B.Sc. Engineering Thesis

Studies on Different Algorithms in Prediction of RNA Secondary Structure

By
Tanzim Ahmed
Student ID: 0805094

Supervised by
Dr. Md. Abul Kashem Mia
Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

A thesis submitted to the
Department of Computer Science and Engineering, BUET in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science and Engineering

Department of Computer Science and Engineering
BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
Dhaka-1000, Bangladesh
July, 2014

Declaration

This is to certify that the work presented in this thesis entitled "**Studies on Different Algorithms in Prediction of RNA Secondary Structure**" is the outcome of the investigation carried out by us under the supervision of **Dr. Md. Abul Kashem Mia**, Professor, Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka. It is also declared that neither this thesis nor any part of this thesis has been submitted or is being currently submitted anywhere else for the award of any degree or diploma.

(Author)
Tanzim Ahmed
Student No: 0805094

Abstract

Secondary structure prediction of nucleic acid molecules is a very important problem in computational molecular biology. The biological role of many RNA crucially depends on their structure. The in depth understanding of the secondary structure of RNA would provide a better insight in to their functionality. Predicting secondary structure of RNA is the most important factor in determining its 3D structure and functions.

Understanding the function of RNA molecules is a key to unlocking the pathways of diseases and biology. Knowledge about this native structure can have an enormous impact on the field of pharmacology, medical science and drug discovery. Main approaches to RNA secondary structure prediction are energy minimization in dynamic programming approach, comparative sequence analysis, parallel algorithm etc.

Keywords--- *RNA, Secondary Structure, ZEM, Nussinov Algorithm, Traceback Algorithm, GTfold, SPfold*

Acknowledgements

First of all I would like to thank my supervisor, Dr. Md. Abul Kashem Mia, for introducing me to the fascinating field of Bioinformatics and guiding me on how to perform my research work. Without his continuous supervision, guidance and valuable advice, it would have been impossible to complete the thesis. We are especially grateful to him for allowing us to choose our course of actions with freedom, for enlightening us during the moments of frustration, for his patience throughout the entire research work and for his moral support to the extra-curricular activities that I accomplished aside from the thesis.

I would also like to thank my friend, Khandoker Md. Nayem, who provided me great support. I am also grateful to all other friends for their continuous encouragement and for helping me in thesis writing.

I would like to express my gratitude to all my teachers. Their motivation and encouragement in addition to the education they provided meant a lot to me.
Last but not the least, I am grateful to my parents and to my family for their patience, interest, and support during my studies.

Contents

List of Figures

Chapter 1

Introduction

One of the most important problems in computational molecular biology is structure prediction of nucleic acids. The prediction of the correct secondary structures of large RNAs is one of the unsolved challenges of computational molecular biology. RNA molecules play a crucial role in cellular processes and their function is directly related to their folding into complex tertiary structures DNA and RNA strands are also used in bio-molecular computations and in self-assembly of nanostructures. Researchers have studied the problem of nucleic acids structure prediction for more than two decades. In this thesis, we start from the state-of-the-art algorithms for nucleic acid secondary structure prediction and extend them to the problems of (1) predicting secondary structure of a pair of nucleic acids and (2) finding, out of a combinatorial set of RNA or DNA short strands, which combination has the most stable secondary structure. In this chapter, we first give background on RNA, DNA and secondary structures. Then, in Section 2, we define the problems that we deal with in this thesis and we give motivations to support why this work is important. A summary of contributions is given in Section 3, and some notations and conventions that we will use throughout this thesis are enumerated in Section 4. The last section gives a brief outline of this thesis.

1.1 Background

The central dogma in molecular biology (Figure 1.1) is that, in an organism, the genes, which constitute the genetic code made of DNA (deoxyribonucleic acid), are first replicated, and then transcribed into RNA (ribonucleic acid). This is premature messenger RNA (pre-mRNA), which in higher organisms (i.e. eukaryotes) are first processed to eliminate some non-coding sequences, called introns, and they become mature messenger RNA (mRNA). In simpler organisms (i.e. prokaryotes), there are no introns, and the genetic code is directly transcribed into mature mRNA. This is translated into proteins, which have well defined functions. DNA and RNA molecules (also known as nucleic acids) are composed of sequences of four types of nucleotides or bases: Adenine (A), Cytosine (C), Guanine (G) and Thymine (T) for DNA or Uracil (U) for RNA. In organisms, the genetic material is usually double-stranded DNA and the RNA is single-stranded. For this reason, RNA is more flexible and can form a much greater variety of complex three-dimensional structures than double-stranded DNA (dsDNA). However, single-stranded DNA (ssDNA), used in in vitro experiments or in DNA computing, can also form complex structures.

The linear sequence of an RNA or DNA strand constitutes the primary structure or sequence. The set of base pairs that form when a nucleic acid sequence folds is called secondary structure. These pairings arise from the Hydrogen-bonding forces between pairs of bases. Thus, in an RNA or ssDNA secondary

structure, each base can be either free (not bonded with any other base) or Hydrogen-bonded to another base. The tertiary structure is the three-dimensional geometry of the arrangement of bases in space. Much research has been done on understanding secondary structures, while the information we currently have about tertiary structures is relatively sparse. It is believed that once secondary structures are known, they can provide useful information about tertiary structures as well.

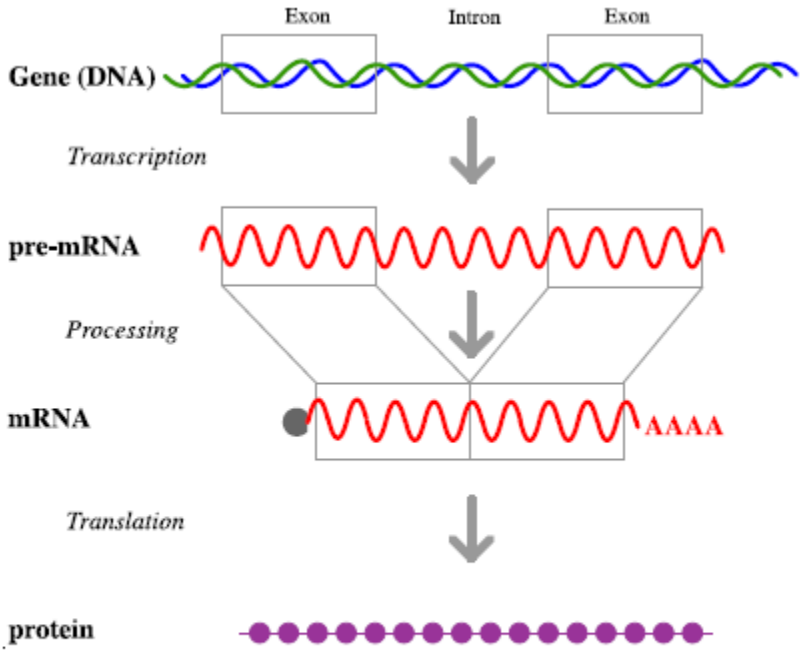

Figure 1.1: Central dogma in molecular biology for eukaryotes.

Although there are other factors that influence secondary structure formation, it is believed that the sequence has the greatest contribution. The most common Hydrogen-bonds which will lead to secondary structure formation are between C and G, and between A and T (or U for RNA). They are called Watson-Crick (WC) bonds. (C.G) pairs are more stable, because they involve three Hydrogen connections, as opposed to (A.T) or (A.U) pairs, which involve two Hydrogen connections.

The sequence orientation of ssDNA and RNA strands is defined by two different strands, called the 5' end and the 3' end. Consecutive base pairs can be formed only if the sequences have opposite orientation. The following is a double stranded DNA primary sequence of 30 nucleotides:

5'-ATGCGCGCTAGCATCGCTCGGCTAGCTGAT-3'
3'-TACGCGCGATCGTAGCGAGCCGATCGACTA-5'

Each base in the second strand is the W-C complement of the corresponding base in the first strand. They can bind and form a double stranded DNA because all the corresponding bases are complementary

and because the strands are in opposite orientation. The same rule is available for RNA or ssDNA. Consider the following simple RNA sequence:

5'-CCCCCCCCCCAAAAAGGGGGGGGGG-3'

It contains 10 C's, 5 A's and 10 G's. The fragment 5'-CCCCCCCCCC-3' can bind to the fragment 3'-GGGGGGGGGG-5' (read from the 3_0 end to the 5_0 end), but it cannot bind to the fragment 5'-GGGGGGGGGG-3' (read from the 5' end to the 3' end), because it does not have the right orientation. Thus, most probably, the first C will bind to the last G, the second C will bind to the G before the last base and so on.

The first step in understanding RNA or ssDNA secondary structures is to identify the substructures of which they are composed. We call elementary structure any substructure which cannot be decomposed in any other substructure. Figure 1.2 shows an example of a complex secondary structure, which contains all elementary structures that we consider in this work. The bullets represent bases; the thin gray line indicates the backbone that holds all the nucleotides together in the molecule, and the thick black lines indicate paired bases.

The 5' and 3' ends are indicated, and a numbering for the base positions is provided, with the first base considered to be at the 5' end. Each base can only participate in at most one base pair. The elementary structures are marked by rectangles and their names are added aside. The elementary structures we consider here follow:

- A hairpin loop or hairpin contains one closing base pair and all the bases between the paired bases are unpaired. The hairpin marked in the figure contains 5 free bases.

- A stacked loop, also called stacked pair, contains two consecutive base pairs. The tuple (I, j) defines a stacked pair if i and j are paired and i + 1 and j - 1 are paired. A stem or helix is made of a consecutive number of stacked loops. The helix marked in the figure has 5 stacked loops.

- An internal loop, sometimes called interior loop, is a loop having two closing base pairs, and all bases between them are free. The tuple (I, j, i', j'), with $i + 1 < i' < j' < j-1$, defines an internal loop if i and j are paired, i' and j' are paired. The asymmetric internal loop marked in the figure has 3 free bases on one side and 4 free bases on the other side.

- A bulge loop, or simply bulge, is a special case of internal loop, which has no free base on one side, and at least one free base on the other side. In fact, a stacked loop is also a special case of internal loop, with no free bases on both sides. In this work, we will consider stacked loops and internal loops to be distinct elementary structures, but we include bulges in the internal loop case, unless otherwise specified.

- A multi-branched loop or multi-loop is a loop which has at least three closing base pairs. The tuple (I, j, i_1; j_1, i_m; j_m), with $m \geq 2$; $i < i_1 < j_1 <\ldots\ldots< i_m < j_m < j$ defines a multi-loop with m + 1 branches if i pairs with j, i_{11} pairs with $j_1\ldots\ldots i_m$ pairs with j_m . The multi-loop marked in the figure has 3 closing base pairs and 6 free bases.

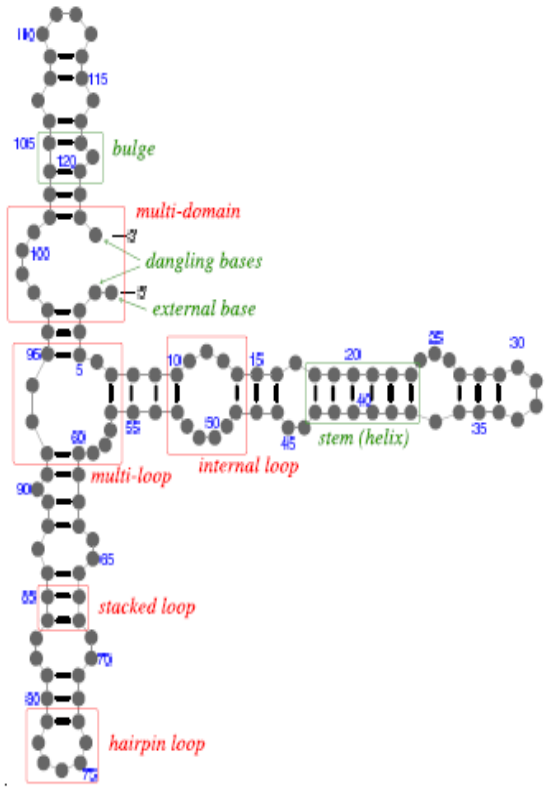

Figure 1.2: Example of a pseudoknot-free secondary structure containing all elementary structures.

- A multi-domain loop or simply multi-domain, is a loop with at least one closing pair. The tuple (I, j, i_1, j_2,…, i_d, j_d), with d ≥ 0; i < j < i_1 < j_2 < …… < i_d < j_d defines a multi-domain if i pairs with j, i_1 pairs with j_1, ……,i_d pairs with j_d and k is a free base. We call domains the whole secondary structures which are closed by the closing pairs of a multi-domain. In other words, the domain closed by (i; j) is the set of all base pairs whose indices are in the interval [i; j]. Note that if we virtually make the sequence circular by merging the 5' and 3' ends together, we obtain a hairpin, internal loop or multi-loop. The multi-domain marked in the figure contains two branches and 7 free bases. These free bases are called external bases, because they are not inside any domain, but they are between domains or between a domain and a molecule end.

Note that, if the whole structure contains only one domain and no external bases, then there is one multi-domain, and the pair (1.n) constitutes its only closing pair. The bases which are neighbors of a closing pair of a multi-domain or multi-loop are called dangling bases. The dangling bases neighboring to the multi-domain marked in the figure have positions 2, 98, 101 and 123. The ones neighboring to the multi-loop have positions 6, 57, 59, 93 and 94.
When we refer to secondary structures, we assume that at least one base pair exists. In some situations, it is possible that no pairing between any two bases exist. In this case we say that the molecule is structure free.

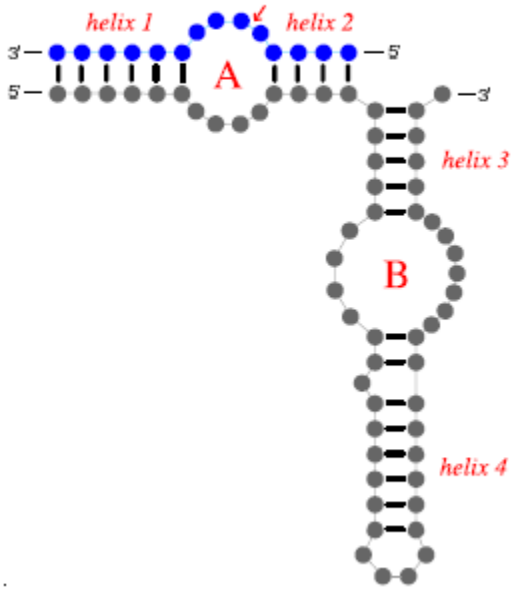

Figure 1.3: Structure of a hairpin ribozyme bound to an mRNA target.

All these elementary structures have been used before with these names except the multi-domain structure, whose exact name we introduce here for convenience later. Another way to understand a secondary structure is to think of it as a set of helices, connected to each other by internal loops, bulges, multi-loops or multi-domains, and some of them ended in hairpins. Note that the secondary structure represented in Figure 1.2 is just a graphical, convenient way to visualize the set of base pairs of the folded molecule. In other words, the geometrical positions of the base pairs and free bases do not have any meaning and do not matter other than for visualization purposes.

Extending the notion of secondary structure for a single molecule, the secondary structure for a pair of molecules S_1 and S_2 is a set of base pairs, with each base of S_1 and S_2 occurring in at most one pair. Figure 1.3 shows an example of a duplex structure, which represents the typical structure of a hairpin ribozyme (the bottom structure), binding to its mRNA target (the top structure). Having this structure, the ribozyme executes its function by cleaving the mRNA target at the site indicated by the red arrow.

The loops that we just enumerated are all elementary structures, which are currently considered by computational programs for secondary structure prediction without pseudoknots. Pseudoknots are very important structures, being sometimes crucial for the RNA function, but making prediction more complicated. The tuple (I, j, I', j') defines a pseudoknot if i and j are pairs, i0 and j0 are pairs and $i < I' < j < j'$. Pseudoknots are not considered in this work. Figure 1.4 shows an example of a simple pseudoknot, marked by a rectangle. It contains two helices, and if we take any pair from the first stem and any pair from the second stem, they satisfy the inequality mentioned above. Inside the pseudoknot and/or outside the pseudoknot, the structure can have any elementary structures discussed before, or even other pseudoknots.

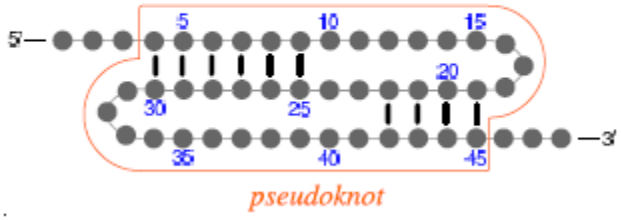

Figure 1.4: A simple pseudo-knotted structure

Tertiary interactions between a base which is already paired and a free base, or between two already paired bases, are possible. These and other interactions, such as Hydrogen bonding between a base and the backbone, between backbone and backbone, binding of metal ions, water interactions, all contribute to the stability of RNA structure. These are part of tertiary structures and are not considered in secondary structure estimations.

Chapter 2

Preliminaries

2.1 Some Definitions

Genome – An organism's genetic material.
Vesicle - a small fluid-filled bladder, sac, cyst, or vacuole within the body
Exon - any nucleotide sequence encoded by a gene that remains present within the final mature RNA product of that gene after introns have been removed by RNA splicing
Intron - any nucleotide sequence within a gene that is removed by RNA splicing while the final mature RNA product of a gene is being generated
Phenotype – the physical traits of an organism
Mutation – a change in a DNA sequence
Organism - an individual animal, plant, or single-celled life form
Progeny – offspring
Collagen - the main structural protein of the various connective tissues in animals.
Nucleotide – basic structural unit of nucleic acids such as DNA
mRNA – this is what is usually being referred to when a Bioinformatician says "RNA". This is used to carry a gene's *m*essage out of the nucleus.
tRNA – *t*ransfers genetic information from mRNA to an amino acid sequence
rRNA – *r*ibosomal RNA. Part of the ribosome which is involved in translation.

Chapter 3

Related Works

RNA folding and secondary structure are very important for RNA function. Experimental studies on RNA secondary and tertiary structures have been carried out continuously after the Watson-Crick binding discovery in 1953. In the last 25 years, applications of computational methods to problems in molecular biology, including RNA folding, became more and more prominent. The quantity of information about RNA structures is rapidly increasing every year.

However, depending on the length of the given RNA sequence, the number of possible secondary structures can be very high. For instance, for a 16S rRNA of 1500 nucleotides, there are approximately 15,000 possible helices (less than 100 will be in the final structure). The maximum number of combinatorial arrangements of all possible helices will eventually lead to different structures. In vitro experiments for some specific RNAs such as hairpin ribozymes show that they behave very similarly to the experiments in vivo. However, there are studies which show that in vitro RNA folding experiments cannot reliably simulate the complex intracellular environments existing in the cell. Thus, there is still a lot of unknown information about what happens in the cell.

The computational methods that currently exist try to capture as much information as possible from the existing knowledge. Still, this knowledge is rudimentary with respect to some rules and contributions of other factors that participate to the folding of RNA. Moreover, in order to ensure that the computational methods are efficient (and thus practical), thermodynamic methods are often highly simplified.

3.1 Secondary structure prediction for single molecules

In this section, several different methods are explained for predicting secondary structures without pseudoknots, and two algorithms for predicting pseudo-knotted structures are summarized. Free energy minimization algorithm A very popular algorithm for finding the minimum free energy (MFE) secondary structure without pseudoknots of an RNA molecule is Zuker and Stiegler algorithm. Its input is the primary RNA sequence, and it uses a dynamic programming algorithm to find the secondary structure with the minimum free energy. The basis of this method is the Nearest Neighbor Thermodynamic Model (NNTM). NNTM contains rules about base pairing formation and tabulated free energy and enthalpy parameters.

Briefly, the main idea of the free energy minimization algorithm is that the bases of the RNA molecule are numbered from 1 to n, starting from the 5’ end and finishing at the 3’ end. Then, for each i and j with $1 \leq i < j \leq n$, the problem is to determine which of the four elementary structures (hairpin loop, stacked loop, internal loop or multi-branched loop), with the exterior pair (I, j), has the lowest free energy. Recurrence relations are applied and a two dimensional matrix with all minimum free energies for each i

and j is filled. As for the standard dynamic programming algorithm, backtracking is necessary to build the path (i.e. the set of base pairs) that gives the MFE secondary structure.

The complexity of Zuker and Stiegler's algorithm is $O(n^4)$ for time and $O(n^2)$ for space. It has been reduced to $O(n^3)$, but the space required increased to $O(n^3)$. Other approaches assume that the free bases on both sides of internal loops are bounded by a constant c (e.g. c = 30). This reduces the time complexity of Zuker and Stiegler's algorithm to O(n3) with no penalty on the space.

Wuchty extended the MFE secondary structure prediction algorithm to generate all suboptimal secondary structures between the MFE and an upper limit. Generating suboptimal structures is important for at least two reasons:

(1) The energy model on which the minimization algorithm relies is imprecise. Also, there are unknown biological constraints, which are not taken into consideration by the energy model. Thus, the true MFE structure might be one of the suboptimal structures with respect to the parameters used.

(2) Under physiological conditions, RNA molecules might fold to alternative structures, whose energy difference is small. Also, it is speculated that specific folding pathways capture molecules in local minima. Mathews show that, on average, the accuracy of the prediction algorithm increases by more than 20% when 750 suboptimal structures are generated, as opposed to generating the MFE structure only.

Different implementations of the free energy minimization algorithm exist. The program mfold was the first one to implement Zuker and Stiegler's algorithm and is available online. It also incorporates Wuchty extension for generating suboptimal structures. The Vienna RNA Package implements Zuker and Stiegler's algorithm, together with the partition function calculation, which will be discussed later in this section. It is available online and is free open source software.

3.2 Partition function algorithm

McCaskill proposed another dynamic programming algorithm for pseudoknot-free folding of an RNA molecule, which permits calculation of probabilities of various structures. The focus is on the equilibrium probabilities of substructures common to an ensemble or class of related structures. These substructure classes are very important because they allow the display of the most significant features of the ensemble. Finally, the equilibrium probability of occurrence for each possible base pair can be calculated, and a mirror image including the base probabilities (one triangle) and the optimal structure (another triangle) can be drawn for good visualization.

McCaskill evaluated his method on four biological RNA sequences with known structures. He showed that the real base pairs have been predicted with high probability, although not always the highest probability. The partition function algorithm has been incorporated in the Vienna RNA Package.

There are two major drawbacks of both the free energy minimization algorithm and the partition function algorithm: (1) they cannot predict pseudo-knotted structures; (2) they heavily rely on the simplified thermodynamic model.

3.3 Comparative analysis methods

Comparative analysis methods predict secondary structures and early stages of tertiary structures of evolutionary related RNA molecules. They overcome the two major drawbacks of the aforementioned methods: pseudo-knotted structures and imprecise thermodynamic model. Gutell Laboratory has started determination of the 16S and 23S rRNA secondary structures since early 1980s, when only two molecules of each class were available.

The comparative analysis method is based on two simple and profound principles:

(1) Different RNA sequences can fold into the same secondary and tertiary structures.
(2) The unique structure and function of an RNA molecule is maintained through the evolutionary process of mutation and selection.

In 1999, 7000 homologous 16S and 1050 23S aligned rRNA sequences where used in covariation-based structure models and the result was compared to the experimentally determined high-resolution crystal structures of the 30S and 50S ribosomal units (which include 16S and 23S rRNAs, respectively). Several methods constitute the class of comparative analysis: (1) covariation analysis predicted 97-98% of the base pairs which are present in the 16S and 23S rRNA crystal structures; (2) tentative covariation-based method predicted about 45% of the base pairs; and (3) motif-based method predicted 70% of the base pairs. In conclusion, these methods predicted nearly all of the standard secondary structure base pairings and helices in the 16S and 23S crystal structures, and they have also identi_ed tertiary base-base interactions. The major drawback of this method is that a large number of evolutionary related sequences are necessary for good accuracy.

Combinations of using the thermodynamic model as well as comparative analysis have been tried. Hofacker presented a method for computing the secondary structure of a set of aligned RNA sequences, using both thermodynamic stability and sequence covariation. They show that only 5 rRNA related sequences and an automatically generated alignment were necessary to correctly predict over 80% of the base pairs. Their program is implemented under the name of RNAalifold, which is available online from the Vienna RNA Package web site.

Chapter 4

Nussinov Algorithm

Researches are going in the field of RNA secondary structure prediction to develop efficient algorithms for the same. One of the most popular dynamic programming algorithm used was Nussinov algorithm.
It is very difficult to compute RNA secondary structure using physical, chemical and biological methods. All computational prediction methods involve some kind of modeling to search for the most stable or possible secondary structure. The criteria for modeling include scores, energy and probabilities.

Information obtained from comparative sequence analysis was used which was based on the fact that same family of RNA share a similar secondary structure. One of the most popular dynamic programming algorithm that was used for secondary structure prediction was Nussinov algorithm.

4.1 Nussinov Algorithm

The Nussinov algorithm tries to maximize the possible number of base pairs of a given sequence [10]. The underlying assumption is that the more base pairs there are in a structure the more stable and more likely the structure is [9]. The algorithm takes advantage of the fact that the optimization problem can be solved by breaking it down into smaller sub problems and solving them [9, 13]. The recursive solution to the problem is to calculate the best structure for a subsequence S[i …j] with $1 \leq i \leq n$ and $i < j \leq n$ where n is the length of S. The result of the maximum number of pairs is stored in a two-dimensional matrix M at. There are four different cases which can occur during the calculation of M[I,j] which are:

1. An unpaired base i together with the best structure for the smaller subsequence S give the best result.

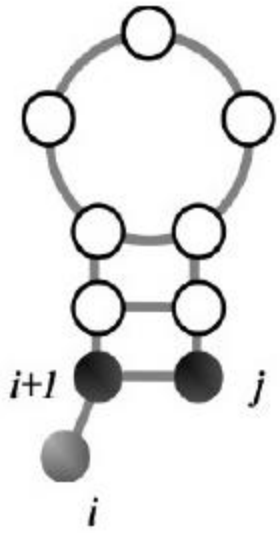

Figure 4.1: i unpaired

2. An unpaired base j together with the best structure for the smaller subsequence S gives the best result.

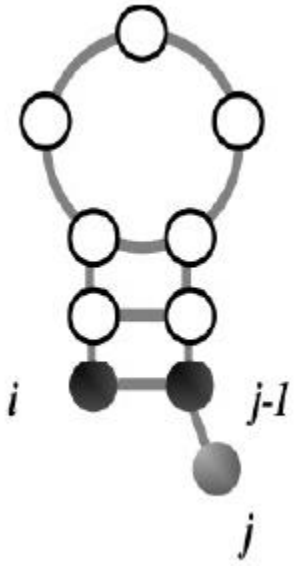

Figure 4.2: j unpaired

3. A base pair (i, j) together with the best structure for the smaller subsequence S[i+1,…, j-1] gives the best result. In this case the new base pair (I, j) is added with the score M[i, j] to M[i+1, j-1] receiving the maximum number of base pairs M.

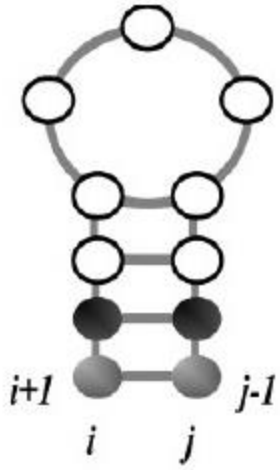

Figure 4.3: i, j paired

4. The combination of the best two structures for the smaller subsequences S[i…k] and S[k+1…j] gives the best result. In this case k has to be found such that M + M is maximal and then the maximum number of base pairs M is M[i, k] + M[k+1, j].

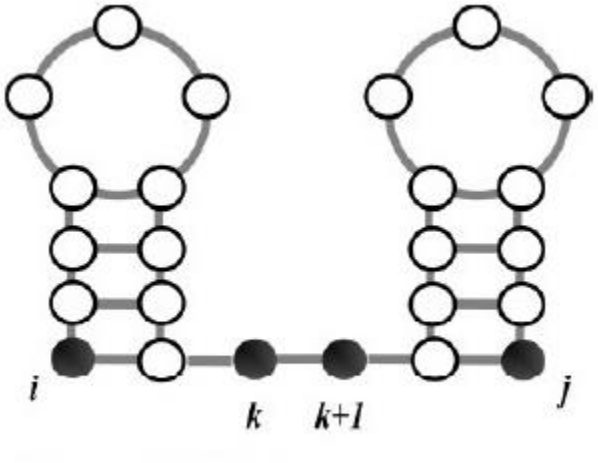

Figure 4.4: Multiloop

After the matrix has been filled the solution can be received via backtracking. Beginning at the maximum number of base pairs for the whole sequence, which is stored in M, one just need to trace back from which of the four possible cells the maximum has been calculated. Then the subsequences are known and for them again backtracking are done. Like this the base pairs of the optimal folding can be recovered. The overall time complexity of the recursion is O and the complexity of space is $O[n^2]$.

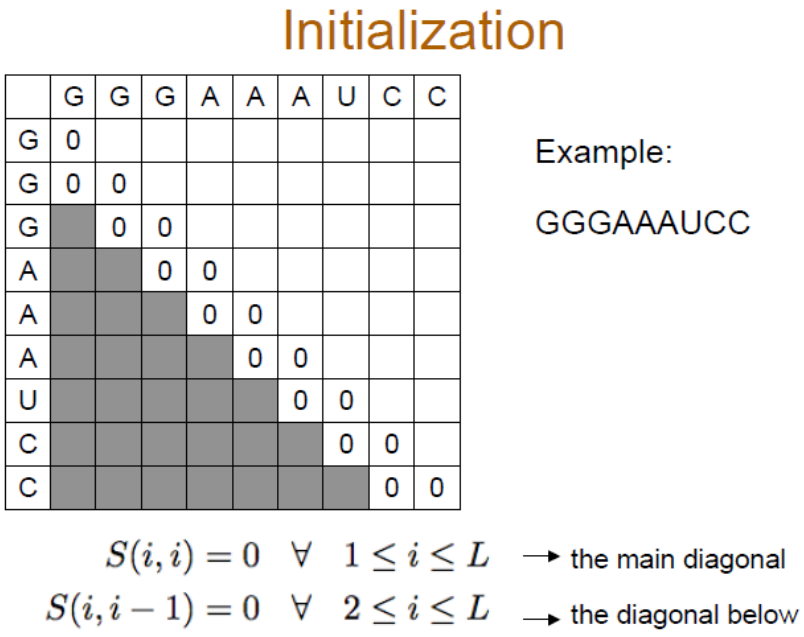

Figure 4.5: Nussinov Algorithm (Initialization)

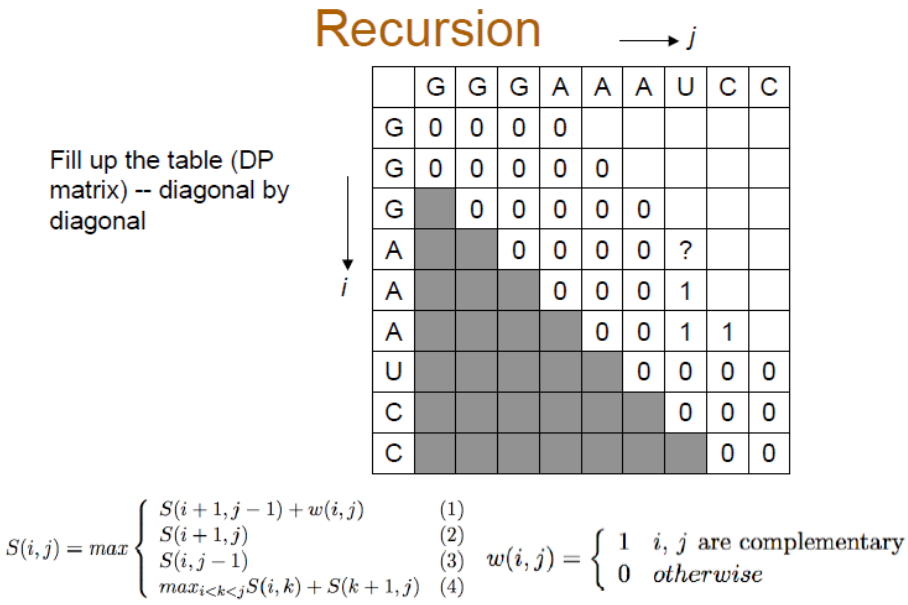

Figure 4.6: Nussinov Algorithm (Recursion)

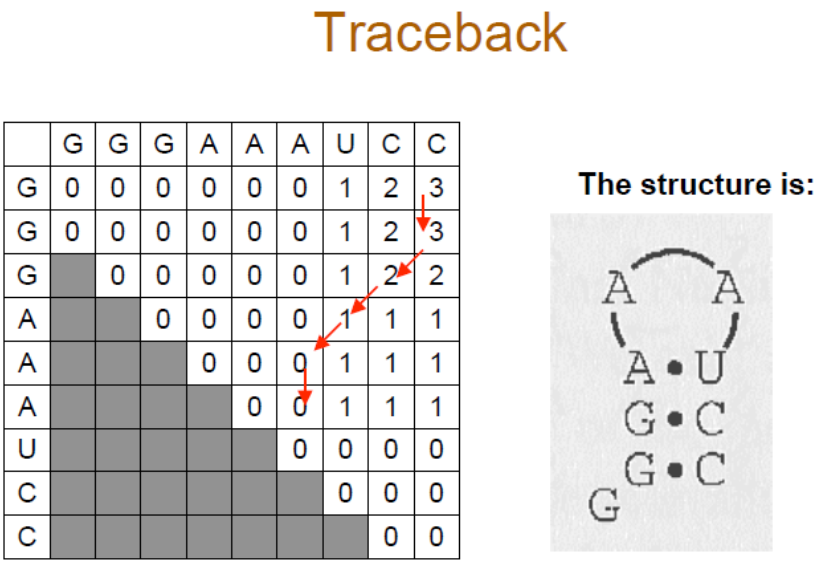

Figure 4.7: Nussinov Algorithm (Traceback)

Even though Nussinov algorithm is a simplistic approach, it does not give accurate structure prediction i.e. nearest neighbor interactions and stacking interactions are not considered here. It also will not necessarily lead to the most stable structure because of the presence of many interior loops and hairpins which are energetically unfavorable. The basic dynamic programming algorithm was then modified to calculate minimum free energy of the structure.

4.2 Drawbacks of Nussinov Algorithm

- Nussinov does not calculate biologically relevant structures because: often there are various possibilities for base-pairing (especially due to pseudoknots), the Nussinov algorithm detects mostly just one variant.
- Stacking of base-pairs is not considered) differences in structure and stability of helices.
- Size of internal loops is not considered.

Chapter 5

Zuker's Energy Minimization Algorithm

ZEM Algorithm was therefore used which takes into account energy of the structure. Random number of sequences is generated for a particular primary sequence of RNA. Energy corresponding to each and every sequence is obtained using ZEM Algorithm. The secondary structure corresponding to the minimum energy is the most appropriate stable structure of RNA.

5.1 ZEM Algorithm

Secondary structure of each RNA sequence is found using ZEM Algorithm. It calculates the minimum free energy and the base pairs of each sequence. *ZEM Algorithm* ZEM states the secondary structure prediction problem as

In the first step of Zukers algorithm, auxiliary numbers $E_{i,j}$ are computed for all fragments, $r_i \ldots r_j$ of the RNA.

$E_{i,j} = 0$ for $j - i < 4$ otherwise

$$E_{i,j} = \min\{\ E_{i+1,j}\ ;\ E_{i,j-1};\ e(i,j) + E_{i+1,j-1};\ \min_{k=i+1}^{j-1}(E_{i,k} + E_{k+1,j})\ \}$$

That is fragments of length ≤ 4 have 0 folding energy, since they cannot fold. Otherwise, r_i is unpaired, or r_j is unpaired, or r_i and r_j pair with each other, or r_i and r_j both pair, but not with each other.

A folding with minimum energy is computed using a traceback algorithm.

Initialization: Set $i = 1$ and $j = n$. Put i and j on to the traceback stack.

- *Steps*

1. If the traceback stack is empty, the traceback terminates. Otherwise, take i and j from the traceback stack.
2. If $E_{i+1,j} = E_{i,j}$ then i is not paired.
   (a) If $j - i > 3$, set $i = i + 1$ and continue with 2.
   (b) If $j - i \leq 3$, continue with 1.

   Otherwise, continue with 3.
3. If $E_{i,j-1} = E_{i,j}$ then j is not paired.
   (a) If $j - i > 3$, set $j = j - 1$ and continue with 3.
   (b) If $j - i \leq 3$, continue with 1.

   Otherwise, continue with 4.
4. If $E_{i,j} = e(i,j) + E_{i+1,j-1}$ then r_i pairs with r_j
   (a) Add i, j to the list of base pairs,
   (b) set $i = i + 1$ and $j = j - 1$ and continue with 2

   Otherwise, continue with 5.
5. If $E_{i,j} = E_{i,k} + E_{k+1,j}$ for some k in (i,j), put the fragment $k + 1, j$ on to the traceback stack (i is $k + 1$ and j is the current j) and deal with fragment $i \ldots k$ by setting j = k and continue with 2. (Note that some k must exist if this point is reached).

In the first step of Zukers energy minimization algorithm an energy matrix showing the dG values of all fragments in the given RNA sequence is obtained. The dG value of the largest fragment in the energy matrix is the minimum dG value of an RNA sequence. This minimum dG value is stored for performing further analysis on it. From the trace back algorithm, base pairs contained in the given RNA sequence are found. These base pairs are used for drawing the secondary structure image.

5.2 Optimization of ZEM Algorithm

- Immediate time complexity of ZEM algorithm is: $O(n^4)$ and space complexity: $O(n^2)$.
- There are $O(n^2)$ matrix entries and $O(n^2)$ interior loops.
- We aim at reducing complexity of limiting case i.e. interior loops by bounding maximal interior loop size (e.g. 30).
- Then the time complexity of ZEM reduces to $O(n^3)$.

5.3 A Genetic Algorithm with Energy Minimization

Chen and coworkers have developed one a comparative sequence analysis algorithm that uses a very different approach to find common RNA secondary structures for a set of RNA sequences (Chen, et al., 2000). Their method is a “genetic algorithm”, which is intended to mimic genetic evolution. Genetic algorithms operate on a population of tentative solutions, each of which has an encoded representation equivalent to the genetic material of an individual in nature. The solutions are modified by mutation

(random changes) and crossover (recombination of features), and the modified solutions are selected by predefined fitness criteria, energy minimization in this case.

Unlike the previously discussed comparative sequence analysis methods, Chen's method does not require an alignment to determine a common structure for a series of RNA sequences. Both the structural energy and structural similarity among sequences of potential solutions are considered. Free energy is minimized by the nearest neighbor approach, with penalties or bonuses for other secondary structure elements (since the focus is on structural similarity, the free energy rules are not as complex as those for recursive algorithms, see below).

The genetic algorithm proceeds as follows. For each sequence, an initial population of *n* structures is generated. Crossover, mutation, and selection are iterated with free energy as the fitness criterion, until the stability criteria of the structures are reached. For each sequence, a conservation score *cons*(*Tp*) is evaluated for each structure *Tp* in the current generation of a sequence *Sp*; stem scores *cons*(*si*) are computed for each stem *si* in each structure *Tp*. Mutations and crossovers are then performed on the current generation. A total of 3*n* structures that satisfy *cons*(*T*) > *hc* and *e*(*T*) < *ec*, where *hc* is a conservation parameter and *e*(*T*) is the free energy of structure *T*, are collected from this iteration. The next generation for each sequence is then selected for by forming a set F from the unique 3*n* structures. A distance function *di* is calculated for each structure *Ti* in F. The distance function is defined as *di* = S*dij*, where *dij*, the distance score between structure *Ti* and *Tj*, is defined by *dij* = 1 – *nij*/*mij*, where *nij* is the number of base pairs in common between the two solutions and *mij* is the maximum number of base pairs of the two structures. The structures are then sorted by ascending values of *sc*(*i*) = (*best_fit* – *cons*(*Ti*)/*di*, and the top *n* structures are selected for the next iteration. After the maximum number of generations and the structures begin to converge, a structure *T'* is eliminated if: (1) *T'* is a substructure of some other structure *T*; (2) *e*(*T'*) < *e*(*T*); (3) *cons*(*T'*) < *cons*(*T*). In this way the most conserved structure with the lowest energy is generated. It is also possible to consider suboptimal structures.

Chapter 6

Parallel Algorithms

Several parallel approaches have been taken for RNA secondary structure prediction. Nakaya presented an approximation approach for generating secondary structures using minimum free energy criterion. The parallel approach enumerates all stacking regions of an RNA sequence and combines the ones which can coexist together to produce multiple secondary structures. Another parallel approximation approach by Taufer samples the RNA sequence systematically and extensively and rebuilds the whole structure by combining the structures of the chunks according to various criteria. However, these approximation approaches do not explore the entire search space and can miss the candidates that do not follow the usual behavior. Also, the success of these kinds of approaches is dependent upon the ability of rebuilding methods to identify motifs correctly by consistently combining the substructures into a full structure. Several distributed memory implementations for RNA secondary structure prediction have been developed which parallelizes the exact dynamic programming algorithm for free energy minimization.

Hofacker partitioned the triangular portion of 2D arrays into equal sectors that are calculated by different processors and data is reorganized after computing each diagonal in order to minimize the space requirements. In this implementation the tables are not stored permanently due to which trace back for all suboptimal secondary structures is not possible.

Fekete uses a similar technique to parallelize the folding procedure and increases communication to store the tables in order to facilitate the full trace back. However, these implementations may not be portable to current parallel computers and also the implementation of the optimized algorithms such as internal loop speedup algorithm whose access pattern differs from the general access pattern become complex for distributed memory environment. In [8], the authors observe that to fold the HIV virus, memory of 1 to 2GB is required, dictating the use distributed memory supercomputers; yet in our work, we demonstrate that this can now be solved efficiently on most personal computers. In our work, for the first time, we give scientists the ability to solve very large folding problems on their desktop by leveraging multicore computing.

Zhou and Lowenthal also studied a parallel, out-of-core distributed memory algorithm for the RNA secondary structure prediction problem including pseudoknots. However, their approach does not implement the full structure prediction but rather studies a synthetic data transformation that improves just one of the dependencies found in the full dynamic programming algorithm.

6.1 Parallelism

The dynamic programming algorithm is computationally intensive both in terms of running time and space. Its space requirements are of $O(n^2)$ as it uses four 2D arrays named $V(i, j)$, $VBI(i, j)$, $VM(i, j)$ and $WM(i, j)$ that are filled up during the algorithm's execution. The main issue is running time rather than memory requirements. For instance, GTfold has a memory footprint of less than 2GB (common in most desktop PCs) even for sequences with 10,000 nucleotides. The filled up arrays are traced in the backwards direction to determine the secondary structures.

The traceback for a single structure takes far less time than filling up these arrays. Time complexity of the dynamic programming algorithm is $O(n^3)$ with the currently adopted thermodynamic model. The two indices i and j are varied over the entire sequence, and every type of loop for every possible base pair (i, j) is calculated. This results in the asymptotic time complexity of $O(n^2)\times$ maximum time complexity of any type of loop for a base pair (i, j).

Computations of internal loops and multiloops are the most expensive parts of the algorithm. To avoid large running time, a commonly used heuristic is to limit the size of internal loops to a threshold k usually set as 30. This significantly reduces running time from $O(n^4)$ to $O(k^2n^2)$. The heuristic is adopted in most of the standard RNA folding programs. Lyngso suggests that the limit is a little bit small for predictions at higher temperatures and give an optimized and exact algorithm for internal loop calculations which has the time complexity of $O(n^3)$ with the same $O(n^2)$ space. The algorithm searches for all possible internal loops closed by base pair (i, j).

Practically, this algorithm is far slower than the heuristic. Choosing one of the options is a tradeoff of running time versus accuracy. In GTfold we provide an option for the user to select the heuristic or internal loop speedup algorithm. Also, our parallelization scheme is valid for both the options. Thermodynamics of multiloops are still not understood fully, but improvements continue to be made. Searching for an optimal multiloop closed by a base pair (i, j) requires searching for all enclosed base pairs which make the loop optimal. To make the multiloop energy function feasible to compute, it may be approximated in $O(n3)$ time. This function has linear dependence upon the number of single stranded bases in the multiloop. Time complexity of the algorithm to implement a relatively more realistic multiloop energy function having logarithmic dependence upon the single stranded bases in the loop is exponential. Also, many other advanced thermodynamic details such as coaxial dangling energies are not implemented in the multiloop energy calculations during the optimization, as it significantly increases the running time.

Both running time and space needs are expected to increase with the use of better thermodynamic models. While memory requirements can be satisfied with today's high-end servers with 256GB or more memory, running time will continue to play as a major prohibitive factor in solving these grand challenge problems. Our parallelization approach in GTfold is designed to keep all these factors in consideration.

6.2 GTfold

The prediction of the correct secondary structures of large RNAs is one of the unsolved challenges of computational molecular biology. Among the major obstacles is the fact that accurate calculations scale

as $O(n^4)$, so the computational requirements become prohibitive as the length increases. Existing folding programs implement heuristics and approximations to overcome these limitations. GTfold is a new parallel multicore and scalable program, which is one to two orders of magnitude faster than the de facto standard programs and achieves comparable accuracy of prediction. Development of GTfold opens up a new path for the algorithmic improvements and application of an improved thermodynamic model to increase the prediction accuracy.

GTfold now optimally folds 11 picornaviral RNA sequences ranging from 7100 to 8200 nucleotides in 8 minutes. GTfold is freely available as open source in websites.

According to the thermodynamic hypothesis, the structure having the minimum free energy (MFE) is predicted as the secondary structure of the molecule. The free energy of a secondary structure is the independent sum of the free energies of distinct substructures called loops. The optimization is performed using the dynamic programming algorithm given by Zuker and Stiegler in 1981 which is similar to the algorithm for sequence alignment but far more complex. The algorithm explores all the possibilities when computing the MFE structure. There are heuristics and approximations which have been applied to satisfy the computational requirements in the existing folding programs.

One potential approach to improve the accuracy of the predicted secondary structures is to implement advanced thermodynamic details and exact algorithms. However, while the incorporation of these improvements can significantly increase the accuracy of the prediction, it also drastically increases running time and space needs for the execution.

GTfold program runs one to two orders of magnitude faster than the current sequential programs for viral sequences on an IBM P5 570, 16 core dual CPU symmetric multiprocessor systems and achieves comparable accuracy in the prediction. We have parallelized the dynamic programming algorithm at a coarse-grain and the individual functions which calculate the free energy of various loops at a fine-grain. GTfold executes exact algorithms in an affordable amount of time for large RNA sequences. Our implementation includes an exact and optimized algorithm in place of the usually adopted heuristic option for internal loop calculations, the most significant part of the whole computation. GTfold takes just minutes (instead of 9 hours) to predict the structure of a *Homo sapiens* 23S ribosomal RNA sequence with 5,184 nucleotides.

Development of GTfold opens up the path for applying essential improvements in the prediction programs to increase the accuracy of the predicted structures. The algorithm has complicated data dependencies among various elements, including five different 2D arrays. The energy of the subsequences of equal length can be computed independently of each other without violating the dependencies pattern introduced by the dynamic programming with a set of five tables. Our approach calculates the optimal energy of the equal length sequences in parallel starting from the smallest to the largest subsequences and finally the optimal free energy of the full sequence. We also describe the nature of individual functions for calculating the energy of various loops and strategies for parallelization.

6.2.1 Dependencies and Access Patterns

Figure below shows a general *ij* plane. A valid base pair is defined as (*i, j*) where $j > i$. Thus, only the upper right triangle is valid. Secondary structures can have only nested base pairings, meaning if there

are two base pairs (*i, j*) and (*i', j')* such that $i < l' < j$ then the constraint $i < l' < j < j'$ is also satisfied. This assumption of nested base pairings results in the general dependency of an element (*i, j*) on the elements in the ΔABC as shown in Figure 6.1. To find the optimal loop formed by a base pair (*i, j*), we need to search for all enclosed base pairs over the subsequence from $i+1$ to $j-1$. In the case of internal loops we need to search for one enclosed base pair while for a multiloop we need to search for more than one base pair. In this fashion, the computation of all types of loops for an element (*i, j*) follows the above dependency pattern.

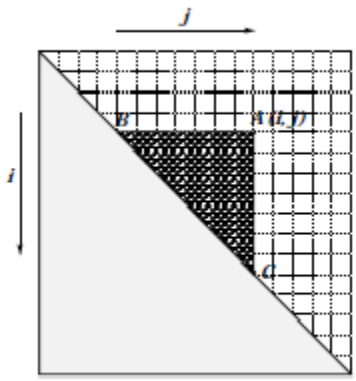

Figure 6.1: The implicit dependency of point A on the elements present in the triangle ABC

The speedup algorithm for internal loop calculations follows the same general technique but its access pattern differs. It updates the elements outside the dependency triangle Δ ABC shown in Figure 3 for the element A. It is an optimized algorithm to reduce the space complexity. We define a *small* internal loop as a loop in which at least one of the sides is less than a constant *c*. The algorithm scores these small internal loops as special cases by applying a function derived from the general internal loop energy function. If an enclosed base pair (*ip, jp*) is better than another candidate of the enclosed base pair (*ip*1*, jp*1) for the closing base pair (*i, j*) where $jp - ip = jp1 - ip1$, then the enclosed base pair (*ip, jp*) will also be better than (*ip*1*, jp*1) for the closing base pairs of the form $(i - b, j + b)$, where *b* is a positive integer such that $1 \leq b \leq min\{i - 1, N - j\}$. This algorithm uses the best enclosed base pair (*ip, jp*) for the closing base pair (*i, j*) to evaluate all internal loops closed by the base pairs of the form $(i - b, j + b)$ at the same time. This way, at element (*i, j*), the elements of the form $(i-b, j+b)$ are also accessed. The access pattern of this algorithm is shown in Figure 6.2 excluding the calculation of special cases.

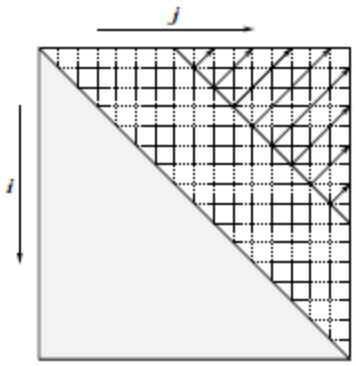

Figure 6.2: The access pattern of $VBI(i, j)$ for the internal loop speedup algorithm

6.2.2 Approach

In the region of the general 2D *ij* plane corresponding to $j > i$, a point (i, j) corresponds to the computation of energy of the subsequence from *i* to *j*. The dependency pattern shown in Figure 3 allows the calculation of all the elements existing on a line $j-i = k$ to be independent of each other, where *k* is any integer from the set $\{0, 1, 2, \ldots, N-1\}$. This way the computation on the line $j - i = k$ can be performed in parallel, and the whole space can be computed by considering subsequent lines from $k = 0$ to $k = N - 1$. Note that the points on one of the lines correspond to the equal length subsequences.

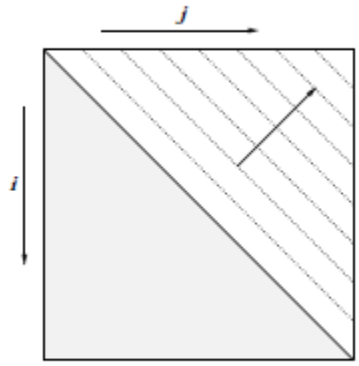

Figure 6.3: Showing the pattern of computation implemented in GTfold

```
input  : Sequence of Length N
output: Optimal Energy of the sequence
begin
    for b ← 0 to N − 1 do
        #pragma omp parallel for schedule (guided)
        for i ← 1 to N − b do
            j ← i + b;
            calcVBI(i, j);
            calcVM(i, j);
            calcV(i, j);
            calcWM(i, j);
        end
        calcW(b + 1);
    end
    return W(N);
end
```

Algorithm 6.1: Main function to compute the secondary structure of an RNA sequence

Algorithm 6.1 arranges the nested *for* loops to compute in the manner described above. The first *for* loop runs for different lines starting from $j = i$ to $j = i + N - 1$ and the second *for* loop calculates all the points on one line in parallel.

Figure 6.3 shows the sequence of these computations. This parallelization strategy is suitable for future improvements to the thermodynamic model or to optimizations for computing the various energy functions. This coarser level of parallelism enables us to exploit more concurrency while offering compatibility for possible future improvements. There are other orderings of the computation that cover the whole space without violating the dependency pattern. One way is to compute the elements column-wise, starting from $j = 1$ to $j = N$. On one column the computation is done for the increasing values of $j - i$, i.e. from row $i = j$ to row $i = 1$. A second way is to compute the elements row-wise, starting from $i = N$ to $i = 1$. On one row the computation is done for the increasing values of $j - i$, i.e. from column $j = i$ to column $j = N$. These two ways achieve a higher degree of spatial locality but they are inherently sequential.

6.2.3 Parallelism at individual functions

Parallelism can also be exploited at the finer level of individual functions which compute the energies for the various kinds of loops for a closing base pair (i, j). The general pattern of different functions for calculating the energy of these kinds of loops is the same except for the function that computes internal loop energies using the speedup algorithm.

The general pattern is to consider various possible options of the corresponding type of loop and select the option that gives the minimum energy. In simplified terms, this pattern of calculation performs minimization over several possible values. These types of calculations are easily done in parallel by assigning equal-sized chunks of minimization work to all threads, collecting the results, and taking the global minimum over all values.

The speedup algorithm for internal loop calculations can be parallelized only for special cases. The computations of general internal loops using the extension principle are inherently sequential. However the generally adopted heuristic option for internal loops has the general minimization pattern, is easily parallelizable, and uses two nested *for* loops which results in a complexity of $O(k^2)$ for a particular (i, j).

Multiloop calculations also follow the general minimization pattern for the *VM* and *WM* arrays, have $O(n)$ time complexity for an element (i, j), and are amenable to parallelization as well.

6.3 SPfold

There are many computations involved in finding the internal loops and multiloops in the proposed algorithm. They are the most expensive parts of the algorithm. This can be noticed from the Secondary Structure Prediction Algorithm described in previous section. In Eq. (3), in the calculation of $OEI(i, j)$, all possible internal loops with the closing base pair (i, j) are considered. This is done by varying indices i' and j' over the subsequence from $i + 1$ to $j - 1$ such that $i' < j'$. This results in the overall time complexity of $O(n4)$. To reduce the complexity we are using speculative parallelization technique, which converts the sequential loops to a form suitable for parallel execution. The code sections are aggressively executed in parallel with speculative parallelization.

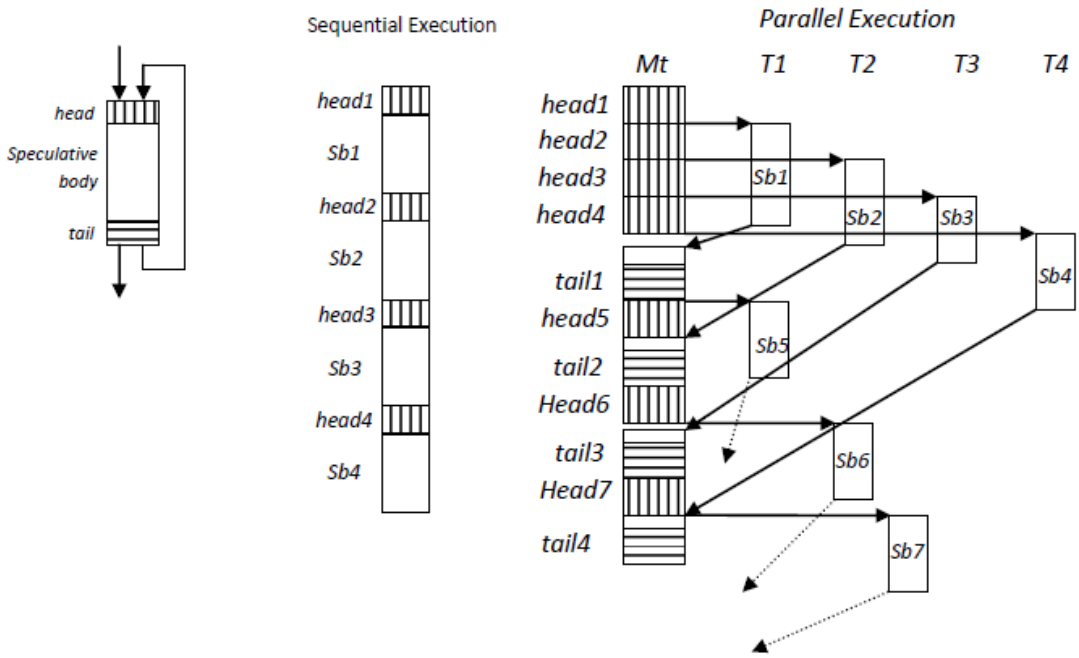

Figure 6.4: Speculative parallelization of loops

Given a series of dependent computation statements S1 → S2 →… Sn, S1 is scheduled to execute non-speculatively, and S2 through Sn are scheduled in parallel with S1 on additional cores. This type of scheduling of S2 through Sn is called as speculative parallelization. The execution model described above has been used to support speculative execution to execute the sequential RNA Secondary Structure Prediction program on multicore.

6.3.1 Approach

In speculative parallelization systems, speculative thread execution is achieved across several processing cores. These processing cores have private caches. For simplicity, assume that the system is only running the program of interest. The processing core running this sequential program guarantees its correctness. In our model, there are two states of computation of the program- the non-speculative state and the speculative state. These two states are maintained separately from each other.

The non-speculative state of the computation is maintained by the main thread of the parallelized application and multiple parallel threads execute parts of the computation using speculatively-read operand values from non-speculative state. This produces parts of the speculative state of the computation. The computed values are held until they are squashed when a dependency with the program is violated, or the final values are not affected. During the speculative execution of S', if a data value is read before it has been computed by S, the dependency is violated. Then mis-speculation occurs. Mis-speculation produces wrong results. So, the values computed during speculative execution of S' must be discarded and S' must be executed again. On the other hand, if mis-speculation does not occur, the execution time of the program is potentially reduced due to parallel execution of the consecutive statements. Of course, speculative execution is only beneficial if the mis-speculation does not occur frequently, committed and used when they are requested by the (non-speculative) program or by other speculative threads.

During the execution of speculative thread, the changes made by stores are hold separately, usually in the private cache of the processing core on which the thread is executing, and providing them to dependent loads.

The program locations accessed and modified by the thread are tagged into write and read sets, respectively, along with the modified architected registers. These are held in private caches or auxiliary storage structures until the speculative thread is invalidated or committed. If the speculative thread had referenced a location during its execution that is modified by the program, it indicates violation of data dependencies. Then, the speculative thread is squashed and its data is discarded [11]. This way of scheduling ensures the isolation of execution of each thread, main or parallel, from execution of all other threads. In case of mis-speculation, the re-execution of other parallel threads is not required.

Suppose, we have the computing statements, S →S' which are dependent. We speculate that no dependencies exist from S to S'. We start speculative execution of S' in parallel with S. We are speculating that the variables being used by speculative threads S' will not be modified by S. At the end of the execution of S', we must check to see if S modified any of the variables that were speculatively read by S'. If this is indeed the case, mis-speculation occurs and recovery is performed by re-executing S'. Otherwise speculation is successful and the results computed by S' are copied back to non-speculative state.

Consider a series of dependent computations S1 →S2 →…Sn such that, while S1 executes non-speculatively, if we speculatively execute S2 through Sn in parallel with S1, then the results of S2 through Sn must be copied to the non-speculative state in-order. This will ensure that if any dependences arise between Si and Sj (for all i < j), they are correctly enforced.

Chapter 7

A Critical Review of Computational Methods for RNA Secondary Structure Prediction

Most RNA secondary structure prediction algorithms perform thermodynamic optimization on a series of plausible structures in order to obtain the structure or structures with the lowest equilibrium free energy. Basic thermodynamic principles indicate that the structure lowest in free energy should be the most stable and barring outside influences, the correct fold. Unfortunately, not all of the factors that determine the energy of a fold are understood, and computational time limitations would make it infeasible to include all of such influences. Current methods ignore free energy contributions from tertiary structure, a reasonable assumption because the forces determining tertiary structure are weaker than those governing secondary. However, tertiary structure contributions may play an increasing role as the length of the RNA strand is increased, because more complex folds are possible. The values for such contributions have not been determined

A major problem with early comparative sequence analysis algorithms was that phylogenic relationships of the aligned sequences and levels of sequence divergence were not considered. One way this problem has been overcome is by generating a pairing parameter l which measures the pattern of nucleotide substitution at paired sites versus those at unpaired sites in order to make quantitative comparisons in evolutionary conservation (Muse, 1995) . This “likelihood-ratio test” (LRT) approach has the severe drawback that its statistical significance is questionable for helices less than 10 base pairs in length (Parsch, et al., 2000). Numerical and probabilistic models can be made to help overcome this problem, however. In Parsch’s algorithm, a complete list of potential RNA helices, along with their l values are generated. Compatible helices are then grouped into subsets, which are combined to form potential secondary structure models. For each set of helices, a total l value is determined by summing the l values for each individual sequence; the optimal structure is the one with the greatest total l value.

Unlike comparative sequence analysis algorithms, which find common structures for a group of sequences, recursive algorithms find optimal structures for a single sequence. These programs typically rely exclusively on free energy minimization for determining the best structures, so the thermodynamic parameters used are of critical importance. Early recursive algorithms were problematic in many regards. The slow speed of computers in the early 1980s was perhaps the biggest setback, but thermodynamic parameters were not very accurate, nor were they always correctly incorporated into algorithms. Thermodynamic data incorporated into the first RNA secondary structure prediction algorithms was thought to have an uncertainty of ±0.2 to 0.5 kcal for basepaired helical regions and ±1 to 2 kcal for loops (Williams and Tinoco, 1986). In some programs, the free energy minimization was designed to make the program obtain results consistent with experiments.

The most reliable method for determining the conserved structure of a series of RNA sequences is to combine comparative sequence analysis with free energy minimization. The Dynalign algorithm is one of the most successful for determining a common structure for two phylogenically related sequences (Mathews and Turner, 2002). It is a dynamic algorithm that aligns two sequences and finds a common structure. It uses thermodynamic parameters described above for the MFOLD algorithm for prediction of free energies. This approach has a number of advantages over genetic algorithms or algorithms based on free energy minimization alone. Though it combines many of the advantages of these methods and eliminates some of the shortcomings, Dynalign is not without its own limitations. Like algorithms based on free energy minimization alone, the ultimate accuracy of Dynalign's structures is dependent on the quality of the thermodynamic parameters. Unlike genetic algorithms, Dynalign is unable to align and determine the structure for more than two common sequences. This is because the dimensionality of the energy functions is equal to the square of the number of sequences being aligned. The algorithm finds the common structure of two sequences using four-dimensional energy functions; to find the structure for three or four sequences it would need to use nine- or sixteen-dimensional functions. It is unlikely that computers will be fast enough or have enough memory to handle this kind of data any time soon.

The best current RNA secondary structure prediction algorithms are around 80% accurate for structural RNAs with known structural information. There are many ways that current algorithms could be improved. I would suggest two very generic improvements that could have very positive impacts on most algorithm types. The issue of accuracy of thermodynamic parameters for free energy minimization algorithms has already been discussed extensively. As mentioned, current thermodynamic parameters were obtained by performing melting studies on small oligonucleotides, but the stabilizing or destabilizing effects of longer sequences are unknown. For example, it is unknown whether there is inherent additional stability for a hairpin containing a 20 base pair stem versus a 5 base pair stem. Modern automated synthesis methods have made it possible to synthesize fairly long (>50 nucleotide) RNAs in modest yields, so experiments of this sort are easily performed. More extensive melting studies, especially on longer loops, to determine additional stabilization values for hairpins and destabilization values for bulges and internal loops, could be very helpful to improve current algorithms. An algorithm that could iteratively consider the effects of increasing loop size in a sequence-specific manner would have a great deal of utility.

Secondly, most current algorithms do not allow input of information obtained from experimental data. For example, biochemical experiments may show that certain sets of bases in an RNA are paired. Rather than simply using this information to confirm the structure, this information could be used to bias the structure. Such restraints could drastically reduce computational time, since the number of possible structures would be greatly reduced. Analogously, a useful additional feature on algorithms that use phylogenic information to determine common structures would be to allow the user to input phylogenic restraints based on known relationships. Even better, the algorithms could be constructed with a database of known phylogenies. This could also drastically reduce computational time, though it runs the risk of missing unexpected relationships.

Chapter 8

Conclusion

RNA molecules serve not only as carriers of information, but also as functionally active units. The three dimensional shape of tRNA molecules plays a crucial role in the process of protein synthesis. RNA is known to exhibit catalytic activity. RNA served both as carrier of genetic information as well as catalytically active substance. RNA may not necessarily have been the first step in prebiotic evolution, but the idea that RNA preceded not only DNA, but also the invention of the translational system, seems widely accepted. Furthermore, RNA provides an ideal, currently the only, system to study genotype-phenotype relationships. Although RNA offers a limited repertoire of catalytic functions, ribozymes gain importance for biotechnological applications, since these molecules are suited for irrational design: Large scale synthesis of RNA molecules underlying mutation and selection experiments, in which the ribozymes are screened for positive catalytic functions, are spreading in use.

RNA secondary structures provide a useful, though coarse grained, description of RNA structure. In many biologically evolved RNA molecules such as viral genomes and tRNA, the secondary structure seems to be more conserved than the sequence. Viruses belonging to the same family show little sequence similarity, yet exhibit strongly conserved secondary structure motifs, e.g. in terminal non-coding regions.

RNA structures play a significant role in a wide range of problems today. The secondary structures provide a convenient way of coarse graining, and their study yields important information about RNA useful in the prediction of the full 3D structures and in the interpretation of the biochemical function either. Secondary structures are discrete and therefore well suited for computational methods.

Most RNA secondary structure prediction algorithms perform thermodynamic optimization on a series of plausible structures in order to obtain the structure or structures with the lowest equilibrium free energy. Detecting RNA secondary structure is an NP-hard problem, especially in pseudo-knotted RNA structures. The detection process is also time-consuming; as a result, an alternative approach such as using parallel architectures is a desirable option.

Chapter 9

References

[1] Amrita Mathuriya, David A. Bader, Christine E. Heitsch, Stephen C. Harvey, *GTfold: A Scalable Multicore Code for RNA Secondary Structure Prediction,* Georgia Institute of Technology Atlanta, GA, USA.

[2] Yuzhen Ye, *RNA folding, RNA secondary structure prediction by dynamic programming algorithms*, School of Informatics, Indiana University.

[3] Martin Fekete, *Prediction of RNA Secondary Structures Using Parallel Computers.*

[4] Narendra Chaudhary, Anjali Mahajan, *SP-fold – Speculative Parallelization for Parallel Algorithm of RNA Secondary Structure Prediction on Multicore,* Priyadarshini Institute of Engineering & Technology, Nagpur, India.

[5] Adam Silverman, *A Critical Review of Computational Methods for RNA Secondary Structure Prediction.*

[6] Mirela Stefania Andronescu, *Algorithms for predicting the secondary structure of pairs and combinatorial sets of nucleic acid strands,* M.Sc., Academy of Economic Studies, Bucharest, Romania, 2000

[7] Angela Brooks, *Molecular Biology Primer.*

[8] Cinita Mary Mathew and G.H. Meera Krishna, *Prediction of RNA Secondary Structure from Random Sequences Using ZEM.*