PREDICTION OF SECONDARY STRUCTURES FOR LARGE RNA MOLECULES

A Thesis
Presented to
The Academic Faculty

by

Amrita Mathuriya

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
Computer Science

Georgia Institute of Technology
May, 2009

PREDICTION OF SECONDARY STRUCTURES FOR LARGE RNA MOLECULES

Approved by:

Professor David A. Bader, Advisor
College of Computing
Georgia Institute of Technology

Professor Christine E. Heitsch, Co-Advisor
School of Mathematics
Georgia Institute of Technology

Professor Richard Vuduc
College of Computing
Georgia Institute of Technology

Professor Stephen C. Harvey
School of Biology
Georgia Institute of Technology

Date Approved: December 19, 2008

To My Family.

ACKNOWLEDGEMENTS

I thank my advisor Professor David A. Bader for his helpful support and guidance in research and course work throughout my MSCS degree. I am deeply grateful to him for providing the great opportunity of working in High Performance Computing area and supporting financially from the beginning to the end, inspite of being a Masters student, so that I could focus on the research work entirely. He always encouraged me, either be it a difficult situation or an accomplishment. I am thankful to him for developing my continued interest in this interdisciplinary area of research. He will continue be an ideal role model for my professional career.

I would like to thank my Co-advisor Professor Christine E. Heitsch for guiding me throughout the entire course of this research. I am grateful to her for helping me in developing the understanding of a new interdisciplinary research area. She is the principal investigator of the NIH grant supporting this work. She devoted significant time for discussing the difficulties, I faced during the work. Her course on "Discrete Mathematical Biology" introduced me to the mathematical aspects of this work and also to the other computationally and mathematically challenging biological problems.

I am thankful to Professor Stephen C. Harvey for introducing me to the biological aspects of the RNA molecules and their secondary structures. His inputs were really helpful for making the work useful for biological community. Discussions with him provided me the most fruitful directions for proceeding ahead. I am grateful to him for being always available for the discussions.

David, Christine and Steve play a very important role in establishing this work as a successful research and it was not possible to have this work done without their

support and guidance.

I am grateful to Professor Richard Vuduc for discussing various approaches for parallelization of GTfold and serving as a committee member. The discussion and his guidance were very helpful in making progress. His course on "Parallel Numerical Algorithms" increased my understanding of handling various issues involved in parallelization.

I would like to express my gratitude towards students in the RNA research group Minmin Pan, Yingying Zeng, Emily Rogers, Swathi Bhat and Mustafa Burak Boz for listening to my presentations and giving comments to improve the work. Discussions in the research group on various aspects of the problem contributed to this research with significant inputs.

Last but not the least, I thank my wonderful friends Asad Malik, Asha Sharma and Fabio Cunial, summer interns from IIT Roorkee Nitesh Agrawal and Manoj Soni, and my lab mates Aparna Chandramowlishwaran, David Ediger, Kamesh Madduri, Karl Jiang, Manisha Gajbe, Seunghwa Kang, Swathi Bhat and Virat Agarwal for discussing research related questions and making my stay memorable in Atlanta. I thank my parents, sisters Anjali and Akanksha and brothers Akash and Aryan for always giving me strength and support to proceed in any kind of difficult situations. In the end, I forward my greatest regards to God.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The prediction of 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. We present a new parallel multicore and scalable program called GTfold, which is one to two orders of magnitude faster than the de facto standard programs mfold and RNAfold for folding large RNA viral sequences and achieves comparable accuracy of prediction. We analyze the algorithm's concurrency and describe the parallelism for a shared memory environment such as a symmetric multiprocessor or multicore chip. We are seeing a paradigm shift to multicore chips and parallelism must be explicitly addressed to continue gaining performance with each new generation of systems.

We provide a rigorous proof of correctness of an optimized algorithm for internal loop calculations called internal loop speedup algorithm (ILSA), which reduces the time complexity of internal loop computations from $O(n^4)$ to $O(n^3)$ and show that the exact algorithms such as ILSA are executed with our method in affordable amount of time. The proof gives insight into solving these kinds of combinatorial problems. We have documented detailed pseudocode of the algorithm for predicting minimum free energy secondary structures which provides a base to implement future algorithmic improvements and improved thermodynamic model in GTfold. GTfold is written in C/C++ and freely available as open source from our website.

CHAPTER I

INTRODUCTION

RNA molecules perform a variety of different biological functions including the role of "small" RNAs (with tens or a few hundred of nucleotides) in gene splicing, editing, and regulation. At the other end of the size spectrum, the genomes of numerous viruses are lengthy single-stranded RNA sequences with many thousands of nucleotides. These single-stranded RNA sequences base pair to form secondary and tertiary structures. Secondary structures of viruses like dengue [5], ebola [29], and HIV [30] are known to have functional significance. Thus, predicting correct secondary structures, and identifying and disrupting functionally significant base pairings in RNA viral genomes becomes a potential method for treating or preventing many RNA-related diseases.

RNA folding is different than DNA folding, in which DNA molecule forms a double stranded helix, whereas RNA molecule remains single stranded and folds in it to have structural forms called secondary and tertiary structures. RNA folding is also very different than protein folding, as RNA molecules have only four kinds of nucleotides, while protein molecules are formed of 20 different amino acids. In comparison to protein folding, the secondary structural elements of RNAs can be separated from tertiary interactions [28] and secondary structures can be helpful for various purposes such as recognizing functionally significant base pairings, predicting tertiary structures etc.

Figure 1 shows the 3D structure of pariacoto virus [26]. Though, the virus is known to form a dodecahedral cage in the views identified using crystallography, but how the RNA genomes reside into the cage is not yet known. Experimental methods for finding out the structures of RNA molecules are too expensive and time taking and

therefore, the computational methods to predict secondary structures are required. Comparative sequence analysis [9, 10] is a computational method for determining secondary structures which predicts highly accurate structures and whose accuracy has been proven using high resolution crystal structures. However, the method needs large datasets for finding a consensus alignment to predict secondary structures. The applicability of this method is limited by the available datasets for many classes of RNAs and other computational methods which predict secondary structures using a single sequence are applied in this situation.

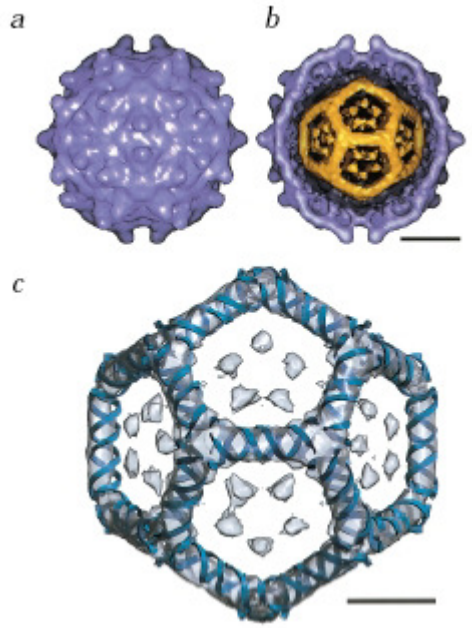

Figure 1: Showing the 3D structure of Pariacoto virus [26].

1.1 RNA Secondary Structure Prediction using Free Energy Minimization

Viral sequences range in length from about 1,000 to over 1,000,000 nucleotides in the recently discovered virophage. Length of the viral sequences poses significant computational challenges for the current computer programs. Free energy minimization excluding pseudoknots is a conventional approach for predicting secondary structures

Figure 2: The optimal secondary structure of an HIV-1 virus with 9,781 nucleotides predicted using GTfold in 84 seconds using 16 dual core CPUs. The minimum free energy of the structure is -2,879.20 Kcal/mole.

from a given sequence. The mfold [19, 37] and RNAfold [13] programs are the standard programs used by the molecular biology community for the last several decades. Recently, other folding programs such as simfold [1] have been developed. These programs predict structures with good accuracy for the RNA molecules having fewer than 1,000 nucleotides. However, for longer RNA molecules, prediction accuracy is very low [7].

According to the thermodynamic hypothesis, the structure having the minimum free energy (MFE) is predicted as the secondary structure of the molecule. The optimization is performed using the dynamic programming algorithm given by Zuker and Stiegler in 1981 [39] which explores the entire search space and finds out the MFE structure. One of the reasons for lower accuracies of the predicted secondary structures as pointed by Mathews and Turner in [17], is the approximations involved in structure prediction algorithms. For example, currently available software mfold, RNAfold and simfold adopt a heuristic option of limiting the size of internal loops to a constant, and simplified energy function for multiloops, to avoid huge computational requirements. While, the incorporation of exact algorithms and advanced thermodynamic model has potential to increase the accuracy of the predicted structures, it also drastically increases running time and space needs for the execution.

1.2 Our Contribution

Current programs use heuristics and approximations to satisfy the computational requirements. We use shared memory parallelism to overcome the computational challenges of the problem. We have designed and implemented a new parallel and scalable program called GTfold for predicting secondary structures of RNA sequences. Our program runs one to two orders of magnitude faster than the current sequential programs for large viral sequences on an IBM P5 570, 16 core dual CPU symmetric multiprocessor system. Figure 2 shows the optimal secondary structure obtained from

GTfold of an HIV-1 sequence (accession number Z11530) having 9,781 nucleotides executed on the system with 32 threads. Structures predicted with GTfold achieves accuracy comparable to the structures predicted with RNAfold and UNAfold which supersedes mfold, for a diverse set of ribosomal RNA sequences having known structures found by more reliable method comparative sequence analysis [9, 10].

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 provides an option for internal loop calculations to select from internal loop speedup algorithm (ILSA) or heuristic options. We demonstrate that GTfold executes exact algorithms in an affordable amount of time for large RNA sequences. GTfold takes just minutes (instead of 9 hours) to predict the structure of a *Homo sapiens* 23S ribosomal RNA sequence with 5,184 nucleotides with the ILSA option. 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. Development of GTfold opens up the path for applying essential improvements in the prediction programs to increase the accuracy of the predicted structures.

The minimization recursion formulas describing the dynamic programming algorithm have been mentioned at various places [2, 16, 18, 19]. Hofacker *et al.* [13] described a brief pseudocode of the algorithm for predicting MFE structures. In this thesis, we document the entire pseudocode of the algorithm which includes thermodynamic details. Pseudocode provided here gives the complete picture of the algorithm and serves as a base for doing performance improvements and incorporating advanced thermodynamic model in GTfold.

Currently, internal loop calculations are the most time consuming part of the whole computation. The naive way of iterating over all possible internal loops to find out the optimal one for every closing base pair has $O(n^4)$ time and $O(n^2)$ space complexities. The optimized algorithm ILSA reduces the time complexity from $O(n^4)$ to $O(n^3)$ with the same space requirements. Lyngsø *et al.* gave an intuitive proof of the correctness for the algorithm [15] by first simplifying the algorithm to be implemented in $O(n^3)$ space and then arguing that the simplified algorithm is same as the speedup algorithm except for the order in which array elements were computed.

In the thesis, we analyze the algorithm in a simplified manner giving the pseudocode and providing a rigorous mathematical proof of the correctness for the algorithm. We explain the algorithm by introducing a concept of gap length which is equal to the length of the subsequence closed by the enclosed base pair. Describing the algorithm with the variable gap instead of the length of internal loops simplifies the explanation of pseudocode and helps describing the proof in a clear way. Our proof starts by sketching the graphical regions in the 2D space of $i_p - j_p$ where (i_p, j_p) is the enclosed base pair, for an arbitrary closing base pair (i, j) showing the special cases which are to be taken care of separately and the region where the extension principle can be applied. We argue that with the algorithm we cover all gap length values for every closing base pair and then for every gap length, we cover all possible enclosed base pairs (i_p, j_p). The proof gives us insight into solving such kinds of algorithmic problems combinatorial in nature and motivates us to do same type of algorithmic improvements for multiloop energy calculations.

CHAPTER II

PREVIOUS WORK

Several parallel and sequential approaches have been taken for RNA secondary structure prediction. There are approaches such as Pfold [14] using stochastic context-free grammars which take many related sequences as input for the prediction. Our focus is on the prediction approaches which predict structures from a single sequence. An another approach contrafold [6] predicts secondary structures using learned thermodynamic parameters from the database of known secondary structures instead of experimentally determined physics based parameters. The approach has outperformed MFE prediction methods for single structure prediction accuracy. However, the parameters learned using the known secondary structures of ribosomal RNAs or other classes may not be suitable for the unrelated class of viral sequences.

Nakaya *et al.* [20] presented an approximation algorithm for generating secondary structures with the 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 approximation algorithm by Taufer *et al.* [27] samples the RNA sequence systematically and extensively, and rebuilds the whole structure by combining the structures of the chunks according to various criteria. Statistical approaches such as RDfolder [35, 36] which builds the secondary structure by combining helical regions based upon probabilistic criteria have also been applied. However, most of these approaches assume the minimum length of a helix equal to three to avoid combinatorial explosion while the helices of length one and two are observed in the real structures. Also, the applicability of all these approaches is limited to sequences shorter than thousand nucleotides.

Evolutionary algorithms(EAs) have also been applied for predicting RNA secondary structures [11, 25, 31, 32, 33]. EAs are in general easily parallelizable. Shapiro *et al.* in 2000 [25] applied massively parallel genetic algorithm for RNA secondary structure prediction on modern workstations. EAs do not explore entire possibilities and may not be able to find the optimal secondary structures. Also, solution quality depends upon various parameters selected for crossover and mutation operators. Performance results of run time and accuracy of these algorithms are presented only for sequences shorter than thousand nucleotides [25, 32, 33, 31]. Here, we are interested in the exact optimization problem of finding the minimum free energy secondary structures of RNA molecules. Exact optimization allows us to explore the entire suboptimal space within a specified energy range during the traceback. In the experimental section we will compare the accuracy of the evolutionary algorithms with GTfold.

Several distributed memory implementations [4, 8, 12, 13] for RNA secondary structure prediction have been developed which parallelize the exact dynamic programming algorithm. Hofacker *et al.* [12, 13] partition the triangular portion of 2D arrays into equal sectors that are calculated by different processors in order to minimize the space requirements and data is reorganized after computing each diagonal. In this implementation the arrays are not stored permanently and because of this, traceback for all suboptimal secondary structures is not possible. Fekete et al. [8] uses a similar technique to parallelize the folding procedure and increases the communication to store the arrays in order to facilitate the full traceback. 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.

Zhou and Lowenthal studied a parallel out of core distributed memory algorithm

for RNA Secondary structure prediction problem including pseudoknots. However the underlying dynamic programming algorithm for secondary structure prediction involves working only with a similar single data dependency pattern in comparison to the complex dependency pattern of the free energy minimization approach and also computational requirements of the two problems are different. In [12], 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.

CHAPTER III

BACKGROUND

3.1 RNA Secondary Structures

RNA molecules are made up of A, C, G, and U, nucleotides which can pair up according to the rules in {(A,U), (U,A), (G,C), (C,G), (G,U), (U,G)}. Nested base pairings of an RNA sequence can be presented in a 2D plane, which is called secondary structure. We take care of only nested base pairings and pseudoknots are not allowed in our model. Pairings among bases form various kinds of loops, which are classified based on the number of branches present in them. Nearest neighbor thermodynamic model (NNTM) provides a set of functions and sequence dependent parameters to calculate the energy of various kinds of loops. The free energy of a secondary structure is calculated by adding up the energy of all loops and stacking present in the structure.

Figure 3 shows an MFE secondary structure predicted using GTfold of an artificial sequence of 79 nucleotides. Various loops annotated in the figure are named as hairpin loops, internal loops, multiloops, stacks, bulges and external loops. Loops formed with two consecutive base pairs are called stacks. Loops having one enclosed base pair and one closing base pair are called internal or interior loops. Internal loops with length of one side as zero are called bulges. Loops with two or more enclosed base pairs and one closing base pair are called multiloops or multibranched loops. The open loop which is not closed by any base pair is called an external or exterior loop. Structures having more than one base pairs present in the external loop are called multidomain structures.

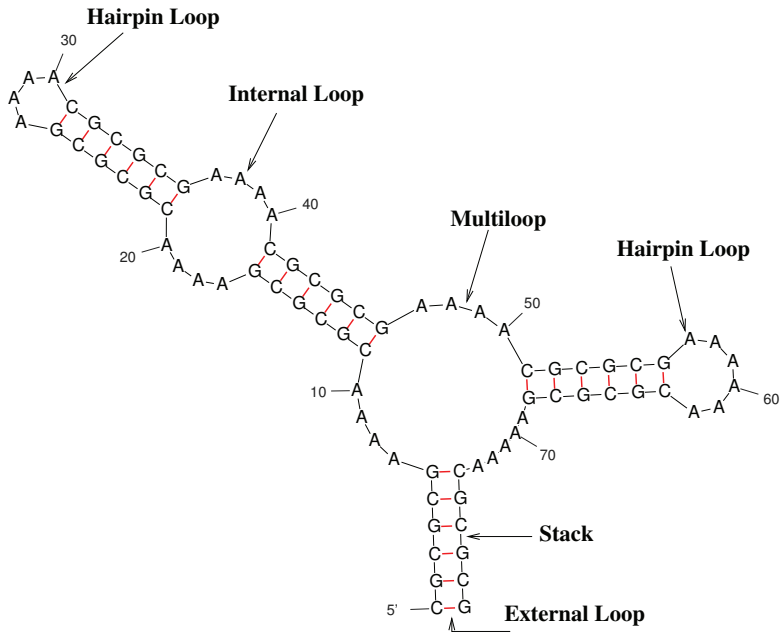

Figure 3: A sample RNA secondary structure with 79 nucleotides.

3.2 Thermodynamic Prediction Algorithm

Prediction of secondary structures with the free energy minimization is an optimization problem like the Smith-Waterman local alignment algorithm. There is a well-defined scoring function which can be optimized via dynamic programming, and structures achieving the optimum can be found through traceback. However, while sequence alignment can be performed with one table and a relatively simple processing order, RNA secondary structure prediction requires five tables with complex dependencies. Each class of loop has a different energy function which is dependent upon the sequence and parameters. For the internal loops and multiloops with one or more branches, all enclosed base pairs need to be searched which makes the loop optimal for the closing base pair.

The algorithm can be defined with recursive minimization formulas. Lyngsø *et al.* [16] described simplified recursion formulas which are reproduced here for convenience. Consider an RNA sequence $S = s_1 s_2 \dots s_N$ and free energy of the subsequence $s_1 s_2 \dots s_j$ to be $W(j)$. Note that the $W(N)$ is the free energy of S and $W(j)$ is given

by the following formula:

$$W(j) = \min\{W(j-1), \min_{1 \le i < j}\{V(i,j) + W(i-1)\}\} \tag{1}$$

In Eq. (1), $V(i,j)$ is the optimal energy of the subsequence $s_i s_{i+1} \dots s_j$ assuming (i,j) forms a base pair and is defined by

$$V(i,j) = \min \begin{cases} eH(i,j) \\ eS(i,j) + V(i+1,j-1) \\ VBI(i,j) \\ VM(i,j). \end{cases} \tag{2}$$

Eq. (2) considers various types of loops that a base pair (i,j) can close. The $eH(i,j)$ function returns the energy of a hairpin loop closed by base pair (i,j). Function $eS(i,j)$ returns the energy of a stack formed by base pairs (i,j) and $(i+1,j-1)$. $VBI(i,j)$ and $VM(i,j)$ are the optimal free energies of the subsequence $s_i s_{i+1} \dots s_j$ in the case when the (i,j) base pair closes an internal loop or a multiloop, respectively.

$$VBI(i,j) = \min_{i < i' < j' < j}\{eL(i,j,i',j') + V(i',j')\} \tag{3}$$

where, $i' - i + j - j' - 2 > 0$. We consider bulge loops as the special cases of internal loops so, the function $eL(i,j,i',j')$ also takes care of these. The formulation of the multiloop energy function has linear dependence upon the number of single stranded bases present in the loop. The standard is to introduce a 2D array WM to facilitate the calculation of VM array. Eq. (4) and (5) shows calculations of $WM(i,j)$ and $VM(i,j)$ respectively.

$$WM(i,j) = \min \begin{cases} V(i,j) + b \\ WM(i,j-1) + c \\ WM(i+1,j) + c \\ min_{i<k\le j}\{WM(i,k-1) + WM(k,j)\} \end{cases} \tag{4}$$

$$VM(i,j) = \min_{i+1<k\leq j-1}\{WM(i+1,k-1) + WM(k,j-1) + a\} \quad (5)$$

3.3 Complexity Analysis and 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 memory footprints of less than 2GB (common in most desktop PCs) even for sequences with 10,000 nucleotides.

The arrays filled up using dynamic programming are traced in the backward 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. We can see from Eq. (3) that, in the calculation of $VBI(i,j)$, all possible internal loops with the closing base pair (i,j) are considered 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 $O(n^4)$. 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.

Lyngsø *et al.* [16, 15] suggest 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.

The 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 $O(n^3)$ 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 strategy in GTfold is designed for reducing the running time and it takes the same amount of space as the sequential algorithm.

CHAPTER IV

PSEUDOCODE

The RNA secondary structure prediction algorithm with free energy minimization is composed of two steps. The fill step is the dynamic programming algorithm given by Zuker and Stiegler [39] which finds out the optimal energy score that can be achieved by a possible secondary structure. The traceback step finds out the optimal structure corresponding to the score. Note that there may be one or more possible secondary structures having the same energy score depending upon the sequence contents and thermodynamic parameters. Here we give a detailed pseudocode of the entire algorithm implemented in GTfold for the fill step. Pseudocode of the GTfold algorithm does not implement coaxial stacking energies. The formulas including coaxial stacking energies for multiloops are presented by Mathews *et al.* [18] in the supporting information. The algorithm for optimal and complete suboptimal traceback is described by Wuchty et al. [34].

The minimization formulas presented in section 3.2 can be implemented recursively as well as iteratively. We implement an iterative formulation of the algorithm. The implementation uses various 1D and 2D arrays corresponding to $W(j)$ and $V(i,j)$, $VBI(i,j)$, $VM(i,j)$, $WM(i,j)$ values. Also we use $calcW(j)$, $calcV(i,j)$, $calcVBI(i,j)$, $calcVM(i,j)$ and $calcWM(i,j)$ functions to calculate the values of $W(j)$, $V(i,j)$, $VBI(i,j)$, $VM(i,j)$ and $WM(i,j)$ array elements. The MFE score of the whole sequence is computed using the function *calculate*() shown in algorithm 1 with the sequence length as input argument. Computation of other functions $calcVBI(i,j)$, $calcWM(i,j)$, $calcVM(i,j)$, $calcV(i,j)$ and $calcW(j)$ is performed as shown in algorithms 2, 3, 4, 5 and 6 respectively. Also pseudocode for internal loop

calculations using the heuristic option is given algorithm 7 and using the speedup algorithm is given in algorithms 8, 9 and 10.

We assume that the free energy of an unfolded sequence is infinity which is represented by a large positive constant INF and all array elements are initialized with INF. In the starting, $readEnergyTables()$ and $readSequence()$ functions read the thermodynamic parameters and sequence as inputs. The function $auPen(i, j)$ returns a constant penalty, if (i, j) is not a G-C or C-G pair otherwise, it returns zero. The function $dangle-5'(i, j, i+1)$ function returns the dangling interaction energy of the single stranded nucleotide s_{i+1} with (i, j) base pair assuming that the nucleotide s_{i+1} is at the 5' end of the base pair. Similarly, $dangle-3'(i, j, j-1)$ returns the dangling interaction energy of single stranded nucleotide s_{j-1} with (i, j) base pair assuming nucleotide s_{j-1} to be at the 3' end of the base pair. Please refer to the practical guide by Zuker *et al.* [38] for the details of functions $eH(i, j), eL(i, j), eS(i, j)$.

One trickier part of including thermodynamic details in the simplified formulas is to include dangling interaction energies. In case of multiloops and external loop, dangling energies are included for the interaction among the base pairs and the adjacent single stranded nucleotides in the loop if they are present. If there is only one single stranded nucleotide between two base pairs then only the lower energy contribution (more negative) is added, i.e. the dangling energy contribution of the single nucleotide is added only for the base pair which contributes lesser free energy. Multiloop contained in the secondary structure shown in Figure 4 have three cases of including dangling energies where 0, 1 and 2 single stranded nucleotides are present between two base pairs. For the base pair $(5, 29)$ nucleotide s_{28} is paired and nucleotide s_6 is unpaired. There are two single stranded nucleotide s_6 and s_7 between base pairs $(8, 18)$ and $(5, 29)$. Therefore, for the base pair (5,29) the dangling energy contribution is added for nucleotide s_6 and not for nucleotide s_{28}.

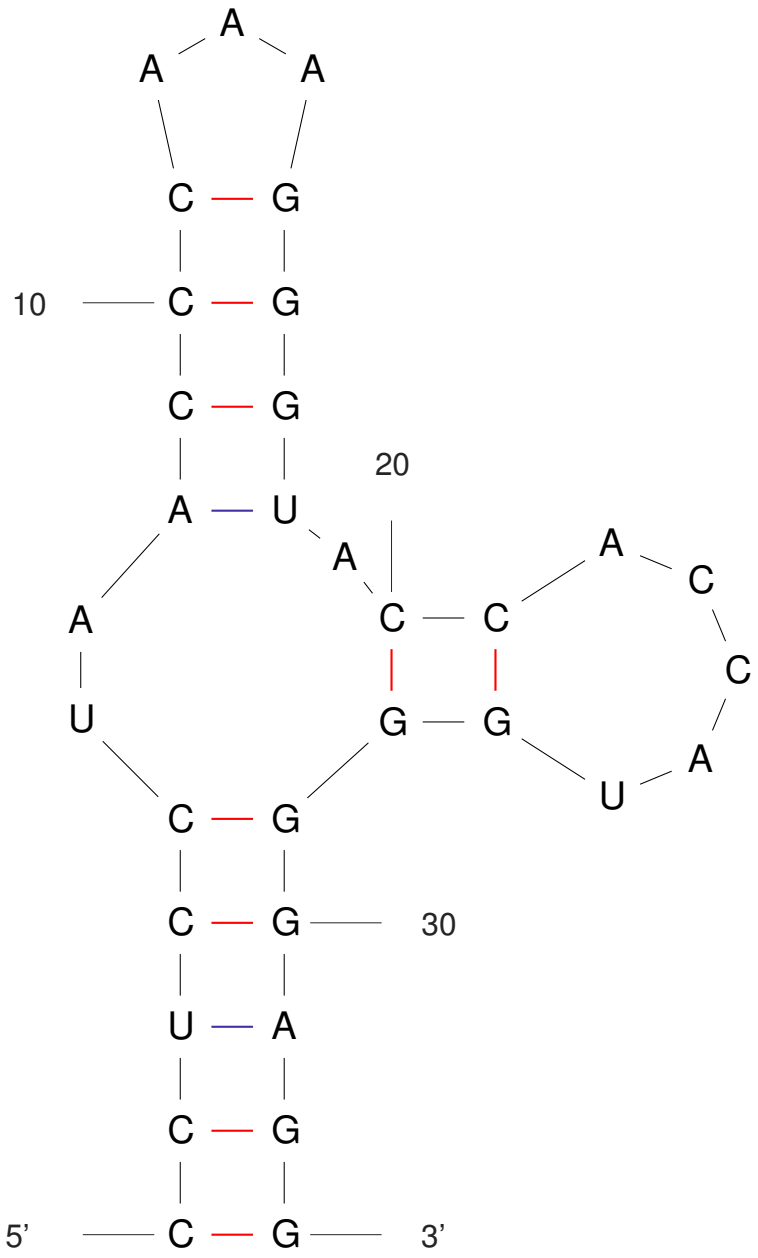

Figure 4: Showing various cases of including dangling interaction energy.

Base pair $(8, 18)$ has single stranded nucleotides present at both the ends. Dangling energy of base pair $(8, 18)$ with single stranded nucleotide s_7 is added for sure and dangling energy of base pair $(8, 18)$ with single stranded nucleotide s_{19} is added only in the case if this is more negative than the dangling energy of base pair $(20, 28)$ with single stranded nucleotide s_{19}. Base pair $(20, 28)$ have single stranded nucleotide only at one side and the corresponding dangling energy will be taken care of similar to the case of base pair $(8, 18)$.

```
input  : Sequence of length N
output: Optimal energy of the sequence
begin
    readEnergyTables();
    readSequence();
    for i ← 1 to N do
        for j ← 1 to N do
            VBI(i, j) ← INF;
            VM(i, j) ← INF;
            V(i, j) ← INF;
            WM(i, j) ← INF;
        end
        W(i) ← INF;
    end
    for b ← 0 to N − 1 do
        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 1: Function *calculate*(N), main function for calculating the free energy.

input : Base indices i and j
output: $VBI(i,j)$
begin
  for $i_p \leftarrow i+1$ **to** $j-2$ **do**
    for $j_p \leftarrow i_p+1$ **to** $j-1$ **do**
      $VBI(i,j) \leftarrow$ `MIN(` $VBI(i,j)$, `eL(`i,j,i_p,j_p`)` $+ V(i_p,j_p)$`)`;
    end
  end
  return $VBI(i,j)$;
end

Algorithm 2: Function $calcVBI(i,j)$, ñaive way of calculating internal loops.

input : Base indices i and j
output: $WM(i,j)$
begin
  `//` b `and` c `are helix penalty and free base penalty for multiloops.`
  $WM_{ij} \leftarrow V(i,j)$+`auPen(`$i,j$`)`$+b$;
  $WM_{idj} \leftarrow V(i+1,j)$+`dangle-3'(`$j,i+1,i$`)`+`auPen(`$i+1,j$`)`$+b+c$;
  $WM_{ijd} \leftarrow V(i,j-1)$+`dangle-5'(`$j-1,i,j$`)` + `auPen(`$i,j-1$`)`$+b+c$;
  $WM_{idjd} \leftarrow V(i+1,j-1)$+`dangle-3'(`$j-1,i+1,i$`)`
  +`dangle-5'(`$j-1,i+1,j$`)` + `auPen(`$i+1,j-1$`)`$+b+2c$;
  $WM(i,j) \leftarrow$ `MIN(`$WM_{ij}, WM_{idj}, WM_{ijd}, WM_{idjd}$`)`;
  for $h \leftarrow i$ **to** $j-1$ **do**
    $WM(i,j) \leftarrow$ `MIN(`$WM(i,j), WM(i,h)+WM(h+1,j)$`)`;
  end
  $WM(i,j) \leftarrow$ `MIN(` $WM(i+1,j)+c, WM(i,j-1)+c, WM(i,j)$`)` ;
  return $WM(i,j)$;
end

Algorithm 3: Function $calcWM(i,j)$

input : Base indices i and j
output: $VM(i,j)$

begin
  $VM_{ij} = VM_{idj} = VM_{ijd} = VM_{idjd} = INF$;
  // a, b, c `are multiloop offset, helix penalty and free base penalty.`
  for $h \leftarrow i+2$ **to** $j-1$ **do**
    $VM_{ij} \leftarrow$ `MIN(` $VM_{ij}, WM(i+1,h-1) + WM(h,j-1)$);
  end
  for $h \leftarrow i+3$ **to** $j-1$ **do**
    $VM_{idj} \leftarrow$ `MIN(`$VM_{idj}, WM(i+2,h-1) + WM(h,j-1)$);
  end
  $VM_{idj} \leftarrow VM_{idj}+$`dangle-5'(`$i,j,i+1$`)`$+c$;
  for $h \leftarrow i+2$ **to** $j-2$ **do**
    $VM_{ijd} \leftarrow$ `MIN(`$VM_{ijd}, WM(i+1,h-1) + WM(h,j-2)$);
  end
  $VM_{ijd} \leftarrow VM_{ijd} +$ `dangle-3'(`$i,j,j-1$`)` $+ c$;
  for $h \leftarrow i+3$ **to** $j-2$ **do**
    $VM_{idjd} \leftarrow$ `MIN(`$VM_{idjd}, WM(i+2,h-1) + WM(h,j-2)$);
  end
  $VM_{idjd} \leftarrow VM_{idjd} +$`dangle-5'(`$i,j,i+1$`)` $+$ `dangle-3'(`$i,j,j-1$`)` $+2c$;
  $VM(i,j) \leftarrow$ `MIN(` $VM_{ij}, VM_{idj}, VM_{ijd}, VM_{idjd}$`)`;
  $VM(i,j) \leftarrow VM(i,j) + a + b +$ `auPen(`i,j`)`;
  return $VM(i,j)$;
end

Algorithm 4: Function $calcVM(i,j)$

input : Base indices i and j
output: $V(i,j)$
begin
  $V(i,j) \leftarrow$ `MIN( eH(`i,j`), eS(`i,j`)` $+V(i+1,j-1), VBI(i,j), VM(i,j)$);
  return $V(i,j)$;
end

Algorithm 5: Function $calcV(i,j)$

input : Base index j
output: Optimal Energy of the sequence $s_i s_{i+1} \dots s_j$, $W(j)$

begin
  for $i \leftarrow 1$ **to** $j-1$ **do**
    $wim1 \leftarrow \texttt{MIN}(0, W(i-1))$
    $W_{ij} \leftarrow V(i,j) + \texttt{auPen}(i,j) + wim1;$
    $W_{idj} \leftarrow V(i+1,j) + \texttt{dangle-3'}(j,i+1,i) + \texttt{auPen}(i+1,j) + wim1;$
    $W_{ijd} \leftarrow V(i,j-1) + \texttt{dangle-5'}(j-1,i,j) + \texttt{auPen}(i,j-1) + wim1;$
    $W_{idjd} \leftarrow V(i+1,j-1) + \texttt{dangle-3'}(j-1,i+1,i) + \texttt{dangle-5'}(j-1,i+1,j) + \texttt{auPen}(i+1,j-1) + wim1;$
    $W(j) \leftarrow \texttt{MIN}(W(j), W_{ij}, W_{idj}, W_{ijd}, W_{idjd});$
  end
  $W(j) = \texttt{MIN}(W(j), W(j-1));$
  return $W(j)$;
end

Algorithm 6: Function $calcW(j)$

input : Base indices i and j
output: $VBI(i,j)$
begin
  $Maxloop \leftarrow 30;$
  for $i_p \leftarrow i+1$ **to** $i + Maxloop + 1$ **do**
    for $j_p \leftarrow j - Maxloop - 1 + (i_p - i - 1)$ **to** $j-1$ & $j_p > i_p$ **do**
      $VBI(i,j) \leftarrow \texttt{MIN}(VBI(i,j), \texttt{eL}(i,j,i_p,j_p) + V(i_p,j_p));$
    end
  end
  return $VBI(i,j)$;
end

Algorithm 7: Function $calcVBI(i,j)$ using the heuristic - limiting the size of internal loops to a constant Maxloop.

input : Indices i and j
output: $VBI(i,j)$

begin
$c \leftarrow 3$;
// bases i_p and j_p forms the interior base pair
// Case 1: When first side < c
for $i_p \leftarrow i+1$ **to** $i+c$ **do**
for $j_p \leftarrow i_p+1$ **to** $j-1$ **do**
$VBI(i,j) \leftarrow$ MIN($VBI(i,j)$,eL$(i,j,i_p,j_p)+V(i_p,j_p)$);
end
end
// Case 2: When first side >= c, and second side < c
for $i_p \leftarrow i+c+1$ **to** $j-2$ **do**
for $j_p \leftarrow j-c$ **to** $j-1$ & $j_p > i_p$ **do**
$VBI(i,j) \leftarrow$ MIN($VBI(i,j)$,eL$(i,j,i_p,j_p) + V(i_p,j_p)$);
end
end
// case 3: General Cases - includes three subcases
// case 3.1: When both sides=c
$i_p = i+c+1$;
$j_p = j-c-1$;
$VBI(i,j) \leftarrow$ MIN($VBI(i,j)$, eL$(i,j,i_p,j_p)+V(i_p,j_p)$);
extend1(i,j,i_p,j_p);
// case 3.2: When First side=c+1, second side=c
$i_p = i+c+2$;
$j_p = j-c-1$;
$E_1 \leftarrow$ eL$(i,j,i_p,j_p)+V(i_p,j_p)$;
// subcase 3.3: When First side=c, second side=c+1
$i_{p1} = i+c+1$;
$j_{p1} = j-c-2$;
$E_2 \leftarrow$ eL$(i,j,i_{p1},j_{p1})+V(i_{p1},j_{p1})$;
$VBI(i,j) \leftarrow$ MIN$(VBI(i,j),E_1,E_2)$;
if $E_1 > E_2$ **then**
$i_p = i+c+1$;
$j_p = j-c-2$;
end
extend2(i,j,i_p,j_p);
return $VBI(i,j)$
end

Algorithm 8: Function $calcVBI(i,j)$ - Calculate internal Loops using Speedup Algorithm, ILSA

```
input: Variable i, j, i_p, j_p
begin
    i_v ← i + c + 1;
    j_v ← j − c − 1;
    for b ← 1 to MIN(i − 1,N − j) do
        VBI(i−b, j+b) ← MIN(VBI(i−b, j+b), eL(i, j, i_p, j_p) + V(i_p, j_p));
        // Two more options
        if VBI(i − b, j + b) > eL(i, j, i_v + b, j_v + b) + V(i_v + b, j_v + b) then
            i_p ← i_v + b;
            j_p ← j_v + b;
            VBI(i − b, j + b) ← eL(i, j, i_p, j_p)+ V(i_p, j_p) ;
        end
        if VBI(i − b, j + b) > eL(i, j, i_v − b, j_v − b)+ V(i_v − b, j_v − b) then
            i_p ← i_v − b;
            j_p ← j_v − b;
            VBI(i − b, j + b) ← eL(i, j, i_p, j_p) + V(i_p, j_p) ;
        end
    end
end
```

Algorithm 9: Function $extend1(i, j, i_p, j_p)$

```
input: Variable i, j, i_p, j_p
begin
    i_v ← i + c + 1;
    j_v ← j − c − 1;
    for b ← 1 to MIN(i − 1,N − j) do
        VBI(i−b, j+b) ← MIN(VBI(i−b, j+b), eL(i, j, i_p, j_p) + V(i_p, j_p));
        // Two more options
        if VBI(i − b, j + b) > eL(i, j, i_v + b + 1, j_v + b) +
        V(i_v + b + 1, j_v + b) then
            i_p ← i_v + b + 1;
            j_p ← j_v + b;
            VBI(i − b, j + b) ← eL(i, j, i_p, j_p)+ V(i_p, j_p)
        end
        if VBI(i − b, j + b) > eL(i, j, i_v − b, j_v − b − 1) +
        V(i_v − b, j_v − b − 1) then
            i_p ← i_v − b;
            j_p ← j_v − b − 1;
            VBI(i − b, j + b) ← eL(i, j, i_p, j_p) + V(i_p, j_p);
        end
    end
end
```

Algorithm 10: Function $extend2(i, j, i_p, j_p)$

CHAPTER V

INTERNAL LOOP SPEEDUP ALGORITHM

An internal loop is associated with the four parameters. Calculating all internal loops with a straight forward approach of searching (i_p, j_p) naively for every (i, j) as shown in algorithm 2 takes $O(n^4)$ amount of time and $O(n^2)$ space. This makes searching for all possible internal loops practically infeasible. Internal loop speedup algorithm (ILSA) described by Lyngsø *et al.* [16, 15], takes the advantage of the current form of internal loop energy function and has been shown to be implementable in $O(n^3)$ time complexity and $O(n^2)$ space for finding all possible internal loops. Also, the algorithm reduces the asymptotic time complexity of the commonly adopted heuristic of limiting the size of internal loops to a constant from $O(k^2n^2)$ to $O(kn^2)$. Here, we explain the algorithm in a simplified manner and provide a sound mathematical proof of its correctness. Our proof gives an insight into solving such kind of combinatorial problems.

Let say the sequence length is N and first side of the internal loop has n_1 and the second side has n_2 lengths respectively. Also, b is a positive integer varying from 1 to $min(i-1, N-j)$ for a closing pair (i, j) which will be used later. The energy function for internal loops depends upon the following terms:

1. Size penalty, where size is $n_1 + n_2$
2. Asymmetry penalty
3. Stacking energy of the closing base pair (i, j) with the adjacent mismatched base pair in the loop
4. Stacking energy of the enclosed base pair (i_p, j_p) with the adjacent mismatched

bases pair in the loop.

Asymmetry Penalty $asym(n_1, n_2)$ is following:

$$asym(n_1, n_2) = \min\{E_{max}, |n_1 - n_2|.f(m)\} \tag{6}$$

Here E_{max} is the maximum asymmetry penalty and f is a function of m where $m = \min\{n_1, n_2, c_1\}$, and c_1 is a small constant. The value of c_1 is given as 5 in [23] and for currently used thermodynamic parameters, it is set to 1 in [24]. For all $n_1 \geq c_1$ and $n_2 \geq c_1$, we can prove that

$$asym(n_1, n_2) = asym(n_1 + 1, n_2 + 1) \tag{7}$$

The subsequence $s_{i+1}s_{i+2}\ldots s_{j-1}$ can be divided in three parts, $s_{i+1}s_{i+2}\ldots s_{i_p}$, $s_{i_p+1}s_{i_p+2}\ldots s_{j_p-1}$ and $s_{j_p}s_{j_p+1}\ldots s_{j-1}$. Length of first and third parts are $n_1 + 1$ and $n_2 + 1$ respectively. We define the length of the second part with a variable gap g with the following relation:

$$(j - i - 1) = g + n_1 + n_2 + 2 \tag{8}$$

The gap g is also equal to $j_p - i_p - 1$. Considering g, instead of size of internal loops makes the algorithm and proof easier to understand. Note that for an arbitrary closing base pair (i, j), we can vary enclosed base pair (i_p, j_p) while keeping g fixed, resulting in the internal loops of size of $(n_1 + n_2)$. This method of keeping the length of the internal loops fix is equivalent to keeping the gap length fixed.

In the current thermodynamic model small internal loops of sizes 1*1, 1*2, 2*1, 2*2 and bulges do not follow the above described form of energy function and are treated specially. Assuming that constant c_1 in Eq. (6) is 1, we can safely apply the extension principle discussed here to the internal loops having both the sides greater than 2. Internal loops with one or both sides shorter than $c = 3$ are calculated naively, treated as special cases and extension principle is applied for others. From now onwards, we will talk only about internal loops that have both sides greater than or equal to $c = 3$. The pseudocode of ILSA is given in algorithm 8, 9 and 10.

5.1 Extension Principle

Consider an arbitrary closing base pair (i, j) and two candidates for the enclosed base pair (i_p, j_p) and (i'_p, j'_p) having same values of g, meaning

$$j_p - i_p = j'_p - i'_p. \tag{9}$$

Let's assume that for the closing base pair (i, j), the enclosed base pair (i_p, j_p) gives the more stable structure in comparison to the other choice (i'_p, j'_p). With this assumption, it can be proved [16] that with the above specified energy function and Eq. (7) and (9), the enclosed base pair (i_p, j_p) will also be better than (i'_p, j'_p) for all possible closing base pairs of the form $(i - b, j + b)$. Note that here, while going from (i, j) to $(i - b, j + b)$ we keep the asymmetry penalty and gap g fixed and increase the size of the loop. While going from (i, j) to $(i - b, j + b)$ for both of the enclosed base pairs, size of the internal loop increases with the same amount, asymmetry penalty terms cancel out due to Eq. (7) and terminal stacking energy of the closing base pair changes in the same manner.

5.2 Algorithm

Using the principal described above, if we know the best enclosed base pair (i_p, j_p) for the closing base pair (i, j), then we can find out the best enclosed base pair (i_{p1}, j_{p1}) for the loop closed by $(i - 1, j + 1)$ in constant time for the same value of gap g. We use this result to evaluate new internal loops using previously computed values. In the algorithm, when we know which enclosed base pair is best for the closing base pair (i, j), we also evaluate internal loops with closing base pair of the form $(i - b, j + b)$ for the same value of g at the same time.

To extend the result of smaller internal loops to the bigger loops and take care of all possible internal loops, we define two base cases for every closing base pair (i, j). The first base case corresponds to the internal loops having both sides equal to c, i.e.

$g = j - i - 2c - 3$. There is only one possible internal loop for this case. The second case corresponds to the two internal loops having one of the sides of length c and the other side of length $c + 1$, i.e. $g = j - i - 2c - 4$. Note, that $g = j - i - 2c - 3$ is the maximum possible value of gap for a closing base pair (i, j). These base cases are extended for all closing base pairs of the form $(i - b, j + b)$. This way, at the time of function call for (i, j), all bigger internal loops have already been evaluated and we can find the optimal internal loop by comparing the previous value of $VBI(i, j)$ with the two base cases.

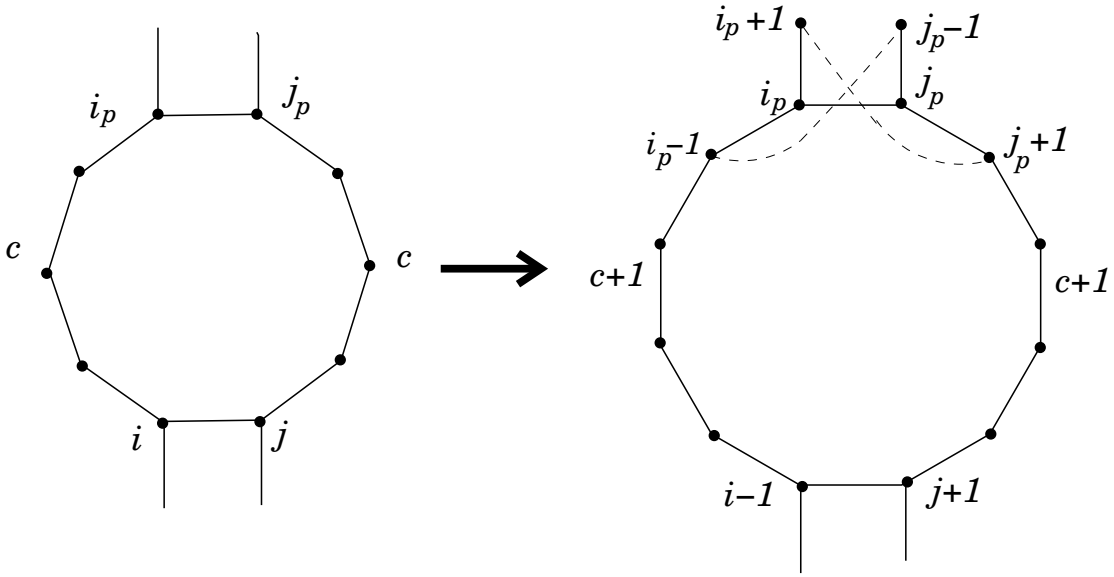

Figure 5: Extension of internal loop from (i, j) to $(i-1, j+1)$ for the first base case of the closing base pair (i, j) for $c = 3$. The length of both sides increases from (c, c) to $(c + 1, c + 1)$.

Figure 5 shows the extension of an internal loop having (i, j) closing base pair to the internal loop with $(i - 1, j + 1)$ closing base pair for the same enclosed base pair (i_p, j_p) for the first base case. To understand how the extensions of the two base cases of (i, j) to $(i-1, j+1)$ can be done in constant time, let's consider an arbitrary closing base pair (x, y) and say that $g = G_1, G_1 - 1$ are the base cases for this, where $G_1 = j - i - 2c - 3$. While extending the internal loop for $(x - 1, y + 1)$ for G_1, the length of the internal loop increases by 2. Therefore, two new candidates of enclosed

base pairs namely $(x+c, y-c-2)$ and $(x+c+2, y-c)$ are possible for closing base pair $(x-1, y+1)$ with gap G_1 for which the resultant internal loops have one side of length c and the other of length c+2. Note that the two candidates were not valid enclosed base pairs for closing base pair (x, y). This way we can get the best enclosed base pair for $[(x-1, y+1), G_1]$ by comparing the energies of the loops corresponding to the best enclosed base pair for $[(x, y), G_1]$ and the two new candidates. Similarly, the best enclosed base pair for $[(x-b, y+b), G_1]$ can be found in constant amount of time with the best enclosed base pair of $[(x-(b-1), y+(b-1)), G_1]$ and two new options which are introduced while extending the loop size from $2c+2b-2$ to $2c+2b$ for G_1. This extension corresponds to the *for* loop in algorithm 9.

The base case of $G_1 - 1$ for closing base pair (x, y) corresponds to two internal loops that have length of $2c+1$, in which one of the sides is c and the other side is $c+1$. Similar to the case described above, while extending the internal loop for $[(x-1, y+1), G_1 - 1]$ with the help of $[(x, y), G_1 - 1]$, the two new enclosed base pairs namely $(x+c, y-c-3)$ and $(x+c+3, y-c)$ are possible. They correspond to internal loops with length of one of the sides equal to c and the other equal to $c+3$, which were not possible earlier. Therefore, we can get the best enclosed base pair for the $[(x-1, y+1), G_1 - 1]$ with the help of the best enclosed base pair for $[(x, y), G_1 - 1]$ and the two newly introduced options. Similarly, the best internal loop for $[(x-(b-1), y+(b-1)), G_1 - 1]$ can be extended to find out the best internal loop for $[(x-b, y+b), G_1 - 1]$ in constant amount of time. This whole extension corresponds to the *for* loop in algorithm 10.

For a closing base pair (i, j), subsequent values of $g = 3, 4, \ldots, j-i-2c-4, j-i-2c-3$ are considered by previous extensions and its own base cases. This way the internal loops for closing base pair (i, j) are considered in the increasing order of gap length g from smallest to largest. The largest values of g are $j-i-2c-4$ and $j-i-2c-3$ which correspond to its base cases. At every base pair (i, j), its base

cases are extended for all base pairs of the form $(i-b, j+b)$. At this time all internal loops with g values corresponding to the base cases are considered for all base pairs $(i-b, j+b)$. This way, we do not need to store the enclosed base pairs for each closing base pair (i, j) and ILSA can be implemented in $O(n^2)$ storage. Also, it results in $O(n)$ computation time for an arbitrary (i, j) and $O(n^3)$ time for computation of all internal loops for every (i, j). Next, we prove that the algorithm computes all possible internal loops.

5.3 Proof of Correctness

5.3.1 Constraints

Now, we are concentrating on the problem of computing the internal loops having both sides at least c. The problem has following constraints:

$$1 \leq i < i_p < j_p < j \leq N \tag{10}$$

$$3 \leq g \leq j - i - 2c - 3 \tag{11}$$

$$i_p \geq i + c + 1 \tag{12}$$

$$j_p \leq j - c - 1 \tag{13}$$

A valid base pair (i, j) satisfies $j > i$, and indices i and j vary over the entire sequence for computing all possible internal loops. Also, base pair (i_p, j_p) is enclosed within the internal loop closed by base pair (i, j), which results into constraint in Eq. 10. The minimum allowed size of a hairpin loop is 3 which might be enclosed by base pair (i_p, j_p) and we assume that the minimum size of an internal loop is 2c leading to constraint in Eq. 11. Constraints in Eq. 12 and 13 result from the condition of having both sides greater than or equal to c.

Figure 6 shows a 2D graph formed with i_p and j_p as X and Y axis respectively for an arbitrary closing base pair (i, j). In the graph, the following two regions bounded

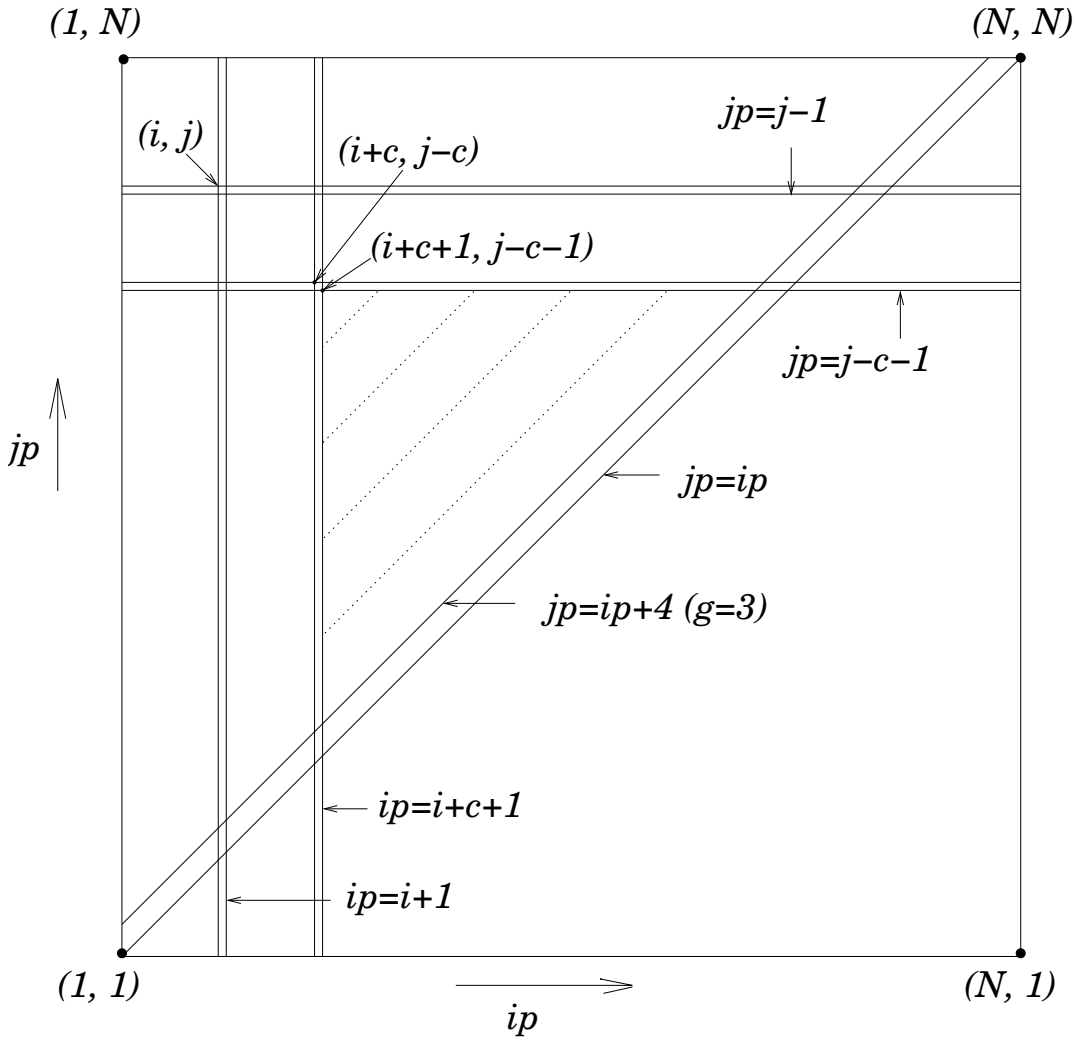

Figure 6: 2D plane $i_p - j_p$ for an arbitrary base pair (i, j) showing the special cases and extendable regions graphically.

by the given lines are taken as special cases and calculated separately.

$$\begin{aligned} i_p &= i+1 \\ i_p &= i+c \\ j_p &= i_p+4 \\ j_p &= j-1 \end{aligned}$$

and

$$\begin{aligned} j_p &= j-1 \\ j_p &= j-c \\ j_p &= i_p+4 \\ i_p &= i+c+1 \end{aligned}$$

The lines are marked in the figure. The first region corresponds to the values of i_p and j_p for which the first side of the internal loop is 0 to $c-1$ and the second side has all allowable sizes. The second region has first side greater than or equal to c and the second side is from 0 to $c-1$. These two regions are taken care of in the case 1 and case 2 of the algorithm 8.

The region corresponding to the "extendable" loops i.e. loops having both the sides greater than or equal to c, is bounded by the following lines.

$$\begin{aligned} i_p &= i+c+1 \\ j_p &= j-c-1 \\ j_p &= i_p+4 \end{aligned}$$

5.3.2 Outline

To show the correctness of the algorithm, we prove that we are calculating all internal loops closed by an arbitrary base pair (i, j). This can be established by the following two steps.

1. We consider all possible values of g for every possible closing base pair (i, j).

2. We consider all possible enclosed base pairs (i_p, j_p) for every possible value of g.

5.3.3 Claim 1

Prove that the algorithm considers every possible value of g, i.e. 3 to $j - i - 2c - 3$ for every possible point (i, j).

To show our claim we first prove that all possible points (i, j) corresponding to valid closing base pairs are situated on one of the lines of the form

$$j = i + 2c + k, \text{with } k \geq 6 \text{ and is an integer} \tag{14}$$

Consider the region bounded by the following lines, as shown in Figure 7.

$$\begin{aligned} i &= 1 \\ j &= N \\ j &= i + 2c + 6 \end{aligned} \tag{15}$$

Assuming the minimum value of $g = 3$, all possible points lie in the region specified above. It is clear that any point (x, y) lying in this region where x and y are integers, satisfies the constraint $y \geq x+2c+k$, where $k \geq 6$ and is an integer. Also the extreme point $(1, N)$ corresponds to the $k = N - 1 - 2c$.

We have shown that all possible points (i, j) lie on one of the lines of the form of Eq. (14), where values of k are taken from 6 to $N - 1 - 2c$. We will prove that the algorithm considers every possible value of g, for any point situated on one of these lines. Figure 7 shows the $i - j$ plane. Let's take an arbitrary value of the closing base pair (i, j) as (x, y) such that it is situated on the line below as shown in Figure 7.

$$j = i + 2c + k, \text{with } k \geq 6$$

This line corresponds to the two base cases for the point (x, y) with $g = k - 3$ and $k - 4$. Consider an another line in the $i - j$ plane

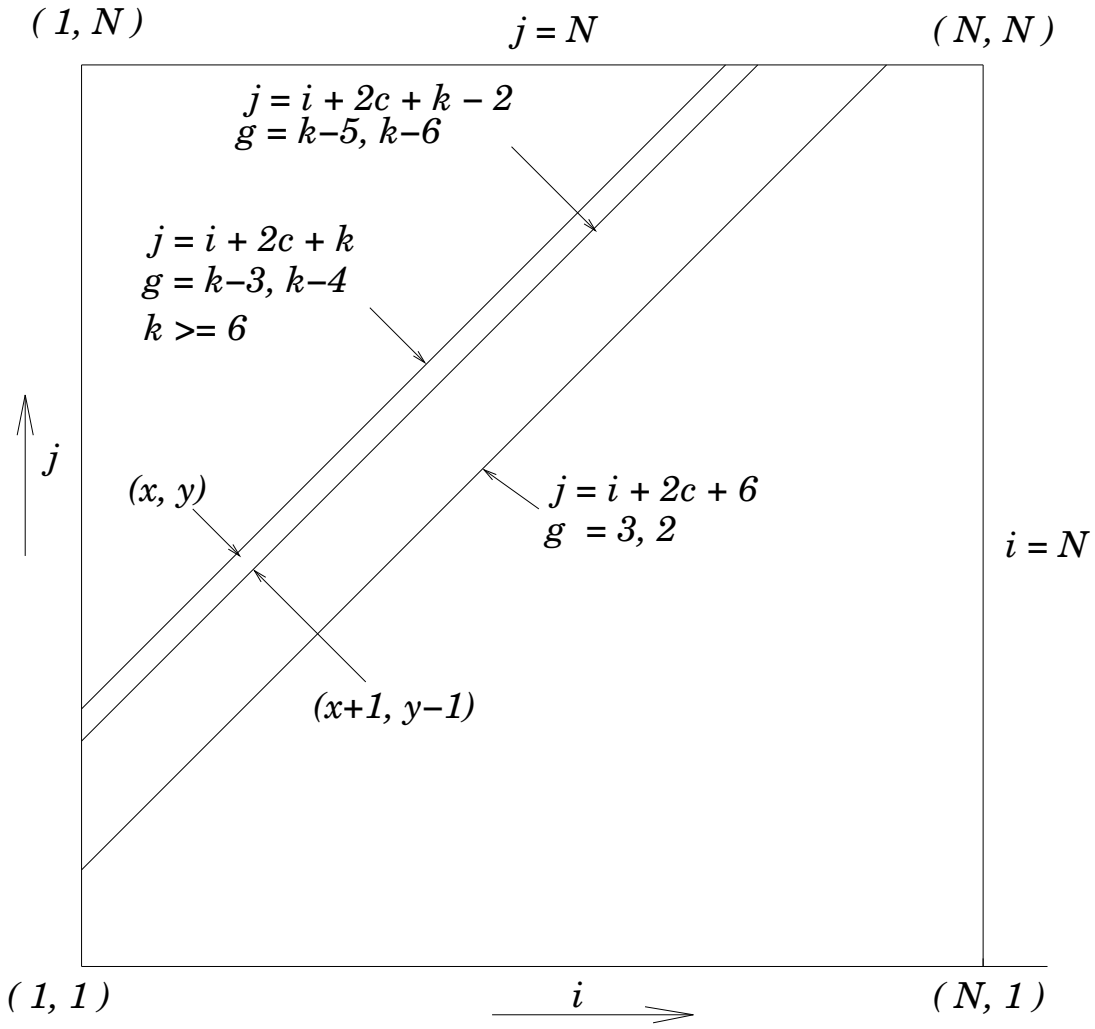

Figure 7: Showing the plane $i-j$. Point $(x+1, y-1)$ situated on line $j = i+2c+k-2$ extends its base case $g = k-5, k-6$ to the point (x, y) situated on line $j = i+2c+k$.

$j = i + 2c + k - 2$

The point $(x+1, y-1)$ is situated on this line and it corresponds to the base cases of $g = k-5, k-6$, for this point. At point $(x+1, y-1)$, the algorithm extends these base case $g = k-5$, $k-6$ for the point (x, y). This way the subsequent lines situated below this one, will extend their base cases for the point (x, y) either going through $g = 4, 3$ or $g = 3, 2$. This proves that the algorithm covers all allowable values of g for an arbitrary point (i, j).

5.3.4 Claim 2

Given (i, j) and a possible value of $g = G_1$, the algorithm covers all possible values of enclosed base pairs (i_p, j_p), or equivalently every possible values of n_1 and n_2.

Let say that an internal loop closed by a base pair (i, j) with gap G_1 is extended from a point $(i + B, j - B)$ for a particular value of $b = B$ such that G_1 is one of the two base cases for $(i + B, j - B)$. Thus, one of the two constraints given below is satisfied:

$$(j - B) = (i + B) + 2c + 3 + G_1$$

or

$$(j - B) = (i + B) + (2c + 1) + 3 + (G_1)$$

The first constraint comes from the base case where both sides of the internal loop are equal to c and second case comes from the base case having one of the sides of the internal loop equal to c and other as $c + 1$.

First case:

In this case, point $(i+B, j-B)$ has two base cases as G_1 and $G_1 - 1$ and the optimal internal loop closed by base pair (i, j) for gap G_1 is extended from the base case corresponding to both sides equal to c. The extended internal loop's both sides are equal to $c + B$. We represent this case as $[c + B, c + B]$. There are $2B + 1$ possible internal loops for closing base pair (i, j) with gap G_1, which we can get by varying length of both sides to the minimum value c. At every iteration in the for loop of

algorithm 9 for point $(i+B, j-B)$, we consider two new options and the loop runs $2B$ times for the closing base pair (i, j). At every step the two new options are taken care of. They correspond to $\{[c+(B-1), c+(B+1)], [c+(B+1), c+(B-1)]\}, \{[c+(B-2), c+(B+2)], [c+(B+2), c+(B-2)]\}, \ldots, \{[c, c+2B], [c+2B, c]\}$. Therefore, all $2B+1$ options of possible enclosed base pairs are considered including the base case for $(i+B, j-B)$.

Second case:

In this case, point $(i+B, j-B)$ has two base cases as G_1+1 and G_1 and the optimal internal loop closed by base pair (i, j) for gap G_1 is extended from the base case corresponding to one of the sides equal to c and other equal to $c+1$. This resultant internal loop has one side as $c+B$ and other side as $c+B+1$ which is represented as $[c+B, c+B+1]$ and $[c+B+1, c+B]$. This case has total number of $2B+2$ distinct possible internal loops for the closing base pair (i, j) with gap G_1. On each subsequent iterations of the *for* loop in algorithm 10 for point $(i+B, j-B)$, we consider two new cases and there are $2B$ iterations for point (i, j). This way all subsequent cases $\{[c+B-1, c+B+2], [c+B+2, c+B-1]\}, \{[c+B-2, c+B+3], [c+B+3, c+B-2]\} \ldots, \{[c, c+2B+1], [c+2B+1, c]\}$ are taken care of. Therefore, we consider all $2B+2$ cases including the two internal loops corresponding to the second base cases of $(i+B, j-B)$.

All these claims lead to the result that we are covering every possible internal loop for every possible closing pair (i, j).

CHAPTER VI

GTFOLD

6.1 Dependencies and Access Patterns

Figure 8 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 for the problem definition. Secondary structures can have only nested base pairings, meaning if there are two base pairs (i, j) and (i', j') such that $i < i' < j$ then the constraint $i < i' < j' < j$ is also satisfied. This assumption of nested base pairings results in the general dependency of point (i, j) on the points in the triangle T as shown in Figure 8. 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.

The speedup algorithm for internal loop calculations ILSA, follows the same general technique but its access pattern differs. It updates the elements outside the dependency triangle T shown in Figure 8 for the point (i, j). The access pattern of this algorithm is shown in Figure 9 excluding the calculation of special cases which belong to internal loops having one or both sides lesser than c. At point (i, j), ILSA updates elements of VBI array of the form $(i - b, j + b)$, where b is a positive integer from 1 to $\min(i - 1, n - j)$.

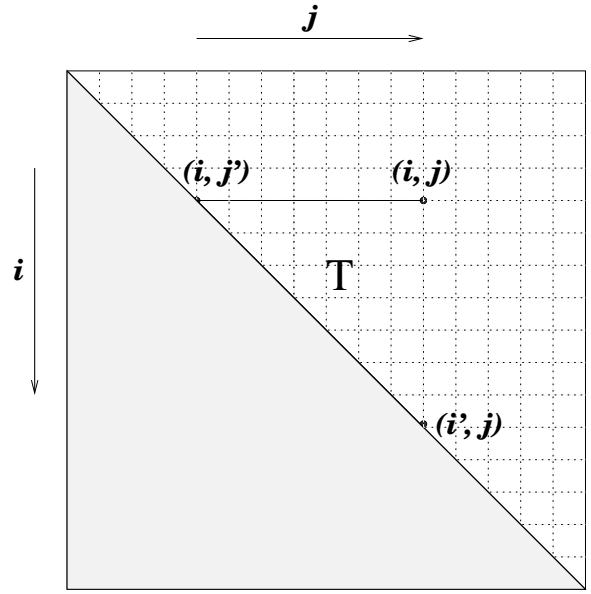

Figure 8: The implicit dependency of point (i, j) on the elements present in the triangle T.

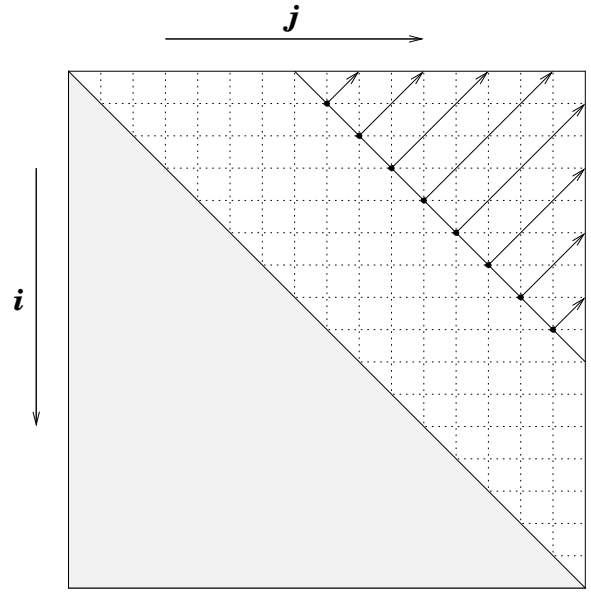

Figure 9: The access pattern of $VBI(i, j)$ for the internal loop speedup algorithm

6.2 Approach

In the region of ij plane having $j > i$, a point (i, j) corresponds to the computation of energy of the subsequence $s_i s_{i+1} \dots s_j$. The dependency pattern shown in Figure 8 allows the calculation of all the elements existing on a line $j - i = k$ to be independent of each other, where k is in the set $\{0, 1, 2, \dots, N-1\}$. 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.

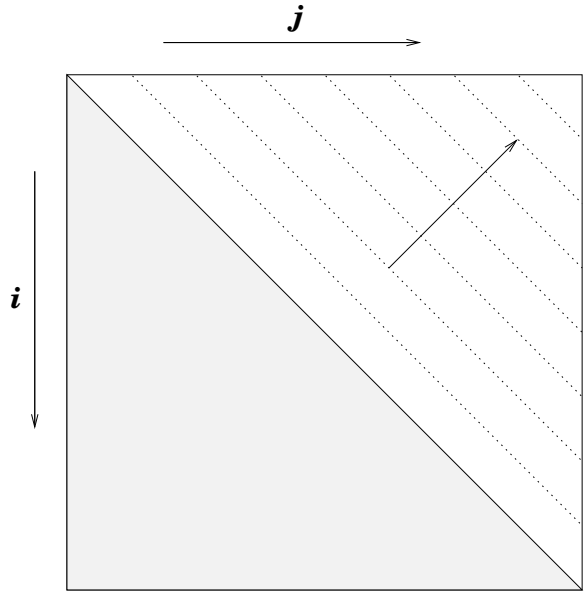

Figure 10: Showing the pattern of computation implemented in GTfold

Algorithm 11 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 10 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.

```
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 11: Main function to compute the secondary structure of an RNA sequence

6.2.1 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 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 ILSA for internal loop calculations can be parallelized for special cases similarly. The computation of general internal loops is done using the extension principal. The *for* loops of algorithm 9 and 10 involve forwarding result from a previous iteration to the next iteration and are sequential in nature. However the generally adopted heuristic option for internal loops as shown in algorithm 2 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 WM and VM arrays shown in algorithms 3 and 4 respectively, have $O(n)$ time complexity for an element (i, j), and are also amenable to parallelization.

6.3 Implementation Details

We use OpenMP [21] to implement shared memory parallelism. All the subsequent diagonals are considered with the upper *for* loop and parallelism is implemented by applying an OpenMP *for* loop pragma over the inner *for* loop to parallelize the computation on the diagonal in consideration as shown in Algorithm 11. The guided scheduling strategy works best for this parallelization. This is because there may not be equal amounts of work for every point on the diagonal. If the bases i and j are not able to make a pair then it is not necessary to carry out the whole calculation. In this case, for the heuristic option of internal loop calculations, only $WM(i, j)$ needs to be calculated and for ILSA $VBI(i, j)$ also needs to be calculated with $WM(i, j)$.

We explore the function level parallelism for the last few diagonals by deciding a threshold variable A with experiments. The parallelism is implemented at the higher level for the diagonals up to $j-i = A$ and the function level parallelism is implemented

starting from the diagonal $j - i = A + 1$ to $j - i = N - 1$. This facilitates the use of more threads to exploit more parallelism at the time when there are not enough points on the diagonals. However this technique did not give us a performance advantage.

6.3.1 Cache locality

For this algorithm the ratio of computation to the memory accesses is low. Energy of a secondary structure is calculated by adding up the energies of various loops present in the structure. Energy of a structure is the sum of various energy terms of which some are read directly from the energy tables and others are calculated by the program. Therefore, large cache sizes and locality in reference for accessing various data elements play an important role in reducing the running time of GTfold. Computing the elements row-wise or column-wise as described in Section 6.2 provides better cache locality than computing the elements on the subsequent diagonals. However, these two ways are inherently sequential.

6.4 Experimental Results

We have performed several experiments to establish that GTfold runs faster than competing folding programs such as mfold and RNAfold and achieves comparable accuracy. For the running time and accuracy comparisons, we are using RNAfold distributed with Vienna RNA Package version 1.7.2 and UNAFold version 3.6, which supersedes mfold.

6.4.1 Energy and Structure Comparison

To establish the accuracy of GTfold, we compare the structures obtained from GTfold, mfold, and RNAfold, with the correct structures determined with the more reliable method of comparative sequence analysis [9, 10]. Comparative sequence analysis requires large data sets for the prediction of secondary structures, and therefore, its application is limited by the availability of the required datasets.

Doshi *et al.* [7] take a phylogenetically diverse dataset of ribosomal RNA sequences and compare the optimal secondary structures predicted using mfold 2.3 and mfold 3.1 with the correct structures. Here we are using the 16S and 23S ribosomal RNA sequences from Figure 1 and Table 4 of their study [7] for accuracy comparisons which are taken from the Gutell database [3]. For predicting structures with UNAFold and RNAfold their command line default options are used. Accuracy of the structures is calculated in the same manner as in [7] with one difference, we include non-canonical base pairs in the comparison instead of excluding them. Accuracy measurement of all three programs is affected in the same manner by excluding non-canonical base pairs because the programs are unable to predict them due to the lack of thermodynamic parameters. We are using sensitivity and specificity as the measures of accuracy. Sensitivity is the percentage of correctly predicted base pairs out of the total base pairs present in the correct secondary structure. Specificity is defined as the number of correctly predicted base pairs out of the total base pairs present in the predicted secondary structure.

Table 1: Free energy (in Kcal/mole) comparison of GTfold, UNAFold and RNAfold for 16S rRNA sequences

Sequence	Length	GTfold	UNAFold	RNAfold
X00794	1962	-741.90	-722.70	-746.60
X54253	701	-149.00	-141.30	-149.03
X54252	697	-142.50	-137.50	-142.52
Z17224	1550	-564.80	-549.10	-565.12
X65063	1432	-582.00	-570.80	-581.94
Z17210	1435	-761.90	-626.60	-762.70
X52949	1452	-802.70	-794.50	-804.40
X98467	1295	-487.00	-460.00	-489.31
Y00266/M24612	1244	-325.60	-317.30	-328.80
X59604	1701	-573.00	-491.40	-574.70
K00421	1474	-687.00	-682.10	-687.01

Table 1 shows the optimal free energy of various 16S ribosomal RNA sequences

Table 2: Accuracy comparison (in percent) of GTfold, UNAFold and RNAfold for 16S rRNA sequences of Table 1. Here Sens. stands for sensitivity and Spec. stands for specificity.

Sequence	GTfold		UNAfold		RNAfold	
	Sens.	Spec.	Sens.	Spec.	Sens.	Spec.
X00794	30.33	22.19	31.65	23.96	27.91	20.65
X54253	25.67	22.12	20.32	18.54	25.13	21.76
X54252	21.16	18.69	21.64	18.60	21.16	18.60
Z17224	26.03	21.88	24.57	21.49	24.57	20.49
X65063	24.09	21.28	22.02	19.27	23.83	20.96
Z17210	24.46	19.98	25.98	21.35	24.71	20.10
X52949	15.07	12.45	16.08	13.42	15.07	12.45
X98467	17.09	16.26	10.97	11.05	16.33	15.65
Y00266/M24612	19.19	18.73	17.30	17.11	18.11	17.82
X59604	27.49	23.22	24.83	20.10	27.49	23.35
K00421	76.42	72.61	75.76	72.44	76.42	72.61

predicted with GTfold, UNAFold and RNAfold. Table 2 shows the accuracy comparison for the three programs for the sequences of Table 1. Similarly Table 3 shows the optimal free energy obtained using the programs for 23S ribosomal RNA sequences, and Table 4 shows the accuracy comparison for the sequences of Table 3.

Table 3: Free energy (in Kcal/mole) comparison of GTfold, UNAFold and RNAfold for 23S rRNA sequences

Sequence	Length	GTfold	UNAFold	RNAfold
X14386	3105	-791.30	-775.10	-792.65
X54252	953	-180.20	-173.50	-179.71
X52392	1621	-395.80	-389.70	-397.85
J01527	3273	-700.60	-684.60	-702.87
K01868	3514	-1328.50	-1294.30	-1333.95
X53361	4052	-1693.60	-1665.60	-1696.49
X52949	2850	-1707.90	-1689.20	-1709.80
M67497	3029	-1666.10	-1647.10	-1668.12

Energy comparisons presented in Tables 1 and 3 show slight differences in the energy obtained from GTfold, RNAfold and UNAfold and our energy scores for the various sequences lie in the very small range of these standard programs. To explain this, we recalculate free energies of the optimal secondary structures obtained from

Table 4: Accuracy comparison (in percent) of GTfold, UNAFold and RNAfold for 23S rRNA sequences of Table 3. Here Sens. stands for sensitivity and Spec. stands for specificity.

Sequence	GTfold		UNAfold		RNAfold	
	Sens.	Spec.	Sens.	Spec.	Sens.	Spec.
X14386	21.77	19.87	18.79	17.25	18.32	16.54
X54252	23.74	18.25	21.92	15.48	23.29	16.50
X52392	24.44	20.96	25.56	22.20	24.16	20.48
J01527	24.72	17.19	30.11	20.78	25.14	17.49
K01868	20.67	13.96	22.01	15.02	17.85	12.10
X53361	22.21	18.09	16.59	13.56	15.44	12.48
X52949	34.44	29.23	31.24	26.96	26.69	22.55
M67497	64.05	56.58	63.60	56.92	63.94	56.31

RNAfold and UNAfold with the GTfold energy function, shown in Tables 5 and 6. These comparisons show that the optimal structures obtained from UNAfold and RNAfold achieve higher score than the score of the GTfold's optimal structure using the energy function of GTfold. In summary, we can say that all three programs are trying to minimize three different objective functions. Details of these objective functions are associated with algorithmic issues and thermodynamic policies chosen by them.

Accuracy comparisons shown in Tables 2 and 4 establish that GTfold achieves accuracy comparable with UNAFold and RNAfold for the diverse dataset chosen. These comparisons for various ribosomal sequences show that in general accuracy of the prediction programs are very low. The prediction accuracy is expected to increase with the inclusion of advanced thermodynamic details that are not presently incorporated due to high computational cost. The development of GTfold facilitates the implementation of these improvements.

Wiese *et al.* [31] took 19 sequences of varying length of different RNA classes and computed the accuracy of their prediction program called RNApredict which uses evolutionary algorithms (EAs). Table 7 compares the accuracy of GTfold and RNApredict for those sequences. The results show that there is no clear argument of

Table 5: Free energy (in Kcal/mole) comparison of GTfold, UNAFold and RNAfold using GTfold energy function for 16S rRNA sequences of Table 1. Columns "UNAfold" and "RNAfold" contain the energy values recalculated using the GTfold energy function and columns "UNAf (T)" and "RNAf (T)" contain the actual energy values predicted by UNAfold and RNAfold respectively.

Sequence	Length	GTfold	UNAfold	RNAfold	UNAf (T)	RNAf (T)
X00794	1962	-741.90	-727.6	-737.2	-722.70	-746.60
X54253	701	-149.00	-143.1	-149.0	-141.30	-149.03
X54252	697	-142.50	-138.7	-142.5	-137.50	-142.52
Z17224	1550	-564.80	-552.6	-558.5	-549.10	-565.12
X65063	1432	-582.00	-574.0	-579.7	-570.80	-581.94
Z17210	1435	-761.90	-708.4	-760.2	-626.60	-762.70
X52949	1452	-802.70	-794.4	-800.0	-794.50	-804.40
X98467	1295	-487.00	-461.2	-484.6	-460.00	-489.31
Y00266/M24612	1244	-325.60	-318.8	-323.3	-317.30	-328.80
X59604	1701	-573.00	-518.1	-571.8	-491.40	-574.70
K00421	1474	-687.00	-684.0	-687.0	-682.10	-687.01

better accuracy for one software versus other. For some of the sequences GTfold performs better and for others RNApredict does. Larger sequences are usually predicted better with RNApredict. However the comparisons involved are not enough to give any judgment for the accuracy of the two approaches.

6.4.2 Running Time Comparison

GTfold implements parallelism for shared memory multiprocessor and multicore systems. Running time experiments are performed on an IBM P5-570 server with 16 dual core 1.9 GHz CPUs and 256 GB of main memory with L2 cache of 1.9 MB per CPU. GTfold is compiled with IBM xlC compiler Enterprise Edition 7.0, with -q64 option for the 64 bit compilation, -O3 level of optimization and -qsmp=omp option for OpenMP support. RNAfold and UNAFold are compiled with their default compiler and compilation options. An additional flag -maix64 is set while compiling UNAFold and RNAfold due to the runtime memory limitations on the system for 32-bit compilations.

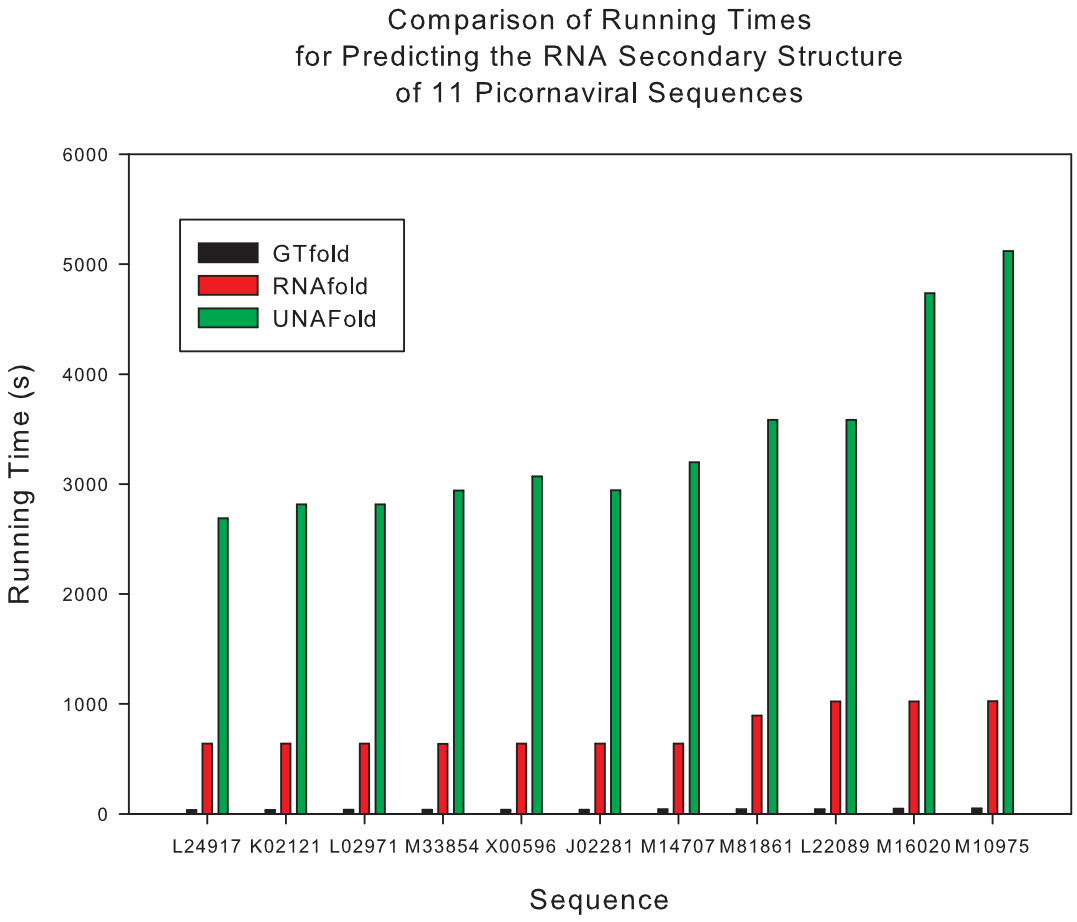

Figure 11: Comparison of running times for predicting the RNA secondary structures of 11 picornaviral sequences. The sequences are arranged in increasing order of length from 7124 to 8214 nucleotides.

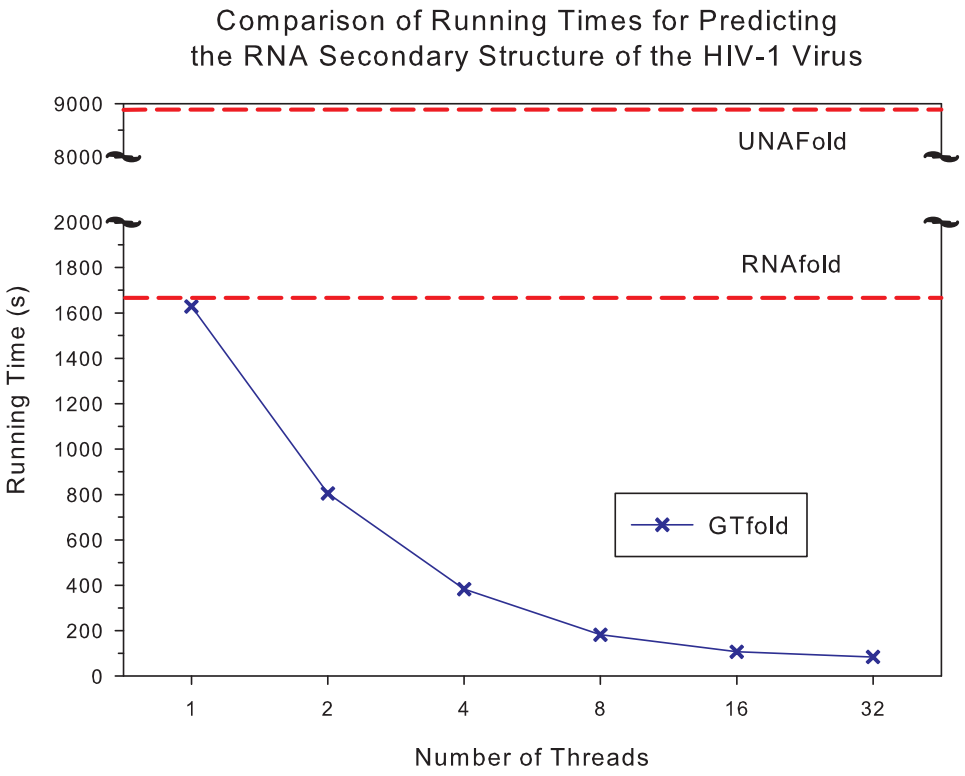

Figure 12: Comparison of running times for predicting the RNA secondary structure of the HIV-1 virus. The dashed horizontal lines represent the sequential running time of UNAFold and RNAfold.

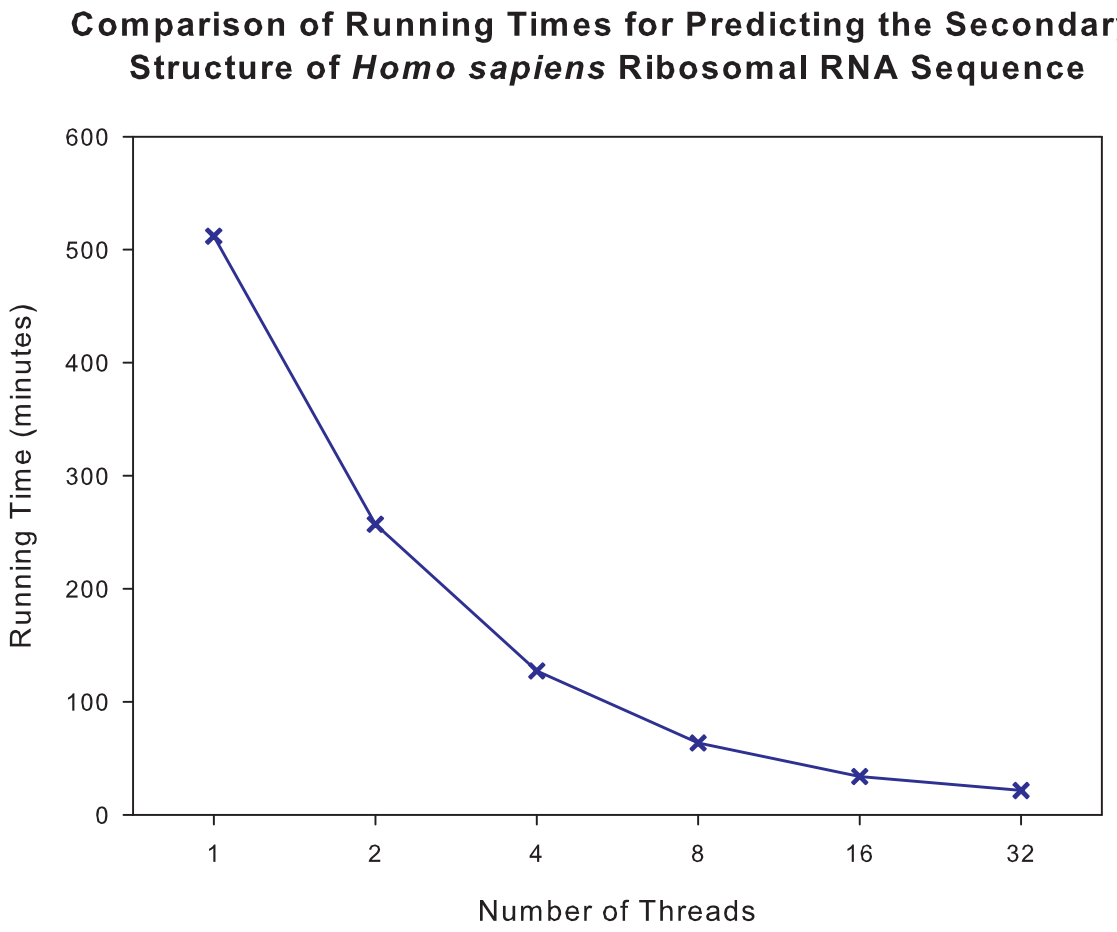

Figure 13: GTfold running time statistics for a *Homo sapiens* 23S ribosomal RNA sequence with accession number J01866/M11167 using the Internal Loop Speedup Algorithm

Table 6: Free energy (in Kcal/mole) comparison of GTfold, UNAFold and RNAfold using GTfold energy function for 23S rRNA sequences of Table 3. Columns "UNAfold" and "RNAfold" contain the energy values recalculated using the GTfold energy function and columns "UNAf(T)" and "RNAf(T)" contain the actual energy values predicted by UNAfold and RNAfold respectively.

Sequence	Length	GTfold	UNAfold	RNAfold	UNAf (T)	RNAf (T)
X14386	3105	-791.30	-779.6	-784.6	-775.10	-792.65
X54252	953	-180.20	-173.4	-179.7	-173.50	-179.71
X52392	1621	-395.80	-394.3	-394.5	-389.70	-397.85
J01527	3273	-700.60	-689.7	-699.6	-684.60	-702.87
K01868	3514	-1328.50	-1301.4	-1316.5	-1294.30	-1333.95
X53361	4052	-1693.60	-1672.1	-1678.0	-1665.60	-1696.49
X52949	2850	-1707.90	-1695.3	-1699.9	-1689.20	-1709.80
M67497	3029	-1666.10	-1654.0	-1664.1	-1647.10	-1668.12

Palmenberg and Sgro in 1997 [22] investigated the optimal and suboptimal secondary structures of 11 picornaviral RNA sequences using mfold version 2.2. The length of the sequences varies from 7124 to 8214 nucleotides. They report that each sequence required 5-7 days of CPU time using a modern workstation so that all 11 sequences took 2 to 3 months of time. In stark comparison, GTfold finishes the execution of this set of sequences in approximately 8 minutes using 32 threads. In Figure 11 we compare the running time of GTfold with 32 threads, UNAFold, and RNAfold, for all the picornaviral sequences on the same IBM machine. We can see that GTfold runs one to two orders of magnitude faster than the standard sequential programs UNAFold and RNAfold.

Hofacker *et al.* in 1996 [12] performed minimum free energy calculations for 22 full length HIV sequences on a modern distributed memory supercomputer using their parallel program reporting the running time of the order of 35 minutes. The sequences are arranged in the order of increasing lengths which range from 9022 to 10269 nucleotides. Accession numbers of these sequences are M38431, U43141, M22639, M17451/M12508, M27323, L02317, K03454/X04414, M62320, M26727, K02013, K03456, M38429, D10112/D00917, M93258, M19921, M93259, K03455/M38432, K02007, M17449,

Table 7: Accuracy comparison (in percent) of GTfold and Evolutionary Algorithms. Here, Sens. stands for sensitivity and Spec. stands for Specificity.

Sequence	Type	Length	Sens.		Spec.	
			GTfold	EA	GTfold	EA
AJ251080	5S rRNA	117	**65.8**	60.5	**73.5**	69.7
X67579	5S rRNA	118	**89.2**	89.2	80.5	**84.6**
V00336	5s rRNA	120	**25.0**	25.0	**26.3**	25.6
AF034620	5S rRNA	122	**76.3**	71.1	85.3	**90.0**
X01590	5S rRNA	123	62.5	**82.5**	71.4	**91.7**
AE002087	5S rRNA	124	**67.5**	62.5	**81.8**	75.8
AF197120	Group I intron, 23S rRNA	394	61.7	**62.5**	**63.2**	62.0
AB058310	Group I intron, 16S rRNA	454	**78.6**	68.3	**69.7**	62.8
AF197122	Group I intron, 23S rRNA	456	21.7	**47.8**	18.2	**40.7**
U40258	Group I intron, 16S rRNA	468	38.9	**60.2**	35.5	**51.9**
L19345	Group I intron, 16S rRNA	543	34.0	**57.2**	27.2	**49.1**
U02540	Group I intron, 16S rRNA	556	51.9	**61.8**	39.1	**50.3**
AF342746	Group I intron, 16S rRNA	605	**60.3**	52.1	39.2	**41.2**
X54252	16S rRNA	697	21.2	**29.1**	18.6	**27.2**
X05914	16S rRNA	784	15.9	**27.9**	15.5	**26.9**
X84387	16S rRNA	940	18.5	**28.5**	21.2	**32.5**
M27605	16S rRNA	945	**36.7**	37.1	36.7	**38.8**
J01415	16S rRNA	954	33.1	**33.5**	34.3	**35.6**
Y08511	16S rRNA	964	17.0	**30.9**	19.6	**33.9**

L20587, L20571, X61240/X16109 in the order. Fig. 14 shows the running time of GTfold with 32 threads for all these 22 sequences on the same IBM machine. GTfold takes from 61 to 90 seconds for different sequences and the average running time is 70 seconds.

Figure 12 compares the running time of GTfold, UNAFold, and RNAfold, for an HIV-1 sequence (accession number Z11530) with 9,781 nucleotides. The secondary structure predicted with GTfold of the viral sequence is shown in Figure 2. All three programs implement the heuristic option and limit the internal loop size to 30. It is clear from the graph that GTfold with one thread performs much better than UNAfold and is comparable with RNAfold. The running time of GTfold decreases with the increasing number of threads. Even with two threads GTfold runs 2.06 times faster than RNAfold. GTfold folds the entire HIV viral sequence in 84 seconds with

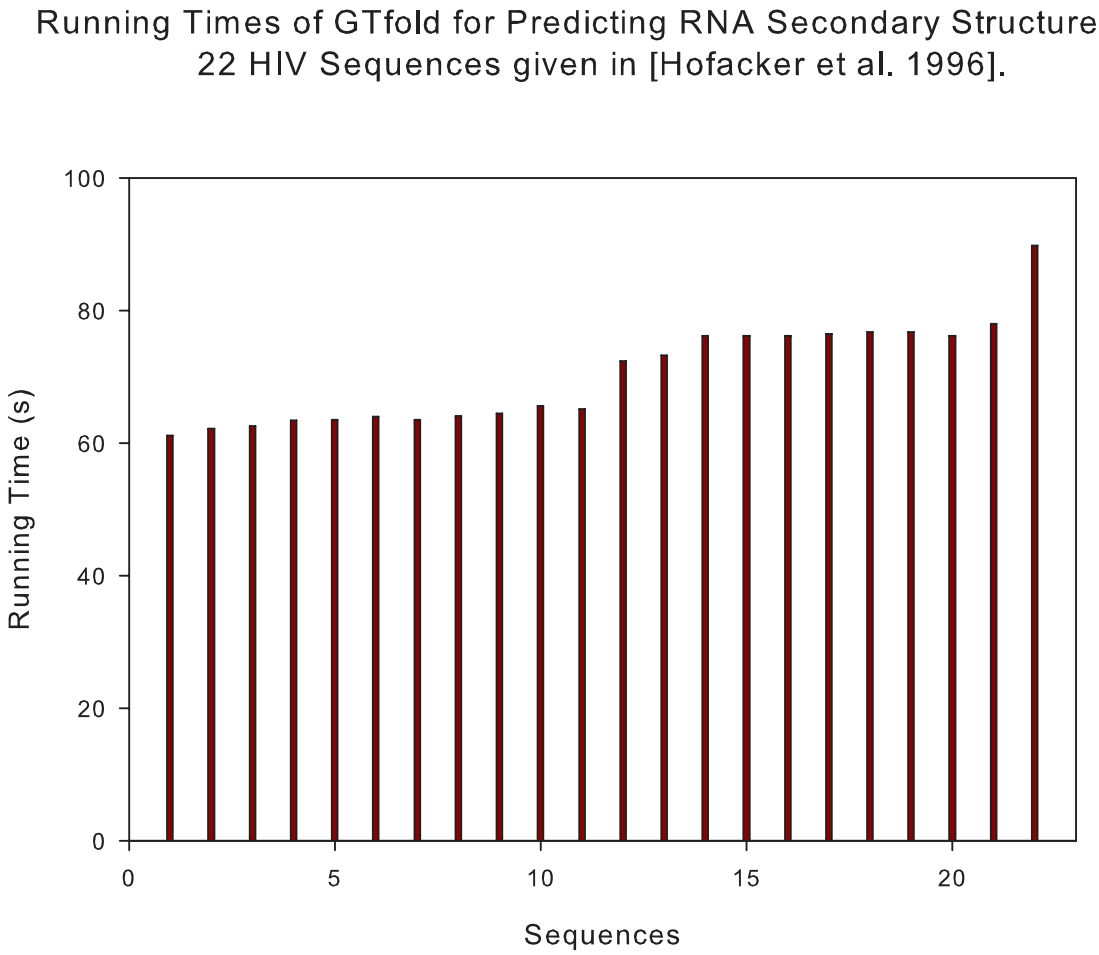

Figure 14: Running time of GTfold with 32 threads for 22 HIV sequences of length from 9022 to 10269 nucleotides used in Hofacker et al. [12]

32 threads in comparison to UNAFold and RNAfold which take approximately 2.4 hours and 27 minutes, respectively.

GTfold also implements an exact algorithm for finding the optimal internal loops called Internal Loop Speedup Algorithm (ILSA). Though internal loops with sizes longer than 30 are observed, they are usually rare. ILSA can catch these exceptional cases occurring with rarity in nature. It is far more expensive to run this algorithm than the commonly used heuristic. Figure 13 shows the running time of GTfold with the varying number of threads for a 5,184 length 23S Ribosomal RNA sequence of *Homo sapiens* with accession number J01866/M11167. GTfold is able to reduce the running time from 512 minutes (approximately 9 hours) to 21.5 minutes by using 32 threads. This way, we show that optimized algorithms such as internal loop speedup algorithm can be executed with GTfold in an affordable time. Please note that UNAFold and RNAfold do not implement the ILSA algorithm and can miss the rare possibilities.

Figure 15 shows the speedup achieved using 2, 4, 8, 16, and 32, threads for GTfold using the internal loop speedup algorithm (ILSA) option for the sequence with accession number J01866/M11167 and the heuristic options with an HIV viral sequence. The maximum speedup achieved in the first and second cases is approximately 23.8 and 19.8, respectively. We have achieved slightly superlinear speedups for 2, 4, and 8, threads in the case of the heuristic option due to the better cache locality when the number of threads is more than one.

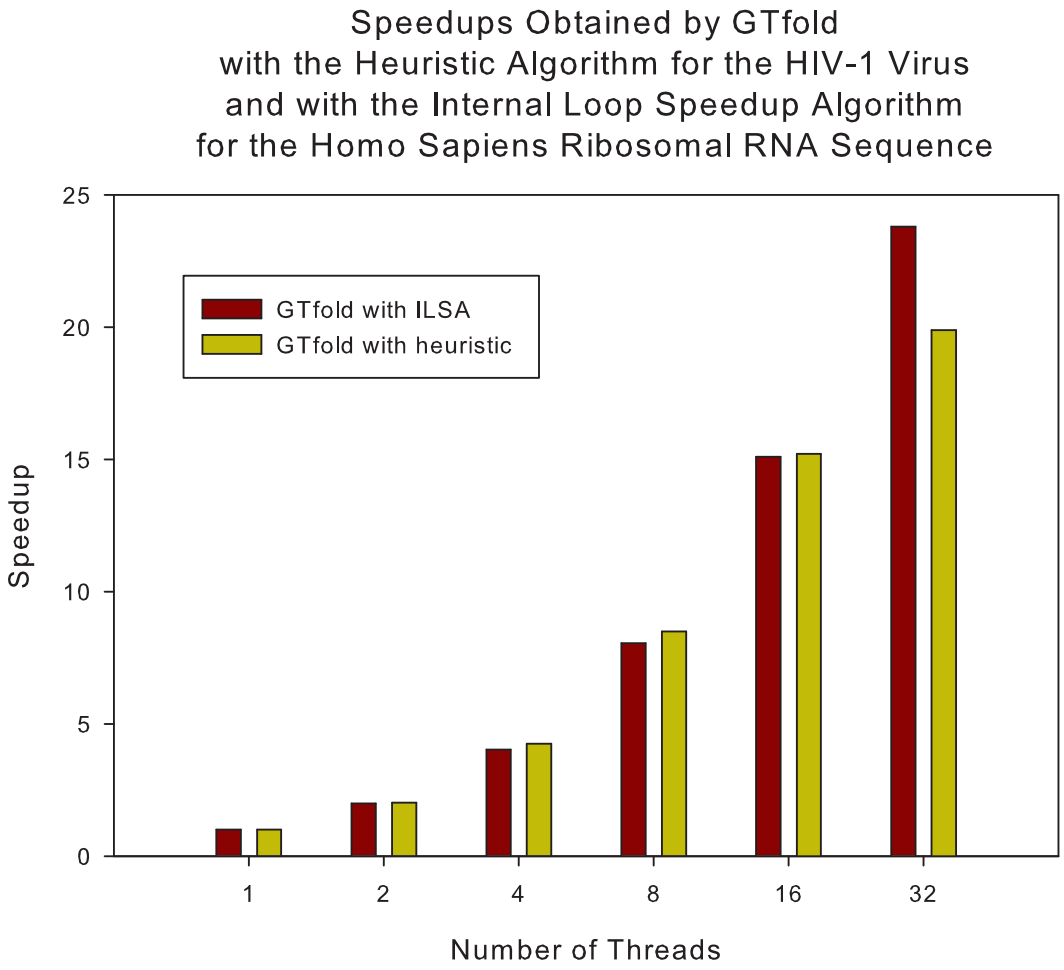

Figure 15: Speedups obtained by GTfold with the heuristic algorithm for the HIV-1 virus and with the internal loop speedup algorithm for the *Homo sapiens* ribosomal RNA sequence. Speedup is with respect to GTfold running on one processor for each series.

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

We have developed GTfold, a parallel and multicore code for predicting RNA secondary structures that achieves 19.8 fold speedups over the current best sequential program for large, important RNA sequences and has accuracy comparable to the existing standard folding programs. GTfold facilitates the implementation of more accurate thermodynamic model and exploring the entire search space. It helps reducing the running time which is the major prohibiting factor for more accurate secondary structure prediction of large RNA sequences, without increasing the space requirements.

Compared to an earlier study [22] for 11 picornaviral sequences which took approximately 2 months of time, GTfold folds these sequences in just 8 minutes. Also, GTfold takes the average running time of 70 seconds for folding 22 HIV sequences used in an another study [12] which took on the order of 35 minutes. GTfold includes an option to select the method of internal loop computations from the heuristic approach of limiting the size of internal loops to a constant and an optimized algorithm, ILSA. We have shown that exact algorithms such as ILSA can be executed in affordable amount of time with GTfold. We analyze computational requirements of the problem and document the detailed pseudocode of the algorithm, which provides a base for incorporating improved thermodynamic model and implementing optimized algorithms. We have given a sound mathematical proof of correctness of ILSA with the simple explanation. The proof gives us insight into solving such kind of combinatorial problems.

As pointed out in the experimental section, accuracy of the optimal structures

predicted with folding programs is very low. Improved thermodynamic model needs to be incorporated in the algorithm for predicting more accurate secondary structures. For example coaxial dangling energies for multiloops and external loop should be added and simplified multiloop energy function should be replaced with the function having logarithmic dependence on single stranded nucleotides. Also, capabilities such as forcing and prohibiting some base pairs to form should be provided in GTfold.

7.1 Suboptimal Secondary Structures

The minimum free energy (MFE) structure predicted by computer programs may not be the native structure of the molecule. The MFE score is found by exploring all the possibilities and it may correspond to more than one completely different secondary structures. The thermodynamic parameters are experimentally determined and are not precisely accurate. Intermolecular interactions among RNA and protein molecules provide additional stability and also tertiary interactions among the secondary structural elements play a role in stabilization. Due to all these reasons, native secondary structure of the molecule may correspond to one of the suboptimal folds within a small energy range of MFE. To find the native structure of the molecule, secondary structures which fall within $\Delta\Delta G$ a small energy range of MFE are predicted. These structures are called suboptimal secondary structures.

Enumerating all secondary structures within a given energy range is a very enabling capability for scientists who want to study the ensemble characteristics. Research in this direction may lead to the knowledge of surprising characteristics of the ensemble of secondary structures. The main problem in producing all suboptimal folds is that the number of structures grows exponentially with the sequence length and the energy range [34]. We are currently working in the direction of developing a new technique to capture the macroscopic information contained in the entire ensemble of secondary structures. Development of GTfold is providing us a base for

implementing new techniques for suboptimal secondary structure prediction.

REFERENCES

[1] Andronescu, M., Aguirre-Hernandez, R., Condon, A., and Hoos, H., "RNAsoft: a suite of RNA secondary structure prediction and design software tools," *Nucleic Acids Research*, vol. 31, no. 13, pp. 3416–3422, 2003.

[2] Andronescu, M. S., "Algorithms for predicting the secondary structure of pairs and combinatorial sets of nucleic acid strands," *M.Sc. Thesis*, 2003.

[3] Cannone, J., Subramanian, S., Schnare, M., Collett, J., D'Souza, L., Du, Y., Feng, B., Lin, N., Madabusi, L., Müller, K., Pande, N., Shang, Z., Yu, N., and Gutell, R., "The Comparative RNA Web (CRW) Site: an online database of comparative sequence and structure information for ribosomal, intron, and other RNAs," *BMC Bioinformatics*, vol. 3, no. 1, 2002.

[4] Chen, J.-H., Le, S.-Y., Shapiro, B. A., and Maizel, J. V., "Optimization of an RNA folding algorithm for parallel architectures," *Parallel Computing*, vol. 24, pp. 1617–1634, 1998.

[5] Clyde, K. and Harris, E., "RNA secondary structure in the coding region of dengue virus type 2 directs translation start codon selection and is required for viral replication," *J. Virol.*, vol. 80, no. 5, pp. 2170–2082, 2006.

[6] Do, C. B., Woods, D. A., and Batzoglou, S., "Contrafold: RNA secondary structure prediction without physics-based models," *Bioinformatics*, vol. 22, no. 14, pp. e90–e98, 2006.

[7] Doshi, K., Cannone, J., Cobaugh, C., and Gutell, R., "Evaluation of the suitability of free-energy minimization using nearest-neighbor energy parameters for RNA secondary structure prediction," *BMC Bioinformatics*, 2004.

[8] Fekete, M., Hofacker, I., and Stadler, P., "Prediction of RNA base pairing probabilities on massively parallel computers," *J. Computational Biology*, vol. 7, no. 1-2, pp. 171–182, 2000.

[9] Gardner, P. and Giegerich, R., "A comprehensive comparison of comparative RNA structure prediction approaches," *BMC Bioinformatics*, vol. 5, no. 140, 2004.

[10] Gutell, R., Lee, J., and Cannone, J., "The accuracy of ribosomal RNA comparative structure models," *Curr. Opin. Struct. Biol.*, vol. 12, no. 3, pp. 301–310, 2002.

[11] Hendriks, A., Deschenes, A., and Wiese, K., "A parallel evolutionary algorithm for rna secondary structure prediction using stacking-energies (INN and INN-HB)," in *Proc. of the 2004 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pp. 223–230, 2004.

[12] Hofacker, I., Huynen, M., Stadler, P., and Stolorz, P., "Knowledge discovery in RNA sequence families of HIV using scalable computers," in *Proc. of the 2nd Int'l Conf. on Knowledge Discovery and Data Mining*, (Portland, OR), 1996.

[13] Hofacker, I., Stadler, P., Bonhoeffer, L., Tacker, M., and Schuster, P., "Fast folding and comparison of RNA secondary structures," *Monatshefte für Chemie*, vol. 125, pp. 167–188, 1994.

[14] Knudsen, B. and Hein, J., "Pfold: RNA secondary structure prediction using stochastic context-free grammars," *Nucleic Acids Research*, vol. 31, no. 13, pp. 3423–3428, 2003.

[15] Lyngsø, R., Zuker, M., and Pedersen, C., "Fast evaluation of internal loops in RNA secondary structure prediction," *Bioinformatics*, vol. 15, pp. 440–445, 1999.

[16] Lyngsø, R., Zuker, M., and Pedersen, C., "Internal loops in RNA secondary structure prediction," in *Proc. of the 3rd Ann. Int'l Conf. on Computational Molecular Biology (RECOMB)*, pp. 260–267, 1999.

[17] Mathews, D. H. and Turner, D., "Prediction of RNA secondary structure by free energy minimization," *Current opinion in structural biology*, vol. 16, no. 3, pp. 270–278, 2006.

[18] Mathews, D., Disney, M., Childs, J., Schroeder, S., Zuker, M., and D.H., T., "Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure," *Proc. of the National Academy of Sciences of the USA*, vol. 101, no. 19, pp. 7287–7292, 2004.

[19] Mathews, D., Sabina, J., Zuker, M., and Turner, D., "Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure," *J. Mol. Biol.*, vol. 288, pp. 911–940, 1999.

[20] Nakaya, A., Yamamoto, K., and Yonezawa, A., "RNA secondary structure prediction using highly parallel computers," *Bioinformatics*, vol. 11, no. 6, pp. 685–692, 1995.

[21] OpenMP Architecture Review Board, *OpenMP Application Program Interface*, 3.0 ed., May 2008.

[22] Palmenberg, A. and Sgro, J.-Y., "Topological organization of picornaviral genomes: Statistical prediction of RNA structural signals," *Sem. Virol.*, vol. 8, pp. 231–241, 1997.

[23] Papanicolaou, C., Gouy, M., and Ninio, J., "An energy model that predicts the correct folding of both the tRNA and the 5S RNA molecules," *Nucleic Acids Research*, vol. 12, pp. 31–44, 2004.

[24] Peritz, A. E., Kierzek, R., Sugimoto, N., and Turner, D. H., "Thermodynamic study of internal loops in oligoribonucleotides: Symmetric loops are more stable than asymmetric loops," *Biochemistry*, vol. 30, no. 26, pp. 6428–6436, 1991.

[25] Shapiro, B. A., Wu, J. C., Bengali, D., and Potts, M. J., "The massively parallel genetic algorithm for RNA folding: MIMD implementation and population variation," *Bioinformatics*, vol. 17, no. 2, pp. 137–148, 2001.

[26] Tang, L., Johnson, K. N., Ball, L. A., Lin, T., Yeager, M., and Johnson, J. E., "The structure of pariacoto virus reveals a dodecahedral cage of duplex RNA," *Nature Structural Biology*, vol. 8, no. 1, pp. 77–83, 2001.

[27] Taufer, M., Solorio, T., Licon, A., Mireles, D., and Leung, M.-Y., "On the effectiveness of rebuilding RNA secondary structures from sequence chunks," in *7th IEEE Int'l Workshop on High Performance Computational Biology (HiCOMB)*, pp. 1–8, 2008.

[28] Tinoco, I. and Bustamante, C., "How RNA folds," *Journal of Molecular Biology*, vol. 293, pp. 271–281, 1999.

[29] Weik, M., Modrof, J., Klenk, H.-D., Becker, S., and Mühlberger, E., "Ebola virus VP30-mediated transcription is regulated by RNA secondary structure formation," *J. Virol.*, vol. 76, no. 17, pp. 8532–8539, 2002.

[30] Westerhout, E., Ooms, M., Vink, M., Das, A., and Berkhout, B., "HIV-1 can escape from RNA interference by evolving an alternative structure in its RNA genome," *Nucleic Acids Research*, vol. 33, no. 2, pp. 796–804, 2005.

[31] Wiese, K. C., Deschenes, A. A., and Hendriks, A. G., "RnaPredict-An evolutionary algorithm for RNA secondary structure predictionrnapredictan evolutionary algorithm for RNA secondary structure prediction," *IEEE/ACM Trasactions on Computational Biology and Bioinformatics*, vol. 5, pp. 25–41, 2008.

[32] Wiese, K. C. and Hendriks, A., "Comparison of P-RnaPredict and mfold-algorithms for RNA secondary structure prediction," *Bioinformatics*, vol. 22, no. 8, pp. 934–942, 2006.

[33] Wiese, K., Hendriks, A., Deschenes, A., and Youssef, B., "P-rnapredict-a parallel evolutionary algorithm for RNA folding: effects of pseudorandom number quality," *IEEE Transactions on NanoBioscience*, vol. 4, pp. 219–227, 2005.

[34] Wuchty, S., Fontana, W., Hofacker, I., and Schuster, P., "Complete suboptimal folding of RNA and the stability of secondary structures," *Biopolymers*, vol. 49, no. 2, pp. 145–165, 1999.

[35] WuJu, L. and JiaJin, W., "Prediction of RNA secondary structure based on helical regions distribution," *Bioinformatics*, vol. 14, no. 8, pp. 700–706, 1998.

[36] Ying, X., Luo, H., Luo, J., and Li, W., "Rdfolder: a web server for prediction of RNA secondary structure," *Nucleic Acids Research, Web Server issue*, vol. 32, pp. W150–W153, 2004.

[37] Zuker, M., "Mfold web server for nucleic acid folding and hybridization prediction," *Nucleic Acids Research*, vol. 31, no. 13, pp. 3406–3415, 2003.

[38] Zuker, M., Mathews, D. H., and Turner, D. H., "Algorithms and thermodynamics for RNA secondary structure prediction: A practical guide," *RNA Biochemistry and Biotechnology, J. Barciszewski B.F.C. Clark, eds., NATO ASI Series, Kluwer Academic Publishers*, 1999.

[39] Zuker, M. and Stiegler, P., "Optimal computer folding of large RNA sequences using thermodynamic and auxiliary information," *Nucleic Acids Research*, vol. 9, no. 1, pp. 133–148, 1981.