CHAPTER 8

RNA SECONDARY STRUCTURE PREDICTION

Wing-Kin Sung
National University of Singapore
ksung@comp.nus.edu.sg

Understanding secondary structures of RNAs helps to determine their chemical and biological properties. Only a small number of RNA structures have been determined currently since such structure determination experiments are time-consuming and expensive. As a result, scientists start to rely on RNA secondary structure prediction. Unlike protein structure prediction, predicting RNA secondary structure already has some success. This chapter reviews a number of RNA secondary structure prediction methods.

ORGANIZATION.

Section 1. We begin by briefly introducing the relevance of RNA secondary structures for applications such as function classification, evolution study, and pseudogene detection. Then we present the different basic types of RNA secondary structures.

Section 2. Next we provide a brief description of how to obtain RNA secondary structure experimentally. We discuss physical methods, chemical methods, and mutational analysis.

Section 3. Two types of RNA secondary structure predictions are then described. The first type is based on multiple alignments of several RNA sequences. The second type is based on a single RNA sequence. We focus on the predicting secondary structure based on a single RNA sequence.

Section 4. We start with RNA structure prediction algorithms with the assumption that there is no pseudoknot. A summary of previous key results on this topic is given. This is then followed by a detailed presentation of the algorithm of Lyngso, Zuker, and Pedersen.[531]

Sections 5– 7. Then we proceed to some of the latest works on RNA structure prediction that allow for pseudoknots. In particular, we present the $O(n^4)$-time algoritm of Akutsu[15] for a restricted kind of pseudoknots and the $1/3$-approximation polynomial time algorithm of Ieong *et al.*[381] for general pseudoknots.

1. Introduction to RNA Secondary Structures

Due to the advance in sequencing technologies, many RNA sequences have been discovered. However, only a few of their structures have been deduced. The chemical and biological properties of many RNAs—like tRNAs—are determined primarily by their secondary structures. Therefore, determining the secondary structures of RNAs is becoming one of the most important topics in bioinformatics. We list a number of applications of RNA secondary structures below:

- Function classification. Many RNAs that do not have similar sequences do have similar functions.[437] An explanation is that they have similar secondary structure. For example, RNA viruses have a high mutation rate. Distant groups of RNA viruses show little or no detectable sequence homology. In contrast, their secondary structures are highly conserved. Hence, researchers classify RNA viruses based on their secondary structure instead of their sequences.
- Evolutionary studies. Ribosomal RNA is a very ancient molecule. It evolves slowly and exists in all living species. Therefore, it is used to determine the evolutionary spectrum of species.[891] One problem in the evolution study is to align the ribosomal RNA sequences from different species. Since the secondary structures of ribosomal RNAs are highly conserved, researchers use the structure as the basis to get a highly accurate alignment.
- Pseudogene detection. Given a DNA sequence that is highly homologous to some known tRNA gene, such a sequence may be a gene or a pseudogene. A way to detect if it is a pseudogene is by computing its secondary structure and checking if that looks similar to some tRNA secondary structure.[525]

Before studying the structures of RNAs, we need to understand the interactions between a pair of RNA nucleotides. RNA consists of a set of nucleotides that can be either adenine (A), cytosine (C), guanine (G), or uracil (U). Each of these nucleotides is known as the base and can be bonded with another one via hydrogen bonds. When this bonding happens, we say that the two bases form a base-pair. There are two types of base-pairs: canonical base-pair and wobble base-pair. The canonical base-pair are formed by a double hydrogen bond between A and U, or a triple hydrogen bond between G and C. The wobble base-pair is formed by a single hydrogen bond between G and U. Apart from these two types of base-pairs, other base pairs like U-C and G-A are also feasible, though they are relatively rare. To simplify the study, we assume only canonical and wobble base-pairs exist.

Unlike DNA, which is double stranded, RNA is single stranded. Due to the extra hydrogen bond in each RNA base, RNA bases in a RNA molecule hybridize with itself and form complex a 3D structure. Biologists describe RNA structures

in three levels: primary structure, secondary structure, and tertiary structure. The primary structure of an RNA is just its sequence of nucleotides. The secondary structure of an RNA specifies a list of canonical and wobble base-pairs that occur in the RNA structure. The tertiary structure is the actual 3D structure of the RNA.

Although the tertiary structure is more useful, such a tertiary structure is difficult to predict. Hence, many researchers try to get the secondary structure instead, as such a secondary structure can already explain most of the functionalities of the RNA. This chapter focuses on the secondary structure of RNAs. Consider a RNA polymer $s_1 s_2 \ldots s_n$ of length n. Generally, the secondary structure of the RNA can be considered as a set S of base pairs (s_i, s_j) where $1 \leq i < j \leq n$ that satisfies the following two criteria:

(1) Each base is paired at most once.
(2) Nested criteria: if $(s_i, s_j), (s_k, s_l) \in S$, we have $i < k < j \Longleftrightarrow i < l < j$.

Actually, a RNA secondary structure may contain base pairs that do not satisfy the two criteria above. However, such cases are rare. If criteria (1) is not satisfied, a base triple may happen. If criteria (2) is not satisfied, a pseudoknot becomes feasible. Figure 1 shows two examples of pseudoknots. Formally speaking, a pseudoknot is composed of two interleaving base pairs (s_i, s_j) and (s_k, s_l) such that $i < k < j < l$.

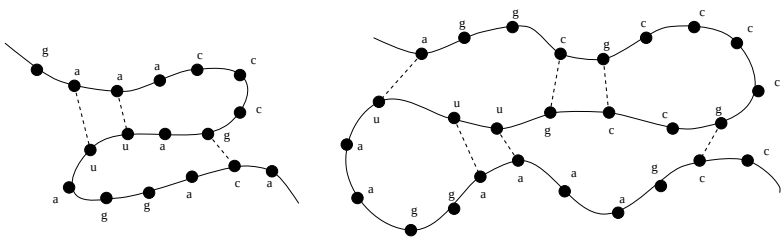

Fig. 1. Pseudoknots.

When no pseudoknot appears, the RNA structure can be described as a planar graph. See Figure 6 for an example. Then, the regions enclosed by the RNA backbone and the base pairs are defined as loops. Based on the positions of the base pairs, loops can be classified into the following five types:

- Hairpin loop—a hairpin loop is a loop that contains exactly one base-pair. This can happen when the RNA strand folds into itself with a base-pair holding them together as shown in Figure 2.

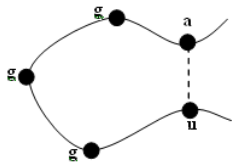

Fig. 2. Hairpin loop.

- Stacked pair—a stacked pair is a loop formed by two base-pairs that are adjacent to each other. In other words, if a base-pair is (i, j), the other base-pair that forms the stacked pair could be $(i+1, j-1)$. Figure 3 shows an example.

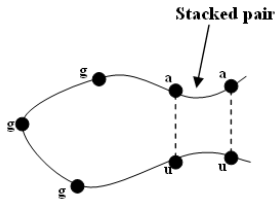

Fig. 3. Stacked pair.

- Internal loop—an internal loop consists of two base-pairs like the stacked pair. The difference between them is that internal loop consists of at least one unpaired base on each side of the loop between the two base-pairs. In short, the length of the two sides of the RNA between the two base-pairs must be greater than 1. This is shown in Figure 4.

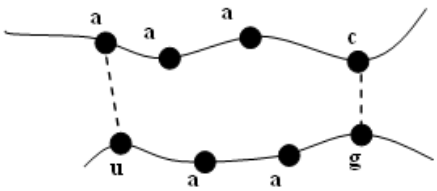

Fig. 4. Internal loop.

- Bulge—a bulge has two base-pairs like the internal loop. However, only one

side of the bulge has unpaired bases. The other side must have two base-pairs adjacent to each other as shown in Figure 5.

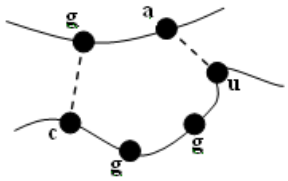

Fig. 5. Bulge.

- Multi-loop—any loop with 3 or more base-pairs is a multi-loop. Usually, one can view a multi-loop as a combination of multiple double-stranded regions of the RNA. Figure 6 shows examples of various loop types.

2. RNA Secondary Structure Determination Experiments

In the literature, there are several experimental methods for obtaining the secondary structure of an RNA, including physical methods, chemical/enzymatic methods, and mutational analysis.

- Physical methods—the basic idea behind physical methods is to infer the structure based on the distance measurements among atoms. Crystal X-ray diffraction is a physical method that gives the highest resolution. It reveals distance information based on X-ray diffraction. For example, the structure of tRNA is obtained using this approach.[431] However, the use of this method is limited since it is difficult to obtain crystals of RNA molecules that are suitable for X-ray diffraction. Another physical method is Nuclear Magnetic Resonance (NMR), which can provide detail local conformation based on the magnetic properties of hydrogen nuclei. Currently, NMR can only resolve structures of size no longer than 30–40 residues.
- Chemical/enzymatic methods—enzymatic and chemical probes[224] which modify RNA under some specific constraints can be used to analyse RNA structure. By comparing the properties of the RNA before and after applying the probes, we can obtain some RNA structure information. Note that the RNA structure information extracted are usually limited as some segments of an RNA polymer is inaccessible to the probes. Another issue is the experiment temperatures for various chemical or enzymatic digestions. A RNA

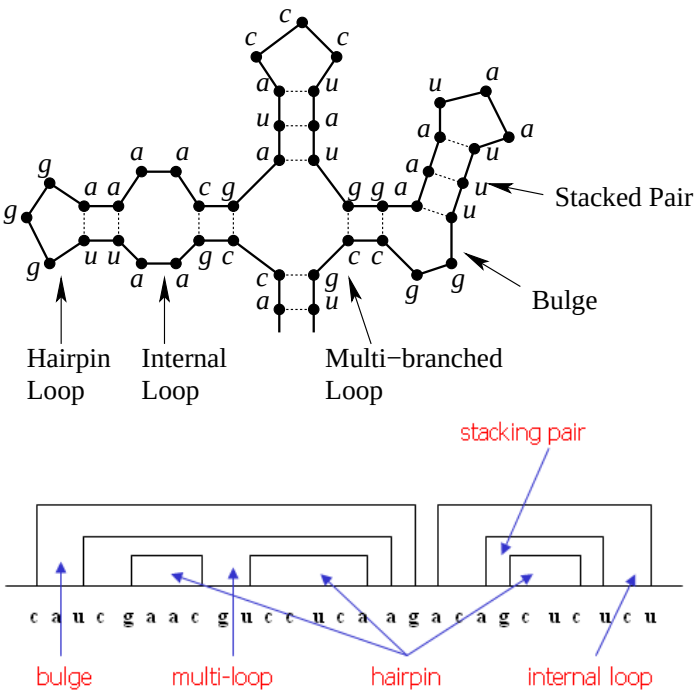

Fig. 6. Different loop types.

polymer may unfold due to high experiment temperature. Caution is required in interpreting such experimental results.

- Mutational analysis—this method makes specific mutation to the RNA sequence. Then, the binding ability between the mutated sequence and some proteins is tested.[8,17] If the binding ability of the mutated sequence is different from the original sequence, we claim that the mutated RNA sequence has structural changes. Such information helps us deduce the secondary structure.

3. RNA Structure Prediction Based on Sequence

Based on laboratory experiment, a denatured RNA renatures to the same structure spontaneously *in vitro*. Hence, it is in general believed that the structure of RNAs are determined by their sequences. This belief motivates us to predict the secondary structure of a given RNA based on its sequence only. There is a growing body of research in this area, which can be divided into two types:

(1) structure prediction based on multiple RNA sequences which are structurally similar; and
(2) structure prediction based on a single RNA sequence.

For the first type, the basic idea is to align the RNA sequences and predict the structure. Sankoff[743] considers the case of aligning two RNA sequences and inferring the structure. The time complexity of his algorithm is $O(n^6)$. (In Computer Science, the big-O notation is used to expressed the worst-case running time of a program. It tries to express the worst-case running time by ignoring the constant factor. For example, if your program runs in at most $10 \times n^2$ steps for an input of size n, then we say the time complexity of your program is $O(n^2)$. Throughout this chapter, we use this notation to express running time.) Corpet and Michot[174] present a method that incrementally adds new sequences to refine an alignment by taking into account base sequence and secondary structure. Eddy and Durbin[220] build a multiple alginment of the sequences and derive the common secondary structure. They propose covariance models that can successfully compute the consensus secondary structure of tRNA. Unfortunately, their method is suitable for short sequences only. The methods above have the common problem that they assume the secondary structure does not have any pseudoknot. Gary and Stormo[274] proposes to solve this problem using graph theoretical approach.

This chapter focuses on the second type. That is, structure prediction based on a single RNA sequence. Section 4 studies RNA structure prediction algorithms with the assumption that there is no pseudoknot. Then, some latest works on RNA structure prediction with pseudoknots are introduced in Sections 5 to 7.

4. Structure Prediction in the Absence of Pseudoknot

This section considers the problem of predicting RNA secondary structure with the assumption that there is no pseudoknot. The reason for ignoring pseudoknots is to reduce computational time complexity. Although ignoring psuedoknots reduces accuracy, such an approximation still looks reasonably good as psuedoknots do not appear so frequently.

Predicting RNA secondary structure is quite difficult. The most naive approach relies on an exhaustive search to find the lowest free-energy conformation. Such an approach fails because the number of conformations with the lowest free-energy is numerous. Identifying the correct conformation likes looking for a needle in the haystack.

Over the past 30 years, researchers try to find the correct RNA conformation based on simulating the thermal motion of the RNA—*e.g.*, CHARMM[108] and AMBER.[653] The simulation considers both the energy of the molecule and the

net force experienced by every pair of atoms. In principle, the correct RNA conformation can be computed in this way. However, such an approach fails because of the following two reasons:

(1) Since we still do not fully understand the chemical and physical properties of atoms, the energies and forces computed are merely approximated values. It is not clear whether the correct conformation can be predicted at such a level of approximation.
(2) The computation time of every simulation iteration takes seconds to minutes even for short RNA sequences. Unless CPU technology improves significantly, it is impossible to compute the structure within reasonable time.

In 1970s, scientists discover that the stability of RNA helices can be predicted using thermodynamic data obtained from melting studies. Those data implies that loops' energies are approximately independent. Tinoco *et al.* [829, 830] rely on this finding and propose the nearest neighbour model to approximate the free energy of any RNA structure. Their model makes the following assumptions:

(1) The energy of every loop—including hairpin, stacking pair, bulge, internal loop, and multi-loop–is independent of the other loops.
(2) The energy of a secondary structure is the sum of all its loops.

Based on this model, Nussinov and Jacobson [623] propose the first algorithm for computing the optimal RNA structure. Their idea is to maximize the number of stacking pairs. However, they do not consider the destabilising energy of various loops. Zuker and Stiegler[944] then give an algorithm that accounts for the various destabilising energies. Their algorithm takes $O(n^4)$ time, where n is the length of the RNA sequence. Lyngso, Zuker, and Pedersen [531] improves the time complexity to $O(n^3)$. Using the best known parameters proposed by Mathews *et al.*, [548] the predicted structure on average contains more than 70% of the base pairs of the true secondary structure. Apart from finding just the optimal RNA structure, Zuker [943] also prposes an approach that can compute all the suboptimal structures whose free energies are within some fixed range from the optimal.

In this section, we present the best known RNA secondary structure prediction algorithm—which is the one proposed by Lyngso, Zuker, and Pedersen. [531]

4.1. *Loop Energy*

RNA secondary structure is built upon 5 basic types of loops. Mathews, Sabina, Zuker, and Turner[548] have derived the 4 energy functions that govern the formation of these loops. These energy functions are:

- $eS(i,j)$—this function gives the free energy of a stacking pair consisting of base pairs (i,j) and $(i+1,j-1)$. Since no free base is included, this is the only loop which can stabilize the RNA secondary structure. Thus, its energy is negative.
- $eH(i,j)$—this function gives the free energy of the hairpin closed by the base pair (i,j). Biologically, the bigger the hairpin loop, the more unstable is the structure. Therefore, $eH(i,j)$ is more positive if $|j-i+1|$ is large.
- $eL(i,j,i',j')$—this function gives the free energy of an internal loop or a budge enclosed by base pairs (i,j) and (i',j'). Its free energy depends on the loop size $|i'-i+1|+|j'-j+1|$ and the asymmetry of the two sides of the loop. Normally, if the loop size is big and the two sides of the loop are asymmetric, the internal loop is more unstable and thus, $eL(i,j,i',j')$ is more positive.
- $eM(i,j,i_1,j_1,\ldots,i_k,j_k)$—this function gives the free energy of a multi-loop enclosed by base pair (i,j) and k base pairs $(i_1,j_1),\ldots,(i_k,j_k)$. The mutli-loop is getting more unstable when its loop size and the value k are big.

4.2. *First RNA Secondary Structure Prediction Algorithm*

Based on the nearest neighbor model, the energy of a secondary structure is the sum of all its loops. By energy minimization, the secondary structure of the RNA can then be predicted. To speedup the process, we take advantage of dynamic programming. The dynamic programming can be described using the following 4 recursive equations.

- $W(j)$—the energy of the optimal secondary structure for $S[1..j]$.
- $V(i,j)$—the energy of the optimal secondary structure for $S[i..j]$ given that (i,j) is a base pair.
- $VBI(i,j)$—the energy of the optimal secondary structure for $S[i..j]$ given that (i,j) closes a bulge or an internal loop.
- $VM(i,j)$—the energy of the optimal secondary structure for $S[i..j]$ given that (i,j) closes a multi-loop.

We describe the detail of the 4 recursive equations below.

4.2.1. $W(j)$

The recursive equations for $W(j)$ are given below. When $j=0$, $W(j)=0$ as the sequence is null. For $j>0$, we have two cases: either $S[j]$ is a free base; or there exists i such that $(S[i],S[j])$ forms a base pair. For the first case, $W(j)=$

$W(j-1)$. For the second case, $W(j) = \min_{1 \le i < j}\{V(i,j) + W(i-1)\}$. Thus, we have the following recursive equations.

$$W(j) = \begin{cases} 0 & \text{if } j = 0 \\ \min\{W(j-1), \min_{1 \le i < j}\{V(i,j) + W(i-1)\}\} & \text{if } j > 0 \end{cases}$$

4.2.2. $V(i,j)$

$V(i,j)$ is the free energy of the optimal secondary structure for $S[i..j]$ where (i,j) forms a base pair. When $i \ge j$, $S[i..j]$ is a null sequence and we cannot form any base pair. Thus, we set $V(i,j) = +\infty$. When $i < j$, The base pair (i,j) should belong to one of the four loop types: hairpin, stacked pair, internal loop, and multi-loop. Thus the free energy $V(i,j)$ should be the minimum of $eH(i,j)$, $eS(i,j)+V(i+1,j-1)$, $VBI(i,j)$, and $VM(i,j)$. Hence we have the following equations.

$$V(i,j) = \begin{cases} +\infty, & \text{if } i \ge j \\ \min \left\{ \begin{array}{ll} eH(i,j) & \text{Hairpin;} \\ eS(i,j) + V(i+1,j-1) & \text{Stacked pair;} \\ VBI(i,j) & \text{Internal loop;} \\ VM(i,j) & \text{Multi-loop.} \end{array} \right\}, & \text{if } i < j \end{cases}$$

4.2.3. $VBI(i,j)$

$VBI(i,j)$ is the free energy of the optimal secondary structure for $S[i..j]$ where the base pair (i,j) closes a bulge or an internal loop. The bulge or the internal loop is formed by (i,j) together with some other base pair (i',j') where $i < i' < j' < j$. The energy of this loop is $eL(i,j,i',j')$. The energy of the best secondary structure for $S[i..j]$ with (i,j) and (i',j') forms an internal loop is $eL(i,j,i',j') + V(i',j')$. By trying all possible (i',j') pairs, the optimal energy can be found as:

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

4.2.4. $VM(i,j)$

$VM(i,j)$ is the free energy of the optimal secondary structure for $S[i..j]$ where the base pair (i,j) closes a multi-loop. The multi-loop is formed by (i,j) together with k base pairs $(i_1,j_1),\ldots,(i_k,j_k)$ where $k > 1$ and $i < i_1 < j_1 < i_2 < j_2 < \ldots < i_k < j_k < j$—see Figure 7 for an example. Similar to the calculation of

$VBI(i,j)$, we get the following:

$$VM(i,j) = \min_{i<i_1<j_1<\ldots<i_k<j_k<j} \left\{ eM(i,j,i_1,j_1,\ldots,i_k,j_k) + \sum_{h=1}^{k} V(i_h,j_h) \right\}$$

4.2.5. *Time Analysis*

Based on the discussion above, computing the free energy of the optimal secondary structure for $S[1..n]$ is equivalent to finding $W(n)$. Such a computation requires us to fill in 4 dynamic programming tables for the 4 recursive equations $W(\cdot)$, $V(\cdot,\cdot)$, $VBI(\cdot,\cdot)$, and $VM(\cdot,\cdot)$. The optimal secondary structure can then be obtained by backtracking. We give below the time analysis for filling in the 4 tables.

- $W(i)$—it is an array with n entries. Each entry requires finding the minimum of n terms, $V(i,j) + W(i-1)$ for i varying from 1 to $j-1$. So, each entry needs $O(n)$ time. As a result, it costs $O(n^2)$ time in total.
- $V(i,j)$—it is an array with n^2 entries. Each entry requires finding the minimum of 4 terms, $eH(i,j)$, $eS(i,j)+V(i+1,j-1)$, $VBI(i,j)$, and $VM(i,j)$. Since each entry can be filled in $O(1)$ time, this matrix can be computed in $O(n^2)$ time.
- $VBI(i,j)$—it is an array with n^2 entries. Each entry requires finding the minimum of n^2 terms: $eL(i,j,i',j') + V(i',j')$ for $i < i' < j' < j$, where both i' and j' vary from 1 to n at most. So, each term needs $O(n^2)$ time. As a result, it costs $O(n^4)$ time in total.
- $VM(i,j)$—it is an array with n^2 entries. Each entry requires finding the minimum of exponential terms: $eM(i,j,i_1,j_1,\ldots,i_k,j_k) + \sum_{h=1}^{k} V(i',j')$ for $i < i_1 < j_1 < \ldots < i_k < j_k < j$. So in total, it costs exponential time.

In summary, the execution time of the algorithm is exponential. The major problem is on those computations pertaining to multi-loops and internal loops, which require time that is exponential and quartic in n respectively. For multi-loops, we assume that the energy of multi-loops can be approximated using an affine linear function, through which we can reduce the time cost of $VM(\cdot,\cdot)$ from exponential time to O(n^3) time. For internal loops, we reduce the overhead of $VBI(\cdot,\cdot)$ to O(n^3) time by using the approximation equation suggested by Ninio.[645] Therefore, we can reduce the overall complexity to O(n^3) time from the original exponential time. The two speed-up methods are discussed in detail in Subsections 4.3 and 4.4.

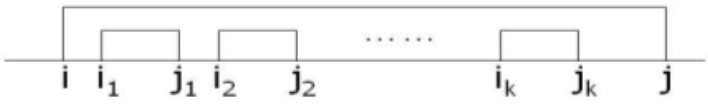

Fig. 7. Structure of a multi-loop.

4.3. *Speeding up Multi-Loops*

4.3.1. *Assumption on Free Energy of Multi-Loop*

To make the problem tractable, the following simplified assumption is made. Consider a multi-loop formed by base pairs (i, j), (i_1, j_1), ..., (i_k, j_k) as shown in Figure 7. The energy of the multi-loop can be decomposed into linear contributions from the number of unpaired bases in the loop, the number of base pairs in the loop, and a constant, that is

$$eM(i, j, i_1, j_1, \ldots, i_k, j_k) = a + b \times k + c \times \begin{pmatrix} (i_1 - i - 1) + \\ (j - j_k - 1) + \\ \sum_{h=1}^{k-1}(i_h + 1 - j_h - 1) \end{pmatrix}$$

where a, b, c are constants; k is the number of base pairs in the loop; and $((i_1 - i - 1) + (j - j_k - 1) + \sum_{h=1}^{k-1}(i_h + 1 - j_h - 1))$ is the number of unpaired bases in the loop.

4.3.2. *Modified Algorithm for Speeding Up Multi-Loop Computation*

Given the assumption above, the RNA structure prediction algorithm can be speeded up by introducing a new recursive equation $WM(i, j)$. $WM(i, j)$ equals the energy of the optimal secondary structure of $S[i..j]$ that constitutes the substructure of a multi-loop structure. Here, inside the multi-loop substructure, a free base is penalized with a score c while each base pair belonging to the multi-loop substructure is penalized with a score b. Thus, we have the following equation.

$$WM(i, j) = \min \begin{cases} WM(i, j-1) + c, & j \text{ is free base;} \\ WM(i+1, j) + c, & i \text{ is free base;} \\ V(i, j) + b, & (i, j) \text{ is pair;} \\ \min_{i<r\leq j} \left\{ \begin{matrix} WM(i, r-1) + \\ WM(r, j) \end{matrix} \right\}, & i \text{ and } j \text{ not free, and} \\ & (i, j) \text{ is not pair} \end{cases}$$

Given $WM(i, j)$, $VM(i, j)$ can be modified as

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

We can find an r between $i+1$ and $j-1$ that divides $S[i..j]$ into 2 parts. The sum of the two parts' energy should be minimal. Then the energy penalty a of the multi-loop is added to the sum to give $VM(i,j)$.

4.3.3. *Time Complexity*

After making these changes, we need to fill in 5 dynamic programming tables, *viz.* $W(i)$, $V(i,j)$, $VBI(i,j)$, $VM(i,j)$, and $WM(i,j)$.

The time complexity for filling tables $W(i)$, $V(i,j)$, and $VBI(i,j)$ are the same as the analysis in Section 4.2.5. They cost $O(n^2)$, $O(n^2)$, and $O(n^4)$ time respectively.

For the table $WM(i,j)$, it has n^2 entries. Each entry can be computed by finding the minimum of 4 terms: $WM(i,j-1)+c$, $WM(i+1,j)+c$, $V(i,j)+b$, and the minimum of $WM(i,k-1)+WM(k,j)$ for $i<k\leq j$. The first 3 terms can be found in $O(1)$ time while the last term takes $O(n)$ time. In total, filling in the table $WM(i,j)$ takes $O(n^3)$ time.

For the table $VM(i,j)$, it also has n^2 entries. But now each entry can be evaluated by finding the minimum of the n terms: $WM(i+1,k-1)+WM(k,j-1)+a$ for $i+1<k\leq j-1$. Thus, filling table $VM(i,j)$ also takes $O(n^3)$ time.

In conclusion, the modified algorithm runs in $O(n^4)$ time.

4.4. *Speeding Up Internal Loops*

4.4.1. *Assumption on Free Energy for Internal Loop*

Consider an internal loop or a bulge formed by two base pairs (i,j) and (i',j') with $i<i'<j'<j$. We assume its free energy $eL(i,j,i',j')$ can be computed as the sum:

$$eL(i,j,i',j') = size(n_1+n_2)+stacking(i,j)+ stacking(i',j')+asymmetry(n_1,n_2)$$

where

- $n_1=i'-i-1$ and $n_2=j-j'-1$ are the number of unpaired bases on both sides of the internal loop, respectively;
- $size(n_1+n_2)$ is an energy function depending on the loop size;
- $stacking(i,j)$ and $stacking(i',j')$ are the energy for the mismatched base pairs adjacent to the two base pairs (i,j) and (i',j'), respectively;
- $asymmetry(n_1,n_2)$ is energy penatly for the asymmetry of the two sides of the internal loop.

To simplify the computation, we further assume that when $n_1 > c$ and $n_2 > c$, it is the case that $asymmetry(n_1, n_2) = asymmetry(n_1-1, n_2-1)$. In practice, $asymmetry(n_1, n_2)$ is approximated using Ninio's equation,[645] *viz.*

$$asymmetry(n_1, n_2) = \min\{K,\ |n_1 - n_2| \times f(m)\}$$

where $m = \min\{n_1, n_2, c\}$, K and c are constants, and $f(m)$ is an arbitrary penalty function that depends on m. Note that $asymmetry(n_1, n_2)$ satisfies the above assumption and c is proposed to be 1 and 5 in two literatures.[645,664]

The above two assumptions imply the following lemma which is useful for devising an efficient algorithm for computing internal loop energy.

Lemma 1: *Consider $i < i' < j' < j$. Let $n_1 = i' - i - 1$, $n_2 = j - j' - 1$, and $l = n_1 + n_2$. For $n_1 > c$ and $n_2 > c$, we have*

$$eL(i, j, i', j') - eL(i+1, j-1, i', j') = size(l) - size(l-2) + stacking(i, j) - stacking(i+1, j-1)$$

Proof: This follows because $eL(i, j, i', j') - eL(i+1, j-1, i', j') = (size(l) + stacking(i, j) + stacking(i', j') + asymmetry(n_1, n_2)) - (size(l-2) + stacking(i+1, j-1) + stacking(i', j') + asymmetry(n_1 - 1, n_2 - 1))$. By the assumption that $asymmetry(n_1, n_2) = asymmetry(n_1 - 1, n_2 - 1)$, we have $eL(i, j, i', j') - eL(i+1, j-1, i', j') = size(l) - size(l-2) + stacking(i, j) - stacking(i+1, j-1)$ as desired. □

4.4.2. *Detailed Description*

Based on the assumptions, $VBI(i, j)$ for all $i < j$ can be found in $O(n^3)$ time as follows.

We define new recursive equations VBI' and VBI''. $VBI'(i, j, l)$ equals the minimal energy of an internal loop of size l closed by a pair (i, j). $VBI''(i, j, l)$ also equals the minimal energy of an internal loop of size l closed by a pair (i, j). Moreover, VBI'' requires the number of the bases between i and i' and the number of the bases between j and j', excluding i, i', j, and j', to be more than a constant c. Formally, VBI' and VBI'' are defined as follows.

$$VBI'(i, j, l) = \min_{\substack{i < i' < j' < j, \\ i' - i - 1 + j - j' - 1 = l}} \{eL(i, j, i', j') + V(i', j')\}$$

$$VBI''(i, j, l) = \min_{\substack{i < i' < j' < j, \\ i' - i - 1 + j - j' - 1 = l, \\ i' - i - 1, j - j' - 1 > c}} \{eL(i, j, i', j') + V(i', j')\}$$

Together with Lemma 1, we have

$$VBI''(i,j,l) - VBI''(i+1,j-1,l) = size(l) - size(l-2) + stacking(i,j) + stacking(i+1,j-1)$$

$$VBI'(i,j,l) = \min \left\{ \begin{array}{l} VBI''(i+1,j-1,l) + \\ \quad size(l) - size(j-2) + \\ \quad stacking(i,j) - stacking(i+1,j-1), \\ \min_{1 \le d \le c} \left\{ \begin{array}{c} V(i+d, j-l-d) + \\ eL(i,j,i+d,j-l+d-2) \end{array} \right\}, \\ \min_{1 \le d \le c} \left\{ \begin{array}{c} V(i+l+d, j-d) + \\ eL(i,j,i+l-d+2,j-d) \end{array} \right\} \end{array} \right\}$$

The last two entries of the above equation handle the cases where this minimum is obtained by an internal loop, in which d is less than a constant c, especially a bulge loop when c is equal to 1, that is at $j' = i+1$ or $j' = j-1$. By definition, we have $VBI(i,j) = \min_l \{VBI'(i,j,l)\}$.

4.4.3. *Time Analysis*

The dynamic programming tables for $VBI'(\cdot,\cdot,\cdot)$ and $VBI''(\cdot,\cdot,\cdot)$ have $O(n^3)$ entries. Each entry in $VBI'(\cdot,\cdot,\cdot)$ and $VBI''(\cdot,\cdot,\cdot)$ can be computed in $O(c)$ and $O(1)$ time respectively. Thus, both tables can be filled in using $O(c \times n^3)$ time. Given $VBI'(\cdot,\cdot,\cdot)$, the table $VBI(\cdot,\cdot)$ can be filled in using $O(n^2)$ time.

Together with filling the tables $W(\cdot)$, $V(\cdot,\cdot)$, $VM(\cdot,\cdot)$, $WM(\cdot,\cdot)$, the time required to predict secondary structure without pseudoknot is $O(n^3)$.

5. Structure Prediction in the Presence of Pseudoknots

Although pseudoknots are not frequent, they are very important in many RNA molecules.[183] For examples, pseudoknots form a core reaction center of many enzymatic RNAs, such as RNAseP RNA[522] and ribosomal RNA.[447] They also appear at the 5'-end of mRNAs, and act as a control of translation. Therefore, discovering pseudoknots in RNA molecules is very important.

Up to now, there is no good way to predict RNA secondary structure with pseudoknots. In fact, this problem is NP-hard.[15, 381, 530] Different approaches have been attempted to tackle this problem. Heuristic search procedures are adopted in most RNA folding methods that are capable of folding pseudoknots. Some examples include quasi-Monte Carlo searches by Abrahams *et al.*,[4] genetic algorithms by Gultyaev *et al.*,[307] Hopfield networks by Akiyama and Kanehisa,[14] and stochastic context-free grammar by Brown and Wilson.[110]

These approaches cannot guarantee that the best structure is found and are unable to say how far a given prediction is from the optimal. Other approaches are based on maximum weighted matching,[274,813]. They report some successes in predicting pseudoknots and base triples.

Based on dynamic programming, Rivas and Eddy,[715] Lyngso and Pedersen,[530] and Akutsu[15] propose three polynomial time algorithms that can find optimal secondary structure for certain kinds of pseudoknots. Their time complexities are $O(n^6)$, $O(n^5)$, and $O(n^4)$, respectively. On the other hand, Ieong *et al.*[381] propose two polynomial time approximation algorithms that can handle a wider range of pseudoknots. One algorithm handle bi-secondary structures—*i.e.*, secondary structures that can be embedded as a planar graph—while the other algorithm can handle general secondary structure. The worst-case approximation ratios are $1/2$ and $1/3$, respectively.

To illustrate the current solutions for predicting RNA secondary structure with pseudoknots, the next two sections present Akutsu's $O(n^4)$-time algorithm and Ieong *et al.*'s $1/3$-approximation polynomial time algorithm.

6. Akutsu's Algorithm

6.1. *Definition of Simple Pseudoknot*

This section gives the definition of a simple pseudoknot.[15] Consider a substring $S[i_0..k_0]$ of a RNA sequence S where i_0 and k_0 are arbitrarily chosen positions. A set of base pairs M_{i_0,k_0} is a simple pseudoknot if there exist j_0, j'_0 such that

(1) each endpoint i appears in M_{i_0,k_0} once;
(2) each base pair (i,j) in M_{i_0,k_0} satisfies either $i_0 \leq i < j'_0 < j \leq j_0$ or $j'_0 \leq i < j_0 < j \leq k_0$; and
(3) if pairs (i,j) and (i',j') in M_{i_0,k_0} satisfy $i < i' < j'_0$ or $j'_0 \leq i < i'$, then $j > j'$.

The first two parts of the definition divides the sequence $S[i_0..k_0]$ into three segments: $S[i_0..j_0]$, $S[j_0..j'_0]$, and $S[j'_0..k_0]$. For each base pair in M_{i_0,k_0}, one of its end must be in $S[j_0..j'_0]$ while the other end is either in $S[i_0..j_0]$ or $S[j'_0..k_0]$. The third part of the definition confines the base pairs so that they cannot intersect each other. Part I of Figure 8 is an example of a simple pseudoknot. Parts II, III, and IV of Figure 8 are some examples that are not simple pseudoknot.

With this definition of simple pseudoknots, a RNA secondary structure with simple pseudoknots is defined as below. A set of base pairs M is called a RNA secondary structure with simple pseudoknots if $M = M' \cup M_1 \cup \ldots \cup M_t$ for some non-negative integer t such that

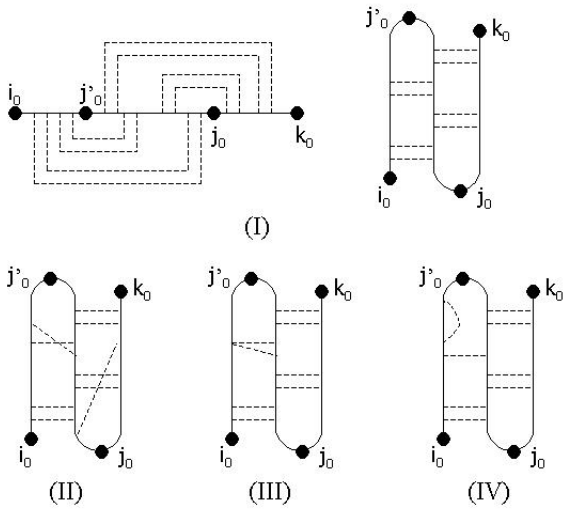

Fig. 8. An illustration of simple pseudoknots.

(1) For $h = 1, 2, \ldots, t$, M_h is a simple pseudoknot for $S[i_h..k_h]$ where $1 \leq i_1 < k_1 < i_2 < k_2 < \ldots < i_t < k_t \leq n$.
(2) M' is a secondary structure without pseudoknot for string S' where S' is obtained by removing segments $S[i_h..k_h]$ for all $h = 1, 2, \ldots, t$.

6.2. *RNA Secondary Structure Prediction with Simple Pseudoknots*

This section presents an algorithm which solves the following problem.

> **Input**: A RNA sequence $S[1..n]$
> **Output**: A RNA secondary structure with simple pseudoknots that maximizes the score.
> **Score**: In this section for simplicity, the score function used is different from that of the RNA secondary structure prediction without pseudoknot. The score function here is the number of the base pairs in $S[1..n]$. In short, we maximize the number of base pairs. Note that the score function can be generalized to some simple energy function.

A dynamic programming algorithm is designed to solve the problem above. Let $V(i, j)$ be the optimal score of an RNA secondary structure with simple pseu-

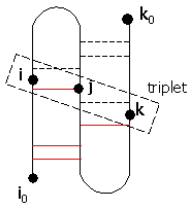

Fig. 9. (i, j, k) is a triplet in the simple pseudoknot. Note that all the base pairs in solid lines are below the triplet.

doknots for the sequence $S[i..j]$. Let $V_{pseudo}(i, j)$ be the optimal score for $S[i..j]$ with the assumption that $S[i..j]$ forms a simple pseudoknot.

For $V(i, j)$, the secondary structure for $S[i..j]$ can be either (1) a simple pseudoknot, (2) (i, j) forms a base pair, or (3) $S[i..j]$ can be decomposed into two compounds. Therefore, we get the following recursive equation.

$$V(i,j) = \max \left\{ \begin{array}{l} V_{pseudo}(i,j), \\ V(i+1,j-1) + \delta(S[i], S[j]), \\ \max_{i<k\leq j}\{V(i,k-1) + V(k,j)\} \end{array} \right\}$$

where $V(i, i) = 0$ for all i. Also, $\delta(S[i], S[j]) = 1$ if $\{S[i], S[j]\} = \{a, u\}$ or $\{c, g\}$; otherwise, $\delta(S[i], S[j]) = -\infty$.

For $V_{pseudo}(i_0, k_0)$, its value can also be computed using a dynamic programming algorithm. To explain the algorithm, we first give some notations. Recall that, in a simple pseudoknot $S[i_0..k_0]$, the sequence is partitioned into three segments $S[i_0..j_0']$, $S[j_0'..j_0]$, and $S[j_0..k_0]$ for some unknown positions j_0 and j_0'. We denote the three segments as left, middle, and right segments, respectively. See Part I of Figure 8 for an example. For a triplet (i, j, k) where $i_0 \leq i \leq j_0'$, $j_0' < j \leq j_0$, and $j_0 < k \leq k_0$, we say a base pair (x, y) is below the triplet (i, j, k) if either $x \leq i$ and $y \geq j$, or $x \geq j$ and $y \leq k$. Figure 9 is an example illustrating this concept of "below". All the base pairs in red color are "below" the triplet (i, j, k).

For a triplet (i, j, k), $S[i]$, $S[j]$, and $S[k]$ should satisfy one of the following relations: (1) (i, j) is a base pair, (2) (j, k) is a base pair, or (3) both (i, j) and (j, k) are not base pair. Below, we define three variables, based on the above three relationships, which are useful for computing $V_{pseudo}(i_0, k_0)$.

- $V_L(i, j, k)$ is the maximum number of base pairs below the triplet (i, j, k) in a pseudoknot for $S[i_0..k_0]$ given that (i, j) is a base pair.
- $V_R(i, j, k)$ is the maximum number of base pairs below the triplet (i, j, k) in

a pseudoknot for $S[i_0..k_0]$ given that (j,k) is a base pair.

- $V_M(i,j,k)$ is the maximum number of base pairs below the triplet (i,j,k) in a pseudoknot for $S[i_0..k_0]$ given that both (i,j) and (j,k) are not a base pair.

Note that $\max\{V_L(i,j,k), V_R(i,j,k), V_M(i,j,k)\}$ is the maximum number of base pairs below the triplet (i,j,k) in a pseudoknot for $S[i_0..k_0]$. Then $V_{pseudo}(i_0, j_0)$ can be calculated as:

$$V_{pseudo}(i_0,k_0) = \max_{i_0 \le i<j<k \le k_0} \{V_L(i,j,k), V_M(i,j,k), V_R(i,j,k)\}$$

6.2.1. $V_L(i,j,k)$, $V_R(i,j,k)$, $V_M(i,j,k)$

We define below the recursive formulae for the 3 variables $V_L(i,j,k)$, $V_R(i,j,k)$, and $V_M(i,j,k)$.

$$V_L(i,j,k) = \delta(S[i],S[j]) + \max \left\{ \begin{array}{l} V_L(i-1,j+1,k), \\ V_M(i-1,j+1,k), \\ V_R(i-1,j+1,k) \end{array} \right\}$$

$$V_R(i,j,k) = \delta(S[i],S[j]) + \max \left\{ \begin{array}{l} V_L(i,j+1,k-1), \\ V_M(i,j+1,k-1), \\ V_R(i,j+1,k-1) \end{array} \right\}$$

$$V_M(i,j,k) = \max \left\{ \begin{array}{l} V_L(i-1,j,k), V_M(i-1,j,k), \\ V_L(i,j+1,k), V_M(i,j+1,k), V_R(i,j+1,k), \\ V_M(i,j,k-1), V_R(i,j,k-1) \end{array} \right\}$$

Here we provide an intuitive explanation for the formulae above. For both $V_L(i,j,k)$ and $V_R(i,j,k)$, the first term represents the number of base pairs on the triplet (i,j,k) while the second term represents the number of base pairs below (i,j,k). For $V_M(i,j,k)$, since there is no base pair on the triplet (i,j,k), the formula only consists of the number of base pairs below the triplet. Note that the two variables, $V_R(i-1,j,k)$ and $V_L(i,j,k-1)$, do not appear in the formula for $V_M(i,j,k)$. This is because $V_M(i,j,k)$ indicates that both (i,j) and (j,k) are not base pair.

6.2.2. *To Compute Basis*

To compute $V_L(i,j,k)$, $V_R(i,j,k)$, $V_M(i,j,k)$, some base values are required.

$$V_R(i_0-1,j,j+1) = \delta(S[j],S[j+1]), \text{ for all } j \quad (1)$$

$$V_R(i_0-1,j,j) = 0, \quad \text{for all } j \quad (2)$$

$$V_L(i,j,j) = \delta(S[i],S[j]), \quad \text{for all } i_0 \le i < j \quad (3)$$

$$V_L(i_0-1,j,k) = 0, \quad \text{for all } k = j+1 \text{ or } k = j \quad (4)$$

$$V_M(i_0-1,j,k) = 0, \quad \text{for all } k = j+1 \text{ or } k = j \quad (5)$$

The base case (3) can be explained by Part I of Figure 10. The base cases (1) and (2) can be explained by Parts II and III of Figure 10 respectively. The base cases (4) and (5) are trivial since they are out of range.

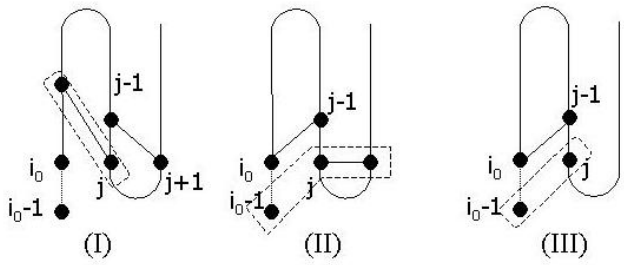

Fig. 10. Basis for the recursive equations V_L, V_M, and V_R.

6.2.3. *Time Analysis*

This section gives the time analysis. First, we analyse the time required for computing V_{pseudo}.

Lemma 2: *$V_{pseudo}(i_0,k_0)$ for all $1 \le i_0 < k_0 \le n$ can be computed in $O(n^4)$ time.*

Proof: Observe that the base cases for $V_L(\cdot,\cdot,\cdot)$, $V_M(\cdot,\cdot,\cdot)$, $V_R(\cdot,\cdot,\cdot)$ only depend on i_0. Thus for a fixed i_0, the values for the base cases of $V_L(\cdot,\cdot,\cdot)$, $V_M(\cdot,\cdot,\cdot)$, and $V_R(\cdot,\cdot,\cdot)$ can be computed in $O(n^2)$ time. Then the values of tables $V_L(\cdot,\cdot,\cdot)$, $V_R(\cdot,\cdot,\cdot)$, and $V_M(\cdot,\cdot,\cdot)$ are independent of k_0, and can be computed in $O(n^3)$ time since each table has n^3 entries and each entry can be computed in $O(1)$ time.

Based on the definition of V_{pseudo}, we have the following recursive equation:

$$V_{pseudo}(i_0, k+1) = \max \left\{ \begin{array}{l} V_{pseudo}(i_0, k), \\ \max_{i_0<i<j<k+1} \left\{ \begin{array}{l} V_L(i, j, k+1), \\ V_R(i, j, k+1), \\ V_M(i, j, k+1) \end{array} \right\} \end{array} \right\}$$

Thus for a fixed i_0, $V_{pseudo}(i_0, k_0)$ for all k_0 can be computed in $O(n^3)$ time. Since there are n choices for i_0, $V_{pseudo}(i_0, k_0)$ for all $1 \leq i_0 < k_0 \leq n$ can be computed in $O(n^4)$ time. □

By the lemma above, the RNA secondary structure with simple pseudoknots for a sequence S can be predicted in $O(n^4)$ time.

Proposition 3: *Consider a sequence $S[1..n]$. We can predict the RNA secondary structure of S with simple pseudoknots in $O(n^4)$ time.*

Proof: Based on the lemma above, $V_{pseudo}(i, j)$ for all i, j can be computed in $O(n^4)$ time. Then, we need to fill in n^2 entries for the table $V(\cdot, \cdot)$ where each entry can be computed in $O(n)$ time. Hence, the table $V(\cdot, \cdot)$ can be filled in using $O(n^3)$ time. In total, the problem can be solved in $O(n^4)$ time. □

7. Approximation Algorithm for Predicting Secondary Structure with General Pseudoknots

The previous algorithm can only handle some special types of pseduoknots. This section addresses general pseudoknots. As the problem of predicting secondary structure with general pseudoknots is NP-hard, we approach the problem by giving an approximation algorithm.[381]

Given a RNA sequence $S = s_1 s_2 \ldots s_n$, this section constructs a secondary structure of S that approximates the maximum number of stacking pairs with a ratio of $1/3$. First, we need some definition. We denote a stacking pair $\{(s_i, s_j), (s_{i+1}, S_{j-1})\}$ as $(s_i, s_{i+1}; s_{j-1}, s_j)$. For a consecutive of q ($q \geq 1$) stacking pairs $(s_i, s_{i+1}; s_{j-1}, s_j), (s_{i+1}, s_{i+2}; s_{j-2}, s_{j-1}), \ldots, (s_{i+q-1}, s_{i+q}; s_{j-q}, s_{j-q+1})$, it is denoted as $(s_i, s_{i+1}, \ldots, s_{i+q}; s_{j-q}, \ldots, s_{j-1}, s_j)$. The approximation algorithm uses a greedy approach. Figure 11 shows the algorithm $GreedySP(\cdot, \cdot)$.

In the following, we analyze the approximation ratio of the algorithm. The algorithm $GreedySP(S, i)$ will generate a sequence of consecutive stacking pairs SP's. Let $SP_1, SP_2, \ldots, SP_h$ be the generated sequence. We have the following fact.

// Let $S = s_1 s_2 \ldots s_n$ be the input RNA sequence.
// Initially, all s_j are unmarked.
// Let E be the set of base pairs output by the algorithm.
// Initially, $E = \emptyset$.

$GreedySP(S, i)$ // $i \geq 3$

(1) Repeatedly find the leftmost i consecutive stacking pairs SP—*i.e.*, find $(s_p, \ldots, s_{p+i}; s_{q-i}, \ldots, s_q)$ such that p is as small as possible—formed by unmarked bases. Add SP to E and mark all these bases.
(2) For $k = i - 1$ downto 2,
Repeatedly find any k consecutive stacking pairs SP formed by unmarked bases. Add SP to E and mark all these bases.
(3) Repeatedly find the leftmost stacking pair SP formed by unmarked bases. Add SP to E and mark all these bases.

Fig. 11. The 1/3-approximation algorithm.

Fact 4: For any SP_j and SP_k, $j \neq k$, the corresponding stacking pairs in SP_j and SP_k do not overlap.

For each $SP_j = (s_p, \ldots, s_{p+t}; s_{q-t}, \ldots, s_q)$, we define two intervals of indices, $\mathcal{I}_j$ and $\mathcal{J}_j$, as $[p..p+t]$ and $[q-t..q]$ respectively. We want to compare the number of stacking pairs formed with that in the optimal case, so we have the following definition.

Definition 5: Let $\mathcal{P}$ be an optimal secondary structure of S with a maximum number of stacking pairs. Let $\mathcal{F}$ be the set of all stacking pairs of $\mathcal{P}$. For each SP_j computed by $GreedySP(S, i)$, we define the set $\mathcal{X}_\beta$, where $\beta = \mathcal{I}_j$ or $\mathcal{J}_j$, as follows.

$$\mathcal{X}_\beta = \left\{ \begin{pmatrix} s_k, s_{k+1}; \\ s_{w-1}, s_w \end{pmatrix} \in \mathcal{F} \;\middle|\; \text{at least one of indices } k, k+1, w-1, w \in \beta \right\}$$

Next, we observe that

Lemma 6: *Let $SP_1, SP_2, \ldots, SP_h$ be the sequence of SP's computed by $GreedySP(S, i)$. Then $\bigcup_{1 \leq j \leq h} \{\mathcal{X}_{\mathcal{I}_j} \cup \mathcal{X}_{\mathcal{J}_j}\} = \mathcal{F}$.*

Proof: We prove it by contradiction. Suppose that there exists a stacking pair $(s_k, s_{k+1}; s_{w-1}, s_w) \in \mathcal{F}$ but not in any of $\mathcal{X}_{\mathcal{I}_j}$ and $\mathcal{X}_{\mathcal{J}_j}$. By Definition 5, none of the indices $k, k+1, w-1, w$, is in any of $\mathcal{I}_j$ and $\mathcal{J}_j$. This contradicts with Step 3 of the algorithm $GreedySP(S, i)$. □

Note that $\mathcal{X}_\beta$'s may not be disjoint.

Definition 7: For each $\mathcal{X}_{\mathcal{I}_j}$, we define $\mathcal{X}'_{\mathcal{I}_j}$ to be $\mathcal{X}_{\mathcal{I}_j} - \bigcup_{k<j}\{\mathcal{X}_{\mathcal{I}_k} \cup \mathcal{X}_{\mathcal{J}_k}\}$; and define $\mathcal{X}'_{\mathcal{J}_j}$ as $\mathcal{X}_{\mathcal{J}_j} - \bigcup_{k<j}\{\mathcal{X}_{\mathcal{I}_k} \cup \mathcal{X}_{\mathcal{J}_k}\} - \mathcal{X}_{\mathcal{I}_j}$.

Let $|SP_j|$ be the number of stacking pairs represented by SP_j. Let $|\mathcal{I}_j|$ and $|\mathcal{J}_j|$ be the number of indices in the intervals $\mathcal{I}_j$ and $\mathcal{J}_j$ respectively.

Lemma 8: *Let N be the number of stacking pairs computed by the algorithm $GreedySP(S, i)$ and N^* be the maximum number of stacking pairs that can be formed by S. If for all j, we have $|SP_j| \geq \frac{1}{r} \times |(\mathcal{X}'_{\mathcal{I}_j} \cup \mathcal{X}'_{\mathcal{J}_j})|$, then $N \geq \frac{1}{r} \times N^*$.*

Proof: By Definition 7, $\bigcup_k\{\mathcal{X}_{\mathcal{I}_k} \cup \mathcal{X}_{\mathcal{J}_k}\} = \bigcup_k\{\mathcal{X}'_{\mathcal{I}_k} \cup \mathcal{X}'_{\mathcal{J}_k}\}$. By Fact 4, $N = \sum_j |SP_j|$, so $N \geq \frac{1}{r} \times |\bigcup_k\{\mathcal{X}_{\mathcal{I}_k} \cup \mathcal{X}_{\mathcal{J}_k}\}|$. Now by Lemma 6, we conclude $N \geq \frac{1}{r} \times N^*$ as desired. □

This brings us to the main approximation result:

Proposition 9: *For each SP_j computed by $GreedySP(S, i)$, we have*

$$|SP_j| \geq \frac{1}{3} \times |\mathcal{X}'_{\mathcal{I}_j} \cup \mathcal{X}'_{\mathcal{J}_j}|$$

Proof: There are 3 steps of the $GreedySP(S, i)$ algorithm to be considered. For each SP_j computed by $GreedySP(S, i)$ in Step 1, we know that $SP_j = (s_p, \ldots, s_{p+i}; s_{q-i}, \ldots, s_q)$ is the leftmost i consecutive stacking pairs—*i.e.*, p is as small as possible. By definition, $|\mathcal{X}'_{\mathcal{I}_j}|$, $|\mathcal{X}'_{\mathcal{J}_j}| \leq i+2$. We further claim that $|\mathcal{X}'_{\mathcal{I}_j}| \leq i+1$. Then, as $i \geq 3$, $|SP_j|/|\mathcal{X}'_{\mathcal{I}_j} \cup \mathcal{X}'_{\mathcal{J}_j}| \geq i/((i+1)+(i+2)) \geq 1/3$.

We prove the claim by contradiction. Assume that $|\mathcal{X}'_{\mathcal{I}_j}| = i+2$. That is, for some integer t, $\mathcal{F}$ has $i+2$ consecutive stacking pairs $(s_{p-1}, \ldots, s_{p+i+1}; s_{t-i-1}, \ldots, s_{t+1})$. Furthermore, none of the bases $s_{p-1}, \ldots, s_{p+i+1}, s_{t-i-1}, \ldots, s_{t+1}$ are marked before SP_j is being chosen. Otherwise, suppose one of such bases, says s_a, is marked when the algorithm chooses SP_l for $l < j$, then the stacking pairs adjacent to s_a do not belong to $\mathcal{X}'_{\mathcal{I}_j}$ and they belong to $\mathcal{X}'_{\mathcal{I}_l}$ or $\mathcal{X}'_{\mathcal{J}_l}$ instead. Therefore, $(s_{p-1}, \ldots, s_{p+i-1}; s_{t-i+1}, \ldots, s_{t+1})$ is the leftmost i consecutive stacking pairs formed by unmarked bases before SP_j is chosen. As SP_j is not the leftmost i consecutive stacking pairs, this contradicts with the algorithm. The claim is thus proved.

For each SP_j computed by $GreedySP(S, i)$ in Step 2, let $|SP_j| = k \geq 2$ and let $SP_j = (s_p, \ldots, s_{p+k}; s_{q-k}, \ldots, s_q)$. By definition, $|\mathcal{X}'_{\mathcal{I}_j}|$, $|\mathcal{X}'_{\mathcal{J}_j}| \leq k+2$. We claim that $|\mathcal{X}'_{\mathcal{I}_j}|$, $|\mathcal{X}'_{\mathcal{J}_j}| \leq k+1$. Then $|SP_j|/|\mathcal{X}'_{\mathcal{I}_j} \cup \mathcal{X}'_{\mathcal{J}_j}| \geq k/((k+1)+(k+1))$, which is at least $1/3$ as $k \geq 2$.

We can prove the claim $|\mathcal{X}'_{\mathcal{I}_j}| \leq k+1$ by contradiction. Assume $|\mathcal{X}'_{\mathcal{I}_j}| = k+2$. Thus, for some integer t, there exist $k+2$ consecutive stacking pairs $(s_{p-1}, \ldots, s_{p+k+1}; s_{t-k-1}, \ldots, s_{t+1})$. Similar to Step 1, we can show that none of the bases $s_{p-1}, \ldots, s_{p+k+1}, s_{t-k-1}, \ldots, s_{t+1}$ are marked before SP_j is chosen. Thus, $GreedySP(S, i)$ should select some $k+1$ or $k+2$ consecutive stacking pairs instead of the chosen k consecutive stacking pairs. Thus, we arrive at a contradiction. We can prove the claim $|\mathcal{X}'_{\mathcal{J}_j}| \leq k+1$ in a similar way.

For each SP_j computed by $GreedySP(S, i)$ in Step 3, SP_j is a leftmost stacking pair—that is, $SP_j = (s_p, s_{p+1}; s_{q-1}, s_q)$. Using the same approach as in Step 2, we can show that $|\mathcal{X}'_{\mathcal{I}_j}|$, $|\mathcal{X}'_{\mathcal{J}_j}| \leq 2$. We further claim that $|\mathcal{X}'_{\mathcal{I}_j}| \leq 1$. Then $|SP_j|/|\mathcal{X}'_{\mathcal{I}_j} \cup \mathcal{X}'_{\mathcal{J}_j}| \geq 1/(1+2) = 1/3$.

To verify the claim $|\mathcal{X}'_{\mathcal{I}_j}| \leq 1$, we consider all possible cases with $|\mathcal{X}'_{\mathcal{I}_j}| = 2$ while there are no 2 consecutive stacking pairs. The only possible case is that for some integers r, t, both $(s_{p-1}, s_p; s_{r-1}, s_r)$ and $(s_p, s_{p+1}; s_{t-1}, s_t)$ belong to $\mathcal{X}'_{\mathcal{I}_j}$. However, this means that SP_j is not the leftmost stacking pair formed by unmarked bases. This contradicts the algorithm and completes the proof. □

By Lemma 8 and Proposition 9, we derive the following corollary.

Corollary 10: *Given a RNA sequence S. Let N^* be the maximum number of stacking pairs that can be formed by any secondary structure of S. Let N be the number of stacking pairs output by $GreedySP(S, i)$. Then $N \geq \frac{1}{3} \times N^*$.*

We remark that by setting $i = 3$ in $GreedySP(S, i)$, we can already achieve the approximation ratio of 1/3. The following lemma gives the time and space complexity of the algorithm.

Lemma 11: *Given a RNA sequence S of length n. The algorithm $GreedySP(S, i)$, where i is a constant, can be implemented in $O(n)$ time and $O(n)$ space.*

Proof: Recall that the bases of a RNA sequence are chosen from the alphabet $\{a, u, g, c\}$. If i is a constant, there are only constant number of different patterns of consecutive stacking pairs that we have to consider. For any $1 \leq k \leq i$, there are only 4^k different strings that can be formed by the four characters $\{a, u, g, c\}$. So for all possible values of k, the locations of the occurrences of these possible

strings in the RNA sequence can be recorded in an array of linked lists indexed by the pattern of the string using $O(n)$ time preprocessing. There are at most 4^k linked lists and there are only n entries in all linked lists.

Now, fix a constant k. In order to locate all k consecutive stacking pairs, we scan the RNA sequence from left to right. For each substring of k consecutive characters, we look up the array to see if we can form k consecutive stacking pairs. By a simple bookkeeping procedure, we can keep track which bases have been used already. Each entry in the linked lists is thus scanned at most once. So the whole procedure takes only $O(n)$ time. Since i is a constant, we can repeat the whole procedure for i different values of k and the total time complexity is still $O(n)$ time. □