2008

A Seeded Genetic Algorithm for RNA Secondary Structural Prediction with Pseudoknots

Ryan Pham
San Jose State University

A Seeded Genetic Algorithm for RNA Secondary Structural Prediction with Pseudoknots

A Writing Project
Presented to
The Faculty of the Department of Computer Science
San Jose State University

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

by
Ryan Pham
December 2008

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Sami Khuri
Professor of Computer Science, San Jose State University, San Jose, CA

Dr. Robert Chun
Professor of Computer Science, San Jose State University, San Jose, CA

Dr. Chris Tseng
Professor of Computer Science, San Jose State University, San Jose, CA

APPROVED FOR THE UNIVERSITY

Abstract

This work explores a new approach in using genetic algorithm to predict RNA secondary structures with pseudoknots. Since only a small portion of most RNA structures is comprised of pseudoknots, the majority of structural elements from an optimal pseudoknot-free structure are likely to be part of the true structure. Thus seeding the genetic algorithm with optimal pseudoknot-free structures will more likely lead it to the true structure than a randomly generated population. The genetic algorithm uses the known energy models with an additional augmentation to allow complex pseudoknots. The nearest-neighbor energy model is used in conjunction with Turner's thermodynamic parameters for pseudoknot-free structures, and the H-type pseudoknot energy estimation for simple pseudoknots. Testing with known pseudoknot sequences from PseudoBase shows that it out performs some of the current popular algorithms.

Table of Contents

Table of Contents 1
1. Introduction 4
1.1 Pseudoknots 4
1.2 Purpose 5
2. Background and Related Works 6
2.1 Stochastic Context-Free Grammar 6
2.2 Hidden Markov Model (context sensitive) 8
2.3 Dynamic programming algorithms 10
2.3.1 Nussinov Algorithm 10
2.3.2 Pseudoknots (Eddy and Rivas) 12
2.3.4 Pseudoknots (Reeder and Giegerich) 17
2.4 Heuristics 19
2.4.1 HotKnots 19
2.4.2 Iterated Loop Matching 22
2.4.3 Genetic Algorithms 25
3. Algorithm 29
3.1 Overview 30
3.2 Generating Seeds and Stem Pools 31
3.2.1 Finding Optimal Pseudoknot-free Structures 31
3.2.2 Generating Stem Pools 32
3.3 Energy Rules 33
3.3.1 Nearest Neighbor Energy Rules 33
3.3.2 Pseudoknot Energy Rules 36
3.4 Genetic Algorithm 40
3.4.1 Solution Representation 40
3.4.2 Mutations 41
3.4.3 Crossovers 42
3.4.4 Selections 42
4. Results 43
5. Future Works 49
6. Conclusions 49
7. References 50

List of Tables

Table 1: Energies for internal, bulge, and hairpin loops from loops.dat (values between 8 and 30 are omitted) [25] 34
Table 2. Energy approximations for H-type pseudoknot loops [17] 38
Table 3. Test sequences from PseudoBase 43
Table 4. Sensitivity of predictions compared with 6 other algorithms 44
Table 5. Specificity of predictions compared with 6 other algorithms 44
Table 6. Sensitivity and specificity of short test sequences (average from 100 runs) 45
Table 7. Sensitivity and specificity of long test sequences (average from 100 runs) 46

List of Figures

Figure 1: A naturally occurring pseudoknot in the human telomerase [22] 5
Figure 2: Planar representation of RNA structural elements. [4] 5
Figure 3: Production rules for capturing a secondary structure and its associated derivation [14] 7
Figure 4: Pseudoknot is broken into two grammars [14] 8
Figure 5: Context-sensitive HMM with a stack(a) and a queue(b) as the memory unit [11] 9
Figure 6: A pseudoknot and its corresponding context-sensitive HMM [11] 10
Figure 7: Recurrence relation for the Nussinov algorithm [8] 11
Figure 8: Dynamic programming matrix [8] 11
Figure 9: Recurrence relation for vx [4] 12
Figure 10: Recurrence relation for wx [4] 13
Figure 11: Using two gap matrices to capture a pseudoknot [4] 14
Figure 12: Matrices used in the pseudoknot algorithm [4] 15
Figure 13: Recurrence relation for whx [4] 16
Figure 14: Updated recurrence relation for wx to include psuedoknots [4] 16
Figure 15: Non-planar pseudoknots [4] 17
Figure 16: Moving boundaries of a pseudoknots [6] 18
Figure 17: Pseudocode for HotKnots [1] 21
Figure 18: Result comparisons for short sequences [1] 22
Figure 19: Regions that do not need to be recomputed (A, B, C) [7] 24
Figure 20: Scoring matrix function for mutual score information [7] 24
Figure 21: Combined scoring functions [7] 25
Figure 22: Extended helix plot function [7] 25
Figure 23: Function for calculating bonus from energy stacking [7] 25
Figure 24: Binary representation of a solution in GA [13] 27
Figure 25: Mutation process [9] 28
Figure 26: Flowchart of the algorithm 30

Figure 27: Recurrence relations for pseudoknot-free structural prediction with free energy [21] 32
Figure 28: Hairpin loop of size 12 34
Figure 29: A bulge loop of size 2 35
Figure 30: An asymmetric internal loop of size 5 35
Figure 31: A multi-branch loop with 4 stems and 8 free bases 36
Figure 32: An H-type pseudoknot forms from two stem-loops [17] 37
Figure 33: H-type pseudoknot in planar representation. (b) Same structure in 3D [18] .. 39
Figure 34: A complex pseudoknot (kissing hairpins) decomposes into 3 separate hairpin loops. 39
Figure 35: A complex pseudoknot decomposes into a two hairpin loops and an internal loop. 40
Figure 36: Tree representation of a secondary structure 41
Figure 37: (A) PKB206 structure predicted by pknotsRG. (B) PKB206 structure predicted by GA and is also the known structure. 45
Figure 38: (A) and (B) are TYMV structures predicted by the seeded GA, (B) is also the known TYVM structure. (C) TYMV structure predicted by pknotsRG. (D) Pseudoknot-free structure predicted by the seeding algorithm. 46
Figure 39: (A) PKB173 structure predicted by pknotsRG and seeding algorithm. (B) Known PKB173 structure. (C) Best structure predicted by seeded GA. 47
Figure 40: (A) PKB72 structure predicted by Seeded GA. (B) known PKB72 structure. (C) PKB72 structure predicted by pknotsRG. 48

1. Introduction

Unlike DNA molecules, which consist of two complementary strands, RNA molecules are generally single stranded. The RNA strand forms its secondary structure when the bases within the molecule pair with each other. RNA's are not just mere vessels for transferring genetic information from DNA to protein; they also act as "catalyst in cellular processes, and mediators in determining the expression level of genes" [1]. Pseudoknots are secondary structural elements; their common function is to induce frameshifts, a function that is found in viruses and is required in all retroviruses to enable their replication and proliferation [2]. Most of the research involving RNA secondary structural predictions has ignored pseudoknots because of their computational complexity. Unlike pseudoknots, all other structural elements, such as loops or helices, form a tree-like hierarchy where these structural elements are either disjoint or contained within another [3]; pseudoknots are formed when there are overlapping regions between structural elements. Consequently, a time and space complexity of $O(n^6)$ and $O(n^4)$, respectively, is required to fold restricted classes of pseudoknots [4] while unknotted structure only requires $O(n^3)$ time and $O(n^2)$ space complexity [5]. Although the running time is still polynomial, it is impractical for sequences over 200 bases long. The general problem of predicting arbitrary pseudoknots for energy based models has been proven to be NP-complete [16]. This work will go over some of the existing computation based prediction algorithms and present a new approach that uses dynamic programming and a genetic algorithm to predict secondary structures with pseudoknots.

1.1 Pseudoknots

Pseudoknots are tertiary structures formed by overlapping secondary structures. The simplest type of pseudoknots is the H-type pseudoknot, which is shown in Figure 1; it consists of two stem-loops, where the loop from the first stem forms part of the second stem. As shown in Figure 2, other structural elements are well nested; pseudoknots are the only structures that exhibit the non-nested characteristic. Pseudoknots have many important biological functions; perhaps, the most important one is their ability to alter gene expression "by inducing ribosomal frameshifting in many viruses" [2].

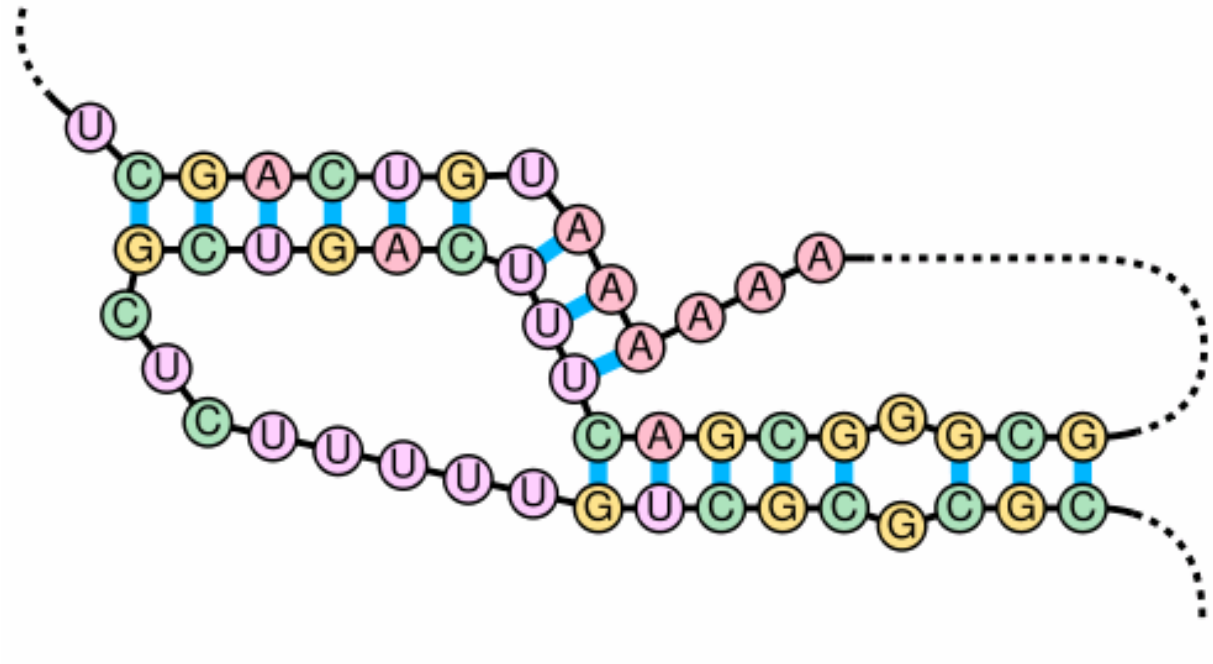

Figure 1: A naturally occurring pseudoknot in the human telomerase [22]

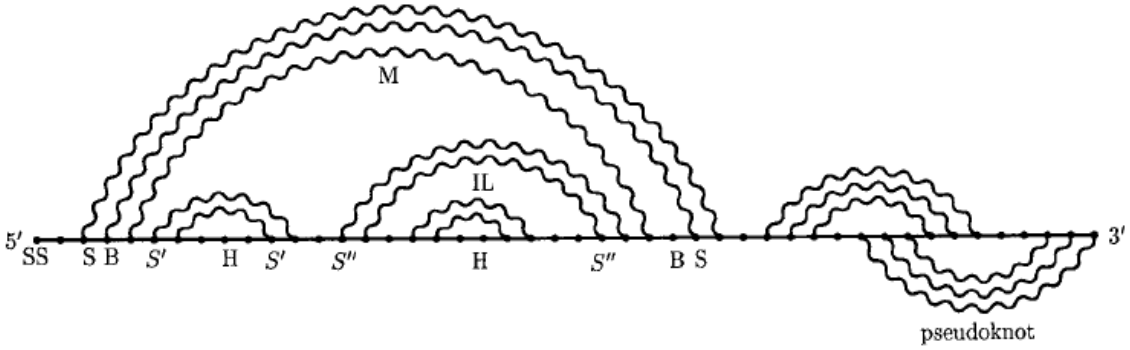

Figure 2: Planar representation of RNA structural elements. [4]

1.2 Purpose

The purpose of this project is to explore a new approach in predicting RNA secondary structures with pseudoknots. One may wonder why we care about pseudoknots in secondary structures since they are considered to be tertiary structural elements. This is because in computation based predictions, the unpaired bases from the pseudoknot will go on to form false structural elements; moreover, the error can propagate and lead to

more false structures. Therefore, prediction algorithms that do not include pseudoknot structures will not be able to achieve high accuracy.

2. Background and Related Works

This section goes over popular approaches that have been proposed or undertaken to predict the secondary structures of RNA's with pseudoknots. Prediction algorithms fall into two main categories, comparative analysis and computation based. With comparative analysis, multiple related sequences are needed since they rely on sequence covariations to predict the structure. Covariations are conserved, complementary regions determined from multiple sequence alignments [21]. Because of this requirement, we have decided to focus on computation based approaches, which relies on free energy estimations.

2.1 Stochastic Context-Free Grammar

A stochastic context-free grammar is an extension on Hidden Markov Model where each production rule is associated with a probability, just as the transition and emission rates in HMMs. Figure 3 shows an example of production rules for recognizing RNA secondary structure.

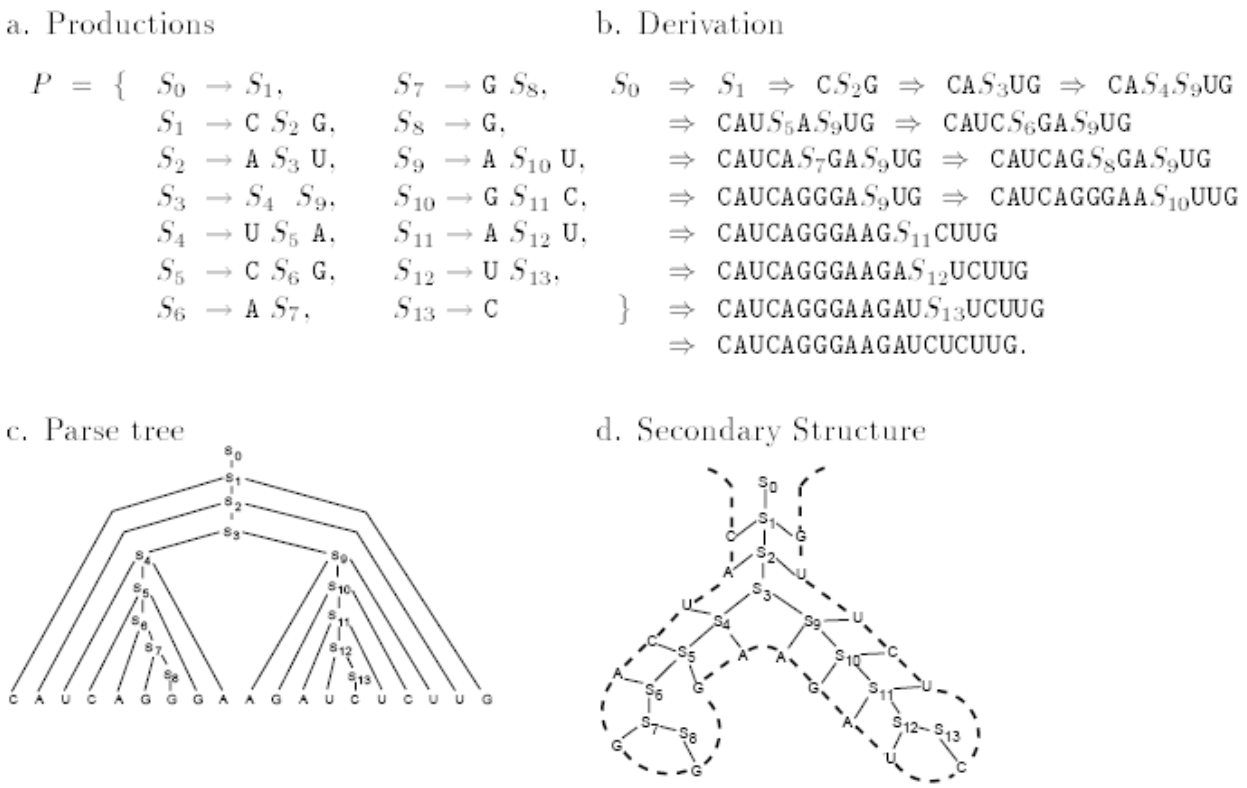

Figure 3: Production rules for capturing a secondary structure and its associated derivation [14]

Due to the non-nested characteristic of pseudoknots, it is impossible to create a context-free grammar that can process pseudoknots. As a result, two context-free grammars are necessary to recognize a pseudoknot; the first language will recognize the first half and the second language will recognize the second half, as shown in Figure 4. The main drawback of using context free grammars for recognizing pseudoknots is that the grammar needs to be tailored for certain families of RNA's. Therefore, a context-free grammar is not the appropriate approach for predicting arbitrary pseudoknots; however, it is perfect for database searches where the main purpose is to find all RNA's that have a certain secondary structure.

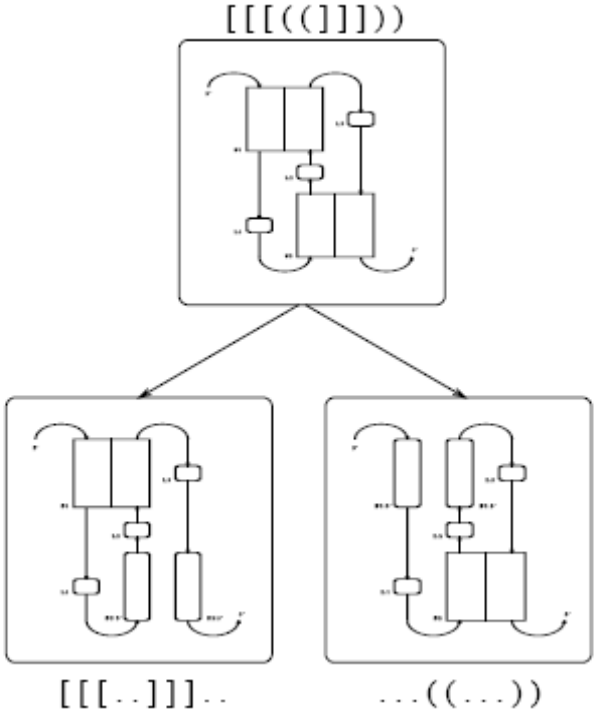

Figure 4: Pseudoknot is broken into two grammars [14]

2.2 Hidden Markov Model (context sensitive)

Traditional Hidden Markov Models are incapable of capturing the long range interaction that occurs in RNA secondary structures. Moreover, it has roughly the same descriptive power as stochastic context-free grammar, which cannot recognize pseudoknots. Context-sensitive Hidden Markov Model, however, is able to capture the long-range interaction with the help of an auxiliary memory unit. Figure 5 shows an example of two context-sensitive HMMs, the first uses a stack and the second uses a queue as the auxiliary memory unit; the first can recognize a palindrome and the second can recognize a single repeat. Figure 6 shows a structure of a pseudoknot and the corresponding context-sensitive HMM for recognizing it. As shown in Figure 6, the topology of the context-sensitive HMM is very similar to the topology of the structure that it is designed to recognize. As a result, an extremely complicated model will be needed to be able to handle the different RNA secondary structural elements, as well as the various types of pseudoknots. The larger model will take longer to process even when the input sequence is short. Furthermore, the HMM will need to be trained; it will be difficult to find training data that is diverse enough for it to recognize the various structural elements, but not enough to generate false positives.

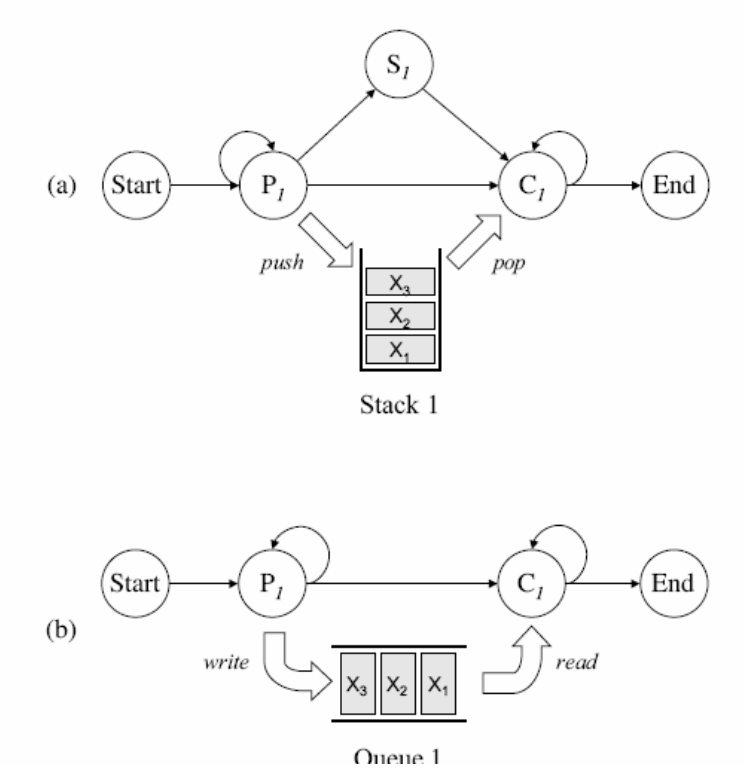

Figure 5: Context-sensitive HMM with a stack(a) and a queue(b) as the memory unit [11]

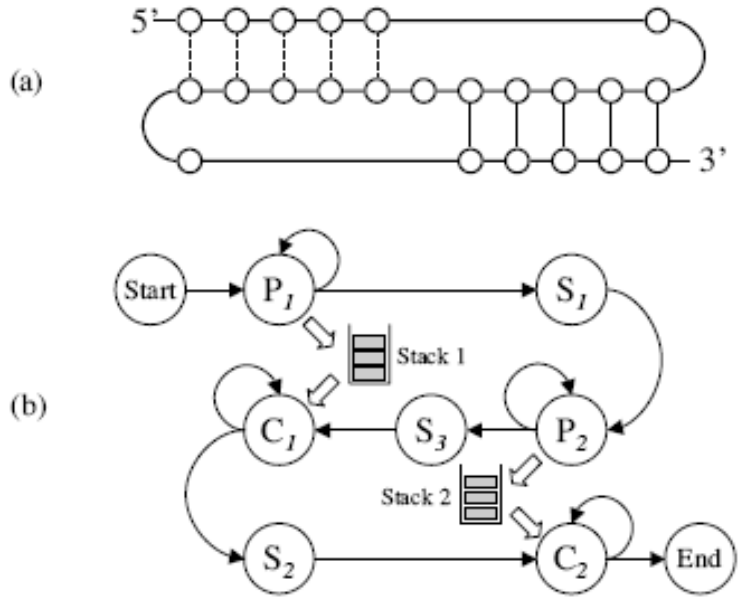

Figure 6: A pseudoknot and its corresponding context-sensitive HMM [11]

2.3 Dynamic programming algorithms

Dynamic programming algorithms are considered to be exact algorithms; they are guaranteed to provide optimal solutions. This section will go over the popular dynamic programming algorithms for RNA structural predictions; starting with the Nussinov algorithm, from which all other dynamic programming algorithms are derived.

2.3.1 Nussinov Algorithm

Nussinov algorithm is a simple dynamic programming algorithm that is capable of finding pseudoknot-free secondary structures with the maximum number of base pairs. Although it is a simple algorithm, it is the foundation for more complex ones. The Nussinov algorithm works in a fashion that is similar to the dynamic programming algorithm for pair-wise sequence alignment; the primary difference is that we are comparing the sequence against itself rather than another sequence; therefore, only half of the dynamic programming matrix is needed, which results in a triangular matrix. The lower left of the dynamic programming matrix is ignored, or initialized to 0 (for all positions where i >= j, where i is the ith row and j is the jth column). The recurrence relation below applies to every cell as we move from the main diagonal to the top-right corner, see Figure 8; the final cell at the top right corner will have the maximal score once the algorithm is completed. The trace-back can applied in $O(n^3)$, and the algorithm has a space complexity of $O(n^2)$.

Recurrence relation:

$$V(i, j) = \max\{\ V(i+1, j),\ (1)\quad V(i, j-1),\ (2)\quad V(i+1, j-1) + 1,\ (3)\quad \mathrm{Max}_{i<k<j}\,[V(i, k) + V(k+1, j)]\ (4)$$

(1) add unpaired position i onto best structure for subsequence $i+1, j$;
(2) add unpaired position j onto best structure for subsequence $i, j-1$;
(3) add i, j pair onto best structure found for subsequence $i+1, j-1$;
(4) combine two optimal substructures i, k and $k+1, j$.

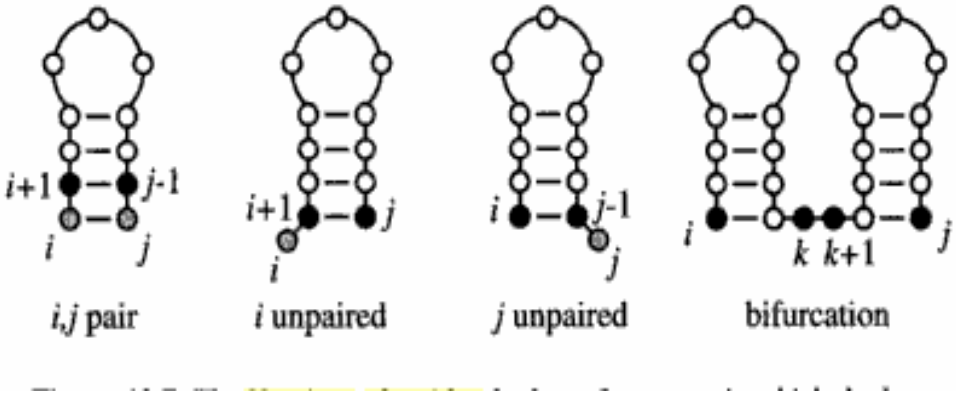

Figure 7: Recurrence relation for the Nussinov algorithm [8]

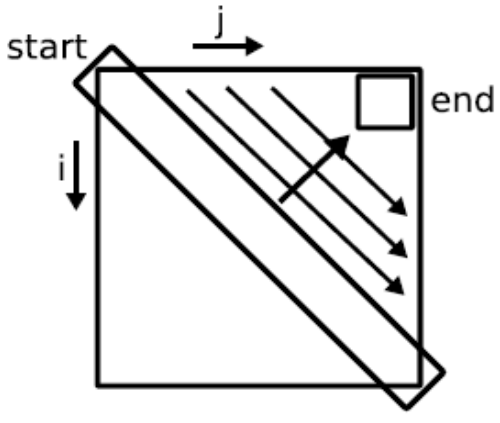

Figure 8: Dynamic programming matrix [8]

The Nussinov algorithm is a straight forward algorithm that is capable of finding an optimal solution with regards to base-pair maximization, which can result in impossible structures; this can easily be resolved by using thermodynamic energy as the scoring function.

2.3.2 Pseudoknots (Eddy and Rivas)

This is perhaps the most complicated algorithm for predicting pseudoknots; it extends upon the Nussinov algorithm to include pseudoknots and uses thermodynamic energy as the scoring function (same extension as Zuker and Stiegler [21]). First, it relies on two dynamic programming matrices to incorporate thermodynamic energies into the Nussinov algorithm. The two triangular matrices are called vx and wx; vx(i,j) is the optimal score between positions i and j in which i and j are paired; wx(i,j) is the optimal score between positions i and j regardless of whether i and j are paired or not. The purpose of using two matrices instead of one is to help with identifying and applying thermodynamic rules to different substructures (hairpin, bulge, internal loop etc.). vx is calculated as:

$$vx(i,j) = \text{optimal}\begin{cases} EIS^1(i,j) &] \quad IS^1 \\ EIS^2(i,j:k,l) + vx(k,l) &] \quad IS^2 \\ P_I + M + wx_I(i+1,k) + wx_I(k+1,j-1) &] \quad \text{multiloop} \end{cases}$$

$$[\forall k,l \quad i \leqslant k \leqslant l \leqslant j]$$

Figure 9: Recurrence relation for vx [4]

The matrix wx_I has the same recursion as wx; however, they have completely different interpretations. The purpose of wx_I is to truncate vx with regards to multiloops. $EIS^n(i_n, j_1: i_2, j_2:\ldots:i_n, j_n)$ "represents the scoring function for an IS order of n, in which i_k is paired to j_k"[4]. IS is short for irreducible surface; a surface is "any alternating sequence of continuous and wavy lines that closes on itself. An irreducible surface is a surface such that if one of the H-bonds (or secondary interaction) is broken, there is no longer a surface contained inside" [4]. In the planar representation (see figure 2), wavy lines between positions i and j in the RNA sequence indicate that the bases at i and j are paired together. Hairpin loops constitute an IS order of one since there are no other substructures if the stem that formed the loop were to unfold; stems, bulges, and internal loops are of IS order two, and multi-branch loops are of IS order larger than two. M is the penalty for generating a multi-branch loop, and P_I is the penalty for each closing pairs in a multi-branch loop. The recurrence relation for wx is:

$$wx(i,j) = \text{optimal}\begin{cases} P + vx(i,j) &] \text{ paired} \\ \left.\begin{array}{l} Q + wx(i+1,j) \\ Q + wx(i,j-1) \end{array}\right] & \text{single-stranded} \\ wx(i,k) + wx(k+1,j) \quad [\forall k, \quad i \leqslant k \leqslant j]. &] \text{ bifurcation} \end{cases}$$

Figure 10: Recurrence relation for wx [4]

The recurrence relation for wx is more or less the same as the recurrence relation in the Nussinov algorithm; P is the score for an external base-pair, and Q is the for a single-stranded nucleotide; both of which are approximate to zero in Turner's thermodynamic parameters, which is being applied by Eddy and Rivas. Performing the trace-back on wx will yield the optimal pseudoknot-free structure. In order to recognize pseudoknots, Eddy and Rivas use one-hole or gap matrices; Figure 11 shows how a pseudoknot can be captured with gap matrices.

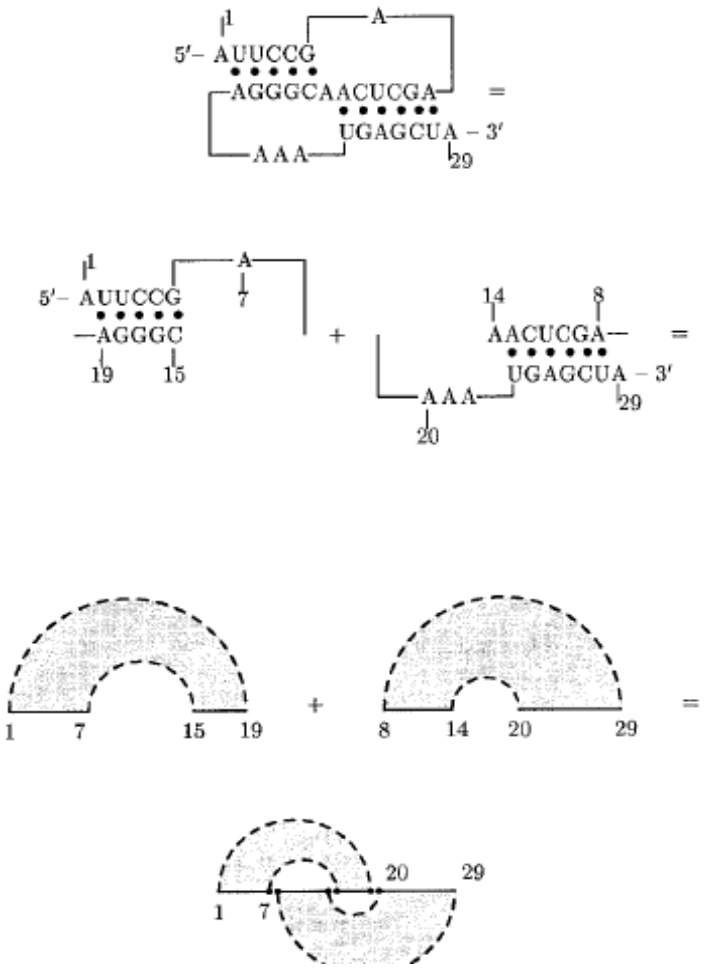

Figure 11: Using two gap matrices to capture a pseudoknot [4]

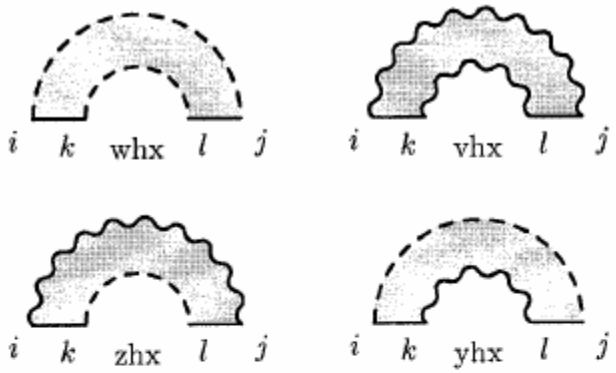

Figure 9. Representation of the gap matrices used in the algorithm for pseudoknots.

Table 1. Specifications of the matrices used in the pseudoknot algorithm

Matrix ($i \leqslant k \leqslant l \leqslant j$)	Relationship i, j	Relationship k, l
$vx(i, j)$	Paired	-
$wx(i, j)$	Undetermined	-
$vhx(i, j:k, l)$	Paired	Paired
$zhx(i, j:k, l)$	Paired	Undetermined
$yhx(i, j:k, l)$	Undetermined	Paired
$whx(i, j:k, l)$	Undetermined	Undetermined

Figure 12: Matrices used in the pseudoknot algorithm [4]

Figure 12 shows the primary matrices used in the dynamic programming algorithm. The matrices vhx, zhx, yhx, and whz are gap matrices; each matrix captures a different configuration, but the matrix whx will contain the maximal score from the configurations captured by the other three matrices. As a result, the recurrence relation for whx is extremely complex, as shown in Figure 13. The matrix wx still remains as the matrix that will yield the secondary structural with minimal free energy; therefore, its recurrence relation must take pseudoknots into account, as shown in Figure 14. Unfortunately, the expansions in gap matrices have to be truncated to keep algorithm polynomial.

$$
whx(i,j:k,l) = \text{optimal}\begin{cases}
\left.\begin{array}{l} 2 * \widetilde{P} + vhx(i,j:k,l) \\ \widetilde{P} + zhx(i,j:k,l) \\ \widetilde{P} + yhx(i,j:k,l) \end{array}\right] & \text{paired} \\
\left.\begin{array}{l} \widetilde{Q} + whx(i+1,j:k,l) \\ \widetilde{Q} + whx(i,j-1:k,l) \\ \widetilde{Q} + whx(i,j:k-1,l) \\ \widetilde{Q} + whx(i,j:k,l+1) \end{array}\right] & \text{single-stranded} \\
\left.\begin{array}{l} wx_I(i,k) + wx_I(l,j) \\ wx_I(i,r) + whx(r+1,j:k,l) \\ whx(i,j:r,l) + wx_I(r+1,k) \\ whx(i,s:k,l) + wx_I(s+1,j) \\ whx(i,j:k,s+1) + wx_I(l,s) \\ yhx(i,j:r,s) + zhx(r,s:k,l) \\ \widetilde{M} + whx(i,j:r,s) + whx(r+1,s-1:k,l) \end{array}\right] & \text{nested bifurcations} \\
\left.\begin{array}{l} G_{wh} + whx(i,s:r,l) + whx(r+1,j:k,s+1) \\ G_{wh} + whx(i,s':k,s) + whx(l,j:s-1,s'+1) \\ G_{wh} + whx(r,j:r',l) + whx(i,k:r-1,r'+1) \end{array}\right] & \text{non-nested bifurcations}
\end{cases}
$$

Figure 13: Recurrence relation for whx [4]

$$
wx(i,j) = \text{optimal}\begin{cases}
\left.\begin{array}{l} P + vx(i,j) \end{array}\right] & \text{paired} \\
\left.\begin{array}{l} Q + wx(i+1,j) \\ Q + wx(i,j-i) \end{array}\right] & \text{single-stranded} \\
\left.\begin{array}{l} wx(i,k) + wx(k+1,j) \end{array}\right] & \text{nested bifurcation} \\
\left.\begin{array}{l} G_w + whx(i,r:k,l) \\ \quad + whx(k+1,j:l-1,r+1) \end{array}\right] & \text{non-nested bifurcation}
\end{cases}
$$

Figure 14: Updated recurrence relation for wx to include psuedoknots [4]

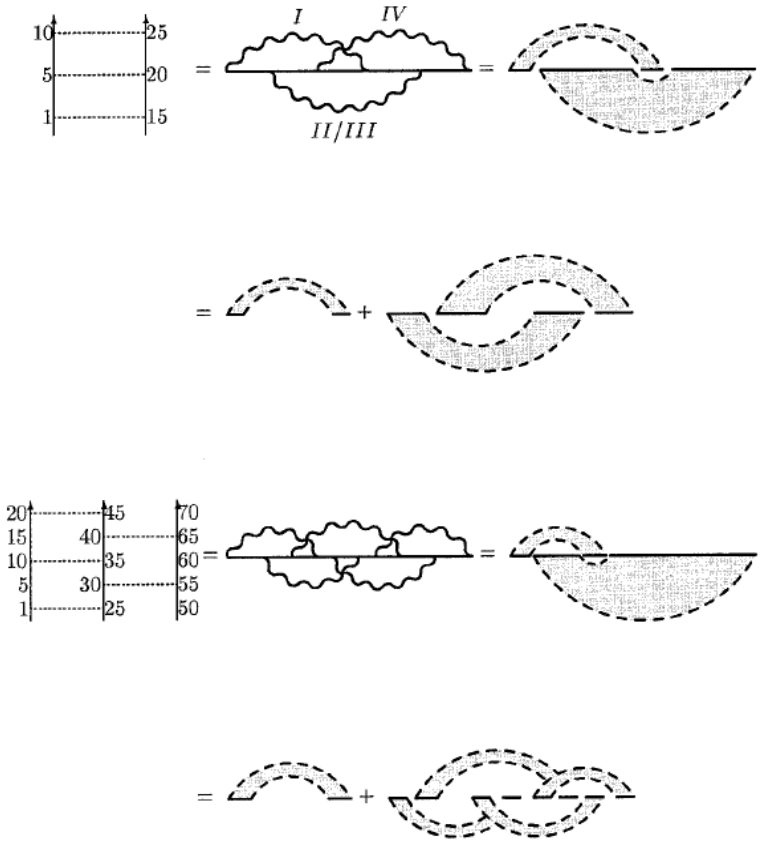

Figure 15: Non-planar pseudoknots [4]

This a very attractive algorithm for predicting secondary structures with pseudoknots since it is capable of finding a solution with the minimal free energy with respect to the energy rules being applied. Since the algorithm has a time and space complexity of $O(n^6)$ and $O(n^4)$, respectively, it cannot handle sequences longer than 150 bases in length. Another limitation of this algorithm is that it cannot recognize certain non-planar pseudoknots, such as the lower pseudoknot in Figure 14. Non-planar pseudoknots are pseudoknots that require lines to be crossed in the planar representation.

2.3.4 Pseudoknots (Reeder and Giegerich)

Reeder and Giegerich added restrictions to the pseudoknot algorithm by Eddy and Rivas to reduce its time complexity. They introduced "canonization rules" to restrict the class of pseudoknots, thereby reducing the search space. The first rule requires both strands in a helix to be of the same length; in short, helices that take parts in pseudoknots must not have bulges. With rule one, two of the eight moving boundaries of the pseudoknots, as shown in Figure 16, can be calculated in term the remaining ones. With rule one applied, boundaries f and h can be calculated as: $f = l – (e – i)$ and $h = j – (g – k)$.

Rule two states that the helices a,a' and b,b' must have maximal length and are fixed once i and l are chosen. The maximal length stacklen(i, l) can be pre-computed in $O(n^2)$. Two more moving boundaries can be removed by rule two since they can be derived from remaining terms: e = i + stacklen(i, l) and g = k + stacklen(k, l). Rule three states that if two helices overlap, when they compete for the same bases, their boundary is fixed at an arbitrary point. Rule three does not reduce any additional boundaries; it helps to reduce computation when there are overlaps between helices.

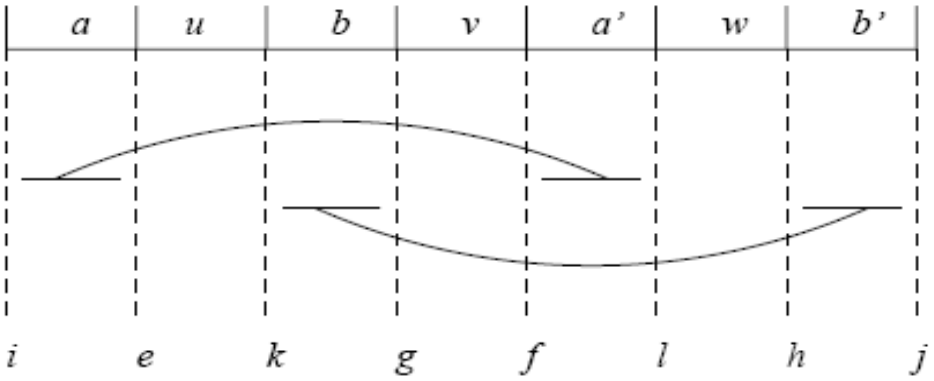

Figure 2
The boundaries of a pseudoknot. Eight moving boundaries delineating a simple recursive pseudoknot.

Figure 16: Moving boundaries of a pseudoknots [6]

Reeder and Giegerich claim that the limitations added by the canonization rules will produce results with similar free energy as the original implementation. Rule one affects length of the helices that can participate in a pseudoknot. Since bulges are ignored, "there must be at least one pair of shorter helices without bulges…which serves as a canonical representative, albeit with somewhat higher free energy" [6]. They claim that rule two is justified by energy models that strongly favor extensions; they recognized that requiring helices to have maximal extents can limit other structures from forming (chain pseudoknots) that can yield a lower energy, but they claim that "the energy of the canonical pseudoknot must be very similar" [6]. Their reasoning for rule three is that an arbitrary decision can be made when helices are competing for the same bases since the difference between having the competed bases with either helix is minimal.

With the removal of the four moving boundaries, Reeder and Giegerich was able to reduce the time and space complexity of Eddy's and Riva's algorithm down to $O(n^4)$ and $O(n^2)$, respectively. However, the dramatic improvement is not without tradeoff; the

algorithm no longer produces an optimal solution. Furthermore, since Reeder and Giergerich failed to provide, or show how much of a deviation from the minimum free energy caused by their canonization rules, solutions provided by this algorithm should fall into the same ranks as ones produced by heuristics. Their result shows comparable results to the implementation by Eddy and Rivas; however, they noticed that they are less reliable as the sequence gets longer. They attributed this to increased search space (exponential increase), and question whether the implementation by Eddy and Rivas would be able to produce better results if their algorithm was able to handle longer sequences.

2.4 Heuristics

Heuristics are algorithms that often provide good solutions, but it cannot provide any guarantees on the solutions produced. Heuristics are often applied to problems that are known to be NP complete since their time complexity is much smaller than those of approximation and exact algorithms.

2.4.1 HotKnots

HotKnots is a heuristic algorithm that is based on the assumption that substructures with low energy are likely to be in the true structure. The algorithm works by finding promising substructures, which are called "hotspots," consisting of simple stem-like structures: "stacked pairs, bulge loops containing one unpaired base, and interior loops with two (opposing) unpaired bases" [1]. First, "a set of hotspots are computed, and each hotspot in the set are used as the basis for expanding a secondary structure" [1]. The algorithm iteratively adds new hotspots to the each of existing secondary structures building a tree-like structure. Each node in the tree will contain a set of hotspots, the number of hotspots each node has is equal to its distance from the root node; the root node has no associated hotspots.

To generate the set of initial hotspots, the algorithm employs the simple local alignment algorithm by Smith and Waterman, in which the two input sequences are the same; the first ordered from 5' to 3', and the second ordered from 3' to 5'; with complementary pairs and the wobble base-pair G-U considered a match. The parameters of thermodynamic energy model are built into the algorithm; extra penalties are added for bulges, identified by deletions and insertions, and internal loops, which are identified as mismatches in the local alignment. Extra penalties are given to insertions/deletions and mismatches to prevent an alignment from getting too long. A restriction is added to prevent subsequences that are no more than three bases apart from aligning with one another; researchers have established that the smallest hairpin loop has at least three bases in the loop. Out of the set of initial hotspots, only k hotspots with lower than -0.4 kcal/mol and have more than two base pairs; in their experiment, Ren et al. chose 20 as their k. This results in a root node with 20 child node, one for each hotspot.

After the initial hotspots have been chosen, additional hotspots are selected by a different process. A dynamic programming algorithm is employed for predicting pseudoknot free secondary structures; this algorithm is similar to the mfold algorithm, by Zuker and Stiegler, with additional constrains. The additional constrains may include that certain bases must remain unpaired. Each node in the tree will invoke this algorithm with the set of hotspots that it already contained as the bases that must remained unpaired; thus, the resulting pseudoknot-free structures are those that do not overlap with existing structures. Applying this algorithm to the root node will yield the optimal pseudoknot-free structure. Of the hotspots generated by the dynamic algorithm, only those with -0.4 kcal/mol or lower are selected. A new child node will be added to the current node for each of the selected hotspots.

To limit the running time of the algorithm, it needs to determine whether the set of hotspots at a given node is promising; if it is promising, then the node will continue its expansion; otherwise, its expansion will be terminated. A set of hotspots is considered to be promising if its energy is no more than 80% higher than the energy of the root node, the pseudoknot-free structure; in addition, the energy cannot exceed 5kcal/mol. The 80% was chosen through preliminary testing. By only adding nodes to the tree for promising hotspots, the search space is reduced, which will help reduce the running time.

```
procedure HotKnots
        input: RNA sequence S
        output: list of structures SecStr(S, H_v) for all nodes v in tree T, ranked in increasing
        order of free energy

        generate an initial list L of hotspots h_1, h_2, ..., h_k ;
        create a tree T with single (root) node r and empty hotspot set;
        add k child nodes to r, with the ith child having initial hotspot set H_v = {h_i};
        build each of the k children (as described below);
        output the list of structures SecStr(S, H_v) for all nodes v in tree T, ranked in
                        increasing order of free energy;
end HotKnots


procedure build
        input: node v of tree T with hotspot set H_v
        select good hotspots that don't overlap with those in H_v and add these to list L;
        for each hotspot h in list L do
                if there is no node w of T with H_w=H_v ∪ {h}
                and SecStr(S, H_v ∪ {h}) is promising then
                   create a new node w, make it a child of v, and set H_w tp be H_v ∪ {h};
                   recursively build node w;
                end if
        end for
end build
```

Figure 17: Pseudocode for HotKnots [1]

The HotKnots algorithm run in a fashion that is very similar to branch-and-bound algorithms. This makes its time complexity difficult to compute; however, since the algorithm made recursive calls to an $O(n^3)$ algorithm, we can be sure that it has a time complexity of $\Omega(n^4)$. Despite its simplicity, HotKnots can produce results that are comparable to those of well known approaches, see Figure 18.

TABLE 2. Sensitivity and specificity of the predictions of six algorithms on our (a) short and (b) long test sequences

(a)

		Sensitivity						Specificity					
Sequence ID	Length	HotKnots	ILM	pknotsRE	STAR	pknotsRG-mfe	NUPACK	HotKnots	ILM	pknotsRE	STAR	pknotsRG-mfe	NUPACK
BR-PrP	45	0.41	**0.83**	0.5	0.33	0.33	0.41	0.38	**0.76**	0.5	0.26	0.26	0.38
BWYV	28	1	0.88	1	1	1	1	1	1	1	1	1	1
DA0260	75	**0.95**	0.68	0.69	0.5	0.77	0.77	0.77	0.68	0.68	0.5	0.85	**0.89**
DA1280	73	1	1	0.76	1	1	1	**0.95**	0.8	0.69	**0.95**	**0.95**	**0.95**
DC0010	73	1	0.9	1	0.95	1	0.8	1	1	1	0.95	1	0.94
DC0262	73	**0.85**	**0.85**	0.61	**0.85**	**0.85**	0.61	**0.78**	0.66	0.52	**0.78**	**0.78**	0.54
DD0260	76	0.28	**0.76**	0.33	0.47	0.28	0.33	0.28	**0.64**	0.29	0.4	0.28	0.26
DY4441	73	0.95	0.76	0.71	1	0.19	0.19	0.95	0.69	0.71	1	0.16	0.17
Ec-alpha	108	0.45	**0.66**	0.45	0.45	0.45	0.45	0.29	**0.4**	0.29	0.3	0.29	0.3
Ec-PK1	30	1	0.36	1	0.36	1	1	1	0.44	1	0.5	1	1
Ec-PK4	52	0.68	0.52	0.68	0.68	0.68	1	1	0.58	0.92	1	1	1
Ec-S15	67	1	0.58	0.94	0.58	0.76	0.88	**0.73**	0.47	0.64	0.62	0.68	0.71
HDV	87	0.4	1	0.46	0.6	0.96	0.63	0.44	0.88	0.46	0.7	**0.93**	0.61
HDV-anti	91	0.16	1	0.41	0.62	0.16	0.41	0.14	**0.66**	0.31	0.6	0.14	0.32
HIVRT32	35	1	1	1	0.9	1	1	1	1	1	1	1	1
HIVRT322	35	1	1	1	0.9	1	1	1	1	1	1	1	1
HIVRT33	35	1	1	1	0	1	0.9	1	1	1	0	1	1
Hs-PrP	45	0	**0.27**	0	0	0	0	0	**0.27**	0	0	0	0
LP-PKI	30	0.5	0.5	0.5	0.5	0.5	**0.8**	1	0.71	0.83	1	1	1
minimalIBV	45	**0.94**	0.88	**0.94**	0.88	**0.94**	**0.94**	0.88	0.88	**0.94**	0.93	**0.94**	**0.94**
MMTV	34	1	0.81	1	1	1	0.45	**0.91**	0.81	**0.91**	**0.91**	**0.91**	0.5
MMTV-vpk	34	1	0.54	1	1	1	1	0.91	0.54	0.91	0.91	0.91	1
pKA-A	36	1	1	1	1	1	1	**0.92**	**0.92**	**0.92**	**0.92**	**0.92**	**0.92**
satRPV	73	0.59	0.77	**0.81**	0.59	**0.81**	0.59	0.68	0.68	**0.85**	0.76	**0.85**	0.68
SRV-1	38	1	0	1	1	1	1	**0.91**	0	**0.91**	**0.91**	**0.91**	**0.91**
T2-gene32	33	1	0.58	1	1	1	1	1	0.7	1	1	1	1
T4-gene32	28	0.63	0.63	1	1	1	1	0.87	1	1	1	1	1
TMV.L	84	0.52	**0.8**	0.52	0.64	**0.8**	0.52	0.61	0.76	0.59	0.69	**0.83**	0.61
TMV.R	105	0.67	0.76	0.44	**0.85**	0.67	0.52	0.74	0.7	0.48	**0.96**	0.74	0.54
Tt-LSU-P3P7	65	**0.95**	0.8	0.9	0.6	0.85	**0.95**	1	0.69	0.85	0.75	1	1
TYMV	86	0.72	**0.88**	0.72	**0.88**	0.76	0.44	0.78	0.75	0.78	**0.88**	0.79	0.5
AVERAGE	58	**0.76**	0.74	0.75	0.71	**0.76**	0.72	**0.77**	0.71	0.74	0.74	**0.77**	0.73

Figure 18: Result comparisons for short sequences [1]

2.4.2 Iterated Loop Matching

Iterated Loop Matching, as the name implies the algorithm adds loops iteratively to form a secondary structure. This algorithm is based on the loop matching algorithm by Nussinov et al. [6], which is a dynamic programming algorithm for finding pseudoknot-free structures. The modified algorithm takes on an additional constrain to ensure that the length of every loop is at least three bases long, which means that the distance between the paired subsequences is at least three bases apart. Without this constrain, the dynamic programming algorithm by Nussinov et al. will generate impossible structures; the minimum length of a loop has been agreed by researchers to be at least three bases long. Furthermore, the loop matching algorithm, in the simplest case, assigns a score of one to Watson-Crick or G-U base pair, and zero to all other matches. This algorithm will favor structures with the maximal number of base pairs. The algorithm also allows for more complex scoring function, through comparative analysis.

The loop matching algorithm is extended to accommodate pseudoknots. Since pseudoknots can be seen as substructures with overlapping regions, and the simplest pseudoknot consists of just two overlapping structures, the loop matching algorithm can simply be ran twice to identify it. Ruan et al. pointed out that by simply taking the secondary structure outputted from the first run of the loop matching algorithm and combining it with the output of the of the second run where the bases paired in the first are ignore produces erroneous because of the false positives generated by the first run. To resolve this problem, the loop matching algorithm needs to run multiple times, and only the base-pairs with the highest score will be selected; here, they assume that substructure with highest score will likely be in the true structure.

The sketch of the algorithm is as follows [7]:

(1) Prepare a base-pairing score matrix B[1..n][1..n] from a sequence or a sequence alignment, where B[i][j] is the score for the i-th base to pair with the j-th base.
(2) Run the basic LM algorithm using matrix B to produce matrix Z and trace-back Z to get a base-pair list L.
(3) Identify all helices in L and combine helices separated by small internal loops or bulges. If no helix is identified, go to step 7.
(4) Assign a score to each helix by summing up the scores of its constitutive base-pairs. Pick the helix H that has the highest score and merge H into the base-pair list S to be reported.
(5) 'Remove' positions of H from the initial sequence. Update the score matrix B accordingly.
(6) Repeat steps 2-5 until no bases remain.
(7) Report base-pair list S and terminate.

All subsequent iterations after the initial run of the loop matching algorithm do not have to recreate everything from scratch. Much of the dynamic programming matrix can be reused. As shown in Figure 19, not every cells in the dynamic programming matrix need to be recomputed; B[i][j] only needs to be recomputed if any of the positions of the chosen structure is between i and j. A subsequent iteration can simply treat the rows and columns corresponding to the bases of the chosen structure as if they have been removed from the dynamic programming matrix. The loop matching algorithm has a time and space complexity of $O(n^3)$ and $O(n^2)$, respectively. Since the algorithm is invoking the loop matching algorithm at every iteration, it will have a time complexity of $O(n^4)$; the space complexity remains the same since the same dynamic programming matrix is being reuse. However, since the number of iterations will be substantially less than n (length of the input sequence) and the sequence length is getting shorter after every iteration, the average case time complexity is close to $O(n^3)$.

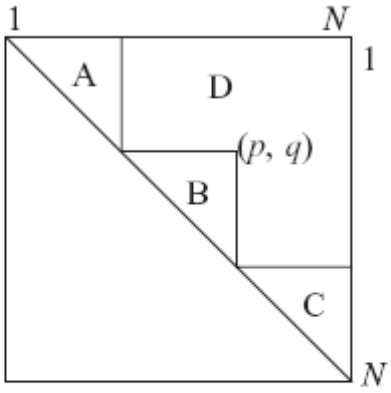

Fig. 4. Three triangle areas of the matrix do not need to be re-computed in each iteration. Let i and j be the row and column index of a cell. (p, q) is the base-pair selected in the previous iteration. A, $i < j < p$; B, $p < i < j < q$; C, $q < i < j$.

Figure 19: Regions that do not need to be recomputed (A, B, C) [7]

If the loop matching algorithm uses base-pair maximization as the scoring function, it would prefer long stems that will often yield false positives due to the energy stacking characteristics of secondary structures. Thus Ruan et al. came up with a new base-pairing score matrix that is based on comparative analysis and thermodynamics, a combination of "mutual information scores" and "helix plot scores." Mutual information scores are calculated from a given multiple sequence alignment; the scoring matrix is calculated as:

$$M_{ij} = \sum_{X,Y} f_{ij}(XY) \log \frac{f_{ij}(XY)}{f_i(X) f_j(Y)}$$

Figure 20: Scoring matrix function for mutual score information [7]

The function in Figure 20 calculates the mutual information score between positions i and j; $f_i(X)$ is the frequency of base X aligned at position i; $f_{ij}(XY)$ is the frequency of finding X at position i and Y at position j. The mutual information scores are combined with the helix plot scores, which is "formed by assigning good-pair scores to cells that represent Watson-Crick or G-U base-pairs, bad pair scores to other base-pairs and penalty scores to gaps…(good-pair score = 1, bad-pair score = 2, paired gap penalty = 3 and helix bonus = 2 x helix length)" [7]. Their combined score is calculated as:

$$B_{ij} = \alpha \times 1000 \times M_{ij} + \beta \times 20 \times HP_{ij}/N$$

Figure 21: Combined scoring functions [7]

HP_{ij} is the helix plot score, N is the number of sequences in the alignment, α *and* β are relative weight whose default values are one. The coefficient 1000 and 20 are there to bring the mutual information and helix plot scores into approximately the same integer range. The resulting scoring function is extended to include RNA-folding thermodynamics by assigning different scores to good pairs, and adding bonus for energy stacking. The extended helix plot score will be calculated as:

$$EXT\_HP_{ij} = GP_{ij} + BONUS_{ij}$$

Figure 22: Extended helix plot function [7]

The function above will replace HP in the Figure 21; GP is the good-pair score between positions i and j ("good-pair scores for G-C, A-U, and G-U are 80, 50, and 30, respectively" [7]). The bonus from energy stacking is calculated as:

$$BONUS_{ij} = 100 \times \frac{Total\ Stacking\ Energy}{\sqrt{Helix\ Length}}$$

Figure 23: Function for calculating bonus from energy stacking [7]

All in all, iterated loop matching is a simple extension of the Nussinov algorithm to include pseudoknots. The main drawback of this algorithm is that it only selects the best helix from each iteration; choosing the wrong helix (local best, but is not global best) will affect the helices produce by subsequent iterations. Furthermore, the algorithm relies on having a multiple sequence alignment available to produce accurate results.

2.4.3 Genetic Algorithms

Genetic algorithm is an optimization method inspired by concepts in biological evolutions. It works by creating new solutions at every generation, and only the more "fit" solutions will progress to the next one. It generates new solutions through mutations and crossovers, and the fitter solutions that will progress to the next generation are chosen by the selection function. Therefore, the three main steps for genetic algorithm are mutation, crossover, and selection. The selection step is the most important step since mutation and crossover steps generate new solutions, but not necessarily better ones. The general steps of genetic algorithm when applied to RNA secondary structural prediction are as follows:

1. create stem pool(s)
2. generate initial population
3. calculate "fitness" of solutions in the initial population
4. repeat
    a. perform mutation and crossover
    b. calculate "fitness" of new solutions produced by mutation and crossover
    c. select set of solution to move to next generation

   until there's no improvement and/or after a certain number of iterations [9]

Genetic algorithm will generally produce suboptimal solutions in term of free energy when compared to dynamic programming approaches. However, known RNA structures are often suboptimal structures in term of free energy; this is mainly a result of inaccurate energy models. Therefore, genetic algorithms can often predict better structures than dynamic programming algorithms. The sections below will summarize different aspects of applying genetic algorithms for RNA secondary structural prediction.

Solution representation

A candidate solution in genetic algorithm usually represented as a binary string, and is often referred to as either a genome or a chromosome. This approach requires all possible stems that can take part in the secondary structure to be identified before the algorithm starts to iterate. The length of the binary string will be as long as the number of possible stems; each position in the binary string will correspond to a stem; a one indicates that the associated stem is part of the solution, and a zero indicates exclusion. The main purpose of using a binary string representation is that it makes task such as mutations and crossovers easier to perform; however, when applied to RNA secondary structures, it makes more difficult to calculate the free energy of the solution as a whole. To overcome this problem, Shapiro and Wu used a tree structure to represent a solution instead. In their implementation, the list of stems is structured into a tree structure; the stems are ordered by their 5' ends. "[T]he stems constitute the edges of the tree, the free strands are contained in the root of the tree (i.e. the 0^{th} node), and the loops of the secondary structures are nodes of the tree" [10]. The tree structure allows the stems and loops in the secondary structure to be precisely determined, which will enable accurate energy rules to be applied.

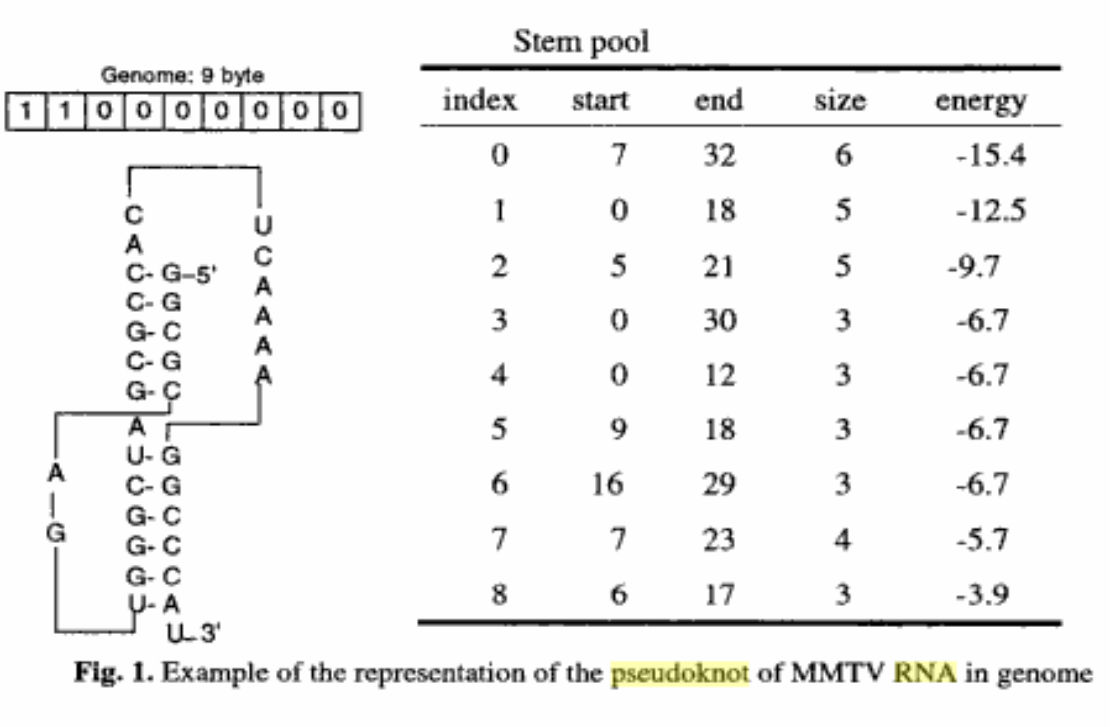

Fig. 1. Example of the representation of the pseudoknot of MMTV RNA in genome

Figure 24: Binary representation of a solution in GA [13]

Generating initial population

There are several ways to generate the initial population; however, each will either generate partial or complete solution. In the approach used by Shapiro and Wu, as well as Gultyaev et al., each solution in the initial population contains only one stem; each stem is randomly chosen based on its energy contribution. In the approach by Gultyaev et al., only stems from the first 20 nucleotides are allow to be chosen to start the initial population. In their implementation, they try to simulate the RNA folding pathway; the length of the sequence is extended after each iteration, which will allow more stems to be considered in subsequent iterations. In another approach (Lee and Han [13]), after stem have been chosen, all possible stems are added to the solution as long as they do not present any conflicts.

Mutations

Mutations are simply random changes to a solution; removal of some stems and additions of new ones. Using the binary representation, a mutation is a random bit flip; in the case where the bit flip includes a new stem, that flip is only allowed if the associated stem do not conflict with existing ones. The probability of a stem being added is based on its energy contribution; Gultyaev et al. uses the ratio between energy gained versus the

destabilizing energy of the new loop. Besides adding new stems to the solutions, perhaps even more important, mutations allow the algorithm to escape from the pitfalls of local minima. Figure 25 shows the process of mutations adopted by Gultyaev and company. First, mutation points are randomly chosen, and mutation regions of random lengths are generated. Any stems that have an intersection with the mutation regions are removed from the solution. The goal any mutation function, besides from adding new stems, is to prevent the algorithm from quickly converging to a local minimum and terminate the algorithm before a better solution can be found.

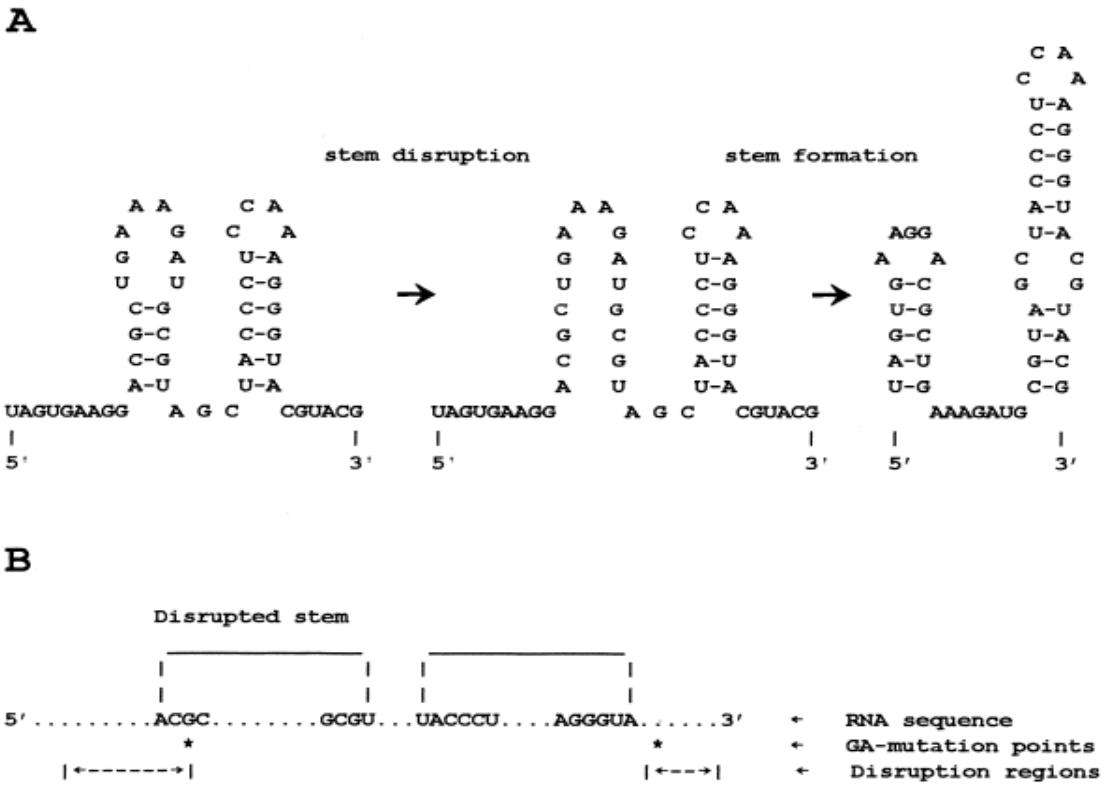

Figure 25: Mutation process [9]

Crossovers

Crossover is a process in which new solutions are produced containing parts from both parental solutions; it often combines favorable stems from parental solutions. Gultyaev and company handled crossover by combining all stems from the current population into one list and a new solution is constructed by iteratively adding each stem to it. The crossover process adopted by Shapiro and Wu is completely different since their algorithm is designed to run on a system with 16,384 processors, in which each processor is connected to eight others and represents a single solution. Each processor will choose two RNA structures from itself and its eight neighbors to perform the crossover; if the

resulting solution has a lower energy than the processor's current one, it will replace it. Since Shapiro's and Wu's algorithm is designed for massively parallel machines, it has a completely different population interaction than other genetic algorithm approaches, which are designed for single-processor machines.

Selections

Selection is the process in which solutions are chosen to move onto the next iteration by a fitness function. There are not many choices for fitness functions when it comes to deciding whether one secondary structure is better than another; the usual ones are minimum free thermodynamic energy, base-pair maximization, and maximum weighted sum. However, thermodynamic energy is generally preferred as the fitness function, since base-pair maximization often result in impossible structures. In a comparative study, Lee and Han [12] found that maximum weighted sum and minimum free yield similar solutions; perhaps, this is largely because only the stacking energies of stems were taken into account. Maximum weighted sum is similar to base-pair maximization with the exception that stems in a pseudoknot are given a better score, or a higher base count. This indicates that a simple and rough fitness function can be a good fitness function when thermodynamic information for a pseudoknot is not available. Currently, thermodynamic energy is only available for the simple H-type pseudoknot, and even that is not very accurate [17]. Selection has the important task of deciding which solution gets to persist to the next generation; its main challenge is to select favorable solutions while maintaining diversity. Through experimentations, Gultyaev et al. discovered that choosing only solutions with lowest energy for crossovers will often "resulted in a rapid convergence of all structures to the same locally favorable solution, which eventually prevented further improvement" [9]. To overcome this, structural differences between solutions were taken into account to help maintain diversity; a new parameter is assigned to each solution; this parameter is calculated as the difference between the energy of the solution and that of the best solution, divided by the number of stems in the solution.

3. Algorithm

Since the problem of RNA secondary structural prediction with pseudoknots has been proven to be NP-complete [16], approximation and heuristic algorithms should be considered instead of exact ones. Furthermore, there are no known exact algorithms for pseudoknot predictions. The dynamic programming algorithm proposed Eddy and Rivas is considered to be comparable to an exact algorithm since the truncations that they introduced only restrict complex pseudoknots whose topologies have never been seen in RNA structures. However, because of its time and space complexities ($O(n^6)$ and $O(n^4)$ [4]), this algorithm is not a viable approach for sequences over 150 bases long. Out of the different heuristics reviewed in section 2, genetic algorithm seems to be one with the most potential. Other heuristics, such as HotKnots and Iterated Loop Matching are

deterministic, greedy algorithms that are prone to local best pitfalls that can greatly reduce their search space. Since genetic algorithms are nondeterministic, they can explore solutions in a much wider search space. However, this is also its main drawback; due to the larger search space and its nondeterministic nature, genetic algorithm approaches are unlike to produce the same solution consistently. Like other heuristics, genetic algorithms cannot provide any guarantees on the quality of the solutions produced. To overcome this problem, the proposed approach uses the free energy of the optimal pseudoknot-free structure as the baseline for the genetic algorithm. Unlike previous genetic algorithm approaches, this one's primary focus is to find structures with pseudoknots that have a lower energy than that of the optimal pseudoknot-free structure. Furthermore, "for known true RNA structures [one] can usually find a set of at least 95% of the base pairs that does not contain any pseudoknot ; on the other hand, almost all RNA structures contain one or more pseudoknots" [3]. Since most of the structural elements of known secondary structures are not pseudoknots, the majority of the structural elements of an optimal pseudoknot-free structure are likely to part of the true structure. Thus generating seeds from an optimal pseudoknot-free structure will help guide the genetic algorithm towards the true structure. The subsections below go over the different aspects of the algorithm.

3.1 Overview

Figure 26 shows the overview of algorithm. Before the genetic algorithm can begin, the initial population and stem pools need to be created. The preprocessing steps need to generate the seeds and stem pools. The subsequent sections will go into how these are accomplished in details.

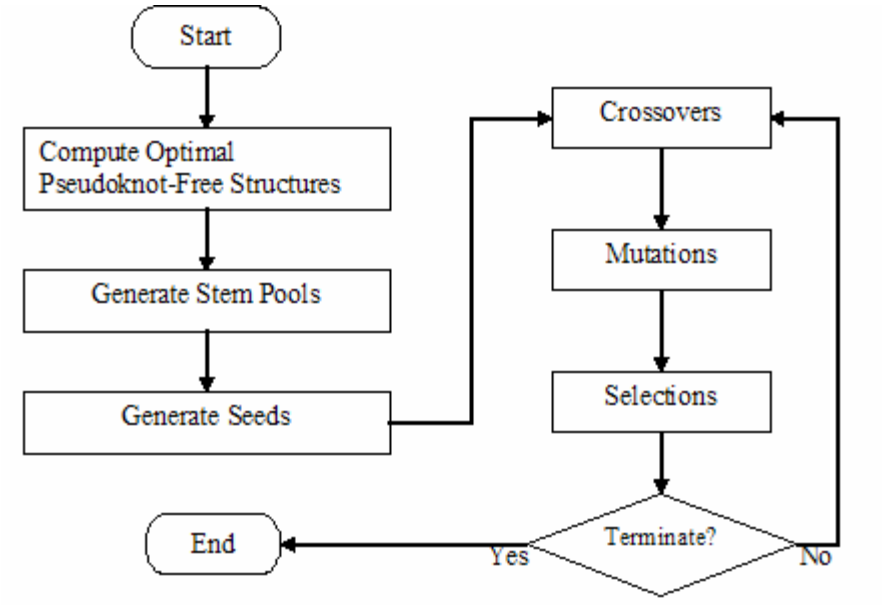

Figure 26: Flowchart of the algorithm

3.2 Generating Seeds and Stem Pools

The intent behind seeding is to provide the genetic algorithm with an initial population that will be more likely to lead to the optimal solution than one that is randomly generated. The main motivation behind seeding with pseudoknot-free structures is that the majority of structural elements of a secondary structure are not pseudoknots. Therefore, it is reasonable to assume that the majority of structural elements predicted for an optimal pseudoknot-free structure are likely to be part of the true structure. To generate the seeds, first the optimal pseudoknot-free structure needs to be predicted. The algorithm that is employed for this task is a dynamic programming algorithm by Zuker and Stiegler [21]; this is the algorithm that Zuker implemented for mfold. After the pseudoknot-free structure has been identified, N copies of it will be made, where N is the size of population that the genetic algorithm will work it. Each copy of the pseudoknot-free structure will be inserted with a different pseudoknot from pseudoknot stem pool. The stem pools will contain all the stems that will be available for the genetic algorithm. The normal stem pool will contain stems of length three and above. The pseudoknot stem pool will contain only H-type pseudoknots; each H-type pseudoknot is made up of two normal stem. The subsequent subsections will go over the pseudoknot-free prediction algorithm, and how the stem pools are generated.

3.2.1 Finding Optimal Pseudoknot-free Structures

The algorithm that is employed to generate the optimal pseudoknot-free structure is a dynamic programming algorithm by Zuker and Stiegler [21]; it is an extension of the Nussinov algorithm that uses energy minimization rather than base-pair maximization. To accomplish this task, another recurrence relation was introduced to keep track of the free energy of each sub-solution. The recurrence relation V(i, j) calculates the energy of all the possible structures between positions i and j where the bases at those positions are forced to be paired; however, if they cannot be paired (not a Watson-Crick or G-U base pair), a score of infinity is assigned. The function e(i, j) is a scoring function that calculates the free energy of having the bases at position i and j closing a hairpin loop. The function e(i, j, k, l) calculates the free energy of having the bases at position i and j closing an internal loop or bulge that is opened by the bases at positions k and l; however, if k is equal to i + 1 and l is equal to j - 1, then it becomes the stacking energy between two consecutive base pairs. The function Wi is the same as the recurrence relation W; however, it is calculated under a different context. It is similar to (7), which handles bifurcations (or branching); in the case of (3), it is for multi-branch loops since the bases at position i and j are forced to pair, forming a loop consisting of at least three stems. The scoring functions use the Nearest Neighbor energy rules described in section 3.3 along with Turner's thermodynamic parameters to handle energy calculations.

$$V(i,j) = \min \begin{cases} e(i, j) & (1) \\ e(i, j, k, l) + V(k, l) & (2) \\ Wi(i+1, k) + Wi(k+1, j-1) & (3) \\ \quad \forall\, k, l \; i <= k <= l <= j & \end{cases}$$

$$W(i, j) = \min \begin{cases} V(i, j) & (4) \\ W(i, j-1) & (5) \\ W(i+1, j) & (6) \\ W(i,k) + W(k+1,j) \quad \forall\, i<=k<=j & (7) \end{cases}$$

Figure 27: Recurrence relations for pseudoknot-free structural prediction with free energy [21]

The recurrence relation W is nearly identical to the one from the Nussinov algorithm; the only difference is that it references the recurrence relation V instead of increasing the base count when the bases at position i and j are paired. Entry (5) handles the case where the base at position i is paired with position j – 1, and the one at position j remains unpaired; similarly, entry (6) is for when i is unpaired. Entry (7) handles bifurcations where i and j are paired, but not with each other.

3.2.2 Generating Stem Pools

A dynamic programming algorithm is employed to generate a set of possible stems. For every possible stems, the algorithm will try to extend it as long as possible; the extension will terminate when it encounter an irregular pairing (not Watson-Crick base pair or the Wobble base pair G-U). During the trace back, if a stem contains consecutive G-U base pairs that gives it destabilizing energy at either ends, those base pairs will be removed. From the list of all possible stems, stems of size two are removed to reduce the number of stems. Qualifying stems of size two will greatly increase the size of the stem pool, and the loss of their energy contribution should be minimal. To further reduce the size of the stem pool any stems of size 3 that contains any base positions that is also part of the stem with the lowest energy are removed; the assumption is that the best stem will generally be part of the true structure. The recurrence relation below is derived from the recurrence relation of the pseudoknot-free prediction algorithm; it only allows energy stacking between consecutive base pairs.

$$V(i, j) = \min\{V(i+1, j-1) + e(i, j, i+1, j-1), 0\}$$

This set of possible stems forms the normal stem pool; the pseudoknot stem pool is formed with the stems from the normal stem pool. The pseudoknot stem pool only

contains the H-type pseudoknots since it is the only pseudoknot type that has an energy model, see section 3.3.2 for more details. Furthermore, the H-type pseudoknot is compact, which means that it does not enclose or overlap another stem. This allows the H-type pseudoknot to be treated as a single stem. All possible H-type pseudoknots that conform to the energy model by Gultyaev et al. are added to the pseudoknot stem pool. This does not mean that the genetic algorithm will only be able to predict the H-type pseudoknots; more complex ones can form with stems from the normal stem pool.

3.3 Energy Rules

This section will go over the energy rules that will be used to determine the free energy of a structure.

3.3.1 Nearest Neighbor Energy Rules

These are the pseudoknot-free energy rules that are implemented in mfold by Zuker [20]. As the name suggests, in the nearest neighbor energy model, the stacking energy given to a base pair is only dependent upon the base pair that precedes it. In other words, the stacking energy of a base pair with bases at position i and j is only dependent on the base pair with bases at position i+1 and j-1. In this energy model, stabilizing energies are given to stems, which are comprised of consecutive base pairs, and dangling bases; destabilizing energies are assigned to loops: hairpin loops, internal loops, bulges, and multi-branch loops. Dangling bases are unpaired bases that are adjacent to a paired base. The nearest neighbor energy model has three restrictions on the type of secondary structural elements that it can be applied to:

1. Hairpin loops need to be at least three bases long
2. Each base can pair with only one other base
3. No pseudoknot structures

The free energy of a secondary structure is the sum of its stabilizing and destabilizing energies. The nearest neighbor energy rules are used in conjunction with thermodynamic data provided by the Turner lab [26]; they perform wet lab tests to estimate energy parameters for the nearest neighbor rules. Turner's thermodynamic parameters provide pre-computed energies for loops of thirty nucleotides or less. For any loop whose length is greater than thirty bases, its energy is computed as

$$\delta\delta G = \delta\delta G_{30} + 1.75 \text{ x } RT \text{ x } \ln(ls/30) \text{ [20]},$$

where R is the universal constant and T is the temperature. The value 1.75 x RT is approximated and provided along with Turner's thermodynamic parameters as 1.079. This interpolation formula applies to hairpin loops, internal loops, and bulge loops. Table 1 shows the free energies loop size from 1 to 30; hairpin loops smaller than 3 are not

allowed. Internal loops of size 1 are not allowed because they considered to be bulge loops of size 1, see Figure 29 and 30 (bulge and internal loops). Free energies for internal loops of size 2 and 3 are placed a separate table where energies are assigned base on the configuration of the loop, as well as the nucleotides involved.

Table 1: Energies for internal, bulge, and hairpin loops from loops.dat (values between 8 and 30 are omitted) [25]

SIZE	INTERNAL	BULGE	HAIRPIN
1	.	3.8	.
2	.	2.8	.
3	.	3.2	5.6
4	1.7	3.6	5.5
5	1.8	4.0	5.6
6	2.0	4.4	5.3
7	2.2	4.6	5.8
8	2.3	4.7	5.4
	...		
30	3.7	6.1	7.7

Below are details on how free energies are calculated for the different type of loops.

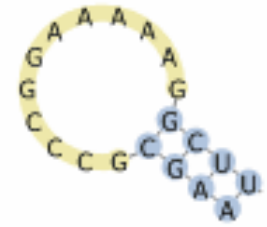

Figure 28: Hairpin loop of size 12

The free energy of a hairpin loop is computed as

$$\delta\delta G_H = \delta\delta G_{H1} + \delta\delta G_{H2} + \delta\delta G_{H3} + \delta\delta G_{H4}\ [23],$$

where $\delta\delta G_{H1}$ is the energy contributed by the size of the loop; $\delta\delta G_{H2}$ is the stacking energies from the two dangling bases inside the loop (only applies to loop whose length is greater than four); $\delta\delta G_{H3}$ is the bonus free energy that applies to certain loops of size 3 and 4 (tri-loops and tetra-loops); $\delta\delta G_{H4}$ is any additional energy that is not already accounted by the previous values. Hairpin loops consisting of only cytosine bases are given additional energy, as well ones with a guanine and uracil (G-U) closing base pairs that is preceded by guanines on either sides.

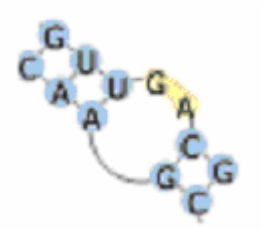

Figure 29: A bulge loop of size 2

The free energy of a bulge loop is just simply the size of the loop in most cases. For bulge loops of size 1, the unpaired base and the opening and closing base pairs determine its energy.

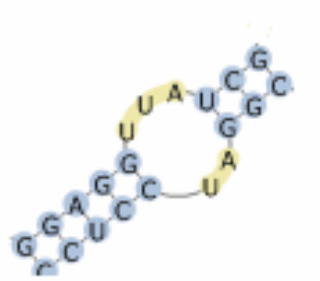

Figure 30: An asymmetric internal loop of size 5

The free energy for an internal loop is computed as

$$\delta\delta G_I = \delta\delta G_{I1} + \delta\delta G_{I2} + \delta\delta G_{I3} + \delta\delta G_{I4}\ [23],$$

where $\delta\delta G_{I1}$ is free energy contributed by the loop; $\delta\delta G_{I2}$ and $\delta\delta G_{I3}$ are stacking energies between the dangling bases and the two base pairs that formed the loop; $\delta\delta G_{I4}$ is the penalty for asymmetric loops.

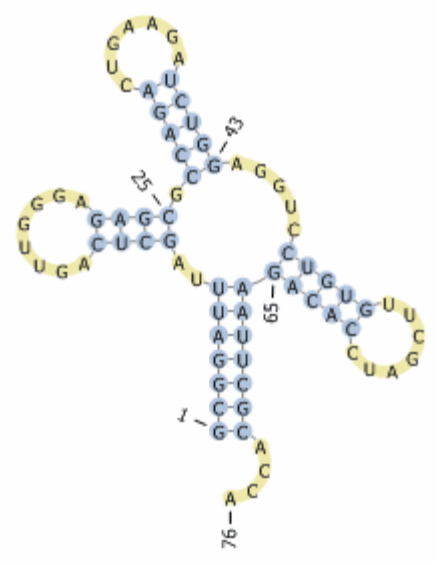

Figure 31: A multi-branch loop with 4 stems and 8 free bases

The free energy of a multi-branch loop is computed as

$$\delta\delta G(L) = a + b \times ls(L) + c \times ld(L) + \delta\delta Gstack \text{ [23]},$$

where *a* is the offset for starting a multi-branch loop; *b* is the free base penalty; *c* is helix penalty; ls(L) is the number of free bases in the loop; ld(L) is the number of stems in the loop; δδGstack is the energy stacking from dangling bases in the loop. The constants *a*, *b*, and *c* are estimated by Turner's thermodynamic parameters as 9.0 kCal/mol, 0, and 0, respectively.

3.3.2 Pseudoknot Energy Rules

One of the primary challenges of RNA secondary structural prediction with pseudoknots is the lack of thermodynamic information available for pseudoknots. The only energy model that is available for pseudoknots only applies to the most simple and common one, the H-type pseudoknot (hairpin pseudoknot). As shown in Figure 32, the H-type pseudoknot consists of only two stem-loop (hairpin) structural elements where the loop from the first stem help forms the second stem.

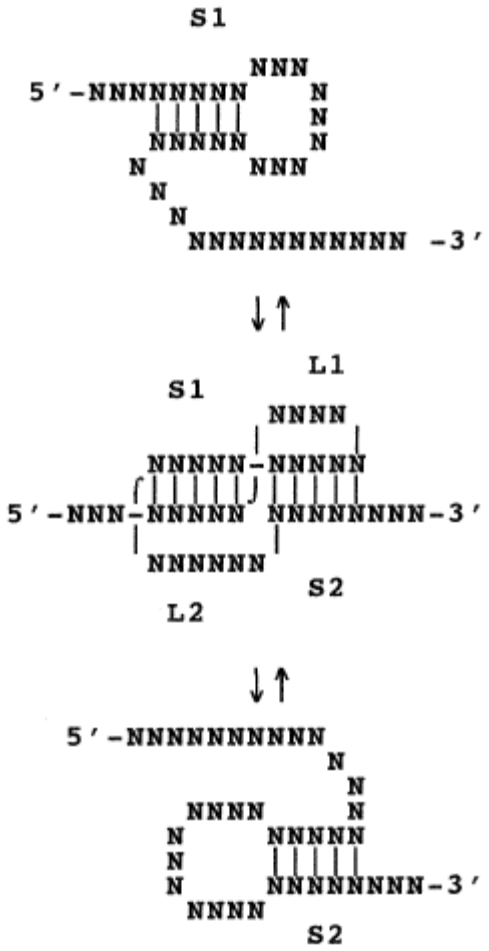

Figure 32: An H-type pseudoknot forms from two stem-loops [17]

The energy contribution of a pseudoknot is the same as any other structures; it gains its stabilizing and destabilizing energies from the stems and loops that forms it. The energy estimation of H-type pseudoknot loops is computed by Gultyaev et al. as

$$G_{L1} = A_{deep}\,(S2) + 1.75 \times RT \times \ln(1 + N - N_{mindeep}(S2))$$

$$G_{L1} = A_{shallow}\,(S2) + 1.75 \times RT \times \ln(1 + N - N_{minshallow}(S1)) \quad [17],$$

where G_{L1} is the energy of the L1 loop and G_{L2} is the energy of the L2 loop from Figure 32. A_{deep} (S2) and $A_{shallow}$(S1) are estimated values Gultyaev et al. derived from the Nearest Neighbor rules and Turner's thermodynamic parameters. The remaining terms are for interpolation purposes when the size of the loop is not listed in Table 2. $N_{mindeep}$(S2) is minimum size of the deep loop L1; for instance, if the stem S2 has a size of 3 or 4, then the minimum size of the deep loop is 2 or 1, respectively. Similarly,

$N_{minshallow}$(S1) is the minimum size of the shallow loop L2. N is the length of the loop. R is the universal gas constants, and T is the temperature.

Table 2. Energy approximations for H-type pseudoknot loops [17]

stem size (bp)	loop size (nt)										
	1	2	3	4	5	6	8	10	15	20	30
(S2)	deep groove (L1)										
2	—	—	—	—	—	—	—	—	—	—	—
3	—	5.0	5.7	6.2	6.5	6.7	7.1	7.4	7.8	8.2	8.6
4	4.7	5.4	5.9	6.2	6.4	6.6	6.9	7.2	7.6	7.9	8.4
5	4.2	4.9	5.4	5.7	5.9	6.1	6.4	6.7	7.1	7.4	7.9
6	3.5	4.2	4.7	5.0	5.2	5.4	5.7	6.0	6.4	6.7	7.2
7	3.5	4.2	4.7	5.0	5.2	5.4	5.7	6.0	6.4	6.7	7.2
8	—	4.2	4.9	5.4	5.7	5.9	6.3	6.6	7.0	7.4	7.8
9	—	4.7	5.4	5.9	6.2	6.4	6.8	7.1	7.5	7.9	8.3
10	—	5.0	5.7	6.2	6.5	6.7	7.1	7.4	7.8	8.2	8.6
(S1)	shallow groove (L2)										
2	—	—	—	—	—	—	—	—	—	—	—
3	—	3.5	4.2	4.7	5.0	5.2	5.6	5.9	6.3	6.7	7.1
4	—	—	4.2	4.9	5.4	5.7	6.1	6.4	7.0	7.3	7.8
5	—	—	—	4.7	5.4	5.9	6.4	6.8	7.4	7.8	8.3
6	—	—	—	5.0	5.7	6.2	6.7	7.1	7.7	8.1	8.6
7	—	—	—	—	5.2	5.9	6.7	7.1	7.8	8.2	8.7
8	—	—	—	—	5.4	6.1	6.9	7.3	8.0	8.4	8.9
9	—	—	—	—	—	5.6	6.8	7.3	8.1	8.4	9.0
10	—	—	—	—	—	5.7	6.9	7.4	8.2	8.6	9.2

Furthermore, the energy estimation only applies to H-type pseudoknots that have a junction size of 1 or 0. The junction is the gap between the stems that form the pseudoknot in planar representation; it is labeled as L2 in Figure 33.

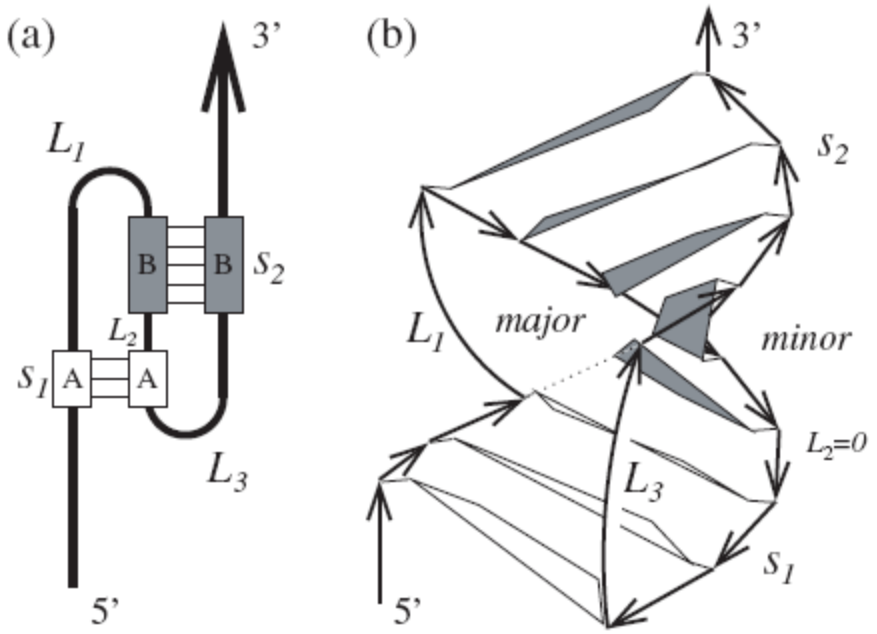

Figure 33: H-type pseudoknot in planar representation. (b) Same structure in 3D [18]

In order to allow more complex pseudoknots to form, we have come up with a simple rule for complex pseudoknots that will still allow the nearest neighbor rules to be applied. Simply calculate the energy for the complex pseudoknot as separate pseudoknot-free structures; Figure 34 and 35 show how a complex pseudoknot is decomposed into separate unknotted structures. The free energy will be higher when compared to energy estimations for the H-type pseudoknots; however, if the structure is stable enough, the stacking energy from the stems should be able to overcome it. This rule only applies to complex pseudoknots; the estimation by Gultyaev et al. will be used for the H-type pseudoknots.

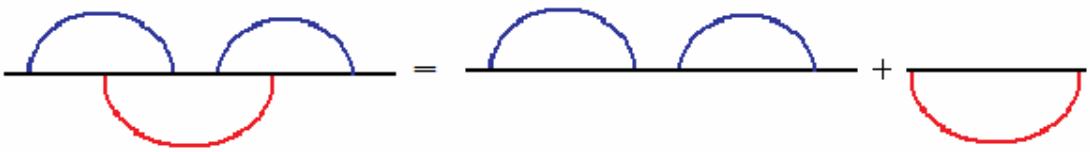

Figure 34: A complex pseudoknot (kissing hairpins) decomposes into 3 separate hairpin loops.

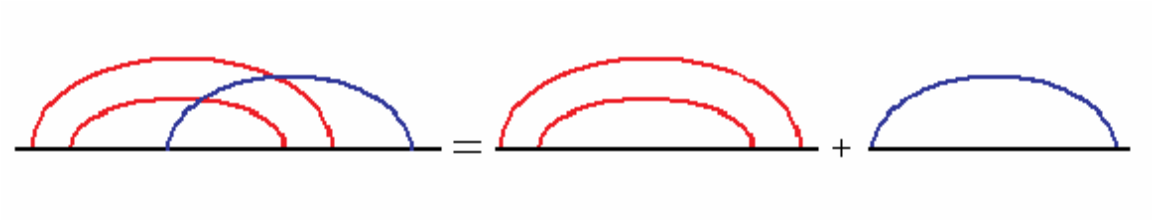

Figure 35: A complex pseudoknot decomposes into a two hairpin loops and an internal loop.

3.4 Genetic Algorithm

This section will go over how different aspects of the genetic algorithm are handled.

3.4.1 Solution Representation

In genetic algorithms, a solution is generally represented as a binary string, which will allow mutations and crossovers to be easily carried out. It is possible for a secondary structure to be represented as a list of stems in a binary string (see section 2.3.4); however, the free energy of the loops formed by the stems cannot be computed with the binary representation. Consequently, genetic algorithm approaches that do use the binary string representation ([9], [13], and [19]), only use the stacking energies of the stems to determine the energy of the structure. Therefore, my approach uses a tree structure instead of the binary string representation; Shapiro and Wu [10] also use a tree structure for the same reason. The tree structure will allow loops to be properly identified, which will enable to the correct energy rules to be applied. Another advantage of using the tree structure is speedy detection of stem collisions and overlaps. Figure 36 shows an example of how a secondary structure in the planar representation is mapped to the tree structure. In this tree structure, complex pseudoknots are treated as separate substructures; the H-type pseudoknot is treated as a single stem that does not contain nor overlap with any other stems.

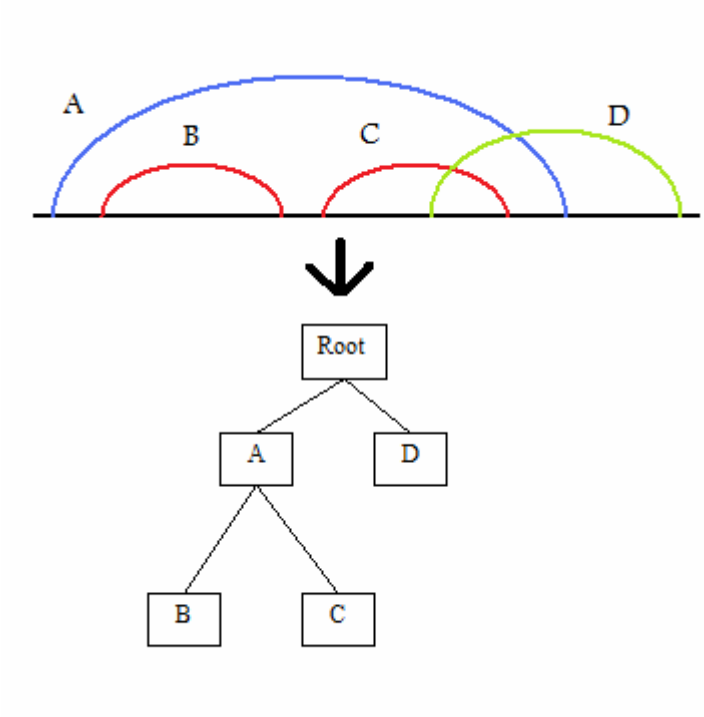

Figure 36: Tree representation of a secondary structure

3.4.2 Mutations

There are two types of mutations possible for a candidate secondary structure, deletion and addition mutations. A deletion mutation will randomly remove a stem from the structure, and an addition mutation will randomly insert a new one. With the binary string representation, a deletion of a stem is accomplished by simply flipping a bit from 1 to 0. Additions are a bit more complex since it requires colliding stems (stems that occupy the same space as the new stem) to be removed. This makes deletions unnecessary since it is an inherent part of additions. Since the genetic algorithm being employed is a steady state genetic algorithm, mutations will not be performed directly on each candidate solution; it will be performed on a copy instead. It starts out by randomly selecting a mutation point along the length of the RNA sequence. After which, a mutation range will be randomly selected (between 1 and 4); the mutation range will extend in both the 5' and 3' directions from the mutation point. Once the mutation range has been established, all stems that occupy the mutation range from both stem pools are retrieved. From the retrieved stems, one is chosen in a roulette wheel fashion, where stems with lower energy will have a higher chance of being selected. The primary task of mutations is to introduce new stems into the population.

3.4.3 Crossovers

Unlike mutations, crossovers generate new variants from the existing elements in population rather than adding new ones. With the binary string representation, crossover just simply cut the two strings in halves at an arbitrary point and swaps them. However, after the crossover has been performed, colliding stems in the newly created structure need to be resolved. Since a tree representation is being used instead of the binary string, crossover is handled differently; however, its concept is still preserved. Crossover is performed on each candidate solution with randomly selected one. The approach I have taken is similar to the one used by Gultyaev et al. [9]; the stems from both solutions will be sorted by their stacking energies and iteratively added to form a new one. The only difference is that Gultyaev et al. used all stems in the current population, which yields only one new variant per iteration of the genetic algorithm.

3.4.4 Selections

Selection has the important task of calculating the fitness of each candidate structure, and deciding which structures will persist to the next iteration. Generally, the effectiveness of a genetic algorithm is determined by how it handles selection. There are a few different flavors of genetic algorithms, and it is decided by the type of selection that is implemented. A genetic algorithm is considered to be generational if an entirely new population is selected at the end of each generation (an iteration of the genetic algorithm). Generational genetic algorithms tend to change more rapidly. On the other side of the spectrum are steady state genetic algorithms, where a portion of the current population will persist to the next generation. As a result, changes in a steady state genetic algorithm occur more slowly. According to the comparative study by Lee and Han [12], steady state genetic algorithms produce better results than generational ones for secondary structure predictions. Thus this is the approach we have taken.

Minimum free energy is used to measure the fitness of a candidate structure, see section 3.3 for more details. The challenge of the selection step is to keep the low energy structures to the next iteration while maintaining diversity to avoid converging to a local best too rapidly. In this pursuit, selection has been divided into three equal segments. The first segment will contain the best 33.3% of the current population. The second segment will contain randomly selected structures from the top 30% of the new candidate structures generate through mutations and crossovers. The last segment will consist of randomly selected structures from the remaining structures. The first segment will ensure that the top structures from the current population will move onto the next iteration. The second segment will ensure that the top structures from the variants generated through mutations and crossovers will have a place in the population. The function of the last segment is to maintain diversity. Furthermore, selection will only allow structures with at least one pseudoknot to be in the first two segments.

4. Results

The algorithm described in section 3 has been implemented in Java and tested with various known pseudoknot structures from PseudoBase [24], which is currently the only repository available for known pseudoknot structures. All secondary structure visualizations were rendered by PseudoViewer [25]. Table 3 contains a set of short sequences that were used for testing; these specific sequences were chosen because they are also used by Ren et al. to compare their HotKnots algorithm against existing ones [1].

Table 3. Test sequences from PseudoBase

PseudoBase ID	RNA Type	Sequence id	organism
PKB207	mRNA	Tb_PrP	Bos Taurus (cow)
PKB71	mRNA	Ec_apha	E.coli
PKB49	tmRNA	Ec_PK1	E.coli
PKB52	tmRNA	Ec_PK4	E.coli
PKB72	mRNA	Ec_S15	E.coli
PKB206	mRNA	Hs_PrP	Homo sapiens
PKB67	tmRNA	Lp_PK1	Legionella pneumophila
PKB5	Viral tRNA-like	TYMV	Turnip yellow mosaic virus
PKB173	Ribozymes	satRPV	Sat. cereal yellow dwarf virus-RPV RNA
PKB73	mRNA	T2_gene32	T2 bacteriophage
PKB74	mRNA	T4_gene32	T4 bacteriophage
PKB77	Ribozymes	Tt-LSU_P3/P7	Tetrahymena thermophila

The accuracy of the predictions is measured by their sensitivities and specificities. Specificity is defined as the ratio between the number base pairs from the predicted structure that matches the known structure (true positives) and the total number of predicted base pairs. Sensitivity is defined as a ratio between the number true positives and the number of base pairs from the known structure. These are the same definitions used by Ren et al. to compare their HotKnots algorithm against five other well known ones. Tables 4 and 5 show how the seeded genetic algorithm matches up with those six algorithms. ILM is the Iterated Loop Matching algorithm by Ruan et al. [7]; pknotsRE is the dynamic programming algorithm by Rivas and Eddy [4]; STAR is a package that uses the genetic algorithm by Gultyaev et al. [9]; pknotsRG is a dynamic programming algorithm by Reeder and Giegerich [5]; NUPACK is a dynamic programming algorithm by Dirks and Pierce. The sensitivity and specificity values for the seeded genetic algorithm are averaged from a 100 different runs for each sequence, in which the structure with the lowest energy is compared with the known structure.

Table 4. Sensitivity of predictions compared with 6 other algorithms

PseudoBase ID	Length	HotKnots	ILM	pknotsRE	STAR	pknotsRG-mfe	NUPACK	Seeded GA
PKB207	45	0.41	0.83	0.5	0.33	0.33	0.41	**1**
PKB71	108	0.45	**0.66**	0.45	0.45	0.45	0.45	0.56
PKB49	30	**1**	0.36	**1**	0.36	**1**	**1**	0.95
PKB52	52	0.68	0.52	0.68	0.68	0.68	**1**	0.82
PKB72	67	**1**	0.58	0.94	0.58	0.76	0.88	0.93
PKB206	45	0	0.27	0	0	0	0	**0.53**
PKB67	30	0.5	0.5	0.5	0.5	0.5	**0.8**	0.75
PKB5	86	0.72	**0.88**	0.72	**0.88**	0.76	0.44	0.77
PKB173	73	0.59	0.77	**0.81**	0.59	**0.81**	0.59	0.72
PKB73	33	**1**	0.58	**1**	**1**	**1**	**1**	**1**
PKB74	28	0.63	0.63	**1**	**1**	**1**	**1**	**1**
PKB77	65	**0.95**	0.8	0.9	0.6	0.85	**0.95**	0.85
Average	55	0.66	0.61	0.71	0.58	0.68	0.71	**0.82**

Table 5. Specificity of predictions compared with 6 other algorithms

PseudoBase ID	Length	HotKnots	ILM	pknotsRE	STAR	pknotsRG-mfe	NUPACK	Seeded GA
PKB207	45	0.41	0.76	0.5	0.26	0.26	0.38	**1**
PKB71	108	0.29	**0.4**	0.29	0.3	0.29	0.3	0.41
PKB49	30	**1**	0.44	**1**	0.5	**1**	**1**	**1**
PKB52	52	1	0.58	0.92	1	1	**1**	**1**
PKB72	67	**0.73**	0.47	0.64	0.62	0.68	0.71	0.71
PKB206	45	0	0.27	0	0	0	**0**	**0.43**
PKB67	30	1	0.71	0.83	1	1	**1**	1
PKB5	86	0.78	0.85	0.78	**0.88**	0.79	0.5	0.75
PKB173	73	0.68	0.68	**0.85**	0.76	**0.85**	0.68	0.74
PKB73	33	**1**	0.7	**1**	**1**	**1**	**1**	**1**
PKB74	28	0.87	1	**1**	**1**	**1**	**1**	**1**
PKB77	65	**1**	0.69	0.85	0.75	**1**	**1**	**1**
Average	55	0.73	0.62	0.72	0.67	0.73	0.71	**0.83**

Overall, the seeded genetic algorithm yields a higher sensitivity and specificity average than any other algorithms for the chosen test sequences. However, the sequence PKB206 and PKB207 are the only sequences in which the seeded genetic algorithm is the clear winner. This is because their known structures yield higher free energy than the optimal pseudoknot-free structure. The primary focus other algorithms is to find the structure with the lowest free energy, whereas the seeded genetic algorithm's primary focus is to find a low free energy structure that contains at least one pseudoknot. Figure 37 shows the secondary structures for PKB206 predicted by pknotsRG and the seeded genetic algorithm. Tables 6 and 7 show the algorithm's performance for short and long sequences. Unfortunately, there are no long sequences available on PseudoBase; the long sequences listed in table 7 are from PseudoViewer's homepage. These sequences are not

ideal candidates since most of their structural elements are pseudoknots; therefore, the assumption that only a small part a secondary structure is made up of pseudoknots does not apply. For these sequences, seeding with pseudoknot-free structures has an adverse effect since the initial population is leading the genetic algorithm in the wrong direction. This is not a problem for shorter sequences since the search space is much smaller; it only takes a few generations for the genetic algorithm to recover.

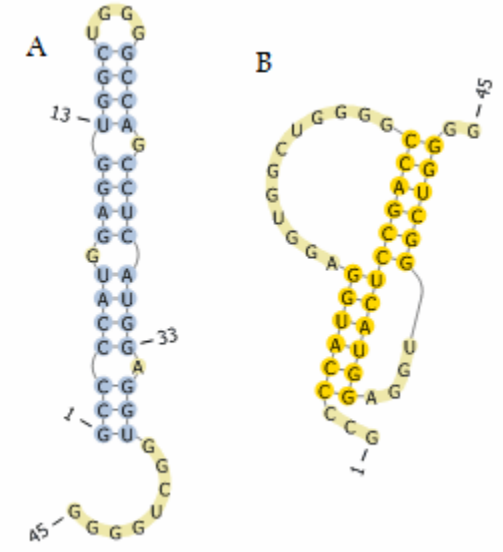

Figure 37: (A) PKB206 structure predicted by pknotsRG. (B) PKB206 structure predicted by GA and is also the known structure.

Table 6. Sensitivity and specificity of short test sequences (average from 100 runs)

sequence	length	sensitivity	specificity	Time (in milliseconds) max	min	average
PKB276	73	0.54	0.57	1609	478	685
PKB207	45	1.00	1.00	662	174	249
PKB74	28	1.00	1.00	690	196	263
PKB72	67	0.93	0.71	523	484	512
PKB71	108	0.56	0.41	1297	433	726
PKB206	45	0.53	0.43	27159	267	733
PKB67	30	0.75	1.00	630	173	240
PKB49	30	0.95	1.00	603	132	186
PKB52	52	0.82	1.00	1240	361	692
PKB173	73	0.72	0.74	1338	394	537
PKB77	65	0.85	1.00	892	309	411
PKB193	82	0.24	0.36	1157	417	581
TYVM(05)	86	0.77	0.75	7385	173	430
PKB243	121	0.57	0.63	1411	595	865
PKB76	89	0.91	0.72	670	571	608

Table 7. Sensitivity and specificity of long test sequences (average from 100 runs)

sequence	length	sensitivity	specificity	Time (in milliseconds) max	min	average
TMV Sat	421	0.15	0.22	10214	3306	5258
E.coli tmRNA	363	0.19	0.33	4651	1852	2681
TMV	214	0.33	0.47	2204	853	1302
DiGIR1	184	0.26	0.37	2633	651	1093
HDV	216	0.37	0.39	2954	897	1446

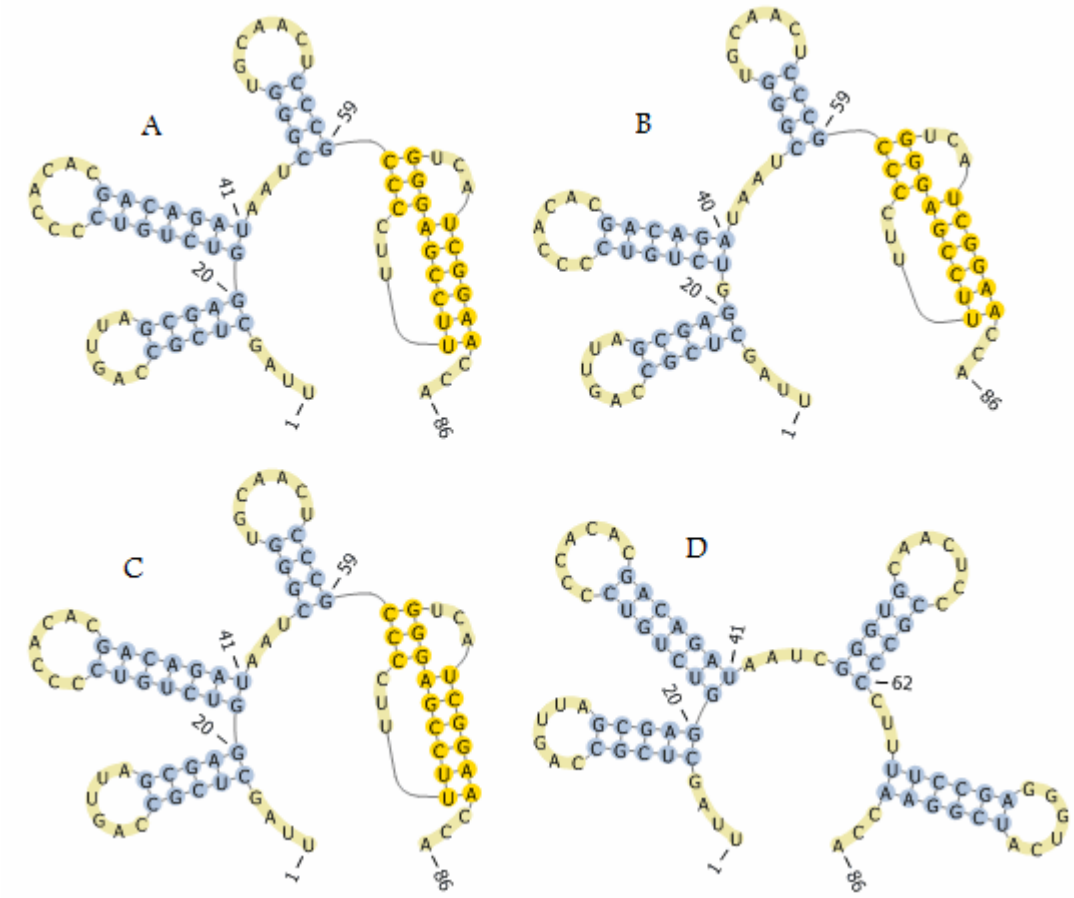

Figure 38: (A) and (B) are TYMV structures predicted by the seeded GA, (B) is also the known TYVM structure. (C) TYMV structure predicted by pknotsRG. (D) Pseudoknot-free structure predicted by the seeding algorithm.

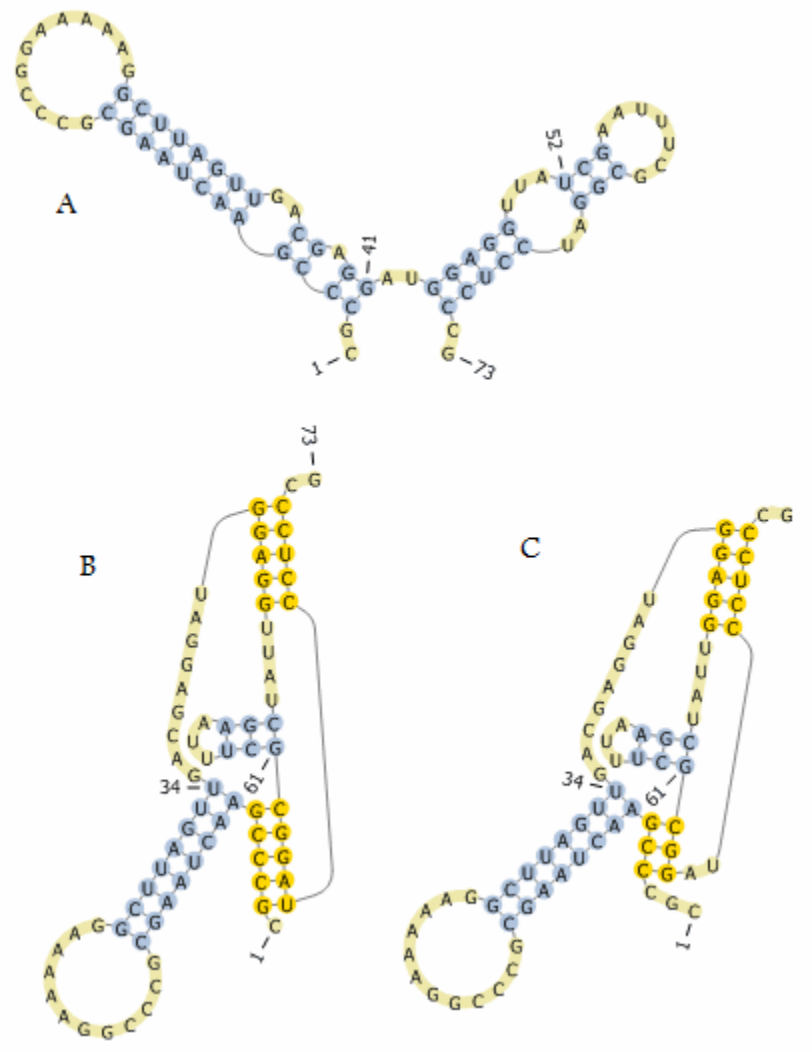

Figure 39: (A) PKB173 structure predicted by pknotsRG and seeding algorithm. (B) Known PKB173 structure. (C) Best structure predicted by seeded GA.

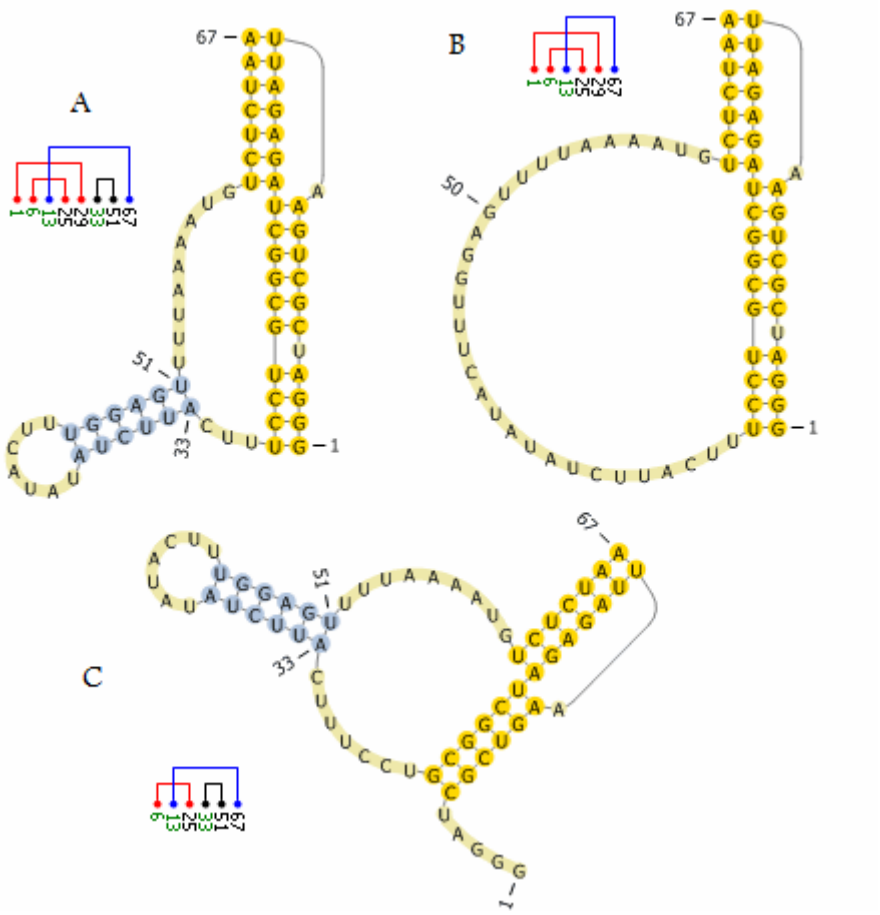

Figure 40: (A) PKB72 structure predicted by Seeded GA. (B) known PKB72 structure. (C) PKB72 structure predicted by pknotsRG.

The structure that a genetic algorithm will be able to predict relies heavily on the contents of the stem pools. As shown in Figure 39, the true structure of sequence PKB173 contains an irregular base pair between C and A (position 3 and 65). None of the computation-based prediction algorithms will be able to identify PKB173's true structure since current energy models do not allow irregular base pairings. Furthermore, to improve accuracy, stems of all sizes need to be considered. However, with the larger stem pools comes a larger search space, which leads to longer execution time and wider differences in results between multiple runs.

The current energy model for pseudoknot is too restrictive since it is only applicable to H-type pseudoknots. Consequently, algorithms that only use the existing model will only be able to predict the H-type pseudoknot, which consists of only two stems. Figure 40 shows a pseudoknot that consists of three different stems. Although, pknotsRG is capable of predicting complex pseudoknots, it prefers simple pseudoknots over complex ones due to its restrictive "canonization" rules. The addition of the new energy rule to

allow complex pseudoknots to form seems to be a good starting point for estimating their free energies.

5. Future Work

The primary challenge in RNA secondary structural predictions is our lack of knowledge of its folding dynamics. Besides pseudoknots, another obstacle for computation based prediction is irregular base pairing. Perhaps, covariations can be included in the stem generation process to allow stems with irregular base pairings. One of the main challenges of genetic algorithms is to prevent rapid convergence of the population that may prevent better solutions from forming. Clustering algorithms can be employed to group candidate structures based on their topology or stems. Clustering will allow the selection process to keep the top scoring and representative solutions from each clusters; this will help the population to maintain its diversity.

6. Conclusions

With our current knowledge of RNA folding, pseudoknots in particular, it is impossible for any computation-based prediction algorithms to produce reliable solutions. Approximation approaches that use dynamic programming algorithms are not efficient at finding pseudoknots because of inaccurate energy estimations. Until we have more accurate energy models, heuristic approaches are more appropriate. The primary advantage that genetic algorithm has over other heuristics is that it is able to explore a much wider search space. However, this is also one of its main drawbacks. With the larger search space, it will take longer to complete and the solutions produced by each run will vary more widely. Since only a small portion of a secondary structure is comprised of pseudoknots, seeding with optimal pseudoknot-free structures will help narrow the search space and guide the genetic algorithm toward the true structure.

The proposed algorithm shows promising results. It out performs the current popular approaches for the set of test sequences from PseudoBase. This is largely because the algorithm favors structures with pseudoknots since its primary function is to add pseudoknots to the pseudoknot-free seeds. Other algorithms ended up predicting the pseudoknot-free structures for some of the test sequences since they yield a lower free energy than the true structures that do contain pseudoknots. Heuristics such as genetic algorithms can often produce better results than dynamic programming approaches because with the existing energy models true structures are often suboptimal structures in term of free energies.

7. References

[1] Ren J., Rastegari B., Condon A., and Hoos H.H. (2005). HotKnots: Heuristic prediction of RNA secondary structures including pseudoknots. *RNA*, 11(10), 1494-504.

[2] Staples, David W., and Butcher, Samuel E. (2005). Pseudoknots: RNA Structures with Diverse Functions. *PLoS Biol*, 3(6), e213, 0956-0959.

[3] Department of Statistics, University of Oxford. RNA Pseudoknot Prediction. Retrieved December 20, 2007 from
http://www.stats.ox.ac.uk/research/genome/projects/rna/pseudoknots_in_rna_secondary_structure/pseudoknots_in_rna_secondary_structure2

[4] Rivas, E. and Eddy, S. (1999). A dynamic programming algorithm for RNA structure prediction including pseudoknots, *Journal of Molecular Biology*, 285(5), 2053-2068.

[5] Reeder, J. and Giegerich, R. (2004). Design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics. *BMC Bioinformatics.* 5:104.

[6] Nussinov, R. and Jacobson, A.B. (1980). Fast algorithm for predicting the secondary structure of single-stranded RNA, *Proceedings of the National Academy of Sciences of the USA*, 77, 6309-6313.

[7] Ruan, J., Stormo G. D., Zhang, W. (2004). An Iterated loop matching approach to the prediction of RNA secondary structures with pseudoknots. *Bioinformatics.* 20(1), 58-66.

[8] Durbin, R., Eddy, S. Krogh, A., and Mitchison, G. (1998). Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press. 269-270.

[9] Gultyaev, A.P., van Batenburg, F.H.D. and Pleij, C.W.A. (1995). The computer simulation of RNA folding pathways using a genetic algorithm. *J. Mol. Biol*., 250, 37-51.

[10] Shapiro, B.A., and Wu, J.C. (1997). Predicting RNA H-Type pseudoknots with massively parallel genetic algorithm, *CABIOS*, 13, 459-471.

[11] Yoon, B. and Vaidyanathan, P.P. (2004) HMM with auxiliary memory: a new tool for modeling RNA secondary structures. *IEEE*, 2(7-10), 1651-1655.

[12] Lee, D. and Han, K. (2002). Prediction of RNA Pseudoknots – Comparative Study of Genetic Algorithms, *Genome Informatics*, 13, 414-415.

[13] Lee, D. and Han, K. (2003). A Genetic Algorithm for Predicting RNA Pseudoknot Structures, *ICCS 2003*, LNCS 2659, 130-139.

[14] Brown, M., and Wilson, C. (1995). RNA Pseudoknot Modeling Using Intersection of Stochastic Context Free Grammars with Applications to Database Search. Retrieved December 20, 2007 from
http://psb.stanford.edu/psb-online/proceedings/psb96/brown.pdf

[15] Matsui, H., Sato, K., and Sakakibara, Y. (2005). Pair stochastic tree adjoining grammars for aligning and predicting pseudoknot RNA structures. *Bioinformatics*. 21(11), 2611- 2617.

[16] Lyngsø, R.B. and Pedersen, C.N.S. (2000). RNA Pseudoknot Prediction in Energy Based Models, *Journal of Computational Biology*, 7(3/4), 409-428.

[17] Gultyaev, A., van Batenburg, A., and Pleij, C. (1999) An Approximation of loop free energy values of RNA H-pseudoknots. *RNA*, 5. 609-617.

[18] Aalberts, D., and Hodas, N. (2005) Asymmetry in RNA pseudoknots: observation and theory. *Nucleic Acids Research*, 33(7), 2210-2214.

[19] van BatenBurd, F., Gultyaev, A., and Pleij, C. (1995) An APL-programmed Genetic Algorithm for the Prediction of RNA Secondary Structure. *J. Theor. Biol*., 174, 269-280.

[20] Zuker, M., Mathews, D.H., and Turner, D.H. (1999). Algorithms and thermodynamics for RNA secondary structure prediction: A practical guide. In *RNA biochemistry and biotechnology* (eds. J. Barciszewski and B. F. C. Clark), 11-43.

[21] Zuker, M., and Stiegler, P. (1981). Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, 9(1), 133-148.

[22] Mount, D. (2001). Bioinformatics: Sequence and Genome Analysis. New York: Cold Spring Harbor Laboratory Press. 205-228.

[23] Pseudoknot. http://en.wikipedia.org/wiki/Pseudoknot

[24] PseudoBase homepage – http://wwwbio.leidenuniv.nl/~Batenburg/PKB.html

[25] PseudoViewer homepage – http://wilab.inha.ac.kr/pseudoviewer/

[26] Turner Lab homepage – http://rna.chem.rochester.edu/

[27] pknotsRG. http://bibiserv.techfak.uni-bielefeld.de/pknotsrg/