i,j,j) = v(a_i,a_j) \quad \forall i < j$$

$$S_R(i_0-1,j,j+1) = v(a_j,a_{j+1}) \quad \forall j$$

$$S_L(i_0-1,j,k) = S_R(i_0-1,j,k) = S_M(i_0-1,j,k) = 0$$

for the other j,k satisfying $k=j$ or $k=j+1$



$S_L(i,j,j) = v(a_i,a_i)$ $S_R(i_0-1,j,j+1) = v(a_j,a_{j+1})$

Figure 2.14: Illustration of the initialization process.

Then, for each pair (i_0, k_0), computes the above scores and obtains the score of a pseudoknot whose endpoints are (i_0, k_0) by :

$$S_{pseudo}(i_0, k_0) = max_{i_0 \le i < j < k \le k_0}\{S_L(i,j,k), S_M(i,j,k), S_R(i,j,k)\}$$

Finally, optimal score $S(1,n)$ is computed by the following recurrence formula:

$$S(i,j) = max\Big\{S_{pseudo}(i,j), S(i+1,j-1) + v(a_i,a_j), max_{i<k\le j}\{S(i,k-1), S(k,j)\}\Big\}$$

Akutsu's method can predict RNA structure with simple pseudoknots. For an input sequence of length *n*, this algorithm requires $O(n^4)$ time complexity and it increases with the coverage of types of pseudoknot. Therefore, improvement can be done in order to decrease the time complexity of this algorithm. Also, there is no established energy function known for pseudoknot structure. This is important for evaluating the energy contribution of loops and stems in pseudoknot structures. Furthermore, it has not been implemented and tested for RNA structure prediction.

2.5.2 pknotsRG

pknotsRG is an pseudoknot prediction algorithm developed by Jens Reeder and Robert Giegerich (Reeder, Steffen & Giegerich 2007). It produces structure

with minimum free energy value based on dynamic programming approach. This algorithm requires $O(n^4)$ time and $O(n^2)$ space for structure prediction which is much reduced while compared to the algorithm developed by Akutsu (Akutsu 2000).

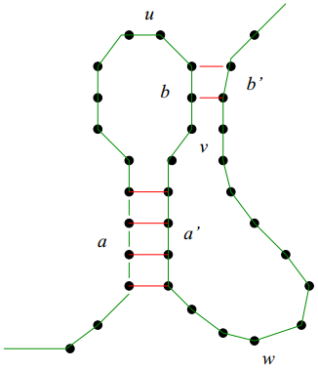


Figure 2.15: Simple pseudoknot which is formed by two stems (*a*-*a'* and *b*-*b'*).

pkntosRG is designed to speed up the prediction of RNA secondary structure with simple pseudoknot (H-type pseudoknot) by the implementation of three canonization rules:

(i) Both strand in a stem must have the same length (|*a*| = |*a'*| and |*b*| = |*b'*|). Therefore, no bulge is allowed in the stem structure.

(ii) Both stems involved in pseudoknot structure (*a*, *a'* and *b*, *b'*) must have the maximal extend.

(iii) If two stems would overlap, their boundary is fixed at an arbitrary point between them.

First rule is to confirm that no bulge can be found in the stem involved in forming pseudoknot structure. Second rule will ensure that base pairings of stems involved in pseudoknot structure are having the maximal extend. This is due to energy model which strongly favors helix extension. Last rule is to draw the border between two stems facing each other and competing for the same bases.

This algorithm considers the class of simple recursive pseudoknot which is further restricted by three rules of canonization, canonized simple recursive pseudoknot (csr-PK). The implementation of canonization reduced the search space and expedites the process of structure prediction while compared to the algorithm developed by Rivas and Eddy (pknotsRE) which considers general classes of pseudoknots (pknotsRG – $O(n^4)$ time and $O(n^2)$ space, pknotsRE - $O(n^6)$ time and $O(n^4)$ space). In addition, evaluation of the coverage of csr-PK on known pseudoknot structures has been done as well. Results show that 135 out of 172 simple recursive pseudoknots are included in the class csr-PK (78.49%).

pknotsRG has been tested on RNA sequences obtained from Pseudobase. Results show that it is good in predicting RNA structure for short sequences. While predicting RNA structure for longer sequences (>400nt), the minimum free energy structure predicted diverged from experimentally

determined structure. Besides, this algorithm requires lengthy execution time for structure prediction.

2.5.3 HotKnots

HotKnots is a heuristic algorithm developed for RNA secondary structure prediction including pseudoknots. This algorithm predicts RNA structure based on the idea of iteratively forming stable stems (Ren et al. 2005).

This algorithm begins with the generation of simple stem-like substructures which are termed as ‘hotspots’. A set of hotspots will be computed as the basis for developing RNA structure by adding substructure one at a time into the partially formed structure. HotKnots maintained multiple partially formed structures and it considers several different addition of substructure in an attempt to produce a tree of candidate structures. Then, standard energy model is used to determine which structures at nodes of the tree have the lowest free energies. Besides, the energy model is also used to determine how to prune the tree of partial structures, so that more alternatives are explored from the most promising partial structures.

HotKnots has been tested on 43 RNA sequences. The length of test sequences can be divided into two categories, short (28-108 nt) and long (210-

400 nt). HotKnots outperformed the other software available (ILM, pknotsRE, STAR and pknotsRG-mfe) while predicting structure for short sequences. For structure prediction using long sequences, performance of HotKnots is better than the other software except STAR. However, HotKnots achieved higher sensitivity value than STAR on five out of twelve sequences in the long category.

Although performance of HotKnots is comparable to the other software available, it can be further improved by modifying the search technique. A more advanced search technique can reduce the execution time of the algorithm. In addition, better energy model can be used for structure determination and selection so as to increase its performance.

2.5.4 Dynamic weighted matching (DWM)

Dynamic weighted matching algorithm is another method being implemented. Liu et al. used a dynamic weight related with stem length and a recursive algorithm to predict RNA secondary structures including pseudoknots. This is done by searching the stem structure with maximum weight summation step-by-step (Liu et al. 2006). The space complexity of this algorithm is $O(n^2)$ and the time complexity is less than $O(n^3 \log n)$.

In this algorithm, RNA structure prediction is regarded as an optimization problem. The author introduced "compound weight" as the optimization criterion. This means that the algorithm looks for structure with maximum whole weight value as the predicted RNA secondary structure. The compound weight means addition of constant weight and dynamic weight which is defined as below while given a section of stem, $Stm(i, j, k)$.

$$W_{(i,j,k)} = \sum_{l=0}^{k-1} w_{i+1,j-1} + \frac{1}{3}(w_{GC} + w_{AU} + w_{GU})\sqrt{k}$$ where $\begin{cases} i & : \text{5'-end initial site.} \\ j & : \text{3'-end terminal site.} \\ k & : \text{Stem length.} \end{cases}$

The first term on the right hand side of equation is the constant weight whereas the following term is the dynamic weight. Constant weight is the sum of weight for every base pair in stem. Dynamic weight is a product of average weight and square root of stem length. A double recursive algorithm is used to search the stems with maximum weight sum and potential pseudoknot.

This algorithm works according to two main principles. First is the whole weight sum maximization and secondly is first-near-last far principle. The second principle means that juxtapose stems are considered first and nested stems are second (Refer to Figure 2.16 for juxtapose and nested stems.). According to van Batenburg et al. (Van Batenburg, Gultyaev & Pleij 1995), it seems like this principle produces better result because it yields structures

which are closer to real structures. On the whole, this algorithm performed well for tRNA sequences but not ideal for ncRNA due to increase of sequence length and longer distance interaction.

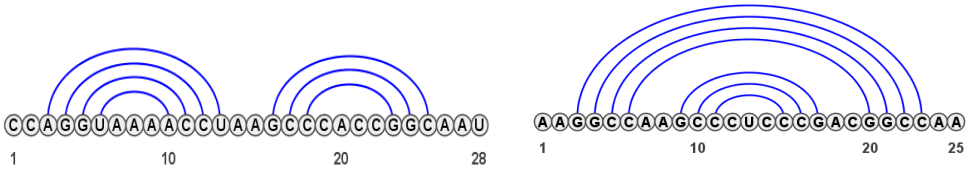


Figure 2.16: Juxtapose and nested stems.

This method is able to predict pseudoknot structure and it is found to be quite fast in structure prediction. The result shows that it is good in tRNA structure prediction with an average value of 95.08 for SN and 95.34 for SP. Although high SP and SN values are achieved, DWM algorithm has only been tested on 8 tRNA sequences downloaded from GenBank database.

2.5.5 CyloFold

CyloFold is an RNA secondary structure prediction algorithm that is not restricted in terms of pseudoknot complexity (Bindewald, Kluth & Shapiro 2010). It is developed by Eckart Bindewald, Tanner Kluth and Bruce A. Shapiro. This algorithm is based on simulation of folding process in a coarse-grained manner by choosing stems based on established energy rules.

CyloFold starts with generating a list of stems structure which contained at least three base pairs. Then, it continues with the folding simulation by picking stems from the list generated with a Boltzmann-weighted probability. Each chosen stem is represented by a very coarse-grained 3D representation in a virtual 3D workspace.

Stem structure is represented by a cylinder that is capped with a half-sphere on both ends. Single strand regions between stems are represented as constraints for the maximum distance between the ends of the capped cylinders. Then, every newly chosen capped cylinder will be placed into the 3D simulation space at a random position such that the distance-constraints are fulfilled. The distance constraints are a function of single-stranded sequence lengths between connected stems. In addition, steric feasibility of structure is check during the folding process. Once no more stems can be placed, a simulation run is completed and output from CyloFold is the overall best scoring structure generated by fifty simulation runs.

CyloFold has been tested on two datasets. First dataset consists of 26 RNA sequences in which their tertiary structure is available in the Protein Data Bank; Second dataset consists of 241 RNA sequences obtained from Pseudobase. Comparison among structures generated by CyloFold and other

existing software (pknotsRG, HotKnots and UNAfold) shows that CyloFold outperformed the others in term of the MCC score for both datasets.

This algorithm simulate the RNA folding process and produce the RNA secondary structure by selecting stems with a probability assigned according to their free energy contribution. Therefore, the structure prediction of CyloFold depends on the energy model employed. Furthermore, fifty simulation runs is required for the structure prediction of an input sequence. Thus, it might acquire lengthy execution time for structure prediction of longer RNA sequences due to the compute-intensive approach.

2.5.6 Stochastic multiple context-free grammar

Stochastic multiple context-free grammar (SMCFG) developed by Nobuyoshi Mizoguchi, Yuki Kato and Hiroyuki Seki is a grammatical approach for ncRNA structure prediction including pseudoknots (Mizoguchi, Kato & Seki 2011). In SMCFG, the right-hand side of a production rule is denoted by function application form. For example, a rule $S \rightarrow aXb$ can be expressed by $S \rightarrow f_1(X)$ where f_1 is the function defined by $f_1[x] = axb$ (x is an arbitrary sequence of terminal symbols) (Mizoguchi, Kato & Seki 2011).

This method assigned application probability to each production rule by the referring to the aligned multiple sequences. Application probability is defined as the product of the transition probability and the paired or unpaired probability. Then, it uses the CYK algorithm to predict the consensus secondary structure according to the grammar shown below:

Type	Rule	Function
E	$W_v \to (\varepsilon, \varepsilon)$	
S	$W_v \to J[W_y]$	$J[(x_1, x_2)] = x_1 x_2$
D	$W_v \to SK[W_y]$	$SK[(x_1, x_2)] = (x_1, x_2)$
U_{1L}	$W_v \to UP_{1L}^{\alpha_i}[W_y]$	$UP_{1L}^{\alpha_i}[(x_1, x_2)] = (\alpha_i x_1, x_2)$
U_{1R}	$W_v \to UP_{1R}^{\alpha_j}[W_y]$	$UP_{1R}^{\alpha_j}[(x_1, x_2)] = (x_1 \alpha_j, x_2)$
U_{2L}	$W_v \to UP_{2L}^{\alpha_k}[W_y]$	$UP_{2L}^{\alpha_k}[(x_1, x_2)] = (x_1, \alpha_k x_2)$
U_{2R}	$W_v \to UP_{2R}^{\alpha_l}[W_y]$	$UP_{2R}^{\alpha_l}[(x_1), x_2] = (x_1, x_2 \alpha_l)$
PL	$W_v \to BP_L^{\alpha_i \alpha_j}[W_y]$	$BP_L^{\alpha_i \alpha_j}[(x_1, x_2)] = (\alpha_i x_1 \alpha_j, x_2)$
PR	$W_v \to BP_R^{\alpha_k \alpha_l}[W_y]$	$BP_R^{\alpha_k \alpha_l}[(x_1, x_2)] = (x_1, \alpha_k x_2 \alpha_l)$
PC	$W_v \to BP_C^{\alpha_i \alpha_l}[W_y]$	$BP_C^{\alpha_i \alpha_l}[(x_1, x_2)] = (\alpha_i x_1, x_2 \alpha_l)$

Example 2 demonstrates the generation of simple RNA pseudoknot structure by using the grammar aforementioned.

Example 2 ： Sequence derivation using MCFG.

Sequence ： GCGAAGCGCGUUG

Structure :

Sequence derivation :

$$W_v \rightarrow UP_{1L}^{G}\left[(\varepsilon,\varepsilon)\right]=(G,\varepsilon)$$

$$W_v \rightarrow UP_{2L}^{G}\left[(G,\varepsilon)\right]=(G,G)$$

$$W_v \rightarrow BP_{C}^{AU}\left[(G,G)\right]=(AG,GU)$$

$$W_v \rightarrow BP_{C}^{AU}\left[(AG,GU)\right]=(AAG,GUU)$$

$$W_v \rightarrow UP_{2R}^{G}\left[(AGG,GUU)\right]=(AAG,GUUG)$$

$$W_v \rightarrow BP_{L}^{GC}\left[(AAG,GUUG)\right]=(GAAGC,GUUG)$$

$$W_v \rightarrow BP_{L}^{CG}\left[(GAAGC,GUUG)\right]=(CGAAGCG,GUUG)$$

$$W_v \rightarrow BP_{L}^{GC}\left[(CGAAGCG,GUUG)\right]=(GCGAAGCGC,GUUG)$$

$$W_v \rightarrow J\left[(GCGAAGCGC,GUUG)\right]=GCGAAGCGCGUUG$$

For SMCFG, it is developed for ncRNA structure prediction. It requires the input file with aligned multiple sequences for the calculation of application probability. Therefore, structure generated by SMCFG depends on the sequence alignment of input sequences. This algorithm has been tested on ncRNA sequences comprise from eight different families. Although the overall performance of SMCFG is almost comparable to existing pseudoknot prediction algorithm (hxmatch and Pair-SMCFG), it achieved low performance while predicting ncRNA from certain family of ncRNA sequences Mizoguchi, Kato & Seki 2011).

2.5.7 DotKnot

DotKnot is an pseudoknot detection algorithm which identifies two types of pseudoknot, that is H-type pseudoknot and intramolecular kissing hairpins (Sperschneider, Datta & Wise 2011). This algorithm is developed by Jana Sperschneider, Amitava Datta and Michael J. Wise.

This algorithm assembles pseudoknots in a constructive fashion from the secondary structure probability dot plot calculated by RNAfold (Hofacker et al. 1994). RNAfold is an RNA secondary structure prediction algorithm which output the minimum free energy structure and base pairing probability matrix. Firstly, a set of promising stems is obtained by setting a low-probability threshold ($1 \times E^{-11}$) so as to discover the potential pseudoknot stems. These stems are stored in a dictionary, D_s. The properties of stems are as follows:

(i) Contain at least 3 base pairs.

(ii) Absolute percentage increase or decrease of stack probabilities for subsequent base pairs in a stem must be lower than a certain threshold, δ.

(iii) Stem weight calculated using simple stacking model, w_{stack} must be lower than 0.0 kcal/mol.

(iv) Stem weight calculated using free energy model proposed by Turner group (Mathews et al. 1999), w must be lower than 4.0 kcal/mol.

Then, maximum weight independent set (MWIS) calculations is used to assemble noncrossing secondary structure elements. MWIS calculation is performed on a list of sorted endpoints for all stems. It will penalize long bulge or internal loops and ensure that confidence of stems is at least $1 \times E^{-3}$. These restrictions reduced the search space and expedite the algorithm while handling long sequences. Those stems which contained bulge or internal loops are stored in a dictionary (D_S^L) while stems which formed multiloops are stored in a different dictionary (D_S^M).

Stems from D_s and D_S^L are used to construct core H-type pseudoknots in which their energies are evaluated by advanced energy models. Restrictions on the structure of core H-type pseudoknots are:

(i) At most one interrupted stem is allowed.

(ii) The maximum and minimum length for each loop region is listed in Table 2.1.

Table 2.1: Restrictions on loop length of core H-type pseudoknot.

Loop	Location in Figure 2.15	Minimum loop length	Maximum loop length
L_1	*u*	1	100
L_2	*v*	0	50
L_3	*w*	2	100

(iii) For interrupted stems with more than 10 base pairs, L_1 must contain at least 2nt while L_3 must contain at least 6nt.

Subsequently, those core H-type pseudoknots with low energy value will be selected for constructing recursive pseudoknots. Each loop region (L_1, L_2 and L_3) is allowed to fold into any secondary structure elements. These structures can be found in D_s, D_s^L and D_s^M. Then, the loop entropy of recursive pseudoknots is recalculated using effective loop length. The effective loop length of a pseudoknot loop with internal structure elements is the number of unpaired nucleotides outside those internal structure elements plus the number of internal structure elements. Finally, those pseudoknots which fulfilled the following two criteria will be stored in a dictionary, D_p before the removal of false positive pseudoknots by using MWIS calculations.

(i) Free energy, $\Delta G(p_i)$ < -5.25 kcal/mol.

(ii) Normalized pseudoknot free energy, $\Delta G(p_i)/l_i \leq \varepsilon$, l_i denotes the length of pseudoknot p_i.

After obtained H-type pseudoknots, kissing hairpin structures are constructed by referring to the list of H-type pseudoknot structures stored in a specific manner. The free energy value for kissing hairpins is estimated by adding the stacking energies, including dangling ends for the three stems involved in kissing hairpins structure plus a length-dependent value for the loop entropies. Properties of stems involved in the formation of kissing hairpins are as follows:

(i) $w_{stack} < -5.0$ kcal/mol and $w < 2.0$ kcal/mol.

(ii) Confidence sum > $1 \times E^{-3}$.

(iii) Normalized kissing hairpins free energy, $\Delta G(k_i)/l_i \leq \varepsilon$.

This algorithm is designed for identifying 2 types of pseudoknot structures, H-type pseudoknot and intramolecular kissing hairpins. Kissing hairpins are restricted to be shorter than 400nt in order to improve the runtime of algorithm. Besides, it provides a number of near-optimal H-type pseudoknot and kissing hairpin candidates as well.

Results show that DotKnot performed better than pknots, FlexStem and RNAfold while predicting RNA structures with pseudoknots. It can predict kissing hairpins correctly and achieve highest MCC score for most test sequences. For pseudoknot-free test set, MCC score of DotKnot is comparable to pknotsRG and RNAfold which implement dynamic programming method for structure prediction (pknotsRG-0.59, RNAfold-0.57, DotKnot-0.55, HotKnots-0.55, FlexStem-0.52).

CHAPTER 3

STEM WEIGHT MAXIMIZATION ALGORITHM

This chapter presents an algorithm which predicts RNA secondary structure based on stem weight maximization. Stem is an important RNA structure as it is made up by continuous base stacking which stabilize the RNA structure. Base stacking of Watson-Crick base pairs (G-C and A-U) are more preferable than Wobble base pair (G-U). Hence, the proposed algorithm always select the stem which possess more preferable base pairs by referring to the stem weight value calculated.

Figure 3.1 shows the general view of stages involved in the proposed algorithm. At the end of this chapter, Example 6 which illustrates the overall structure prediction of stem weight maximization algorithm is presented.

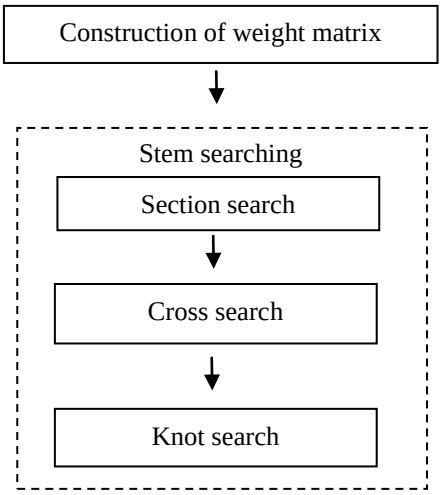


Figure 3.1: Flow chart of the proposed algorithm.

The proposed algorithm begins with the construction of weight matrix. The weight matrix constructed will be used for stem searching in which selected stem is the one with highest stem weight value. Stem searching is subdivided into three stages. Each stage will search for stems in different regions. Searching regions for "section search" are those remaining unpaired regions of previously found stem (Refer to Figure 3.4 for the searching regions of "section search".). Subsequently, the proposed algorithm will continue with "cross search" which searches for base pairing in the unpaired regions after "section search" (Refer to Figure 3.7 for determining searching regions of "cross search".). Finally, the "knot search" will search for pseudoknots if there are valid regions for the formation of pseudoknots (Refer to section 3.2.3 for the details of "knot search").

3.1 Construction of weight matrix

In this research, sequences used for structure prediction are obtained from FRABASE (Popenda et al. 2010), RNA STRAND (Andronescu et al. 2008) and CompaRNA (Puton et al. n.d.). Sequences downloaded from these databases include various types of RNA such as tRNA, ribosome and Ribonuclease P RNA.

The proposed algorithm begins with the construction of weight matrix which will be used for stem searching in the following stage. Weight matrix is a two dimensional matrix used to store the weight value assigned for each base pair. The proposed algorithm includes three types of base pair (G-C, A-U and G-U). These three types of base pair are assigned with a value each as its weight that is 11, 8 and 3 respectively (Liu et al. 2006). Higher value is assigned for Watson-Crick base pairs (G-C and A-U) due to their stabilization effect towards the RNA structure. Wobble pair (G-U) is assigned with lower weight value because it is thermally less stable than Watson-Crick base pairs. For other combination of base pairs, they are assigned with the value zero.

Given an input sequence of length N, a weight matrix of size $N \times N$ is created, $wMatrix[N][N]$. First, construction of weight matrix begins with the initialization of diagonal to zero value. Then, fill up the matrix according to the base pair's weight value as follows:

$$wMatrix(a_m, a_n) = \begin{cases} 11 \text{ , if } a_m = \text{G and } a_n = \text{C or vice versa.} \\ 8 \text{ , if } a_m = \text{A and } a_n = \text{U or vice versa.} \\ 3 \text{ , if } a_m = \text{G and } a_n = \text{U or vice versa.} \\ 0 \text{ , otherwise.} \end{cases}$$

m and n refer to the row and column of weight matrix. a_m refers to the nucleotide at row m while a_n refers to the nucleotide at column n.

Example 1 : Construction of weight matrix.

Input sequence : GGGCGACGCAGAAAAGAGGUGCACUUAUCUUU

Length (N) : 32

Matrix created : $wMatrix[N][N]$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
	G	G	G	C	G	A	C	G	C	A	G	A	A	A	A	G	A	G	G	U	G	C	A	C	U	U	A	U	C	U	U	U	
0	G	0	0	0	11	0	0	11	0	11	0	0	0	0	0	0	0	0	0	3	0	11	0	11	3	3	0	3	11	3	3	3	0	G
1	G	0	0	11	0	0	11	0	11	0	0	0	0	0	0	0	0	0	0	3	0	11	0	11	3	3	0	3	11	3	3	3	1	G
2	G	0	11	0	0	11	0	11	0	0	0	0	0	0	0	0	0	0	0	3	0	11	0	11	3	3	0	3	11	3	3	3	2	G
3	C	0	11	0	0	11	0	0	11	0	0	0	0	0	0	11	0	11	11	0	11	0	0	0	0	0	0	0	0	0	0	0	3	C
4	G	0	0	11	0	11	0	11	0	0	0	0	0	0	0	0	0	0	0	3	0	11	0	11	3	3	0	3	11	3	3	3	4	G
5	A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	8	8	0	8	0	8	8	8	5	A
6	C	0	11	0	0	11	0	0	11	0	0	0	0	0	0	11	0	11	11	0	11	0	0	0	0	0	0	0	0	0	0	0	6	C
7	G	0	11	0	0	11	0	0	11	0	0	0	0	0	0	11	0	11	11	0	11	0	0	0	0	0	0	0	0	0	0	0	7	G
8	C	0	11	0	0	11	0	0	11	0	0	0	0	0	0	11	0	11	11	0	11	0	0	0	0	0	0	0	0	0	0	0	8	C
9	A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	8	8	0	8	0	8	8	8	9	A
10	G	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	11	0	11	3	3	0	3	11	3	3	3	10	G
11	A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	8	8	0	8	0	8	8	8	11	A
12	A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	8	8	0	8	0	8	8	8	12	A
13	A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	8	8	0	8	0	8	8	8	13	A
14	A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	8	8	0	8	0	8	8	8	14	A
15	G	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	11	0	11	3	3	0	3	11	3	3	3	15	G
16	A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	8	8	0	8	0	8	8	8	16	A
17	G	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	11	0	11	3	3	0	3	11	3	3	3	17	G
18	G	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	11	0	11	3	3	0	3	11	3	3	3	18	G
19	U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	8	0	0	0	0	8	0	0	0	0	19	U
20	G	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	11	0	3	3	0	3	11	3	3	3	20	G
21	C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	C
22	A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	8	0	8	0	8	8	8	22	A
23	C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23	C
24	U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	U
25	U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	U
26	A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	8	0	8	0	8	8	8	26	A
27	U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	U
28	C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	C
29	U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	U
30	U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	U
31	U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	U

Example 1 shows the construction of weight matrix for the input sequence, GGGCGACGCAGAAAAGAGGUGCACUUAUCUUU. Input sequence has the length of 32, therefore dimension of weight matrix created should be $wMatrix[32][32]$. Subsequently, initialize the diagonal with value zero and start to fill up weight matrix at the position $wMatrix[0][31]$ which is circled in Example 1. At this position, G is paired with U, thus it is filled with the value '3'. The process of filling up weight matrix continues until all position which satisfies the condition of $m < n$ is filled with the corresponding weight value. The completed weight matrix would be an upper triangular matrix.

3.2 Stem searching

After construction of weight matrix, the next step would be searching for stem. Stem searching is looking for continuous base pairing of nucleotides which form a stem (Refer to Figure 3.2 for the structure of stem in weight matrix.). In the weight matrix, stem is the continuous regions which contained nonzero entries diagonally.

The stem searching process is divided into three stages, which search for stem in different regions:

(i) Section search

(ii) Cross search

(iii) Knot search

These three searching processes are dependent. The first search (section search) must be completed before carry on to the next search (cross search) and finally the last search (knot search).

"Section search" is searching for stem in three different regions (Refer to Figure 3.4 for the illustration of searching regions.). It will select the stem with maximum stem weight value from each region. "Cross search" will identify possible base pairing in the unpaired region (Refer to Figure 3.8 for determining searching region of "cross search".). "Knot search" is the final stage of stem searching. It will search for pseudoknot structures in between loop and unpaired regions.

In the proposed algorithm, "section search" and "knot search" consider only those stems which possess stem length greater than 2nt. In addition, loop length of stems found in "section search" must be greater than 2nt as well. For "cross search", it is searching for possible base pairing in the bulge and internal loop regions. Hence, the stem length is set to be not greater than 2nt in this case. Besides, it will search for possible stem if there exists a pair of unpair regions which are located at the beginning and ending of input sequence. Here, the

minimum stem length is 3nt. The following subsections will discuss about each of these searching stages in detail.

3.2.1 Section search

"Section search" is the first stage of stem searching. It searches for stems which possess maximum stem weight value in the specified regions of weight matrix constructed. The stem weight value is the summation of successive base pairing diagonally in the weight matrix.

Since the formation of Watson-Crick base pairs stabilize the RNA secondary structure, "section search" begins with identifying the stems with maximum stem weight value from the entire weight matrix constructed in the earlier stage. Stems are marked by the regions perpendicular to the diagonal with continuous non-zero entries in the weight matrix. In Figure 3.2, it shows all possible stems for the input sequence, GGGCGACGCAGAAAAGAGGUGCACUUAUCUUU.

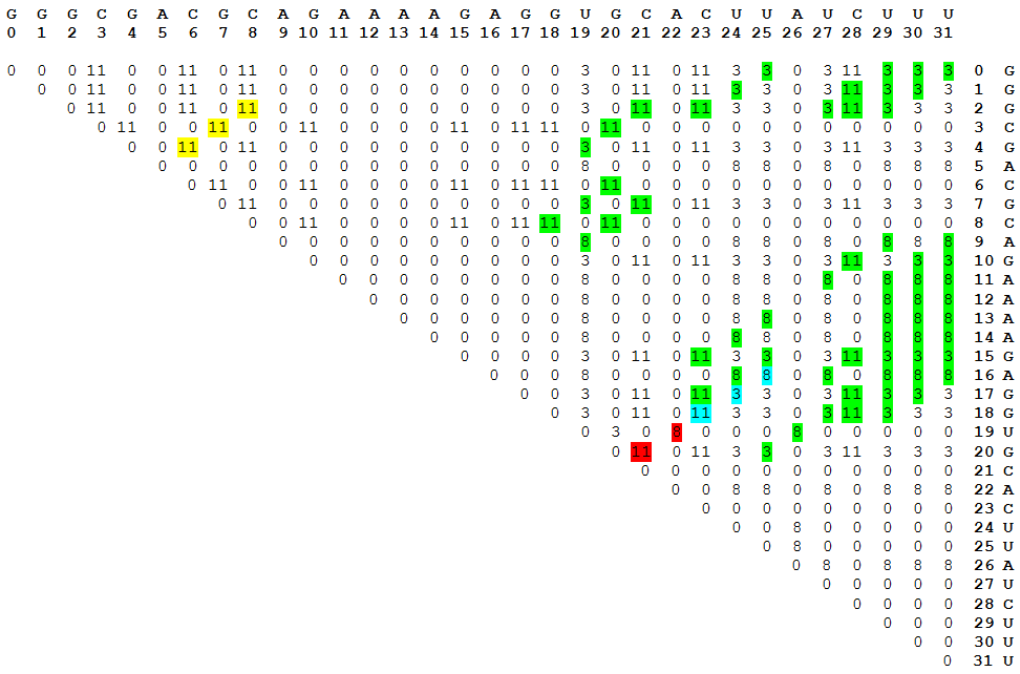


Figure 3.2: Stems found in the weight matrix. Valid stems are highlighted in green and blue color while invalid stem is highlighted in yellow color.

In “section search”, stem and loop length of stems must be greater than 2nt. For those stems which do not possess loop length greater than 2nt, the proposed algorithm will unpair the innermost base pairing of nucleotides in order to form a valid stem. Thus, the stem length is deducted by one. For example, two innermost base pairs of stem (highlighted in red color) highlighted in blue color has been unpaired in order to become valid stems with loop length of 4nt. The stem weight of this stem is 22 which can be obtained by summing up the values involved in the stem formation (8, 3 and 11 diagonally).

In Figure 3.2, the stem highlighted in yellow color is invalid because its loop length is less than 3nt. The proposed algorithm cannot unpair the innermost base pair and make it become valid stem due to its stem length which is equal to the minimum value, 3. Figure 3.3 illustrates the process of stem modification aforementioned.

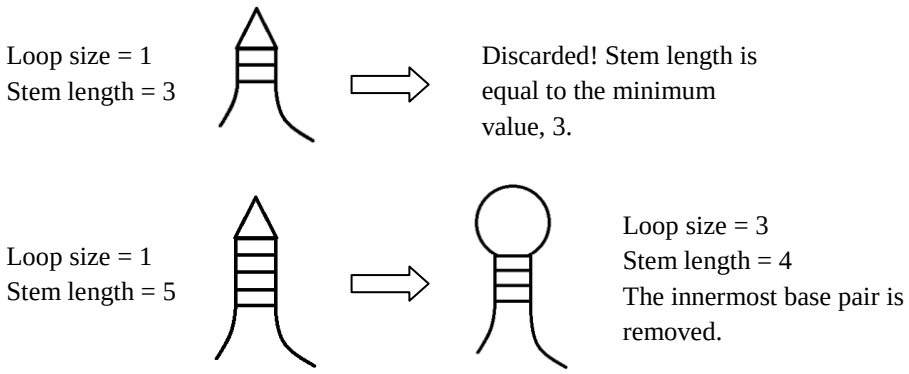


Figure 3.3: Stem adjustment when loop size is less than 3nt.

Details for “section search” are as follows:

1. Search for valid stems within the specified region in weight matrix. The initial searching region for “section search” is the entire weight matrix.
2. Calculate the weight of each valid stem and store them into a list. The stem weight, w of a stem with length k is obtained by summing up those weight values recorded in the weight matrix, starting at the first position with nonzero entry as follows:

$$w = \sum_{k} wMatrix(m+k, n-k)$$

Note that stem is enclosed by neighboring entries with zero weight value that is both $wMatrix(m-1, n+1)$ and $wMatrix(m+k+1, n-k-1)$ are zero.

3. Stem selection.
    a) If stem which involved base pairing between the first and last nucleotides of input sequence exist, select it.
    b) Else, select the stem with maximum stem weight value from the list of stem weight. If there are several stems which possess the maximum stem weight value, select the one with the shortest stem length.
4. Divide the selected stem into three regions and repeat step 1-3 for each region.

First stem for any input sequence would be the one with the highest stem weight value found within the entire weight matrix. Then this stem will be divided into three regions and "section search" continues looking for stem with similar properties in each region. Before searching for stem in each region, searching regions of at least 6nt must be fulfilled because minimum six nucleotides are required to form a stem with at least three base pairs in length.

Figure 3.4 shows three regions of stem. Region 1 is the unpair region before the stem while region 3 is the unpair region after the stem. Region 2 is the loop region of the stem. For each stem found, “section search” will divide it into three regions and continue to search for new stem in each region recursively. This process continues until no more stem can be found.

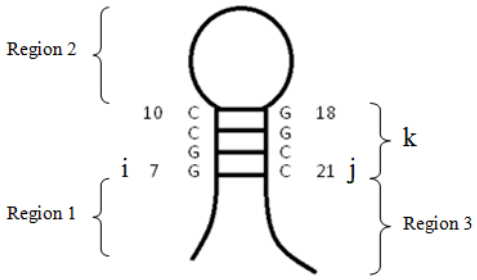
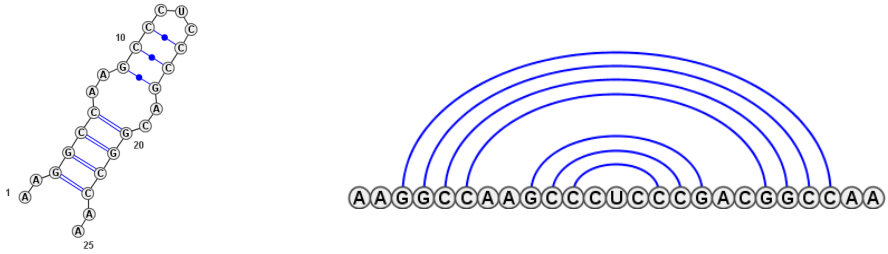


Figure 3.4: Three regions of a stem.

During the first stem searching process of “section search”, if there is a stem which includes base pairing between the first and last nucleotides of input sequence, it will be selected and stored in `stem_section` without further evaluation. This is due to the preference of the proposed algorithm which favors nested stems rather than juxtapose stems. Observations from the experimental structure obtained from FRABASE shows that RNA structure preferred nested stems while compared to juxtapose stems (64 structures with nested stem structures; 7 structures with juxtapose stem structures.). Figure 3.5 shows the configuration of nested stems and juxtapose stems.

(a) Juxtapose stem



(b) Nested stem

Figure 3.5: Juxtapose and nested stems.

When there are several stems with maximum stem weight value, the proposed algorithm will select the one which possess the shortest stem length (Refer to Figure 3.6 for the illustration of stem selection.). Consequently, RNA structure predicted by the proposed algorithm contains more Watson-Crick base pairs (G-C and A-U) which stabilize the structure.

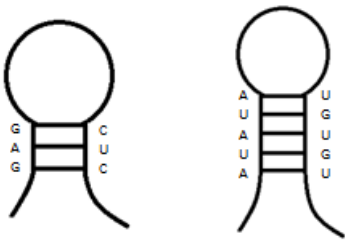


Figure 3.6: Stem selection while both stems are having equal stem weight value, 30. Stem on the left is selected due to its shorter stem length, 3.

Every selected stem will be stored as a triplet, `[i, j, k]`. `i` indicates the starting position of stem, `j` is the ending position and `k` is the length of stem. Position of `i`, `j` and `k` can be found in Figure 3.4. For "section search", the proposed algorithm will discard those stems which do not possess stem length and loop length of at least three base pairs and three nucleotides respectively. All triplet of stems found in "section search" are stored in a list named `stem_section`.

Example 2 : Stem searching of "section search".

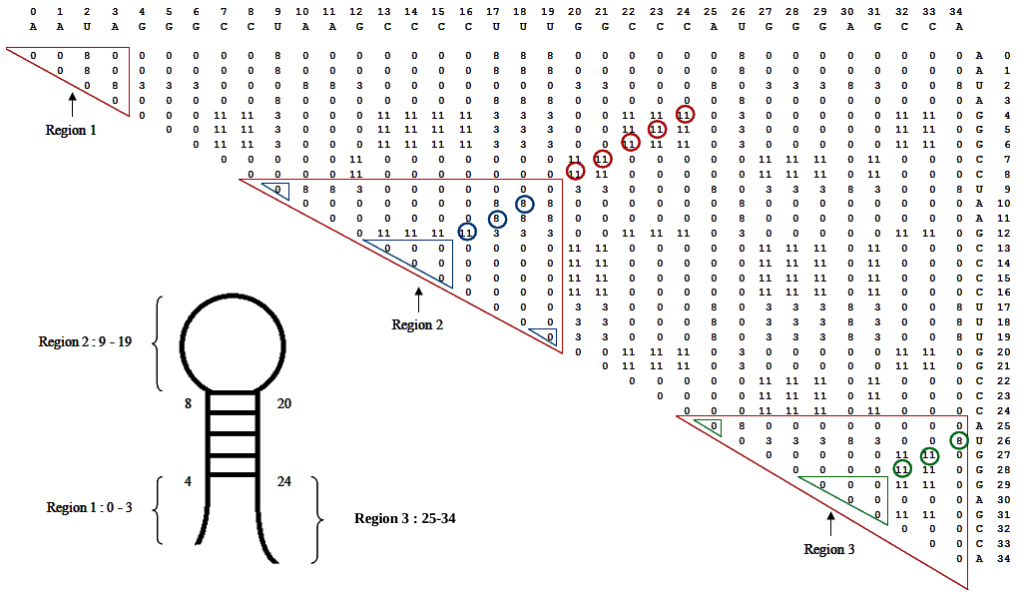
Input sequence : AAUAGGGCCUAAGCCCCUUUGGCCCAUGGGAGCCA

Stem found (`stem_section`) : `[[4, 24, 5], [10, 18, 3],`

`[26, 34, 3]]`

Length (N) : 35

Weight matrix :



"Section search" starts with finding the first stem with maximum stem weight value in weight matrix. In Example 2, first stem found is `[4, 24, 5]` with weight value 55 (11+11+11+11+11). The maximum stem weight value is obtained by summing up those values highlighted with red circle in the weight matrix. Then, this stem is divided into three regions. Region 1 ranges from 0-3, Region 2 ranges from 9-19 and Region 3 ranges from 25-34. These three regions can be found in the stem diagram (lower left corner) of Example 2. Also, these three regions are highlighted with red triangle in the weight matrix.

Then, "section search" continues searching for stem in each of the three regions. The second stem found is `[10, 18, 3]` (circled in blue) and three regions of this stem are highlighted using blue triangle. These three blue triangles are invalid regions for stem searching because none of them fulfilled the minimum length of searching region, 6nt. As a result, "section search" does not perform stem searching in these regions. After that, "section search" carry on its stem searching process in Region 3 of first stem, `[4, 24, 5]`. Here, it found another stem, `[26, 34, 3]` (circled in green). Subsequently, no more stem can be found and "section search" halt at this point.

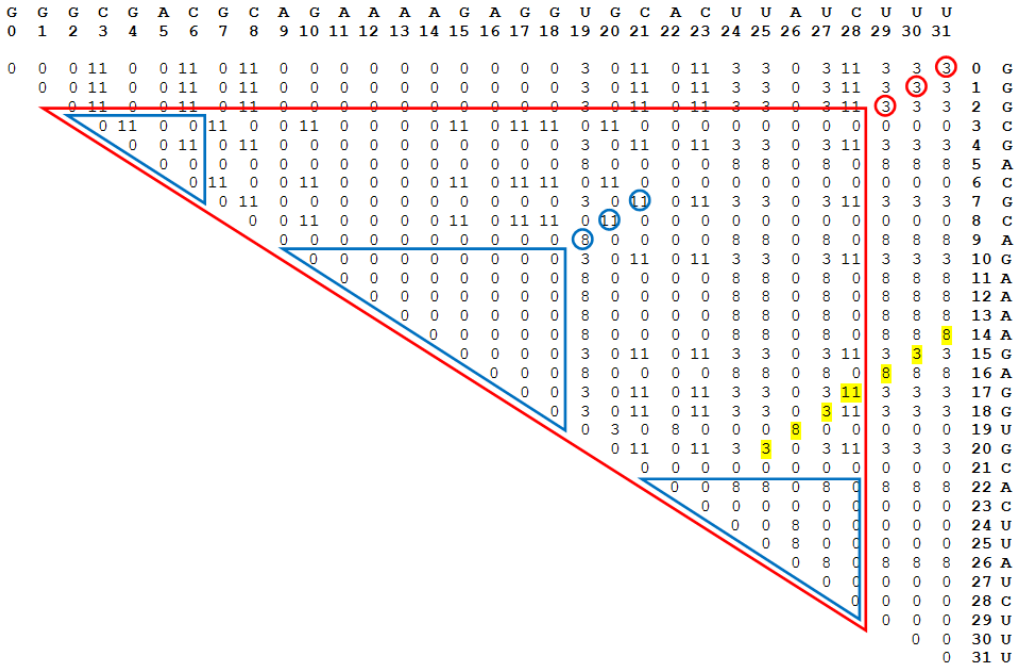
Example 3 : Stem searching of "section search" involving base pairing of first and last nucleotide.

Input sequence : GGGCGACGCAGAAAAGAGGUGCACUUAUCUUU

Stem found (`stem_section`): `[[0, 31, 3], [7, 21, 3]]`

Length (*N*) : 32

Weight matrix :



Example 3 shows the process of “section search” which involved base pairing of starting and ending nucleotides. In step 3(a) (Refer to page 49.), "section search" will check on all valid stems before select the one with maximum stem weight value during first run. Therefore, the first stem being selected in Example 3 is `[0, 31, 3]` (circled in red) instead of `[14,31,7]`(highlighted in yellow color).

Since the stem found begins at the position ‘0’ and ends at the position ‘31’, this means that the next search do not involve Region 1 and Region 3. Hence, “section search” continue to search for stems in Region 2, which is the area ranges from 3 to 28 (Region bounded with red triangle in Example 3.). Stem found in this region is `[7, 21, 3]` (circled in blue). After this, “section search” continue with stem searching in regions bounded with blue triangle. No stem can be found in any of these regions, therefore “section search” stop at this moment.

Although “section search” and DWM are similar but the process of stem selection is different. DWM algorithm selects stem with the maximum whole weight value which is the compound weight (Refer to section 2.5.4 for the detail of DWM). For the proposed algorithm, it searches for potential stem before the calculation of stem weight instead of compound weight for DWM algorithm. Besides, stem length and loop size must be at least 3nt in order to be a potential stem.

3.2.2 Cross search

“Cross search” is searching for stem with maximum stem weight value as well. It differs from “section search” in the searching region. “Cross search” looks for stem in the unpaired regions. “Unpaired region” refers to the region which contained unpaired nucleotides excluding hairpin loop. Unpaired regions can

be determined by exploring the dot bracket representation of `stem_section` (Refer to Figure 3.7 for searching of unpaired regions for "cross search".) . The minimum size of unpaired region is set to be at least 3nt.

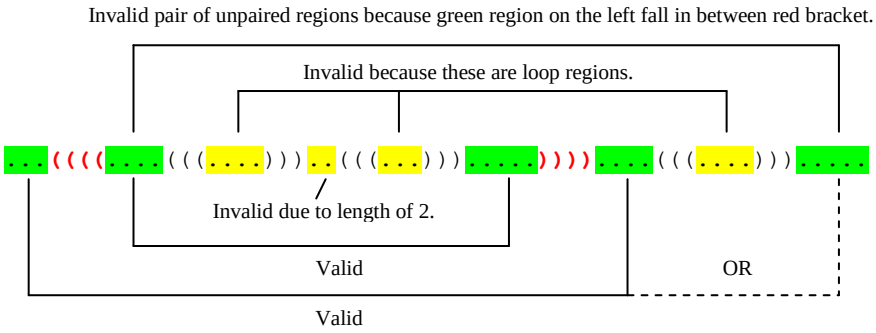


Figure 3.7: Determine unpaired regions for "cross search".

Unpaired regions are stored in a list, S . Each region is represented using two values which indicate the starting and ending positions of region. After determined the unpaired regions, "cross search" will begin the stem searching process. In general, the stem searching of "cross search" is summarized as follows:

1. Select the first region, S_0 and last region, $S_{len(S)-1}$ from S .
2. Determine the validity of regions selected in step 1.
    a) Ensure that the first region is not identical with the second region. If both regions are identical, remove the first region from S and repeat step 1.

b) Search for those stems which enclosed the first region and store them into a list. Repeat this step for the second region.

c) Check whether the list of stems for both regions are identical.

- If both lists are identical, proceed with step 3
- Else, change the second region to $S_{len(S)-2}$. Repeat step 2 for the pair of newly selected regions.

3. Search for base pairing in between both regions.

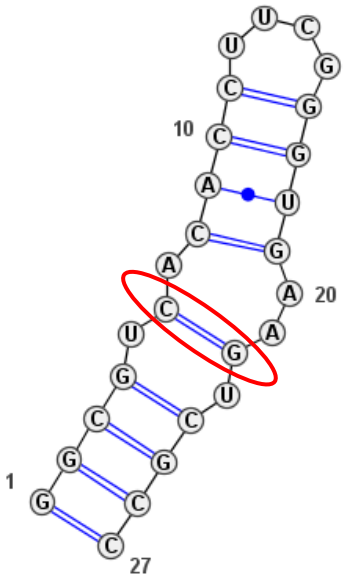
a) If stems can be found, select the one with maximum stem weight value and remove both selected regions from S. Then, repeat step 1.

b) Else, change the second region to $S_{len(S)-2}$. Repeat stem 2 for the pair of newly selected regions. Otherwise, remove the first region from S and repeat step 1.

"Cross search" begins by selecting the first and last regions from the list S. For every pair of dissimilar selected regions, "cross search" will determine whether they are valid for stem searching by compiling a list of stems for each region. The list contains those stems which enclosed by the selected region. If both lists are not identical, the second region which is the last region in S will be changed to the next region before it, $S_{len(S)-2}$.

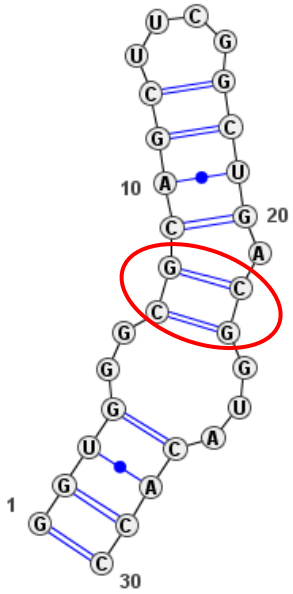
Subsequent to the existence of valid regions for stem searching, “cross search” will search for stems in between the regions and select the one which possess maximum stem weight value. If the pair of valid regions include first and last nucleotides of input sequence, the minimum length of stem is set to be at least 3nt while less than 3nt otherwise. The proposed algorithm discards short stems which formed by base pairing involving the beginning and ending of input sequence because isolated base pairs destabilize the RNA structure. Figure 3.8 shows two examples of stem found by “cross search”.

1BYJ



GGCGUCACACCUUCGGGUGAAGUCGCC
((((.(.((((....)))).).)))).

1EBR



GGUGGGCGCAGCUUCGGCUGACGGUACACC
((((..((((((....)))).))...)))).

Figure 3.8: Base pairing found by “cross search”.

Example 4 : Stem searching of “cross search”.

Input sequence : GGGCGACGCAGAAAAGAGGUGCACUUAUCUUU

Stem found : `stem_section - [[0, 31, 3], [7, 21, 3]]`

`stem_cross - [[4, 28, 2]]`

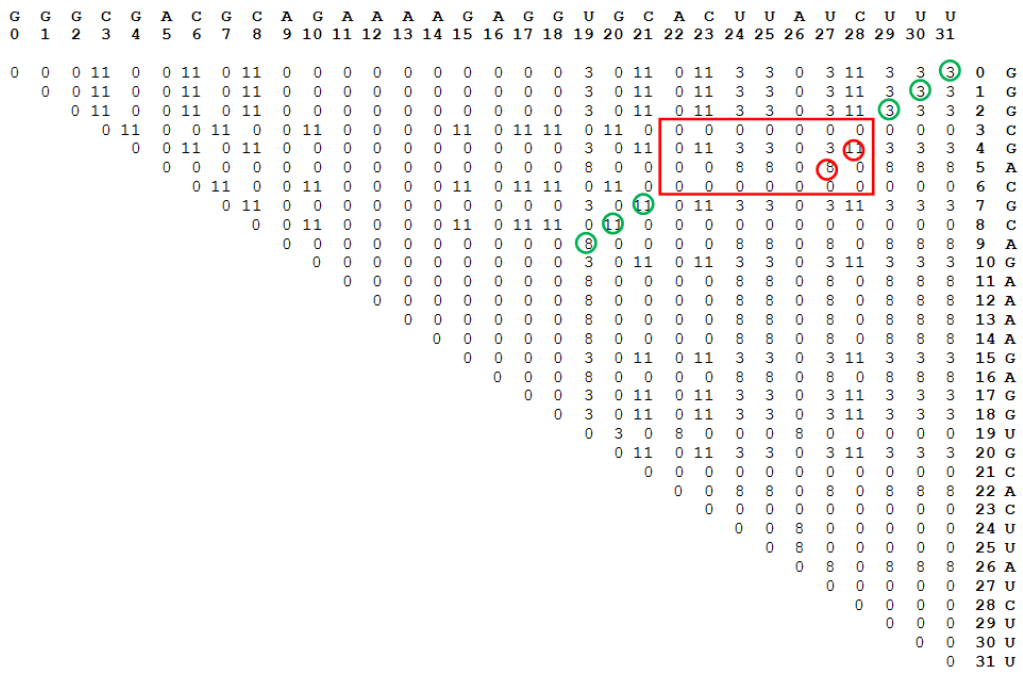
Dot-bracket representation of `stem_section` :

`(((....(((.........)))).......)))`

Length (*N*) : 32

Valid regions : `[[3, 6], [22, 28]]`

Weight matrix :



Example 4 shows the stem searching for “cross search”. Those stems found in “section search” are stored in `stem_section` (`[[0, 31, 3], [7, 21, 3]]`). Thus, the proposed algorithm continues with “cross search” by determining the valid regions (`[3, 6], [22, 28]`) before stem

searching. A stem is found in these two regions, that is `[4, 28, 2]`. Then, "cross search" stop because no more valid regions are available for stem searching.

3.2.3 Knot search

"Knot search" which is the final stage of stem searching, will identify the existence of pseudoknot structures which are found to be functionally important in some cellular activities (Draper, Gluick & Schlax 1998). The proposed algorithm can identify two types of pseudoknot which are H-type pseudoknot and kissing hairpins. H-type pseudoknot is form by the base pairing between hairpin loop and unpaired region. Kissing hairpins is form by the base pairing between two hairpin loops.

Hence, this stage begins with identifying suitable regions for forming pseudoknot structures. "Knot search" requires two types of valid regions which are loop regions and unpaired regions. Loop regions are stored in `loop` while unpaired regions are stored in `unpair`. These two regions are determined by exploring the dot-bracket representation of stems found in previous stem searching processes ("section search" and "cross search"). Stems found in "knot search" are stored in `stem_knot`. Figure below illustrates how to determine the searching regions for "knot search".

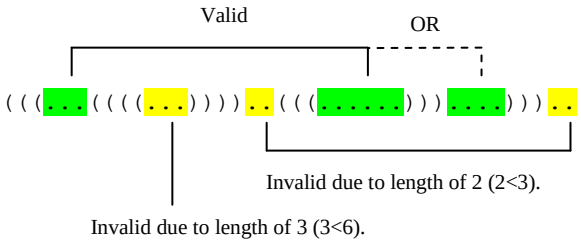


Figure 3.9: Determine searching regions for knot search.

By observing those experimentally determined structures obtained from FRABASE, all pseudoknots are found in large hairpin loop regions. Therefore, the proposed algorithm will not consider small loop regions while searching for pseudoknot structures. In the proposed algorithm, the loop length for pseudoknot structures formation must be at least 6nt. In addition, unpaired regions which contain less than 5nt are excluded as well.

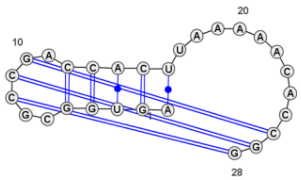
If both loop regions, L and unpaired regions, S are available, "knot search" will search for potential pseudoknot structures as follows:

1. Search for H-type pseudoknot.
    a) Select L_0 and S_0 from L and S respectively.
    b) If stems can be found, select the one with maximum stem weight value as the pseudoknot structure. Remove selected regions from L and S. Then, return to step 1.
    c) Else, select the next region in S and repeat previous step.

2. Search for kissing hairpins.
    a) Select L_i from L in which $L_i \neq L_0$.
    b) If stems can be found, select the one with maximum stem weight value as the pseudoknot structure. Remove selected regions from L. Then, repeat step 1.
    c) Else, return to step 2(a) by selecting the other loop region as L_i.
3. Remove L_0 from L and repeat step 1.

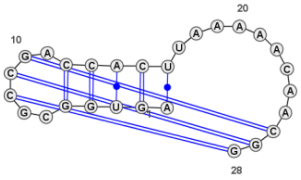
First, "knot search" will identify potential H-type pseudoknot structures. Therefore, it selects first region from L and second region from S to search for pseudoknot structure. When no more unpaired regions are available for constructing H-type pseudoknot, "knot search" will continue by selecting the next region in L as the second region. Hence, it is looking for kissing hairpins. The proposed algorithm prefers H-type pseudoknot due to its higher occurrence while compared to kissing hairpin structures. Among 15 FRABASE structures with pseudoknots, 11 of them are found to be H-type pseudoknot. Figure 3.10 shows two examples of pseudoknot structures found.

1YG4



AGUGGCGCCGACCACUUAAAAACACCGG
((((((..[[[.))))).........]]]

1D0T



AGUGGCGCCGACCACUUAAAAACAACGG
((((((..[[[.))))).........]]]

Figure 3.10: Pseudoknots found by “knot search”.

Example 5 ： Stem searching of “knot search”.

Input sequence ： GGGCGACGCAGAAAAGAGGUGCACUUAUCUUU

Stem found : `stem_section - [[0, 31, 3], [7, 21, 3]]`

`stem_cross - [[4, 28, 2]]`

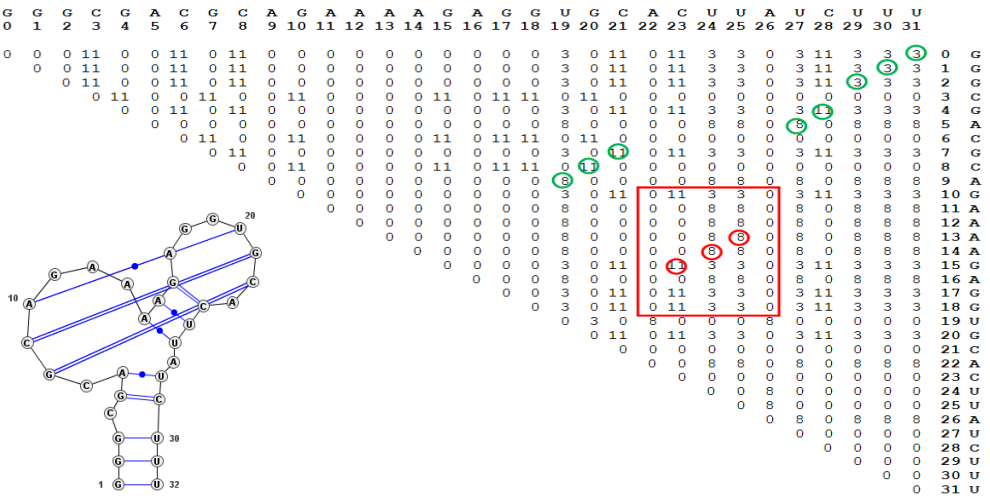
Dot-bracket representation of stem found in previous stem searching processes :

(((.((.(((.........))).....)))))

Length (*N*) ： 32

Valid regions ： `loop - [[10, 18]], unpair - [[22, 26]]`

Weight matrix ：



Example 5 illustrates the stem searching of “knot search”. “Knot search” found the stem, `[13, 25, 3]` in the valid regions and stored it into `stem_knot`. This is a H-type pseudoknot because it involves base pairing between a loop ([10, 18]) and unpaired region ([22, 26]). In “knot search”, when a stem is found, the corresponding regions involved in base pairing will be removed from their corresponding list in which it is obtained. Subsequently “knot search” keeps on searching for stems if valid regions are available.

Once the stem searching process has been completed, the proposed algorithm will revise the dot-bracket representation by adding in the pseudoknot structures obtained by “knot search”. Pseudoknot structures which involved cross pairing of previously found stems will be represented using square bracket (‘[‘ and ‘]’). The bottom left figure of Example 5 shows the final structure obtained.

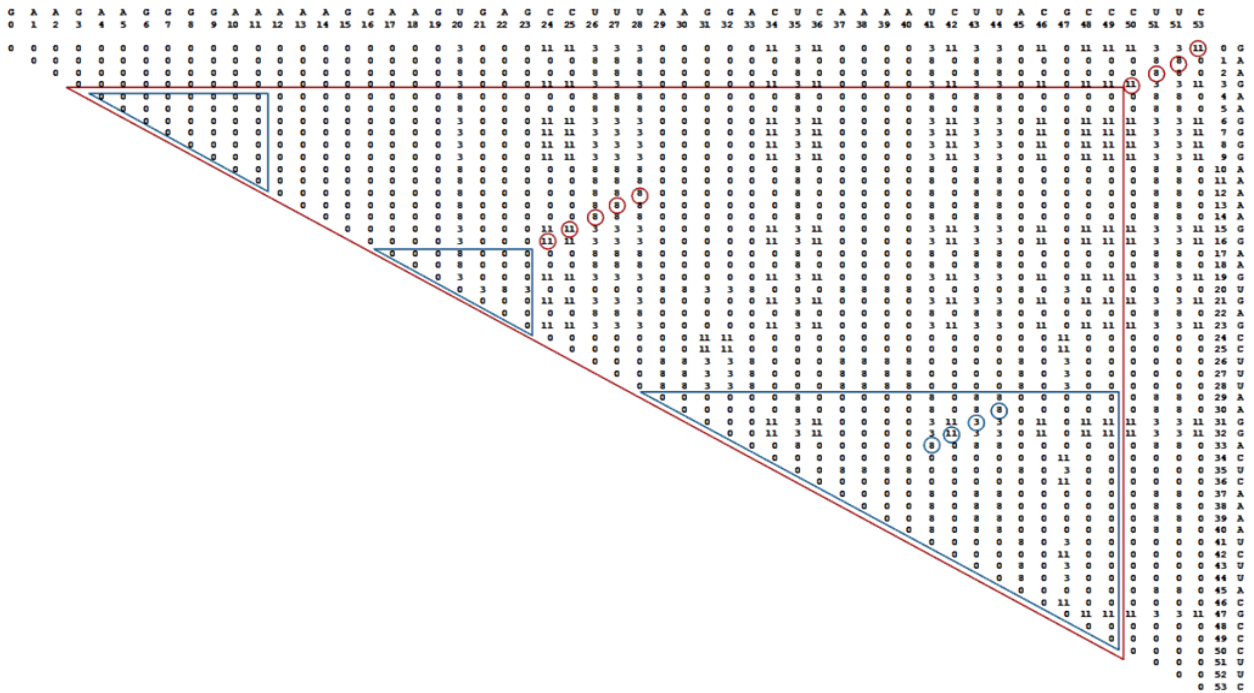
Example 6 ： Structure prediction of the proposed algorithm.

Input sequence ： GAAGAAGGGGAAAAAGGAAGUGAGCCUUUAAGGACUCAAAAUCUUACGCCCUUC

Section search ： stem_section - [[0, 53, 4],[12,28,5],[30,44,4]]

Structure ： ((((.........(((((.......)))))).((((.......))))).....))))

Weight matrix ：

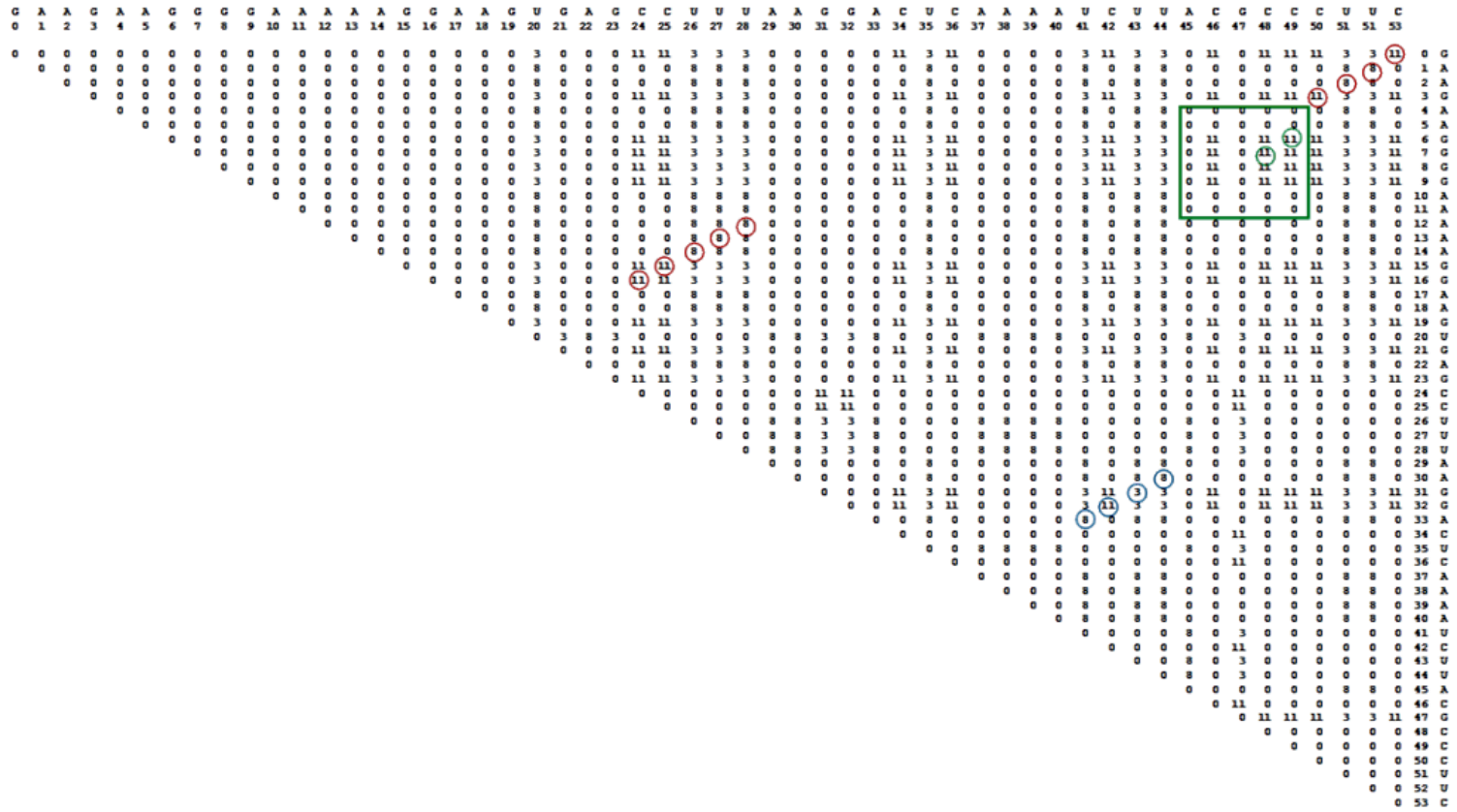


Cross search : `stem_cross - [[6, 49, 2]]`

Valid regions : `[[4, 11],[45,49]]`

Structure : `(((((..((....(((((.......)))))).((((.......)))))...))))))`

Weight matrix :



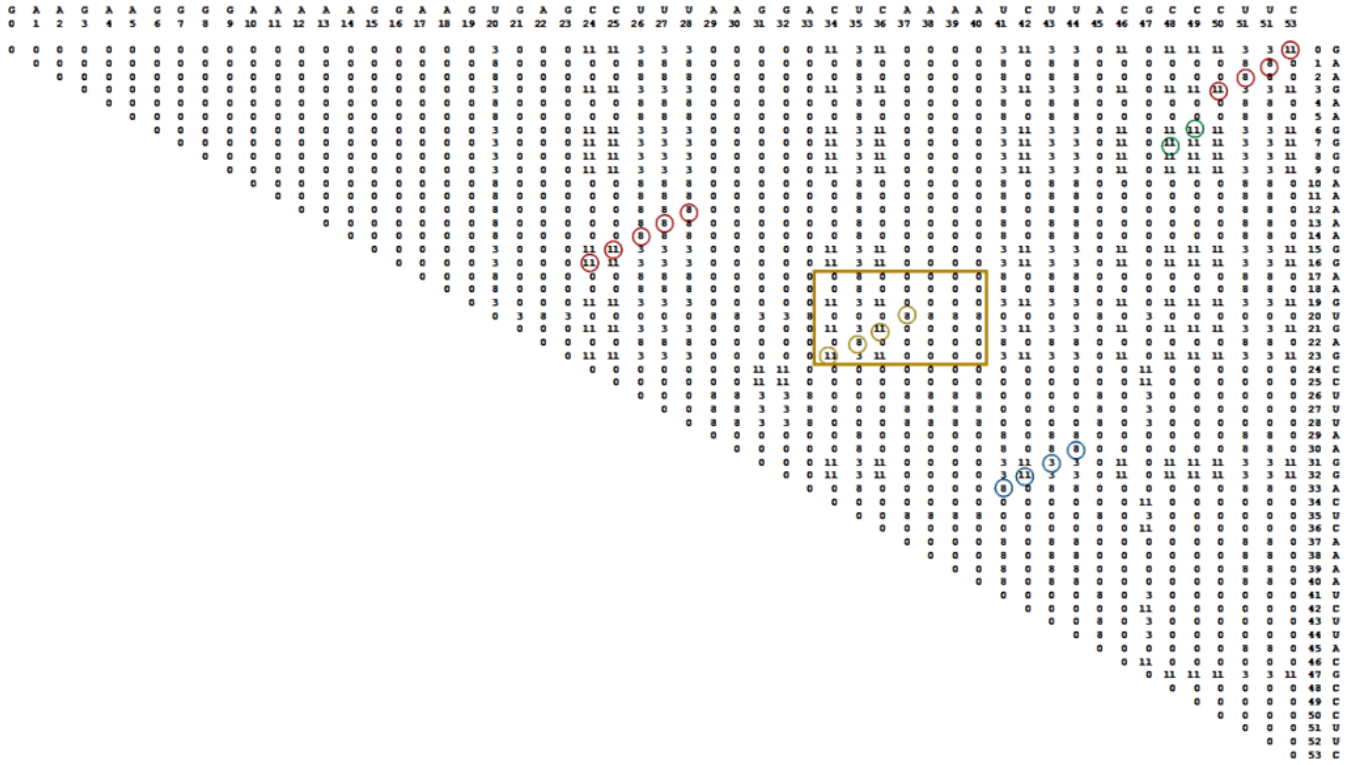
Knot search : `stem_knot - [[20, 37, 4]]`

Valid regions : `loop - [[17, 23],[34,40]],`

`unpair - [[8,11],[17, 23],[34,40],[45,47]]`

Structure : `((((..((....((((((...[[[[))))))..((((]]]]...))))...))))))`

Weight matrix :



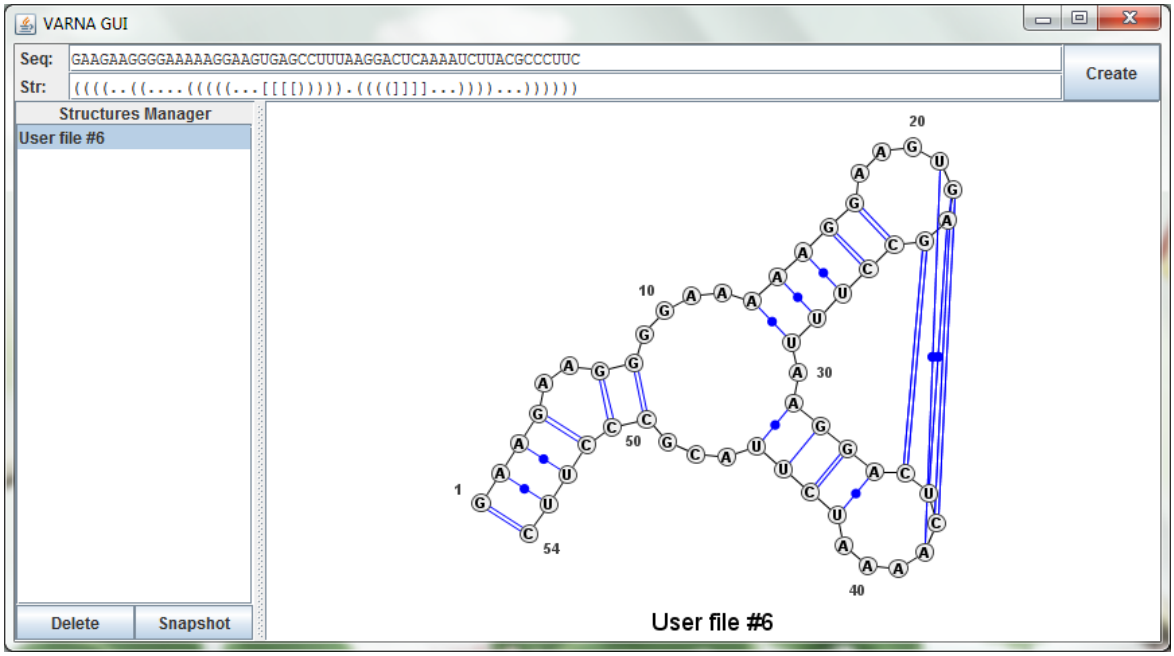


Figure 3.11: RNA structure predicted in Example 6.

CHAPTER 4

ANALYSES

This chapter discusses the performance of stem weight maximization algorithm presented in Chapter 3. RNA structures are obtained from three databases (FRABASE, RNA STRAND and CompaRNA). Performance of the proposed algorithm is evaluated by calculating the specificity and sensitivity values. Then, execution time of the proposed algorithm is recorded as well while predicting structures of RNA STRAND and CompaRNA.

4.1 FRABASE

This section will discuss about the performance of the proposed algorithm by testing on 110 sequences downloaded from FRABASE (Popenda et al. 2010). The length of these sequences are range between 12 – 76 nt. Structures of these sequences are obtained through experimental method like NMR (97 structures), X-ray diffraction (12 structures) and electron microscopy (1 structure).

4.1.1 General performance

Table 4.1 shows the total of specificity (SP) and sensitivity (SN) obtained by the proposed algorithm. The table listed the total of SP and SN values fall in each category (Refer to Appendix B for the complete result.). Since all value obtained are higher than 70%, the category defined in the table starts with the range 70-79%. The SP and SN values in percentage form are calculated using the formulae shown below:

$$SP = \frac{TP}{TP + FP} \times 100 \qquad SN = \frac{TP}{TP + FN} \times 100$$

TP is the amount of correctly predicted base pairs; FP is the amount of incorrectly predicted base pairs; FN is the amount of known base pairs which have not been correctly predicted. SP indicates the proportion of known base pairs that have been correctly predicted. SN value shows the ability of algorithm in identifying known base pairs. Higher SP and SN values indicate better structure prediction by having higher similarity between database structure and predicted structure. The structure determined through electron microscopy (2J28) is not shown in Table 4.1. The SP and SN values for 2J28 are 100 for both.

Table 4.1: Total of SP and SN in each category (FRABASE).

Category (%)	NMR				X-ray diffraction				Total	
	SP	%	SN	%	SP	%	SN	%	SP	SN
70 - 79	5	5.2	2	2.1	2	16.7	0	0	7	2
80 - 89	9	9.3	2	2.1	8	66.7	1	8.4	17	3
90 - 99	8	8.2	8	8.2	1	8.3	7	58.3	9	15
100	75	77.3	85	87.6	1	8.3	4	33.3	76	89
Total	97				12				109	

Majority of sequences obtained 100% for both SP and SN values. For SP value, there are 76 sequences achieved 100%. This is about 70% from the total sequences. Meanwhile, there are 89 sequences which achieved 100% for SN value (82%). Among these sequences, a total of 73 are perfect prediction (achieved 100% for both SP and SN). This means that structure predicted from the proposed algorithm are exactly the same with structure downloaded from FRABASE. Generally, total sequences with high (≥80%) SP value is 102 (94%) and 107 (98%) for SN value.

From Table 4.1, there are 97 structures determined through NMR method and 12 structures determined through X-ray diffraction method. From the table, it shows that the proposed algorithm performed better for structures determine through NMR method while compared to those structures determine through X-ray diffraction method. Majority of SP and SN values are fall in the category of 100% for NMR structure prediction.

Among the 110 sequences downloaded, there are 15 sequences with pseudoknot structures in which 7 are determined by NMR method while those remaining are determined by X-ray diffraction method. Table 4.2 shows the total of SP and SN values in each category. In general, performance of the proposed algorithm in pseudoknot prediction is quite good because majority of SP and SN values obtained are high (≥80%).

Table 4.2: Total of SP and SN in each category for pseudoknot structures (FRABASE).

Category (%)	NMR		X-ray diffraction		Total	
	SP	SN	SP	SN	SP	SN
70 – 79	2	0	1	0	3	0
80 – 89	1	1	7	1	8	2
90 – 99	1	1	0	4	1	5
100	3	5	0	3	3	8
Total	7		8		15	

4.1.2 Performance of the proposed algorithm compared to other algorithms

Results obtained by the proposed algorithm are compared with four pseudoknot prediction algorithms, HotKnots, pknotsRG, DotKnot and CyloFold. HotKnots is a heuristics algorithm based on the idea of iteratively forming stable stems. It explores many alternative secondary structures and selects the one with minimum free energy value (Ren et al. 2005). pknotsRG is a pseudoknot prediction algorithm which predicts the minimum free energy RNA structure based on Turner energy rules (Reeder, Steffen & Giegerich 2007, Mathews et al. 1999). DotKnot is a pseudoknot detection algorithm (Sperschneider, Datta & Wise 2011). CyloFold is simulating a folding process in the coarse-grained manner by selecting helices based on established energy rules (Bindewald, Kluth & Shapiro 2010).

MCC (Matthew's Correlation Coefficient) is calculated in order to know the performance of various algorithms. MCC is defined as below:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TN + FN)(FP + TN)(TP + FN)}}$$

If one of the four summation inside the square root is equal to zero, then denominator is arbitrarily set to one. MCC value ranges from -1 to +1. Number which closer to +1 means better prediction and +1 indicates perfect prediction.

Table 4.3 summarizes the overall performance of the proposed algorithm and three other algorithms while Table 4.4 shows the total of SP and SN in each category for all of them. Generally, performance of the proposed algorithm is good although its MCC score is slightly lower than HotKnots. The minimum value for SP is 72.22% and 71.43% for SN (Refer to Appendix B-1 for the minimum SP and SN values of the other algorithms.). The minimum SP value achieved is the highest value while compared to the other algorithms.

Table 4.3: Performance of the proposed algorithm, Hotknots, pknotsRG, DotKnot and CyloFold (FRABASE).

	Average value		
	SP	SN	MCC
Proposed algorithm	95.60	98.18	0.9122
HotKnots	96.86	98.70	0.9396
pknotsRG	95.16	97.02	0.8994
DotKnot	95.59	95.79	0.8884
CyloFold	95.79	93.66	0.8594

Table 4.4: Total of SP and SN in each category for the proposed algorithm, Hotknots, pknotsRG, DotKnot and CyloFold (FRABASE).

Category (%)	Proposed algorithm		HotKnots		pknotsRG		DotKnot		CyloFold	
	SP	SN	SP	SN	SP	SN	SP	SN	SP	SN
40-49	0	0	0	0	0	0	0	0	1	0
50-59	0	0	0	0	3	3	0	0	0	3
60-69	0	0	1	0	1	1	7	1	1	5
70-79	7	2	3	1	5	2	2	10	4	7
80-89	17	3	8	5	9	5	10	10	10	11
90-99	9	15	14	7	14	8	11	5	16	10
100	77	90	85	97	78	91	80	84	78	74
Total	110									

4.2 RNA STRAND

RNA STRAND is a database which collects known RNA secondary structures from various databases (Andronescu et al. 2008). 30 RNA structures with pseudoknot are downloaded from this database. The source of these sequences is Ribonuclease P Database (Brown 1999) in which structures are obtained by comparative sequence analysis method. The length for these structures is range between 229 - 457 nt.

4.2.1 General performance

Figure 4.1 shows the performance of the proposed algorithm in predicting RNA STRAND structures. It shows the difference between SP and SN values of the proposed algorithm in predicting 30 structures downloaded from the database. Majority of the difference between these two values are always negative. This

shows that SP values are higher than SN values for most of the cases. This indicates that the proposed algorithm obtained higher value of FN while compared to FP for this dataset.

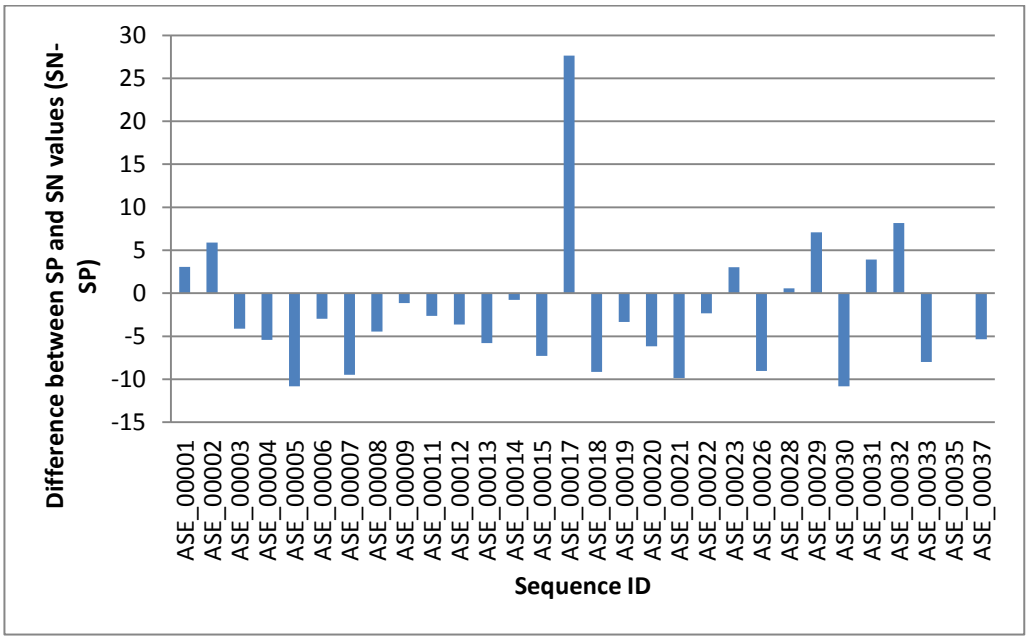


Figure 4.1: Difference of SN and SP values (SN – SP) for 30 structures downloaded from RNA STRAND.

The highest peak in Figure 4.1 is 27.63. This value corresponds to the difference between SN and SP of ASE_00017. The SP value is much lower than SN in this case due to high FP value obtained. A large segment (75nt) of ASE_00017 which contained no base pairs is the main cause of the high FP value obtained.

4.2.2 Performance of the proposed algorithm compared to pknotsRG and DotKnot

The average values of SP and SN of the proposed algorithm in predicting RNA STRAND structures are 74.62% and 72.51% respectively. Table 4.5 shows that most of the SP and SN values for the proposed algorithm and DotKnot fall in the category 70 – 79% while 80 – 89% for pknotsRG. The average SP and SN values for pknotsRG (SP - 81.33%, SN - 80.03%) and DotKnot (SP - 79.28%, SN - 76.15%) are higher than the proposed algorithm for this dataset.

Table 4.5: Total of SP and SN in each category for the proposed algorithm, pknotsRG and DotKnot (RNA STRAND).

Category (%)	Proposed algorithm		pknotsRG		DotKnot	
	SP	SN	SP	SN	SP	SN
40 – 49	1	0	1	0	1	0
50 – 59	0	0	0	0	0	1
60 – 69	3	12	0	1	0	1
70 – 79	19	16	7	10	14	20
80 – 89	7	2	20	19	15	8
90 – 99	0	0	2	0	0	0

Although performance of the proposed algorithm is not as good as pknotsRG and DotKnot, but the execution time of the proposed algorithm is much lower than both of them. This is illustrated in Figure 4.2. The average execution time of the proposed algorithm is 1.80s whereas pknotsRG and Dotknot required 16.19s and 16.23s respectively on the whole.

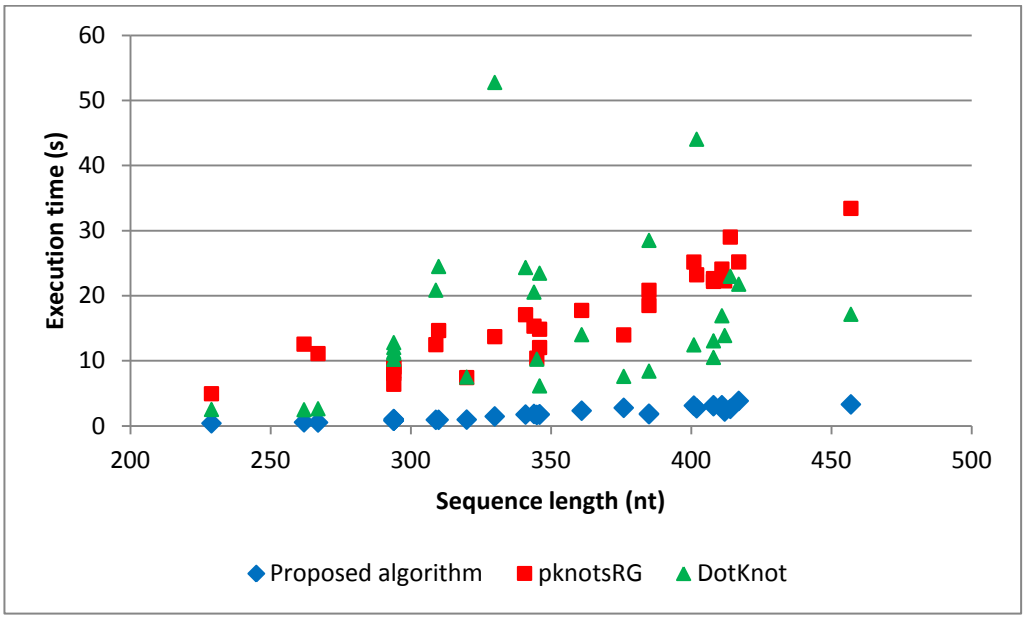


Figure 4.2: Execution time of the proposed algorithm, pknotsRG and DotKnot (RNA STRAND).

4.3 CompaRNA

CompaRNA is a server which performs benchmarking of various RNA structure prediction algorithm (Puton et al. n.d.). The complete pseudoknot dataset of CompaRNA has been downloaded. This dataset contained 92 RNA structures with pseudoknot. The length of these structures ranges between 27 - 3174 nt.

4.3.1 General performance

Figure 4.3 shows the box plot for SP and SN values achieved by the proposed algorithm. From the box plot, it can be observed that there are some outliers for SP while no outliers are found for SN. The minimum value achieved by the proposed algorithm is 19.57% for SP and 48.51% for SN. For maximum value, both SP and SN values achieved 100%.

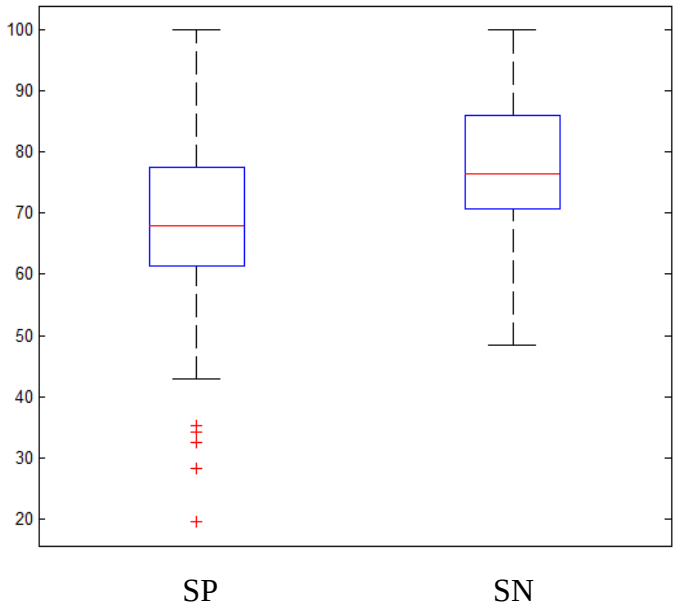


Figure 4.3: Box plot for SP and SN values achieved by the proposed algorithm (CompaRNA).

For this dataset, length of sequences can be classified into 3 categories, that is short ($25 < length < 400$), middle ($1400 < length < 1800$) and long ($2700 < length < 3200$). The average performance of the proposed algorithm for each of these categories is summarized in Table 4.6. Table 4.6 shows that

performance of the proposed algorithm decrease significantly from short to middle category. This shows that the proposed algorithm perform better in the structure prediction of short sequences. For longer sequences which range between 1400-3200nt, the performance of the proposed algorithm has not much difference. This indicates that performance of the proposed algorithm is quite stable for structure prediction of longer sequences.

Table 4.6: Performance of the proposed algorithm in predicting RNA sequences with various length.

Category	Length (nt)	Average value	
		SP	SN
Short	25 - 400	73.3	83. 2
Middle	1400 - 1800	61.0	71.3
Long	2700 - 3200	62.2	71.4

Additionally, performance of the proposed algorithm in terms of execution time required for structure prediction is recorded as well. Generally, execution time of the proposed algorithm increased exponentially with length of sequences. Figure 4.4 shows the execution time of the proposed algorithm while predicting sequences obtained from STRAND. The figure shows that execution time increased significantly for every increase of sequence length by 1000nt.

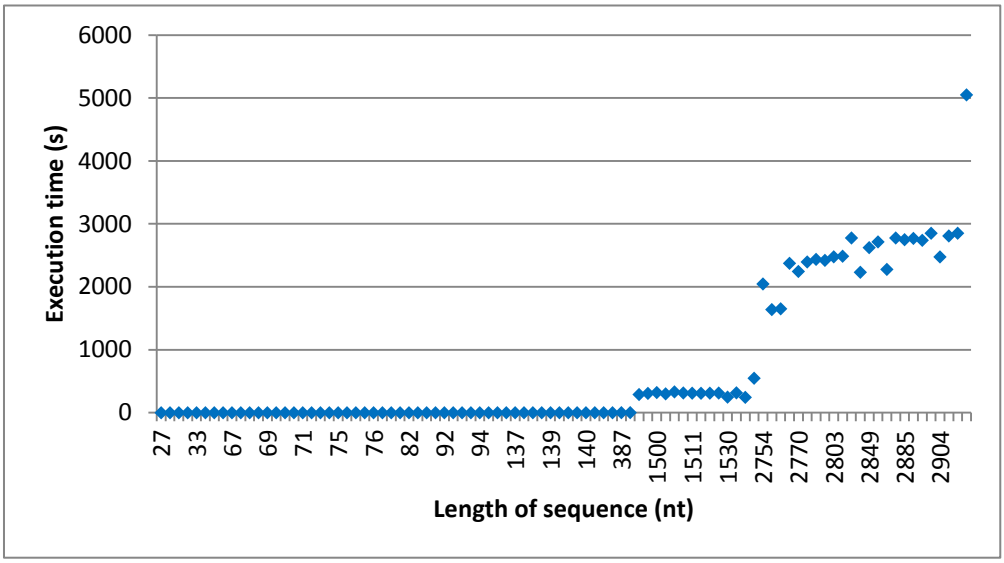


Figure 4.4: Execution time required for the structure prediction of CompaRNA structures.

4.3.2 Performance of the proposed algorithm compared to pknotsRG

Table 4.7 summarized the performance of the proposed algorithm and pknotsRG. Although total SP values which fall in the category of 100% is higher for pknotsRG, but performance of the proposed algorithm is better than pknotsRG on the whole. The average SP and SN values achieved by the proposed algorithm are 68.54% and 78.31% respectively. For pknotsRG, the average SP and SN values achieved are 59.63% and 61.04% respectively.

Table 4.7: Performance of the proposed algorithm and pknotsRG (CompaRNA).

Category (%)	Proposed algorithm		pknotsRG	
	SP	SN	SP	SN
10 – 19	1	0	1	1
20 – 29	1	0	6	3
30 – 39	3	0	7	6
40 – 49	5	1	28	16
50 – 59	10	1	14	28
60 – 69	31	16	6	7
70 – 79	22	37	3	13
80 – 89	10	24	12	8
90 – 99	2	9	3	8
100	7	4	12	2

4.4 Comparison among three databases

Here, discussion will focus in the comparison of performance of the proposed algorithm while predicting structures downloaded from three different databases. Since the length of sequences obtained from CompaRNA is ranged between 27 - 3174nt while the maximum length of sequences obtained from FRABASE and RNA STRAND are 76nt and 457nt respectively, the range of sequence length included in this section would be 12 - 457nt.

From Figure 4.5 and Figure 4.6, performance of the proposed algorithm is the best while predicting FRABASE structures. For SP values, performance of the proposed algorithm in predicting CompaRNA structures is not consistent due to the bigger fluctuation of SP values achieved. Conversely, majority of SP values achieved for predicting RNA STRAND structures are more consistent by ranging between 68.5 - 87.5%.

Generally, SN values achieved for structure prediction are more consistent while compared to SP values. SN values achieved are ranged within 60-100 with two exceptional cases which are 48.51% and 53.7%. On the whole, performance of the proposed algorithm in ascending order is as follow: CompaRNA (SP-68.54%, SN-78.31%), RNA STRAND (SP-74.62%, SN-72.51%) and FRABASE (SP-95.60%, SN-98.18%).

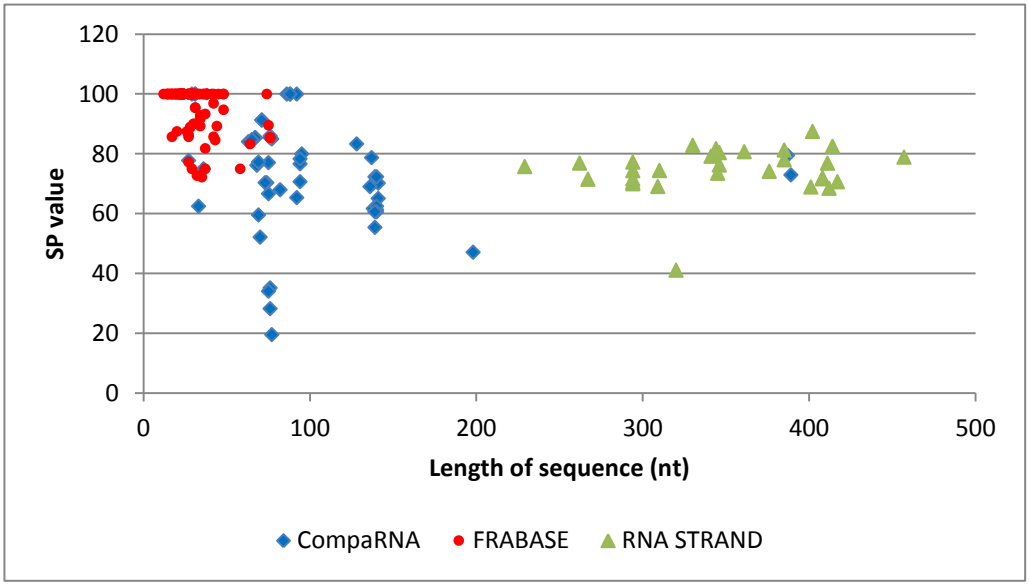


Figure 4.5: SP values achieved for predicting structures obtained from RNA STRAND, CompaRNA and FRABASE.

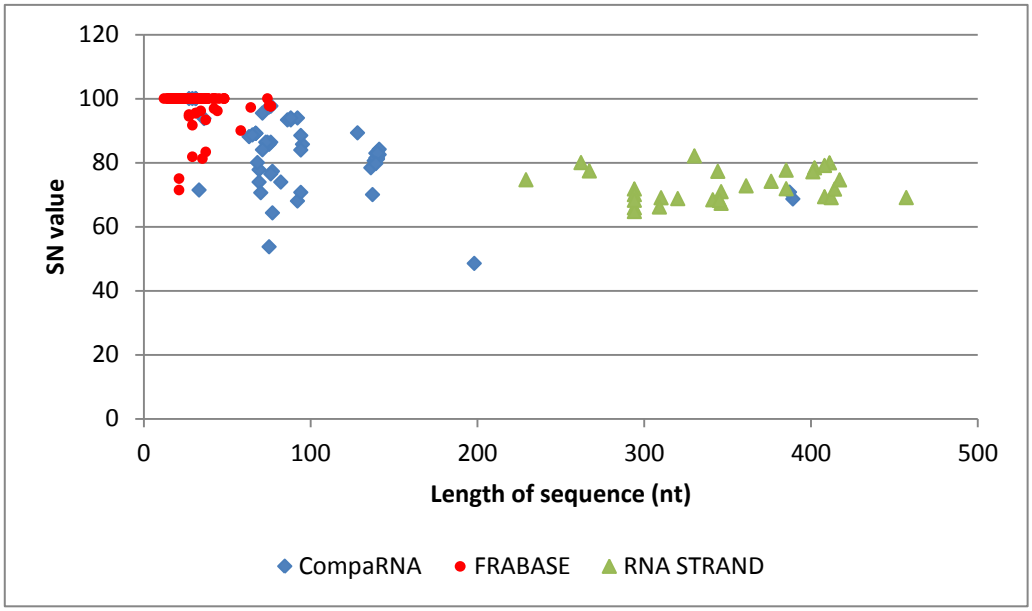


Figure 4.6: SN values achieved for predicting structures obtained from RNA STRAND, CompaRNA and FRABASE.

CHAPTER 5

CONCLUSION AND FUTURE WORK

During the development of the pseudoknot prediction algorithm, various problems have been encountered. In this chapter, some unsolved problems are outlined and discussed. The main problem of the proposed algorithm is having extra base pairs. These extra base pairs generally occurred at the starting position of stem and pseudoknot structure. Besides, contributions and future works are presented as well.

5.1 Superfluous base pair

Basically, there are three problems found in structure prediction by the proposed algorithm. First, structure predicted will contain extra base pairs while compared to FRABASE structure. These extra base pairs are in fact valid pairing but they are found to be not paired in the database. Some examples of structure which contained extra base pairs are shown in Figure 5.1.

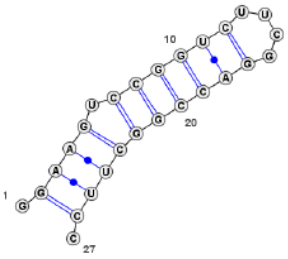
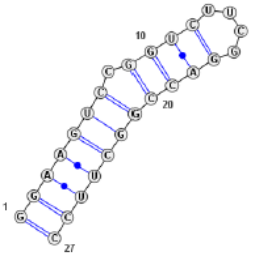
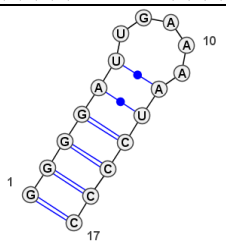
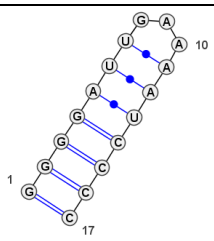
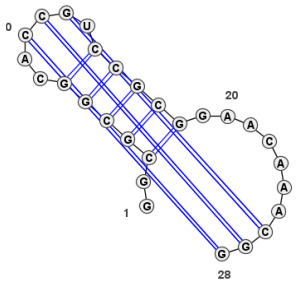
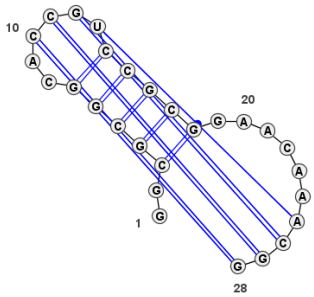
FRABASE structure	Predicted structure
1F7F 	
. ((((. (((((. . . . .))))))))) .	(((((((. ((((. . . .)))))))))))
1J4Y 	
((((((. . . . .))))))	(((((((. . .)))))))
1L2X 	
. . (((((. . [[[.))))) . . . .]]]	. . (((((. . [[[[ ))))) . . . . . .]]]]

Figure 5.1: Extra base pairing found in structure predicted by the proposed algorithm.

Besides, pseudoknots found in FRABASE always form by a base pairing only. In the proposed algorithm, “knot search” does not consider stem with length smaller than 3nt. As a result, pseudoknots predicted by the proposed algorithm always contained additional base pairs and it might not form within the same regions as in FRABASE structure. This is shown in Figure 5.2.

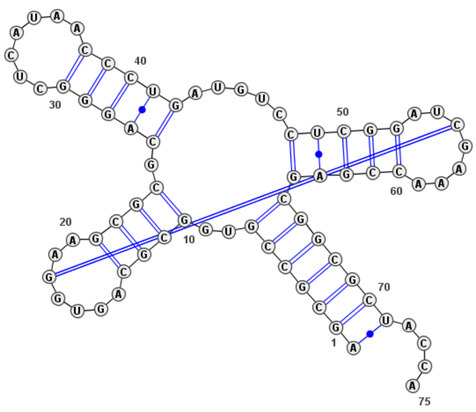
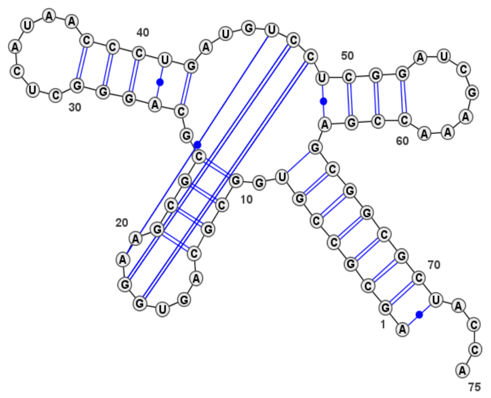
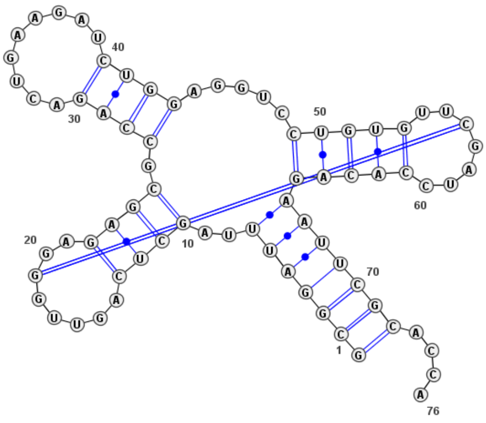
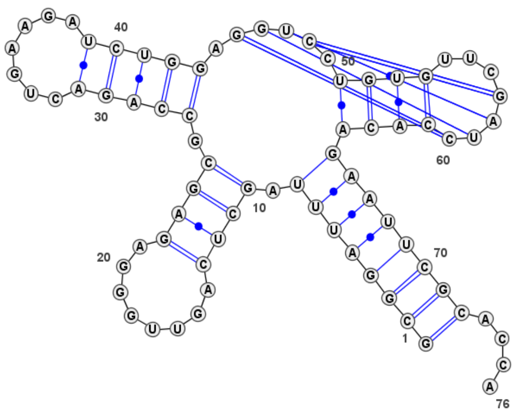
FRABASE structure	Predicted structure
1YFG  (((((((..((((....[..)))).(((((.......))))).....(((((..]... .))))))))))))....	 ((((((((.((((...[[[.)))).(((((.......)))))...]]]((((...... .))))))))))))....
4TNA  (((((((..((((.....[..)))).((((.........)))).....(((((..].. ..))))))))))))....	 ((((((((.((((........)))).(((((.......))))).[[[[.((((...]]]]))))))))))))....

Figure 5.2: Extra base pairing found in pseudoknot predicted by the proposed algorithm.

Furthermore, the proposed algorithm favors stem with maximum stem weight value. Thus, it might produce structure which is different from FRABASE structure. This case is observable while FRABASE structure is having lots of base pairs with stem length less than 3nt. Figure 5.3 shows that RNA structure might contain a few stems which is short in length whereas the proposed algorithm does not prefer these stems. Therefore, structure predicted is diverse from FRABASE structure. Table 5.1 shows the total occurrence of problems discussed in this section.

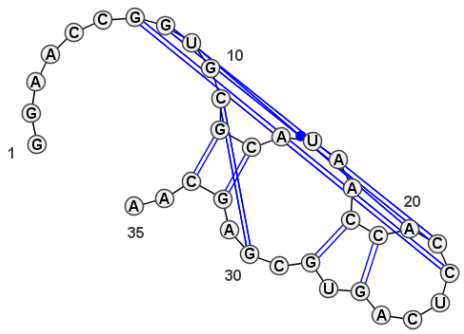
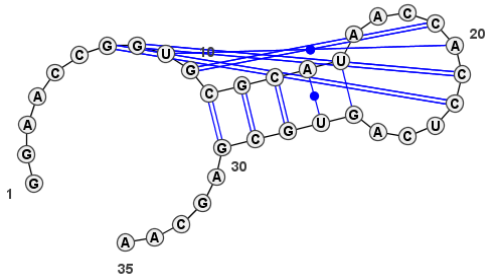
FRABASE structure	Predicted structure
1DDY	
	
......(((.[{{....[[)))...].].] .}}..	......[[[[(((((...]]]]...))))) .....

Figure 5.3: Problems of stem weight maximization implemented by the proposed algorithm.

Table 5.1: Total occurrence of various problems.

Problems	Total
Extra base pairing	20
Pseudoknot	7
Stem selection	6
Short stem	2

5.2 Contributions

This research has developed an RNA pseudoknot prediction algorithm by extending the DWM algorithm (Liu et al. 2006). Stem selection criteria of the proposed algorithm is different from DWM algorithm. The proposed algorithm prefers stem with maximum stem weight value. In addition, stems which does not fulfilled those criteria discussed in Section 3.2.1 are filtered out before the calculation of stem weight.

Moreover, the proposed algorithm can predict two types of pseudoknot structures that is H-type pseudoknot and kissing hairpins. Currently, majority of pseudoknot prediction algorithms developed are target on the common type of pseudoknot that is H-type pseudoknot whereas the proposed algorithm can predict an additional type of pseudoknot, kissing hairpins.

In Chapter 4, the proposed algorithm is shown to produce reasonably accurate structures. Comparison of structures predicted by the proposed algorithm with FRABASE structures produced high SP and SN values. This shows that structures predicted by the proposed algorithm have high similarity with experimentally determined RNA structures.

In addition, the proposed algorithm can handle long sequences and perform structure prediction in a short time frame. Hitherto, the maximum length of RNA sequences that has been tested on the proposed algorithm is 3174nt. For sequences up to 1500nt, it requires about 5 minutes for structure prediction. From Figure 4.2, it shows that the execution time of the proposed algorithm is much reduced while compared to pknotsRG and DotKnot.

In this research, performance of the proposed algorithm is evaluated by structure comparison between predicted structure and database structure. FRABASE is one of the databases used for evaluating the performance of the proposed algorithm. This database contains only RNA structures determined through experimental like NMR, X-ray diffraction and electron microscopy. Therefore, comparison with structures obtained from FRABASE provides a reliable indication on how well is the performance of the proposed algorithm.

5.3 Future work

Since the proposed algorithm always contain extra base pairing in pseudoknot, therefore implementation of post processing for removing these extra base pairs might be considered as a way to overcome this problem.

Besides, the proposed algorithm can be modified for the prediction of triple helix interaction which is another type of pseudoknot as well. Triple helix interaction is formed when base pairing occurs in between the loop regions of H-type pseudoknot. Figure below shows the structure of triple helix interaction.

Figure 5.4: Triple helix interaction.

REFERENCES

Akutsu, T. 2000, 'Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots', *Discrete Applied Mathematics*, vol. 104, pp. 45-62.

Andronescu, M., Bereg, V., Hoos, H.H. and Condon, A. 2008, 'RNA STRAND: The RNA Secondary Structure And Statistical Analysis Database', *BMC Bioinformatics*, vol. 9, no. 1, pp. 340.

Bindewald, E., Kluth, T. and Shapiro, B.A. 2010, 'CyloFold: secondary structure prediction including pseudoknots', *Nucleic Acids Research*, vol. 38, pp. w368-w372.

Brown, J.W. 1999, 'The Ribonuclease P Database', *Nucl. Acids Res.*, vol. 27, pp. 314.

Chen, Q. and Chen, Y.P. 2009, 'Discovery of Structural and Functional Features in RNA Pseudoknots', *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 7, pp. 974-984.

Doudna, J.A. and Cech, T.R. 2002, 'The chemical repertoire of natural ribozymes', *Nature*, vol. 418, no. 6894, pp. 222-228.

Draper D.E., Gluick T.C. and Schlax P.J. 1998, 'Pseudoknots, RNA folding and translational regulation in RNA structure and function', *Cold Spring Harbor Laboratory Press*, pp. 415-436.

Eddy, S.R. 2004, 'How do RNA folding algorithms work?', *Nature Biotechnology*, vol. 22, no. 11, pp. 1457-1458.

Eddy, S.R. 2005, 'Computational Analysis of RNAs', *BMC Bioinformatics*, vol. 6, pp. 63.

Giedroc D.P., Theimer C.A. and Nixon P.L. 2000, 'Structure, stability and function of RNA pseudoknots involved in simulating ribosomal frameshifting', *J. Mol. Biol.*, vol. 298, pp. 167-185.

Hofacker, I.L., Fontana, W., Stadler, P.F., Bonhoeffer, S., Tacker, M. and Schuster, P. 1994, 'Fast folding and comparison of RNA secondary structures', *Monatsh Chem*, vol. 125, pp. 167-188.

Krane, D.E. and Raymer, M.L. 2003, *Fundamental concepts of bioinformatics*. Pearson Education, Benjamin Cummings, 1301 Sansome Street, San Francisco.

Liu, Haijun, Xu, Dong, Shao, Jianling, and Wang, Yifei 2006, 'An RNA folding algorithm including pseudoknots based on dynamic weighted matching', *Computational Biology and Chemistry*, vol. 30, pp. 72-76.

Liu, Q., Ye, X., and Zhang, Y. 2006, 'A Hopfield Neural Network Based Algorithm for RNA Secondary Structure Prediction', *IMSCCS*, vol. 1, pp. 10-16.

Lyngso, R.B. and Pederson, C.N. 2000, 'RNA pseudoknot prediction in energy-based models', *J. Comput. Biol.*, vol. 7, pp. 409-27.

Mathews, D.H. and Turner, D.H. 2006, 'Prediction of RNA secondary structure by free energy minimization', *Curr Opin Struct Biol*, vol. 16, pp. 270-8.

Mathews, D.H., Sabina, J., Zuker, M. and Turner, D.H. 1999, 'Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure', *J. Mol. Biol.*, vol. 288, pp. 911-940.

Meister, G. and Tuschi, T. 2004, 'Mechanisms of gene silencing by double-stranded RNA', *Nature*, vol. 431, pp. 343-349.

Meyer, I.M. and Miklós, I. 2007, 'SimulFold: Simultaneously Inferring RNA Structures Including Pseudoknots, Alignments, and Trees Using a Bayesian MCMC Framework', *PLoS Computational Biology*, vol. 3, no. 8, pp. 1441-1454.

Mihalusova, M., Wu, J.Y. and Zhuang, X. 2011, 'Functional importance of telomerase pseudoknot revealed by single-molecule analysis', *PNAS Early Edition*.

Mizoguchi, N., Kato, Y. and Seki, H. 2011, 'A Grammar-Based Approach to RNA Pseudoknotted Structure Prediction for Aligned Sequences', *IEEE International Conference on Computational Advances in Bio and medical Sciences*, pp. 135-140.

Nussinov, R., Peiczenik, G., Griggs, J.R. and Kleitman, D.J. 1978, 'Algorithms for Loop Matchings', *SIAM J. Appl. Math*, vol. 35, pp. 68-81.

Popenda, M., Szachniuk, M., Blazewicz, M., Wasik, S., Burke, E.K., Blazewicz, J. and Adamiak, R.W. 2010, 'RNA FRABASE 2.0: an advanced web-accessible database with the capacity to search the three-dimensional fragments within RNA structures', *BMC Bioinformatics*, vol. 11, pp. 231.

Puton T., Rother K., Kozłowski Ł., Tkalińska E. and Bujnicki J.M. n.d., *CompaRNA: a server for continuous benchmarking of automated methods for RNA structure predictions*. Available from: <http://iimcb.genesilico.pl/comparna/ >.

Reeder, J., Höchsmann, M., Rehmsmeier, M., Voss, B. and Giegerich R. 2006, 'Beyond Mfold: Recent advances in RNA bioinformatics', *Journal of Biotechnology*, vol. 124, pp. 41-55.

Reeder, J., Steffen, P. and Giegerich, R. 2007, 'pknotsRG: RNA pseudoknot folding including near-optimal structures and sliding windows', *Nucleic Acids Research*, vol. 35, pp. w320-w324.

Ren, J., Rastegari, B., Condon, A. and Hoos, H.H. 2005, 'HotKnots: Heuristic prediction of RNA secondary structures including pseudoknots', *RNA*, vol. 11, pp. 1494-1504.

Rivas, E. and Eddy, S.R. 1999, 'A dynamic programming algorithm for RNA structure prediction including pseudoknot', *J. Mol. Biol.*, vol. 285, pp. 2053-2068.

Roy, A., Panigrahi, S., Bhattacharyya, M. and Bhattacharyya, D., 2008, 'Structure, Stability and Dynamics of Canonical and Noncanonical Base Pairs: Quantum Chemical Studies', *J. Phys. Chem. B.*, vol. 112, pp. 3786-3796.

Sperschneider, J., Datta, A. and Wise, M.J. 2011, 'Heuristic RNA pseudoknot prediction including intramolecular kissing hairpins', *RNA*, vol. 17, no. 1, pp. 27-38.

Staple, D.W. and Butcher, S.E. 2005, 'Pseudoknots: RNA Structures with Diverse Functions', *PLoS Biology*, vol. 3, no. 6, pp. e213.

Storz, G. 2002, 'An expanding universe of noncoding RNAs', *Science*, vol. 296, pp.1260-1263.

Tabaska, J.E., Cary, R.B., Gabow, H.N. and Stormo, G.D. 1998, 'An RNA folding method capable of identifying pseudoknots and base triples', *Bioinformatics*, vol. 14, no. 8, pp. 691-699.

Tahi, F., Engelen, S. and Rgnier, M. 2003, 'A fast algorithm for RNA secondary structure prediction including pseudoknots', *Proceedings of the Third IEEE Symposium on Bioinformatics and Bioengineering (BIBE'03)*, pp. 0-7695-1907-5/03.

Uemera, Y., Hasegawa, A., Kobayashi, S. and Yokomori, T. 1999, 'Tree adjoining grammars for RNA structure prediction', *Theoret. Comput. Sci.*, vol. 210, pp. 277-303.

Vaish, N.K., Kore, A.R. and Eckstein, F. 1998, 'Survey and summary Recent developments in the hammer-head ribozyme field', *Nucleic Acids Research*, vol. 26, no. 23, pp. 5237-5242.

Valencia-Sanchez, M.A. Liu, J, Hannon, G.J. and Parker, R. 2006, 'Control of translation and mRNA degradation by miRNAs and siRNAs', *Genes Dev.*, vol. 20, pp. 515-524.

Van Batenburg, F., Gultyaev, A. and Pleij, C. 1995, 'An APL-programmed genetic algorithm for the prediction of RNA secondary structure', *J. Theor. Biol.*, vol. 174, pp. 269-280.

Wilm, A., Higgins, D.G. and Notredame, C. 2008, 'R-Coffee: a method for multiple alignment of non-coding RNA', *Nucleic Acids Res.*, vol. 36, no. 9, pp. e52.

Witwer, C., Hofacker, I.L. and Stadler, P.F. 2004, 'Prediction of Consensus RNA Secondary Structures Including Pseudoknots', *IEEE Transactions on Computational Biology and Bioinformatics*, vol. 1, no. 2, pp. 66-76.

Zuker, M. and Stiegler, P. 1981, 'Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information', *Nucleic Acids Res.*, vol. 9, pp. 133-148.

Zuker, M. 1989, 'On Finding All Suboptimal Folding of an RNA Molecule', *Science*, vol. 244, pp. 48-52.

APPENDIX A

STEM WEIGHT MAXIMIZATION ALGORITHM

```
import sys
from math import sqrt
from numpy import *

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
#                          FUNCTIONS                                      #
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

def fill_wMatrix ():
    for i in range (N):
        for j in range (N-i):
            wMatrix[i][i+j] = pairing[seq[i]+seq[i+j]]
    return


def save ():
    outputFile.write(seq_ID)
    predicted = ''
    print >> outputFile, "\tLength = ", len(seq)
    for i in range (len(pre_str)):
        predicted += pre_str[i]
    print >> outputFile, "Pre : ",predicted
    print >> outputFile, "\n"
    print >> outputFile, "-"*73
    return


def bracket ( stem, bracket_type ):
    for i in range(len(stem)):
        for j in range(stem[i][2]):
            if bracket_type == 1:
                pre_str[stem[i][0]+j] = '('
                pre_str[stem[i][1]-j] = ')'
            else:
                pre_str[stem[i][0]+j] = '['
                pre_str[stem[i][1]-j] = ']'
    return


def unpair_region ( loc ):
    len_UR = 1
    while ( loc+1 < N and pre_str[loc+1] == '.' ):
        len_UR += 1
        loc    += 1
    return len_UR


def searching_region ():
    loop = []; unpair = []; loc = []
    if pre_str[0] == '.':
        loc.append(0)
    for i in range(N):
        if i!=0:
            if ( pre_str[i] == '.' ):
                if pre_str[i-1] == '(' or pre_str[i-1] == ')' or pre_str[i-1] == '[' or
pre_str[i-1] == ']':
                    loc.append(i)
    for j in range(len(loc)):
        len_UR = unpair_region ( loc[j] )
        if len_UR >= 3:
            if pre_str[loc[j]-1] == '(' and pre_str[loc[j]+len_UR] == ')':
                if (loc[j]+len_UR-1) - (loc[j]-1) > 5:
                    loop.append( [loc[j], loc[j]+len_UR-1] )
            else:
                unpair.append( [loc[j], loc[j]+len_UR-1] )

    return loop, unpair
```

```
def pairing_CS ( search_space, row, col, min_len, stem, list_wt, reg ):
    count = 0
    if row[1]-row[0] >= col[1]-col[0]:
        SR = col[1]-col[0]
    else:
        SR = row[1]-row[0]
    if reg == 1:
        cond = row[1]-SR
    else:
        cond = col[0]+SR
    for k in range(len(search_space)):
        stem_len = 0; stem_wt = 0
        if reg == 1:
            if search_space[k][0] > cond:
                count += 1
        else:
            if search_space[k][1] < cond:
                count += 1
        for l in range(SR+1-count):
            if ( wMatrix[search_space[k][0]+l][search_space[k][1]-l] >= 3 ):
                stem_len += 1
                stem_wt  += wMatrix[search_space[k][0]+l][search_space[k][1]-l]
            else:
                if stem_len >= min_len:
                    stem.append(   [(search_space[k][0]+l)-stem_len,   (search_space[k][1]-
l)+stem_len, stem_len] )
                    list_wt.append( stem_wt )
                stem_len = 0; stem_wt = 0
        if (  (search_space[k][0]+l) == row[1] or (search_space[k][1]-l) == col[0] ):
            if stem_len >= min_len:
                stem.append(   [(search_space[k][0]+l)-(stem_len-1),   (search_space[k][1]-
l)+(stem_len-1), stem_len] )
                list_wt.append( stem_wt )
            stem_len = 0; stem_wt = 0
    return stem, list_wt


def potential_stem ( row, col, min_len, stem, list_wt ):
    stem_len = 0
    stem_wt  = 0
    for i in range(((col-row)/2)+2):
        if wMatrix[row+i][col-i] >= 3:
            stem_len += 1
            stem_wt  += wMatrix[row+i][col-i]
        else:
            if stem_len >= min_len:
                loop_size = ((col-i) - (row+i))+1
                if stem_len == min_len:
                    if loop_size >= 3:
                        stem.append([row+i-stem_len, col-i+stem_len, stem_len])
                        list_wt.append(stem_wt)
                else:
                    if loop_size >= 3:
                        stem.append([row+i-stem_len, col-i+stem_len, stem_len])
                        list_wt.append(stem_wt)
                    else:
                        if loop_size > 0:
                            stem.append([row+i-stem_len, col-i+stem_len, stem_len-1])
                            list_wt.append(stem_wt-wMatrix[row+i-1][col-i+1])
                        else:
                            if stem_len > 4:
                                stem.append([row+i-stem_len, col-i+stem_len, stem_len-2])
                                list_wt.append(stem_wt-wMatrix[row+i-1][col-i+1]-
wMatrix[row+i-2][col-i+2])
            stem_len = 0
            stem_wt  = 0
    return stem, list_wt


def diagonal_search ( LB, UB, min_len ):
    stem    = []
    list_wt = []
    stem, list_wt = potential_stem( LB, UB, min_len, stem, list_wt )
    for i in range(UB-LB-4):
        stem, list_wt = potential_stem( LB, UB-i-1, min_len, stem, list_wt )
        stem, list_wt = potential_stem( LB+i+1, UB, min_len, stem, list_wt )
    return stem, list_wt


def cross_pairing ( reg1, reg2, min_len ):
    stem  = []; list_wt  = []
    stem1 = []; list_wt1 = []
    stem2 = []; list_wt2 = []
    potential_stem = []; potential_len = []
    search_space1 = []; search_space2 = []
```

```
    if reg1[0] < reg2[0]:
        row = reg1
        col = reg2
    else:
        row = reg2
        col = reg1
    for i in range( row[1]-row[0]-1 ):
        search_space1.append([row[0]+i,col[1]])
    for j in range ( col[1]-col[0]-1 ):
        search_space2.append([row[0],col[1]-j])
    search_space2.remove(search_space2[0])
    stem1, list_wt1 = pairing_CS( search_space1, row, col, min_len, stem1, list_wt1, 1 )
    stem2, list_wt2 = pairing_CS( search_space2, row, col, min_len, stem2, list_wt2, 2 )
    stem            = stem1+stem2
    list_wt         = list_wt1+list_wt2
    if stem == []:
        return
    else:
        for k in range( len(list_wt) ):
            if list_wt[k] == max(list_wt):
                potential_stem.append(stem[k])
                potential_len.append(stem[k][2])
        selected_stem = potential_stem[potential_len.index(min(potential_len))]
        if ( min_len == 1 ):
            stem_cross.append(selected_stem)
        else:
            stem_knot.append(selected_stem)
    return selected_stem


def section_search ( LB, UB ):
    selected_stem = []; potential = []; stem_len = []
    if UB-LB < 6:
        return
    stem, list_wt = diagonal_search( LB, UB, 3 )
    if stem == []:
        return
    priority = 0
    for i in stem:
        if i[0] == 0 and i[1] == N-1:
            stem_section.append(i)
            selected_stem = i
            priority += 1
    if priority == 0:
        for j in range (len(list_wt)):
            if list_wt[j] == max(list_wt):
                potential.append(stem[j])
                stem_len.append(stem[j][2])
        if len(potential) == 1:
            stem_section.append(potential[0])
        else:
            stem_section.append(potential[stem_len.index(min(stem_len))])
        selected_stem = stem_section[len(stem_section)-1]
    if selected_stem != []:
        d, e, f = selected_stem
        section_search( LB, d-1 )
        section_search( d+f, e-f )
        section_search( e+1, UB )
    return


def cross_search ():
    loop, unpair = searching_region()
    count    = len(unpair)
    while len(unpair) > 1:
        reg1 = []; reg2 = []
        region1 = unpair[0]
        region2 = unpair[count-1]
        if region1 != region2:
            for i in stem_section:
                if region1[0] < i[1] and region1[0] > i[0]:
                    reg1.append(i)
                if region2[0] < i[1] and region2[0] > i[0]:
                    reg2.append(i)
            if reg1 == reg2:
                if (region1[0] == 0 or region1[0] == 1) and (region2[1] == N-2 or region2[1]
== N-1):
                    stem = cross_pairing( region1, region2, 3 )
                else:
                    stem = cross_pairing( region1, region2, 1 )
                if stem != None:
                    unpair.remove(region1)
                    unpair.remove(region2)
                    count = len(unpair)
                else:
                    count -= 1
            else:
                count -= 1
```

```
        else:
            unpair.remove(region1)
            count = len(unpair)
    return


def knot_search ():
    loop, unpair = searching_region()
    count1 = 0; count2 = 0
    for i in unpair:
        if (i[1]-i[0])+1 < 5:
            unpair.remove(i)
    if ( len(loop) > 0 and len(unpair) > 0 ):
        region1 = loop[0]
        region2 = unpair[0]
    else:
        return
    while ( len(loop) > 0 and len(unpair) > 0 ):
        stem    = cross_pairing( region1, region2, 3 )
        if stem != None:
            loop.remove(region1)
            try:
                value = loop.index(region2)
            except ValueError:
                value = -1
            if value >= 0:
                loop.remove(region2)
            else:
                unpair.remove(region2)
            if ( len(loop) > 0 and len(unpair) > 0 ):
                region1 = loop[0]
                region2 = unpair[0]
            count1  = 0
            count2  = 0
        else:
            if ( count1+1 < len(loop) ):
                region1 = loop[count1+1]
                region2 = unpair[count2]
                count1 += 1
            else:
                if ( count2+1 < len(loop) ):
                    region1 = loop[count2]
                    region2 = loop[count2+1]
                    count2 += 1
                else:
                    loop.remove(loop[0])
                    if ( len(loop) > 0 and len(unpair) > 0 ):
                        region1 = loop[0]
                        region2 = unpair[0]
                    count1 = 0
                    count2 = 0

    return


#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
#                           MAIN                                    #
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

wGC = 11; wAU = 8; wGU = 3
pairing = {'AA':0,   'AC':0,   'AU':wAU, 'AG':0,
           'CA':0,   'CC':0,   'CU':0,   'CG':wGC,
           'GA':0,   'GC':wGC, 'GU':wGU, 'GG':0,
           'UA':wAU, 'UC':0,   'UU':0,   'UG':wGU}
inputFile  = open('input.txt','r')
allData    = inputFile.readlines()
inputFile.close()
outputFile = open('bracket.txt','w')
count_file = 0; stem_section = []; stem_cross = []; stem_knot = []
print 'Total sequence : ',len(allData), '\n'
for data in allData:
    stem_section  = []; stem_cross = []; stem_knot = []
    seq_ID        = data.split()[0]
    seq           = (data.split()[1]).upper()
    N             = len(seq)
    pre_str       = ['.']*N
    wMatrix       = (array([0]*(N**2))).reshape(N,N)
    count_file   += 1
    print 'Seq ', str(count_file).ljust(4), ': ', seq_ID
    fill_wMatrix()
    section_search(0,N-1)
    bracket( stem_section, 1 )
    cross_search()
    bracket( stem_cross, 1 )
    knot_search()
    bracket( stem_knot, 2 )
    save()
outputFile.close()
```

APPENDIX B-1

Result for FRABASE dataset

ID	Length (nt)	Proposed algorithm		Hotknots		pknotsRG		DotKnot		Cylofold	
		SP	SN	SP	SN	SP	SP	SP	SN	SP	SN
17RA	21	100	75	100	100	100	100	100	75	100	75
1A60	44	89.29	96.15	92.86	100	89.29	89.29	89.29	96.15	89.29	96.15
1A9L	38	100	100	100	100	100	100	100	100	100	100
1AFX	12	100	100	100	100	100	100	100	100	100	100
1AJF	18	100	100	100	100	100	100	100	100	100	100
1AJU	30	100	100	100	100	100	100	100	100	100	100
1AQO	29	100	91.67	100	91.67	100	100	100	100	100	91.67
1ARJ	29	100	100	100	100	100	100	100	100	100	100
1ATO	19	100	100	100	100	100	100	100	100	100	100
1ATV	17	100	100	100	100	100	100	100	100	100	100
1ATW	15	100	100	100	100	100	100	100	100	100	100
1BN0	20	100	100	100	100	100	100	100	100	100	100
1BVJ	23	100	100	100	100	100	100	100	100	100	62.5
1BYJ	27	100	100	100	100	100	100	100	88.89	100	88.89
1CQ5	43	84.62	100	91.67	100	80	80	84.62	100	80	72.73
1D0T	21	100	100	100	100	100	100	100	75	100	75
1DDY	35	72.22	81.25	72.22	81.25	72.22	72.22	68.75	68.75	72.22	81.25
1E95	36	100	100	100	100	100	100	100	100	100	100
1EBQ	29	100	100	100	100	100	100	100	100	100	100
1EBR	30	100	100	100	100	81.82	81.82	100	80	81.82	90
1EHZ	76	85.42	97.62	95.24	95.24	95.24	95.24	69.57	76.2	95.24	95.24
1EOR	22	100	100	100	100	100	100	100	100	100	100
1ESY	19	100	100	85.71	100	100	100	85.71	100	100	66.67
1F1T	38	100	100	100	100	100	100	100	84.62	100	84.62
1F7F	27	77.27	94.44	81.82	100	81.82	81.82	81.82	100	81.82	100
1FEQ	17	100	100	100	100	100	100	100	100	100	100
1FQZ	27	85.71	100	100	100	42.86	42.86	100	100	42.86	50
1FYO	27	100	100	100	100	100	100	100	88.89	100	88.89
1HLX	20	87.5	100	87.5	100	87.5	87.5	87.5	100	87.5	100
1HWQ	30	90	100	90	100	90	90	90	100	90	100
1I3X	19	100	100	100	100	100	100	100	100	100	100
1IE1	22	100	100	100	100	100	100	100	100	100	100
1J4Y	17	85.71	100	85.71	100	85.71	85.71	85.71	100	85.71	100
1JUR	22	100	100	100	100	100	100	100	100	100	100
1JTW	16	100	100	100	100	100	100	100	80	100	100
1JTJ	23	100	100	100	100	100	100	100	100	100	100
1JP0	21	100	71.43	100	100	100	100	100	71.43	100	71.43
1K4A	14	100	100	100	100	100	100	100	100	100	100
1K4B	14	100	100	100	100	100	100	100	100	100	100
1K5I	23	100	100	100	100	100	100	100	100	100	100
1K6G	22	100	100	100	100	100	100	100	100	100	100
1K6H	22	100	100	100	100	100	100	100	100	100	100
1KAJ	32	72.73	100	72.73	100	72.73	72.73	100	100	72.73	100
1KKS	24	100	100	100	100	100	100	68.75	91.67	100	100
1KP7	30	100	100	100	100	80	80	100	77.78	80	88.89
1L1W	29	100	100	100	100	100	100	100	100	100	100
1L2X	28	88.89	100	88.89	100	88.89	88.89	88.89	100	88.89	100
1LC6	24	100	100	100	100	100	100	100	100	100	100
1LDZ	30	90	100	90	100	90	90	90	100	90	100
1LUU	17	100	100	100	100	100	100	100	100	100	100
1LVJ	31	100	100	100	100	100	100	100	100	100	100
1MFJ	20	100	100	100	100	100	100	100	100	100	100
1MFK	23	100	100	100	100	94.44	94.44	100	100	94.44	94.44
1MNX	42	85.71	100	100	100	100	100	100	100	100	100
1MT4	24	100	100	100	100	100	100	100	100	100	100
1N8X	36	100	100	100	100	100	100	100	100	100	100
1NC0	24	100	100	100	100	100	100	100	100	100	100
1NEM	23	100	100	100	100	100	100	100	100	100	100
1NZ1	24	100	100	100	100	100	100	100	100	100	100
1OQ0	15	100	100	100	100	100	100	100	100	100	83.33
1OW9	23	100	100	100	100	100	100	100	100	100	100
1P5N	34	72.73	100	80	100	80	80	66.67	75	80	100
1PJY	22	100	100	100	100	100	100	100	100	100	100
1Q8N	38	100	100	100	100	100	100	100	84.62	100	84.62
1QD3	29	100	100	100	100	100	100	100	100	100	100
1R2P	34	89.29	96.15	92.86	100	92.86	92.86	92.86	100	92.86	100

ID	Length (nt)	Proposed algorithm		Hotknots		pknotsRG		DotKnot		Cylofold	
		SP	SN	SP	SN	SP	SP	SN	SP	SN	SP
1R7W	34	100	100	100	100	100	100	100	100	100	100
1RAW	37	100	100	75	75	75	75	75	75	75	75
1RFR	30	100	100	100	100	100	100	100	100	100	100
1RNK	34	91.67	100	91.67	100	91.67	91.67	91.67	100	91.67	100
1S34	23	100	100	100	100	100	100	100	100	100	66.67
1SY4	24	100	100	100	100	100	100	100	100	100	100
1TJZ	22	100	100	100	100	100	100	100	100	100	100
1TXS	38	100	100	100	100	100	100	100	100	100	100
1U3K	38	100	100	100	100	100	100	100	100	100	100
1XHP	32	100	100	100	100	100	100	100	100	66.67	54.55
1XSG	27	100	100	100	100	100	100	100	100	100	100
1XSH	27	86.36	95	90.91	100	90.91	100	90.91	100	90.91	100
1YG4	28	100	100	100	100	100	100	100	100	100	100
1YNC	31	95.45	95.45	100	100	100	100	95.45	95.45	100	100
1YNG	31	95.45	95.45	100	100	100	100	95.45	95.45	93.33	63.64
1YSV	27	100	100	100	100	100	100	100	100	100	100
1Z2J	45	100	100	100	100	100	100	100	100	100	100
1ZC5	41	100	100	100	100	100	100	100	100	100	100
2A43	26	87.5	100	87.5	100	87.5	100	87.5	100	87.5	100
2A9L	38	100	100	100	100	100	100	100	100	100	100
2AHT	27	100	100	100	100	100	100	100	100	100	100
2AP0	28	100	100	100	100	100	100	100	100	100	100
2F88	34	92.86	100	92.86	100	92.86	100	92.86	100	92.86	100
2HNS	22	100	100	100	100	100	100	100	100	100	100
2JUK	22	100	100	100	100	100	100	100	100	100	100
2JXV	33	100	100	100	100	100	100	100	100	100	100
2K5Z	29	100	81.82	100	81.82	100	100	100	81.82	100	81.82
2K63	29	75	100	100	100	100	100	100	100	100	100
2KE6	48	94.74	100	94.74	100	92.11	97.22	94.74	100	93.75	83.33
2KEZ	24	100	100	100	100	100	100	100	100	100	100
2KFC	37	75	83.33	62.5	83.33	75	83.33	62.5	83.33	100	55.56
2KUW	48	100	100	100	100	97.37	97.37	100	100	100	84.21
2KUV	48	100	100	100	100	97.37	97.37	100	100	100	84.21
2KX8	42	96.88	96.88	100	100	100	100	100	100	100	100
2L2J	42	100	100	100	100	100	100	100	100	100	100
3EGZ	64	83.33	97.22	100	88.89	97.06	91.67	93.75	83.33	97.06	91.67
437D	28	88.89	100	88.89	100	88.89	100	88.89	100	88.89	100
1D6K	37	81.82	100	100	100	81.82	100	100	100	100	100
1I6U	37	93.33	93.33	93.33	93.33	93.33	93.33	93.33	93.33	93.33	93.33
1QA6	58	75	90	93.33	93.33	76.67	76.67	76.67	76.67	91.67	73.33
2J28	74	100	100	100	96.3	100	100	100	100	97.5	72.22
1TN2	76	85.42	97.62	95.24	95.24	54.76	54.76	69.57	76.19	78	92.86
1YFG	75	89.58	97.73	86.36	86.36	86.36	86.36	86.36	86.36	97.62	93.18
4TNA	76	85.42	97.62	95.24	95.24	54.76	54.76	69.57	76.19	95.24	95.24
Average	31	95.60	98.18	96.86	98.70	95.16	97.02	95.59	95.59	95.79	93.66
Max.	76	100	100	100	100	100	100	100	100	100	100
Min.	12	72.22	71.43	62.5	75	54.76	54.76	62.5	68.75	42.86	50.00

APPENDIX B-2

Result for RNA STRAND dataset

ID	Length (nt)	Proposed algorithm			pknotsRG			DotKnot		
		SP	SN	Execution time (s)	SP	SN	Execution time (s)	SP	SN	Execution time (s)
ASE_00001	262	76.92	80	0.53	78.48	82.67	12.561	71.79	74.67	2.49
ASE_00002	267	71.52	77.4	0.56	76.97	80.14	11.11	75.97	80.14	2.64
ASE_00003	294	70	65.88	0.81	83.13	81.17	8.052	79.07	80	12.05
ASE_00004	310	74.44	69.01	0.9	82.18	85.57	14.652	82.45	79.9	24.49
ASE_00005	341	79.21	68.38	1.684	84.84	77.78	17.085	89.25	70.94	24.33
ASE_00006	309	69.1	66.13	0.94	80.1	84.41	12.482	79.78	76.34	20.85
ASE_00007	346	80.41	70.91	1.75	85.44	80	14.854	84.85	76.36	23.46
ASE_00008	344	81.73	77.27	1.81	85.86	77.27	15.322	86.06	81.36	20.55
ASE_00009	229	75.76	74.63	0.37	93.65	88.06	4.948	88.64	87.31	2.53
ASE_00011	294	74.39	71.76	0.87	78.16	80	9.02	75.88	75.88	12.8
ASE_00012	385	81.28	77.64	1.78	85.34	80.49	20.829	80.93	77.64	28.5
ASE_00013	294	70.51	64.71	0.81	81.82	84.71	8.442	77.06	77.06	10.24
ASE_00014	330	82.86	82.08	1.44	80.09	81.6	13.715	80.66	80.66	52.76
ASE_00015	294	77.27	70	1.03	73.33	77.65	9.066	79.07	80	11.37
ASE_00017	320	41.12	68.75	0.94	48.42	71.88	7.429	42.59	53.91	7.5
ASE_00018	402	87.5	78.36	2.65	92.75	85.82	23.215	85.83	81.34	44.04
ASE_00019	294	71.6	68.24	1.01	8146	85.29	6.399	79.88	77.06	11.14
ASE_00020	385	78.02	71.83	1.81	84.27	82.94	18.504	83.61	78.97	8.42
ASE_00021	457	78.93	69.06	3.31	84.35	77.5	33.416	82.39	73.13	17.15
ASE_00022	408	71.67	69.35	3.06	78.99	75.81	22.632	77.59	72.58	13.07
ASE_00023	411	76.89	79.92	3.21	84.25	84.25	24.09	80.88	86.61	16.92
ASE_00026	346	76.29	67.27	1.72	81.98	82.73	12.04	80.09	78.64	6.15
ASE_00028	412	68.5	69.05	2.22	82.66	81.35	22.282	78.57	74.21	13.884
ASE_00029	408	72.01	79.1	3.12	84.65	79.1	22.201	76.03	75.41	10.52
ASE_00030	414	82.54	71.72	2.56	88.89	82.76	29.009	87.7	76.21	22.98
ASE_00031	417	70.68	74.6	3.84	82.94	82.94	25.19	75.82	73.41	21.77
ASE_00032	401	68.94	77.12	3.06	80.63	75.85	25.17	74.77	67.8	12.45
ASE_00033	361	80.73	72.73	2.21	85.44	72.73	17.74	79.73	73.14	14.02

ID	Length (nt)	Proposed algorithm			pknotsRG			DotKnot		
		SP	SN	Execution time (s)	SP	SN	Execution time (s)	SP	SN	Execution time (s)
ASE_00035	376	74.15	74.15	2.28	71.1	65.68	13.984	80.84	73.31	7.61
ASE_00037	345	73.53	68.18	1.84	76.92	72.73	10.41	80.73	70.45	10.25
Average	349	74.62	72.51	1.80	81.33	80.03	16.19	79.28	76.15	16.23
Max.	457	87.50	82.08	3.84	93.65	88.06	33.42	89.25	87.31	52.76
Min.	229	41.12	64.71	0.37	48.42	65.68	4.95	42.59	53.91	2.49

APPENDIX B-3

Result for CompaRNA dataset

ID	Length (nt)	SP	SN	Execution time (s)
2KFC_0_A	36	75	93.75	0
2RP0_23_A	27	77.78	100	0
2WDJ_0_A	2809	62.77	70.03	2777
2WH3_0_W	76	35.19	86.36	0.06
2WH4_0_A	2889	65.41	73.78	2739.63
2WWL_0_A	1530	67.95	73.43	248.26
2WWQ_0_B	2904	63.92	69.28	2475.8
2X9R_0_V	75	66.67	85.71	0.06
2X9T_0_A	1509	55.85	68.85	314.47
2X9U_0_A	2901	67.17	76.19	2852
2ZM5_0_C	74	70.37	86.36	0.06
2ZM5_0_D	69	59.57	77.78	0.06
2ZM6_0_A	1511	58.45	65.1	311
2ZUF_0_B	76	86	97.73	0.06
2ZZN_0_C	71	91.3	95.45	0.06
3A2K_0_D	77	85	77.27	0.03
3A3A_0_A	86	100	93.33	0.09
3ADC_0_C	88	100	93.33	0.06
3ADC_0_D	92	100	93.94	0.09
3ADD_0_D	88	100	93.94	0.09
3CXC_0_0	2754	66.34	73.71	2046.04
3DS7_0_A	67	85.42	89.13	0.03
3E1A_0_A	75	77.08	97.37	0.06
3FIC_0_Y	68	76.19	80	0.03
3FIH_0_Y	76	85.42	97.62	0.06
3FIN_0_A	2855	61.95	70.31	2714.75
3FO4_0_A	63	84.09	88.1	0.03
3FU2_0_B	31	100	100	0
3FU2_0_C	31	100	100	0
3FWO_0_A	2770	48.04	62.96	2244.88
3G8S_0_S	138	61.7	80.56	0.19
3G8T_0_P	140	61.32	83.33	0.16
3G8T_0_R	141	65.09	84.15	0.16
3G9C_0_P	140	62.5	81.08	0.19
3G9C_0_Q	141	70.21	82.5	0.16
3G9C_0_R	140	60.64	81.43	0.16
3G78_0_A	389	72.97	68.64	2.12
3G96_0_S	139	60.64	81.43	0.22
3GCA_0_A	33	62.5	71.43	0
3GER_0_A	67	85.42	89.13	0.03
3GES_0_A	67	85.42	89.13	0.03
3GOG_0_A	65	84.78	88.64	0.06
3GX2_0_A	95	80	85.71	0.09
3GX6_0_A	94	76.67	88.46	0.09
3GX7_0_A	94	78.33	83.93	0.09
3HHN_0_E	137	78.75	70	0.19
3HL2_0_E	82	68	73.91	0.06
3HUW_0_A	1500	53.56	64.74	307.48
3HUX_0_A	2768	64.79	71.73	2374.39
3HUZ_0_A	2780	54.91	64.6	2437.1
3I1M_0_A	1534	65.28	71.02	245.03
3I1P_0_A	2844	61.34	66.71	2231.61
3I1R_0_A	2855	65.31	70.86	2277.6
3I8F_0_A	2909	72.09	76.05	2809.19
3I8G_0_A	1516	71.89	77.34	311.38
3I8H_0_A	1515	71.27	77.02	309.6
3I8I_0_A	2913	68.97	72.24	2852.43
3I9B_0_A	1517	71.78	77.23	313.06
3I9C_0_A	2885	71.87	76.04	2749.54

ID	Length (nt)	SP	SN	Execution time (s)
3I9D_0_A	1532	68.99	74.58	316.56
3I9E_0_A	2886	73.02	76.48	2770.08
3I55_0_0	2755	61.8	68.98	1651.73
3I56_0_0	2754	65.11	74.68	1639.47
3IGI_0_A	387	79.52	70.76	1.75
3IIN_0_B	198	47.12	48.51	0.34
3IRW_0_R	92	65.38	68	0.09
3IVK_0_C	128	83.33	89.29	0.16
3IVN_0_A	69	77.27	73.91	0.06
3IWN_0_A	94	70.69	70.69	0.06
3JYV_0_7	70	52.17	70.59	0.03
3JYV_0_A	1759	46.26	73.35	548.37
3JYX_0_5	3174	32.53	73.38	5050.45
3K1V_0_A	29	100	100	0.03
3KIQ_0_w	77	19.57	64.29	0.06
3KIT_0_A	2849	66.82	73.49	2623.55
3KNI_0_A	2804	53.33	64.83	2486.86
3KNJ_0_A	1502	42.95	64.99	300.52
3KNJ_0_W	75	34.09	53.7	0.06
3KNJ_0_Y	76	28.26	76.47	0.03
3KNK_0_A	2803	49.94	66.9	2477.28
3KNL_0_A	1500	56.85	67.13	321.27
3KNM_0_A	2789	66.67	76.46	2421.37
3KNN_0_A	1497	53.63	68.21	289.97
3KNO_0_A	2776	64.62	74.69	2395.73
3L0U_0_A	73	70.37	86.36	0.06
3L3C_0_P	136	69.05	78.38	0.16
3L3C_0_Q	139	72.34	82.93	0.22
3L3C_0_R	140	72.34	82.93	0.19
3L3C_0_S	139	55.43	79.69	0.19
3LA5_0_A	71	91.3	84	0.06
3MR8_0_A	1505	69.11	75.37	331.5
3MRZ_0_A	2880	65	70	2777.02
Average	1034.5	68.54	78.31	360.61
Max.	3174	100	100	5050.45
Min.	27	19.57	48.51	0