Algorithms for predicting the secondary structure of pairs and combinatorial sets of nucleic acid strands

by

Mirela Ştefania Andronescu

M.Sc., Academy of Economic Studies, Bucharest, Romania, 2000

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

We accept this thesis as conforming
to the required standard

The University of British Columbia

August 2003

Abstract

Secondary structure prediction of nucleic acid molecules is a very important problem in computational molecular biology. In this thesis we introduce two new algorithms for: (1) secondary structure prediction of pairs of nucleic acid molecules (*PairFold*), and (2) finding which sequences, formed from a combinatorial set of nucleic acid strands, have the most stable secondary structures (*CombFold*). Our algorithms run in polynomial time in the sequences lengths and are extensions of the free energy minimization algorithm [72] for secondary structure prediction without pseudoknots, using the nearest neighbour thermodynamic model. Predicting hybridization of pairs of molecules is motivated by important applications such as ribozyme - mRNA target duplexes, primer binding prediction and DNA code design. Finding the most stable concatenations in combinatorial sets of strands is useful for SELEX experiments and for testing whether sets in DNA computing or tag libraries concatenate without secondary structure.

Our results for *PairFold* predictions show over 80% accuracy for sequences of up to 100 nucleotides. The performance goes down as the sequences increase in length and as the number of non-canonical base pairs, pseudoknots and tertiary interactions, none of these considered here, increases. The accuracy of *CombFold* is similar to that of the free energy minimization algorithm for single strands, being just a polynomial method for structure prediction of a combinatorial set of strands. We show that although complex, *CombFold* can quickly predict large concatenations of sets drawn from the literature. In the future, these two algorithms can be combined to predict the most stable duplexes formed by two combinatorial sets.

Contents

List of Tables

List of Figures

Acknowledgements

First of all, I would like to thank my supervisor Anne Condon, who has guided me through research in bioinformatics since I did not even know what it meant. I am deeply grateful to her encouragements and her enthusiasm in dealing with challenging problems, her support, great research ideas she shared with me, and her responsiveness and open-mindedness to my opinions.

I would like to thank Raymond Ng for being my second reader and for useful comments. Thanks to Holger Hoos for wonderful discussions on various bioinformatics topics. Many thanks go to Dan Tulpan, for helping me when I was stuck, his friendship and being the third (informal) reader of my thesis. I want to thank the people in Beta Lab and other collaborators who helped me throughout: Sanja Rogic, Viann Chan, Rosalia Aguirre-Hernandez, Jérémy Barbay, Greg Lakatos.

I am grateful to JyBy for his friendship, for giving me joy and happiness. Last but not the least, I want to thank my parents, my aunt Adi and my brother Călin for their support and trust.

Mirela Ştefania Andronescu

The University of British Columbia
August 2003

To my grandpa.

Chapter 1

Introduction

One of the most important problems in computational molecular biology is structure prediction of nucleic acids. RNA molecules play a crucial role in cellular processes and their function is directly related to their folding into complex tertiary structures [48]. DNA and RNA strands are also used in biomolecular computations and in self-assembly of nanostructures [9, 10]. Researchers have studied the problem of nucleic acids structure prediction for more than two decades. In this thesis, we start from the state-of-the-art algorithms for nucleic acid secondary structure prediction and extend them to the problems of (1) predicting secondary structure of a pair of nucleic acids and (2) finding, out of a combinatorial set of RNA or DNA short strands, which combination has the most stable secondary structure.

In this chapter, we first give background on RNA, DNA and secondary structures. Then, in Section 2, we define the problems that we deal with in this thesis and we give motivations to support why this work is important. A summary of contributions is given in Section 3, and some notations and conventions that we will use throughout this thesis are enumerated in Section 4. The last section gives a brief outline of this thesis.

1.1 Background

The central dogma in molecular biology (Figure 1.1) is that, in an organism, the genes, which constitute the genetic code made of DNA (*deoxyribonucleic acid*), are first replicated, and then transcribed into RNA (*ribonucleic acid*). This is *premature messenger RNA* (pre-mRNA), which in higher organisms (i.e. *eukaryotes*) are first processed to eliminate some non-coding sequences, called *introns*, and they become *mature messenger RNA* (mRNA). In simpler organisms (i.e. *prokaryotes*), there are no introns, and the genetic code is directly transcribed into mature mRNA. This is translated into proteins, which have well defined functions [19].

DNA and RNA molecules (also known as nucleic acids) are composed of sequences of four types of *nucleotides* or *bases*: Adenine (A), Cytosine (C), Guanine (G) and Thymine (T) for DNA or Uracil (U) for RNA. In organisms, the genetic material is usually double-stranded DNA and the RNA is single-stranded. For this reason, RNA is more flexible

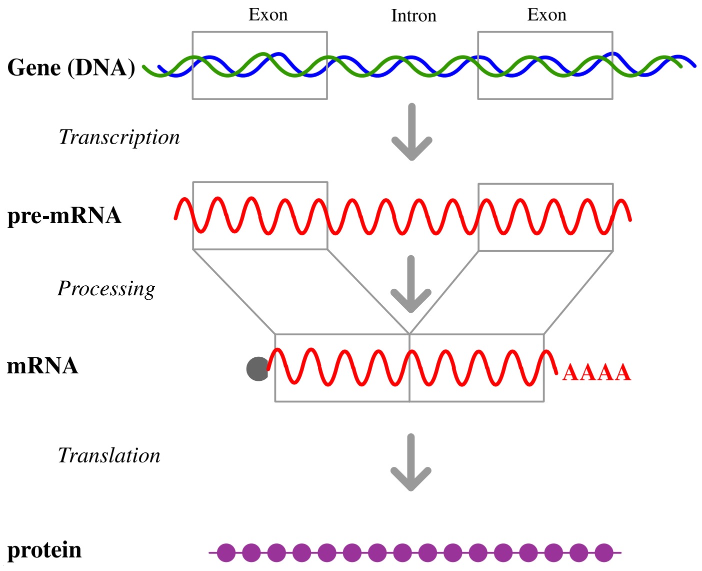


Figure 1.1: Central dogma in molecular biology for eukaryotes.

and can form a much greater variety of complex three-dimensional structures than double-stranded DNA (dsDNA). However, single-stranded DNA (ssDNA), used in *in vitro* experiments or in DNA computing, can also form complex structures.

The linear sequence of an RNA or DNA strand constitutes the *primary structure* or *sequence*. The set of base pairs that form when a nucleic acid sequence folds is called *secondary structure*. These pairings arise from the Hydrogen-bonding forces between pairs of bases. Thus, in an RNA or ssDNA secondary structure, each base can be either free (not bonded with any other base) or Hydrogen-bonded to another base. The *tertiary structure* is the three-dimensional geometry of the arrangement of bases in space. Much research has been done on understanding secondary structures, while the information we currently have about tertiary structures is relatively sparse. It is believed that once secondary structures are known, they can provide useful information about tertiary structures as well.

Throughout this thesis, we focus on predicting secondary structures from primary structures of RNA or ssDNA molecules. Although there are other factors that influence secondary structure formation, it is believed that the sequence has the greatest contribution. The most common Hydrogen-bonds which will lead to secondary structure formation are between C and G, and between A and T (or U for RNA). They are called Watson-Crick (W-C) bonds. $(C.G)$ pairs are more stable, because they involve three Hydrogen connections, as opposed to $(A.T)$ or $(A.U)$ pairs, which involve two Hydrogen connections.

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

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

Each base in the second strand is the W-C complement of the corresponding base in the first strand. They can bind and form a double stranded DNA because all the corresponding bases are complementary and because the strands are in opposite orientation. The same rule is available for RNA or ssDNA. Consider the following simple RNA sequence:

```
5'-CCCCCCCCCCAAAAAGGGGGGGGGG-3'
```

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

The first step in understanding RNA or ssDNA secondary structures is to identify the substructures of which they are composed. We call *elementary structure* any substructure which cannot be decomposed in any other substructure. Figure 1.2 shows an example of a complex secondary structure, which contains all elementary structures that we consider in this work. The bullets represent bases, the thin gray line indicates the backbone that holds all the nucleotides together in the molecule, and the thick black lines indicate paired bases. The 5' and 3' ends are indicated, and a numbering for the base positions is provided, with the first base considered to be at the 5' end. Each base can only participate in at most one base pair. The elementary structures are marked by rectangles and their names are added aside. The elementary structures we consider here follow:

- A *hairpin loop* or *hairpin* contains one closing base pair and all the bases between the paired bases are unpaired. The hairpin marked in the figure contains 5 free bases. Formally, the tuple (i, j) defines a hairpin loop in a given secondary structure if i and j are paired, and k is a free base, $\forall k, i < k < j$;

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

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

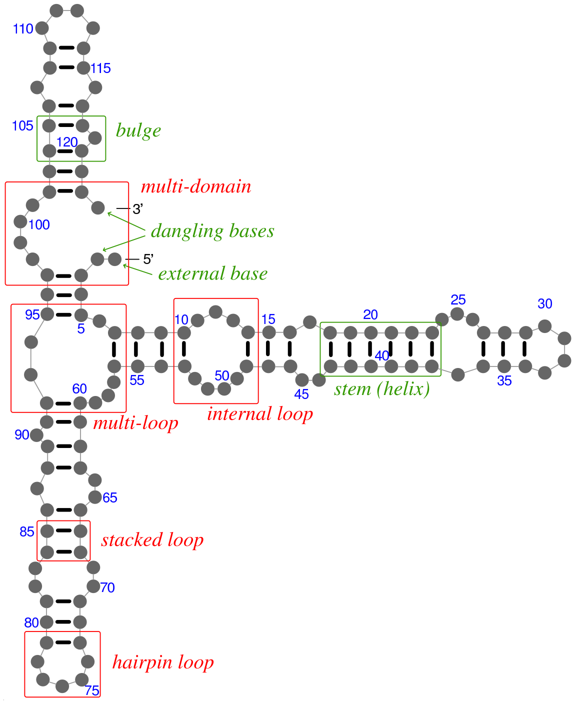


Figure 1.2: Example of a pseudoknot-free secondary structure containing all elementary structures.

A *bulge loop*, or simply *bulge*, is a special case of internal loop, which has no free base on one side, and at least one free base on the other side.

Note that, in fact, a stacked loop is also a special case of internal loop, with no free bases on both sides. In this work, we will consider stacked loops and internal loops to be distinct elementary structures, but we include bulges in the internal loop case, unless otherwise specified;

- A *multi-branched loop* or *multi-loop* is a loop which has at least three closing base pairs. The tuple $(i, j, i_1, j_1, \dots i_m, j_m)$, with $m \geq 2, i < i_1 < j_1 < \dots < i_m < j_m < j$ defines a multi-loop with $m + 1$ branches if i pairs with j, i_1 pairs with $j_1, \dots, i_m$ pairs with j_m and k is a free base, $\forall k, i < k < i_1$, $j_1 < k < i_2$, …, $j_m < k < j$. The multi-loop marked in the figure has 3 closing base pairs and 6 free bases;

- A *multi-domain loop*, or simply *multi-domain*, is a loop with at least one closing pair. The tuple $(i, j, i_1, j_2, \dots, i_d, j_d)$, with $d \geq 0, i < j < i_1 < j_2 < \dots < i_d < j_d$ defines a multi-domain if i pairs with j, i_1 pairs with j_1, …, i_d pairs with j_d and k is a free base $\forall k, 1 \leq k < i$, $j < k < i_1$, …, $j_d < k \leq n$, where n is the length of the sequence. We call *domains* the whole secondary structures which are closed by the closing pairs of a multi-domain. In other words, the domain closed by (i, j) is the set of all base pairs whose indices are in the interval $[i, j]$. Note that if we virtually make the sequence

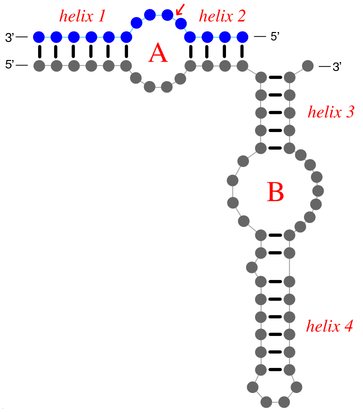


Figure 1.3: Structure of a hairpin ribozyme bound to an mRNA target.

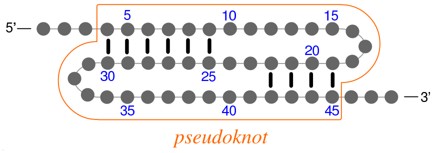


Figure 1.4: A simple pseudoknotted structure.

> circular by merging the 5' and 3' ends together, we obtain a hairpin, internal loop or multi-loop. The multi-domain marked in the figure contains two branches and 7 free bases. These free bases are called *external bases*, because they are not inside any domain, but they are between domains or between a domain and a molecule end.
>
> Note that, if the whole structure contains only one domain and no external bases, then there is one multi-domain, and the pair *(1.n)* constitutes its only closing pair.
>
> The bases which are neighbours of a closing pair of a multi-domain or multi-loop are called *dangling bases*. The dangling bases neighbouring to the multi-domain marked in the figure have positions 2, 98, 101 and 123. The ones neighbouring to the multi-loop have positions 6, 57, 59, 93 and 94.
>
> When we refer to secondary structures, we assume that at least one base pair exists. In some situations, it is possible that no pairing between any two bases exist. In this case we say that the molecule is *structure free*.

All these elementary structures have been used before with these names [24, 33, 71], except the *multi-domain* structure, whose exact name we introduce here for convenience later.

Another way to understand a secondary structure is to think of it as a set of helices, connected to each other by internal loops, bulges, multi-loops or multi-domains, and some of them ended in hairpins. Note that the secondary structure represented in Figure 1.2 is just a graphical, convenient way to visualize the set of base pairs of the folded molecule. In other words, the geometrical positions of the base pairs and free bases do not have any meaning and do not matter other than for visualization purposes. In our representation of secondary structures, if the backbone location is straightforward, we omit it. The structure used in Figure 1.2 will be used again in Chapter 3, together with its RNA sequence (Figure 3.2).

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

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

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

An example of a long, complicated structure determined by comparative sequences analysis [22] in Gutell Lab [21] is given in Figure 1.5. It contains all elementary structures, three pseudoknots and two tertiary interactions.

In the analysis of the algorithms we propose, we will refer to different types of RNAs. Thus, more background describing the most important RNA classes is necessary. Depending on its type, RNA molecules can have different functions. RNAs can be grouped in two general classes [19]: (1) The *informational RNA* class is formed of messenger RNA (mRNA and pre-mRNA), the intermediary between the genes and the proteins, which are illustrated in Figure 1.1. (2) *Functional RNAs*, which are not translated into proteins, but are active as RNA. Although the genes that encode functional RNAs are relatively few, the active RNAs count for a large percentage of the RNA in the cell. They are more stable than informational RNAs, and they need abundance to carry out their function. Three very important classes of functional RNAs that we will refer to later in the thesis are: (1) *transfer RNAs* (*tRNAs*) are short RNA molecules (less than 100 bases long), which have an important role in mRNA translation into proteins; (2) *ribosomal RNAs* (*rRNAs*), some of them exceeding 4000 bases in length, which also have a central role in the translation machinery. They are part of ribosomes, which are large macromolecular assemblies composed of several types of rRNAs and about 100 different proteins. (3) *ribozymes* are RNA molecules which act as enzymes and catalyze a specific biological reaction. Recent studies of last ten years reveal that

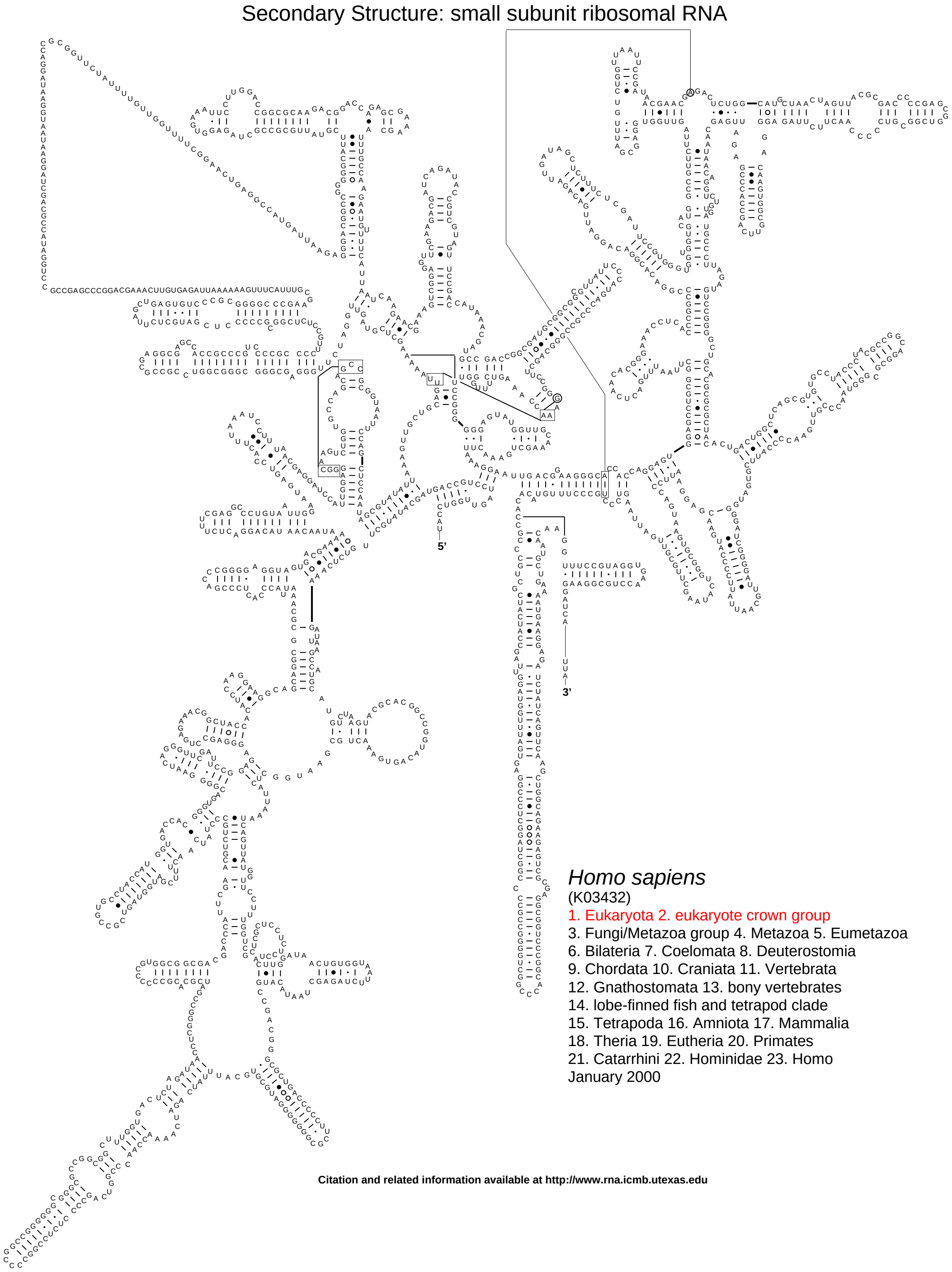

Secondary Structure: small subunit ribosomal RNA
5'
3'
Homo sapiens
(K03432)
1. Eukaryota 2. eukaryote crown group
3. Fungi/Metazoa group 4. Metazoa 5. Eumetazoa
6. Bilateria 7. Coelomata 8. Deuterostomia
9. Chordata 10. Craniata 11. Vertebrata
12. Gnathostomata 13. bony vertebrates
14. lobe-finned fish and tetrapod clade
15. Tetrapoda 16. Amniota 17. Mammalia
18. Theria 19. Eutheria 20. Primates
21. Catarrhini 22. Hominidae 23. Homo
January 2000
Citation and related information available at http://www.rna.icmb.utexas.edu

Figure 1.5: Secondary structure of a 16S rRNA from Gutell Database [21]. The structure has been determined by comparative sequence analysis.

functional RNAs have a much more important role than originally thought, when proteins were considered to do nearly all functional tasks in the cell.

1.2 Problems and motivations

In this thesis, we try to solve two main problems: (1) secondary structure prediction of a pair of nucleic acid molecules and (2) secondary structure prediction of combinations formed from a combinatorial set of nucleic acid molecules. In this section, we briefly explain the problem statements and motivate their study.

Secondary structure prediction of a pair of molecules

We define the *basic problem of secondary structure prediction of a pair of nucleic acid molecules* as follows: given two RNA sequences S_1 and S_2 and a thermodynamic model M, find the pseudoknot-free secondary structure R with the smallest free energy change under the model M, in which S_1 and S_2 can fold. Note that the solution R may include base pairing between the two molecules, as well as base pairing within each molecule.

The algorithm we propose for solving this problem is called *PairFold* and is presented in detail in Chapter 5. It is motivated by predicting interactions between any two DNA or RNA molecules. Examples include: (1) a ribozyme and an RNA target [25, 27, 43, 47, 53, 60, 68]; (2) a probe or primer and a target RNA molecule [68]; (3) pairs of DNA strands in DNA code design [55]; (4) pairs of strands in biomolecular nanostructures [39, 69]; (5) molecular tags in a polymer library [10]. Concrete examples where *PairFold* is useful will be given in Section 5.5.

Secondary structure prediction of a combinatorial set of molecules

Consider the following sets of short RNA strands (also called *words*):

	S_1	S_2	S_3	S_4	S_5
1	UAGCGA	CAGCGUAAUAU	AUGCG	AUAGCGGUA	AUCG
2	AUAGAU	AGAUGCGCGGU		GAGCGCAAG	CUGC
3		UAGGCUAGCGU			GCGA

If we take one word w_{ij} from line j of each set S_i and concatenate them together, we obtain what we call *combinations*. For example, the sequence UAGCGA CAGCGUAAUAU AUGCG AUAGCGGUA AUCG is the combination $w_{11}w_{21}w_{31}w_{41}w_{51}$. Note that the number of possible combinations is exponential in the number of sets which contain at least two words.

The input set of sets will be called *Input-Set*, and the set of all possible combinations will be called *Combinatorial-Set*. We define the *basic problem of secondary structure prediction of a combinatorial set of nucleic acid molecules* as follows: given an *Input-Set*

IS and a thermodynamic model M, predict which *combination*, out of all elements of the *Combinatorial-Set* CS formed from IS, folds to a pseudoknot-free secondary structure with the lowest minimum free energy.

The algorithm we propose is called *CombFold*, and is described in detail in Chapter 6. Applications where *CombFold* is useful include: (1) biochemical experiments altering a consensus nucleic acid sequence, such as directed mutagenesis or SELEX experiments [38]; (2) information storage in DNA computations, where all combinations of a combinatorial set are used. It is important that none of the strands fold on themselves in the temperature range at which they are used [9, 10, 14]; (3) combinatorial sets are also used as molecular bar codes in applications such as massive parallel signature sequencing [10].

1.3 Contributions

This section describes the main contributions of this thesis in the order they are discussed:

1. The Nearest Neighbour Thermodynamic Model (*NNTM*) for RNA secondary structure prediction has been described before by Zuker et al. [71]. However, their report misses some cases when compared to their implementation of *mfold* and the similar program from the Vienna RNA package [24, 61]. Thus, a new implementation of the RNA secondary structure prediction algorithm by Zuker and Stiegler [72] would be difficult without consulting previous implementations. The description of NNTM that we give in Chapter 3 is meant to be a complete description of this model. A thorough understanding of it is necessary for new implementations or extensions of the secondary structure prediction algorithm;

2. A thorough analysis of the accuracy of our program *SimFold* and of two popular secondary structure prediction programs (*mfold* [35, 70] and *RNAfold* [24, 61]) is the first one showing more characteristics about the data sets and also pointing out that the performance on some tRNA sequences, in terms of percentage of base pairs found, appears to be poorer than reported earlier by Mathews et al. [33] (Chapter 4);

3. An algorithm similar to our method underlying *PairFold*, to predict secondary structures of pairs of nucleic acid molecules, has been briefly described by Mathews et al. [32], but no analysis on real biological data has been performed. We give a thorough analysis of *PairFold* on sets of ribozymes found in the literature and on primer binding prediction. We also discuss several other useful applications of *PairFold*, such as prediction of DNA or RNA duplex formation and DNA code design. Moreover, we created the RNAsoft web site [41], which offers online access to PairFold. Information about this web site have been published in a special issue on Web based software of Nucleic Acids Research journal [5];

4. Our algorithm underlying *CombFold* has been developed in parallel by Cohen and Skiena [12], but for a different problem. We offer an analysis of data sets from the

DNA computing literature, and give other useful applications, such as for SELEX experiments. Part of this work has been presented at the DNA 8 Conference (Eight International Meeting on DNA Based Computers, Hokkaido, Japan, 2002) and appeared in the conference proceedings [6]. A full version of this paper has been accepted to appear in the journal Natural Computing [6]. Online access to *CombFold* is also available on the RNAsoft web site [41], and information about this software have been published in the Nucleic Acids Research Journal [5];

5. Finally, our extension to *CombFold* for calculating suboptimal combinations is the first one that we are aware of. There are at least two reasons for which this is useful information: (1) for both DNA computation and SELEX experiments, where knowing the next most stable combinations is important; and (2) returning the k most stable combinations can cover some of the impreciseness of the energy model we are using.

1.4 Notations and conventions

Throughout this thesis, the following notations and conventions will be used:

- Given a sequence S, its default orientation is from the $5'$ end (left) to the $3'$ end (right), unless otherwise stated;

- To represent secondary structures, sometimes we use the *dot-parenthesis* (or *dot-brackets*) format. This contains three characters: "(", ")" and ".". A left bracket corresponds to a base which is paired to a base upstream, a right bracket denotes a base paired to a base downstream, and a dot denotes a free base. For example, the secondary structure of the simple RNA sequence given in Section 1.1 can be depicted as follows:

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

  Note that this representation is only valid for pseudoknot-free structures.

- Two nucleotides between round brackets and separated by a point (e.g. *(A.U)*) signify a base pair.

1.5 Thesis outline

The most important work related to the research reported in this thesis is first presented in Chapter 2. The basis of the secondary structure prediction calculations considered in this work resides in the standard Nearest Neighbour Thermodynamic Model, which will be described in great detail in Chapter 3. Our two algorithms: *PairFold* and *CombFold*, are extensions of the free energy minimization algorithm [72]. Full understanding of this method is necessary for understanding our proposed solutions. We call this basic algorithm *SimFold*, and we explain its underlying equations in Chapter 4. A discussion on complexity and a thorough performance evaluation are given. Chapter 5 explains our algorithm for structure prediction of pairs of molecules, *PairFold*, and gives several applications and performance evaluations on biological data. Chapter 6 proposes a polynomial time algorithm for prediction of the most stable combination of a combinatorial set, *CombFold*. An extension, to predict the k most stable combinations, is included. Two heuristic approaches are discussed, along with practical applications. Finally, we conclude the thesis in Chapter 7 and we present ideas of how this work can be continued in several ways. Some details additional to data analysis discussed in Chapters 4 and 5 are given in Appendices A and B.

Because the two algorithms we propose are based on the Nearest Neighbour Thermodynamic Model, which is complex and not very well explained in the literature, we dedicate the whole Chapter 3 to detailed explanations of this model, although this is not our work (other than collecting the proper information). Similarly, since our two solutions are in fact extensions of the Zuker and Stiegler's [72] free energy minimization algorithm for secondary structure prediction, we explain it in detail in Section 4.1. Our work starts with showing that our implementation of Zuker and Stiegler's algorithm is reliable (Sections 4.2-4.4) and continues with our proposed solutions, in Chapters 5 and 6.

Chapter 2

Related Work

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

The quantity of information about RNA structures is rapidly increasing every year. However, depending on the length of the given RNA sequence, the number of possible secondary structures can be very high. For instance, for a 16S rRNA of 1500 nucleotides, there are approximately 15,000 possible helices (less than 100 will be in the final structure). The maximum number of combinatorial arrangements of all possible helices, which will eventually lead to different structures, is about 4.3×10^{393} [22]. *In vitro* experiments for some specific RNAs such as hairpin ribozymes show that they behave very similarly to the experiments *in vivo* [48]. However, there are studies which show that *in vitro* RNA folding experiments cannot reliably simulate the complex intracellular environments existing in the cell [48]. Thus, there is still a lot of unknown information about what happens in the cell. The computational methods that currently exist try to capture as much information as possible from the existing knowledge. Still, this knowledge is rudimentary with respect to some rules and contributions of other factors that participate to the folding of RNA. Moreover, in order to ensure that the computational methods are efficient (and thus practical), thermodynamic methods are often highly simplified.

Section 1 of this chapter describes previous work related to the secondary structure prediction for single RNA or DNA molecules. This work is related to all three algorithms described in this thesis: *SimFold*, *PairFold* and *CombFold*. Section 2 presents work for predicting secondary structures for pairs of molecules, and this is relevant to our *PairFold* algorithm. Section 3 gives information about algorithms for predicting secondary structures of combinatorial sets, which is directly related to *CombFold*.

2.1 Secondary structure prediction for single molecules

In this section, several different methods are explained for predicting secondary structures without pseudoknots, and two algorithms for predicting pseudoknotted structures are summarized.

Free energy minimization algorithm

A very popular algorithm for finding the minimum free energy (MFE) secondary structure without pseudoknots of an RNA molecule is Zuker and Stiegler algorithm [72]. This method is the basis of the algorithms proposed in this thesis, and will be described in great detail in Chapter 4. Its input is the primary RNA sequence, and it uses a dynamic programming algorithm to find the secondary structure with the minimum free energy. The basis of this method is the Nearest Neighbour Thermodynamic Model (NNTM), which will be described in Chapter 3. NNTM contains rules about base pairing formation and tabulated free energy and enthalpy parameters.

Briefly, the main idea of the free energy minimization algorithm is that the bases of the RNA molecule are numbered from 1 to n, starting from the $5'$ end and finishing at the $3'$ end. Then, for each i and j with $1 \leq i < j \leq n$, the problem is to determine which of the four elementary structures (hairpin loop, stacked loop, internal loop or multi-branched loop), with the exterior pair $(i.j)$, has the lowest free energy. Recurrence relations are applied and a two dimensional matrix with all minimum free energies for each i and j is filled. As for the standard dynamic programming algorithm, backtracking is necessary to build the path (i.e. the set of base pairs) that gives the MFE secondary structure.

The complexity of Zuker and Stiegler's algorithm is $O(n^4)$ for time and $O(n^2)$ for space. It has been reduced to $O(n^3)$ for time by Lyngsø et al. [31], but the space required increased to $O(n^3)$. Other approaches [24] assume that the free bases on both sides of internal loops are bounded by a constant c (e.g. $c = 30$). This reduces the time complexity of Zuker and Stiegler's algorithm to $O(n^3)$ with no penalty on the space.

Wuchty et al. [66] extended the MFE secondary structure prediction algorithm to generate all suboptimal secondary structures between the MFE and an upper limit. Generating suboptimal structures is important for at least two reasons [66]: (1) The energy model on which the minimization algorithm relies is imprecise. Also, there are unknown biological constraints, which are not taken into consideration by the energy model. Thus, the true MFE structure might be one of the suboptimal structures with respect to the parameters used. (2) Under physiological conditions, RNA molecules might fold to alternative structures, whose energy difference is small. Also, it is speculated that specific folding pathways capture molecules in local minima [20]. Mathews et al. [33] show that, on average, the accuracy of the prediction algorithm increases by more than 20% when 750 suboptimal structures are generated, as opposed to generating the MFE structure only.

Different implementations of the free energy minimization algorithm exist. The program *mfold* [70, 71] was the first one to implement Zuker and Stiegler's algorithm and is available online [35]. It also incorporates Wuchty et al.'s [66] extension for generating suboptimal structures. The Vienna RNA Package [24] implements Zuker and Stiegler's algorithm, together with the partition function calculation, which will be discussed later in this section. It is available online and is free open source software [61].

Partition function algorithm

McCaskill [34] proposed another dynamic programming algorithm for pseudoknot-free folding of an RNA molecule, which permits calculation of probabilities of various structures. This involves calculation of the partition function:

$$Q = \sum_S e^{-\Delta G(S)/RT}$$

from statistical mechanics, where the sum goes over all possible structures in which the RNA molecule can fold. Although this sum has a number of terms that is exponential in the molecule length n, the partition function calculation can be done in time $O(n^3)$. Once the partition function Q is calculated, we can calculate the probability of a given structure S: $P(S) = \frac{1}{Q}e^{-\Delta G(S)/RT}$. However, what is more relevant for biological function of RNA structures is an *ensemble* of related structures (also called kinetically clustered objects) interchanging more or less rapidly between each other [34]. The focus is on the equilibrium probabilities of substructures common to an ensemble or class of related structures. These substructure classes are very important because they allow the display of the most significant features of the ensemble. Finally, the equilibrium probability of occurrence for each possible base pair can be calculated, and a mirror image including the base probabilities (one triangle) and the optimal structure (another triangle) can be drawn for good visualization. McCaskill [34] evaluated his method on four biological RNA sequences with known structures. He showed that the real base pairs have been predicted with high probability, although not always the highest probability. The partition function algorithm has been incorporated in the Vienna RNA Package [24, 61].

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

Comparative analysis methods

Comparative analysis methods predict secondary structures and early stages of tertiary structures of evolutionary related RNA molecules. They overcome the two major drawbacks of the aforementioned methods: pseudoknotted structures and imprecise thermodynamic model. Gutell Laboratory [11, 21] has started determination of the 16S and 23S rRNA secondary structures since early 1980s, when only two molecules of each class were available.

The comparative analysis method is based on two simple and profound principles [22]: (1) “different RNA sequences can fold into the same secondary and tertiary structures”; (2) “the unique structure and function of an RNA molecule is maintained through the evolutionary process of mutation and selection”. In 1999, 7000 homologous 16S and 1050 23S aligned rRNA sequences where used in covariation-based structure models [22] and the result was compared to the experimentally determined high-resolution crystal structures of the 30S and 50S ribosomal units (which include 16S and 23S rRNAs, respectively). Several methods constitute the class of comparative analysis: (1) covariation analysis predicted 97-98% of the base pairs which are present in the 16S and 23S rRNA crystal structures; (2) tentative covariation-based method predicted about 45% of the base pairs; and (3) motif-based method predicted 70% of the base pairs. In conclusion, these methods predicted nearly all of the standard secondary structure base pairings and helices in the 16S and 23S crystal structures, and they have also identified tertiary base-base interactions. The major drawback of this method is that a large number of evolutionary related sequences is necessary for good accuracy.

Combinations of using the thermodynamic model as well as comparative analysis have been tried. Hofacker et al. [23] presented a method for computing the secondary structure of a set of aligned RNA sequences, using both thermodynamic stability and sequence covariation. They show that only 5 rRNA related sequences and an automatically generated alignment were necessary to correctly predict over 80% of the base pairs. Their program is implemented under the name of *RNAalifold*, which is available online from the Vienna RNA Package web site [61].

Pseudoknotted secondary structure prediction

Predicting RNA secondary structures *including pseudoknots* from the primary sequence of a molecule and using a thermodynamic model is a great challenge. Firstly, the pseudoknots can be very complex; secondly, the forces that drive the formation of pseudoknots are not well understood, thus the model being a rough approximation of what is believed to happen. Lastly, and most importantly, it has been proved that finding a minimum energy structure among all possible pseudoknots is NP-hard, although the energy function used in the proof is highly idealized [30].

Rivas and Eddy [40] proposed a free energy minimization dynamic programming algorithm which, apart from the elementary structures considered by Zuker and Stiegler [72], also includes pseudoknots. This was considered to be the first algorithm to optimally determine a large class of pseudoknots. The algorithm is complex and its worst case complexity is $O(n^6)$ for time and $O(n^4)$ for space. They show that their algorithm does not predict spurious pseudoknots for a set of tRNA. They also predict 54 out of 63 SELEX-selected HIV-1-RT-ligant simple pseudoknots, and most pseudoknots in short viral RNAs.

The recent work of Dirks and Pierce [13] introduces a partition function algorithm for nucleic acids secondary structures which contain the most physically relevant pseudoknots. The algorithm has a complexity of $O(n^5)$ for time and $O(n^4)$ for space. Although the class of predicted pseudoknots is more restricted than in [40], this algorithm has the advantage of permitting study of conformational ensembles of most relevant structures. On the class of pseudoknots they can predict, they show their results are slightly worse for the false positives and slightly better for the true positives when compared to Rivas and Eddy's algorithm. The thermodynamic parameters for the pseudoknots have been generated individually by each group.

2.2 Secondary structure prediction of pairs of molecules

The closest related work to *PairFold* for predicting pseudoknot-free secondary structures for pairs of nucleic acids is by Mathews et al. [32]. They predict equilibrium affinity of complementary DNA or RNA oligonucleotides to an RNA target. Briefly, given a structured long RNA molecule $S = s_1 s_2 s_3 \dots s_n$ and the oligomer length l, their first program, called *OligoWalk*, generates the oligos which are Watson-Crick complements of every window of length l from S. There are $n - l + 1$ such possible oligos, for example the first one being $3' - \bar{s_1}\bar{s_2} \dots \bar{s_l} - 5'$, where $\bar{s_i}$ is the Watson-Crick complement of s_i. The target polymer can be in the folded or unfolded state, and the same for the oligomer, which can also be unimolecular or bimolecular. Assuming that the oligo will disrupt preexisting structure in the region of complementarity and will form a helix, OligoWalk calculates the standard free energy change of the duplex, which they call ΔG_3°. Then, they calculate "the overall free energy change of binding", $\Delta G_{overall}^\circ$, which takes into consideration the stability of the newly formed helix and the total concentration of the oligonucleotides. Performing a walk of $n-l+1$ steps along the target RNA, they try to find correlations between ΔG_3° or $\Delta G_{overall}^\circ$ and three types of experiments drawn from the literature. They find correlations suggesting that OligoWalk can be useful for designing oligomers capable of binding to targets.

The drawback of this method is that there is no guarantee whether or not the oligomer will disrupt the local structure of the target RNA. Even if the free energy of the duplex is low, thus the duplex being considered to be stable, there might be another region in the target RNA to which the target site would prefer to bind. In primer design and other oligomer-target binding problems, one would like to predict whether a given oligomer will be able to bind to that specific target site, and eventually use this information for primer design.

In the same paper, Mathews et al. [32] report a second program, an extension of *mfold* [70], but for bimolecular secondary structure prediction. Given two nucleic acids S_1 and S_2, a third sequence S is created by concatenating the two sequences and by adding a 3-nucleotide "molecular linker" between them. The three nucleotides will be restricted not to bind to any other bases in the two given sequences, and the free energy of the loop thus created will be an intermolecular initiation penalty plus the dangling ends free energies.

The intermolecular linker can appear in hairpin loops, bulges or internal loops, multi-loops or external loops. The Zuker and Stiegler's [72] algorithm is modified to deal with these situations.

The two programs: OligoWalk and the bimolecular secondary structure prediction program (which we refer to as BSSP) are incorporated in the *RNAstructure package* [42], a software package for Windows. However, only OligoWalk out of the two seems to be functional at the date of this thesis.

The algorithm behind the BSSP program seems to lead to the same results as what we propose as *PairFold*. However, *PairFold* is a more elegant extension of the Zuker and Stiegler's algorithm. Since, to our best knowledge, no evaluation of the BSSP program has been performed, it is not clear whether the results of the two programs are always the same.

A software tool close to OligoWalk is *ProbeSelect* by Li and Stormo [29], which designs probe oligomers for DNA microarrays. The best probes are selected based on the most favorable free energy, and also on maximization of the difference between this free energy and the free energy of the probe binding to any other mismatched target. To calculate this free energy quickly, they created a heuristic which is assuming the perfect alignment of alternatives. They expect their program to provide a good approximation to the optimal probes set for a complete genome.

Another, simpler extension of the *mfold* program is the "2-state hybridization server" [70], which is available online on Mfold web page [35]. It also adds a linker between the two input sequences, and it forces the bases in the linker not to pair. To our understanding, only the hairpin loop special case is covered. The free energy, enthalpy, entropy and melting temperatures are returned, but no information about the secondary structure is output by the program.

HyTher software tool, available online at [26], takes as input two RNA/RNA, RNA/DNA or DNA/DNA sequences of equal length. It only calculates the free energy of stacked pairs or mismatches at the corresponding positions in the two input sequences. No minimization algorithm for finding the most favorable duplex structure is performed, the assumption being that the two input sequences will have matches or mismatches at corresponding positions. Thus, the free energy returned with this assumption may be greater than the free energy returned by a minimization algorithm (such as *PairFold*). Also, input sequences of different length are not accepted.

2.3 Secondary structure prediction of combinatorial sets

An $O(n^3)$ algorithm for finding the combination with the smallest MFE structure out of all possible combinations in a combinatorial set has been developed by Cohen and Skiena [12]. The problem they tried to solve was determining, among all RNA sequences coding for a specific protein, which has the most stable secondary structure. They were interested in finding whether the real mRNA sequence was close to the RNA sequence they found. Their underlying algorithm is very similar with our *CombFold* algorithm, it uses the nearest neigbour thermodynamic model and is based on Zuker and Stiegler's [72] algorithm. Applying their algorithm on 200 short microbial RNA sequences, they found that a minimized sequence has, on average, a predicted MFE of 2.657 times lower than the naturally occurring sequence.

Chapter 3

The Nearest Neighbour Thermodynamic Model

All living organisms consume energy continuously. Their metabolism transforms energy to heat, which is dissipated to the environment. **Thermodynamics** originates from the Greek words *therme* (heat) and *dynamis* (power) and is the science which describes the relationship between different forms of energy. In thermodynamics, the part of interest (e.g. an organism) is seen as a *system*, and the rest of its universe is defined as the *surroundings.* Whether or not a system can exchange energy with its surroundings, it is said to be *open* or *closed.* All living entities are open systems.

The algorithms described in this thesis use the *Nearest Neighbour Thermodynamic Model (NNTM)*. This model assumes that the stability of a specific base pair depends on the neighbouring bases. Base pair stability is measured by the *standard free energy change* ΔG°. It is believed the most stable secondary structure of an RNA molecule or single-stranded DNA molecule is the one which has the lowest possible free energy change. In other words, the lower the free energy change, the more stable the secondary structure is.

First, we have to mention that none of what is explained in this chapter is our work. Instead, it is a thorough explanation of the *NNTM* model, which is heavily used in the algorithms we propose in this thesis. Section 1 describes the background on which the *NNTM* model is based. Section 2 explains the thermodynamic parameters determined by Turner's Lab [57]. The way these parameters are actually used for calculating the free energy of a secondary structure is explained in Section 3. An example of such calculation is given in Section 4. The model and parameters described in this chapter can also be used for pairs of RNA/RNA or DNA/DNA sequences. This particular situation is described in Section 5. We conclude the chapter with a discussion and some limitations of this model.

3.1 Background for NNTM

Calculation of standard free energy, enthalpy and entropy changes, equilibrium constant of the reaction and melting temperature are very important for determining characteristics of folded RNA or DNA. This section gives the background necessary to understand the role and the meaning of these parameters and the mathematical relations that connect them to each other.

It is important to mention that throughout this thesis we do not consider the dynamics of the RNA or DNA folding process. We only look at the RNA or DNA in the folded state, i.e. after the folding process reached an equilibrium and its state does not change any more (or the changes are insignificant).

Free energy, enthalpy and entropy

Free energy indicates the direction of a spontaneous change. It was introduced by J. W. Gibbs in 1878, and it is abbreviated G. ΔG represents the work done by a system at constant temperature and pressure when undergoing a *reversible* process. This system will spontaneously evolve in the direction that minimizes the Gibbs function:

$$\Delta G = \Delta H - T \cdot \Delta S, \tag{3.1}$$

where G is the free energy, H is the enthalpy, T is the absolute temperature (in degrees Kelvin (K)) and S is the entropy.

The quantity ΔG (measured in *kcal/mol*) is negative for "energy-releasing" processes, and positive for "energy-consuming" reactions. Sometimes we use the notation ΔG_T to denote the free energy change at a specific temperature T.

Enthalpy (H) is a measure of the heat flow that occurs in a process. The enthalpy change (ΔH) for an exothermic reaction (i.e. the heat flows from the system to the surroundings) is negative. The enthalpy change for an endothermic reaction (i.e. the heat flows from the surroundings to the system) is positive. The enthalpy (or enthalpy change) is measured in *kcal/mol*.

Entropy (S) is a thermodynamic function which measures the disorder of a system [54]. Thus, the entropy change ΔS measures the change in the degree of disorder. If ΔS is positive, it means there was an increase in the level of disorder. A negative value indicates a decrease in disorder. The entropy (or entropy change) is measured in $kcal/(mol\cdot K)$ or *entropy units* ($1eu = 1cal/(mol\cdot$ K)).

It is worth mentioning that enthalpy and entropy do depend on the temperature, but the dependency is very small, at least for short molecules [46], and they are not taken into consideration, but considered fixed for any temperature between 0°C and 100°C. Also, the standard free energy and entropy changes depend on the salt concentration [45], which in our study is assumed to be 1 M NaCl, and cannot be changed.

ΔG°, ΔH° and ΔS° denote standard free energy, enthalpy and entropy changes, i.e. measured at standard conditions such as pressure (1 *atm*), and the temperature of interest. For nucleic acids, it is very important to determine thermodynamic parameters at the human body temperature [46] as accurately as possible. It has been shown the measurements for free energy change are more accurate than for enthalpy and entropy change (a standard error of 2-5% for ΔG°_{37}, versus 5-8% for ΔH° and ΔS°) [46].

Since we look at the RNA or DNA secondary structure only when the folding process reached its equilibrium, we measure the *standard* free energy, enthalpy and entropy changes. If, at some temperature, enough energy was consumed by the process, then interactions between the nucleotides of the molecule can happen. In this case, we say the molecule *has secondary structure*. To get to this state, the process was energy-releasing and exothermic (hence $\Delta G^\circ < 0$ and $\Delta H^\circ < 0$) and it tended to get ordered (hence $\Delta S^\circ < 0$). If not enough energy was consumed, then there are no bondings between the bases. In this situation, we say that the molecule is *structure free*, i.e it does not have secondary structure. The standard free energy, enthalpy and entropy changes will be 0.

Equilibrium constant

Consider the general reaction scheme $aA + bB \rightleftharpoons cC + dD$. Let C_A, C_B, C_C and C_D denote the concentrations of the reactants A, B and the products C, D, measured in *mol/l* (or *Molar* (M)). $k = \frac{C_C^c \cdot C_D^d}{C_A^a \cdot C_B^b}$ is known as the equilibrium constant of the reaction. The free energy change is given by:

$$\Delta G = \Delta G^\circ + R \cdot T \cdot ln(k),$$

where R is the gas constant (1.98717 $cal/(mol\cdot$ K)), and T is the absolute temperature. If the system is at equilibrium, $\Delta G = 0$. Hence, ΔG° can be calculated if the concentration of the reactants and products is known:

$$\Delta G^\circ = -R \cdot T \cdot ln(k) \tag{3.2}$$

Depending on the reactants' type and their concentration, the equlibrium constant k can get one of the following values:

- $k = 1/(C_A + C_B)$ for self-complementary oligonucleotides[1];
- $k = 4/(C_A + C_B)$ for non-self complementary molecules if $C_A = C_B$;
- $k = 1/(C_A + C_B/2)$ for non-self complementary molecules if $C_A > C_B$.

[1] Short sequences ($\sim$ 20 bases or less), which perfectly fold to themselves at the midpoint. For example, the DNA oligonucleotide `CGATAATCG` is self-complementary since the first base can pair with the last base, the second base can pair with the base before the last base etc.

Melting temperature

From equations 3.1 and 3.2, we can determine the melting temperature T_m of different types of molecules: (1) The melting temperature of double-stranded DNA or an RNA molecule bound to a complementary molecule is defined as the temperature at which 50% of the strands are in the double-helical state and 50% are in the unfolded state [45]; (2) The melting temperature of a long RNA or single-stranded DNA molecule is the temperature at which 50% of the base pairs have been denatured, leading to molecules containing alternating stems and denatured regions (loops) [64].

$$T_m = \frac{\Delta H^\circ}{\Delta S^\circ - R \cdot ln(k)} - 273.15 \tag{3.3}$$

This formula gives the melting temperature in degrees Celsius and assumes an ionic concentration $[Na^+]$ of 1M. Hence, it does not show the dependence of the melting temperature on the ionic concentration. The following formula takes it into consideration [64]:

$$T_m = \frac{\Delta H^\circ}{\Delta S^\circ - R \cdot ln(k)} + 16.6 \cdot log_{10} \frac{[Na^+]}{1.0 + 0.7[Na^+]} - 269.3 \tag{3.4}$$

The standard free energy changes at 37°C (ΔG°_{37}) and the standard enthalpy changes ΔH° were experimentally determined by Turner's Lab for RNA [49], and by SantaLucia Lab for DNA [45]. Parameters for RNA have been refined by Mathews et al. [33] mainly by knowledge based methods. Using equation 3.1, one can determine the standard free energy change at any temperature between 0°C and 100°C. Also, using equations 3.3 and 3.4 input concentrations, and reactants type, one can determine the melting temperature of the given molecule(s). Note that in the following sections, when we refer to free energy or energy, we mean *standard free energy change* (ΔG°). Similarly, when we refer to enthalpy or entropy, we mean *standard enthalpy change* ΔH° and *standard entropy change* ΔS°.

3.2 Thermodynamic parameters

Thermodynamic parameters for RNA and DNA folding have been determined by different methods such as optical melting methods [46, 67], absorbance melting curves, microcalorimetry [46] and knowledge-based methods using databases of known structures [33]. All parameters for RNA that we use have been published [16, 33, 49, 57, 58, 59, 65]. For DNA, only parameters for stacked loops [45], single mismathes [1, 2, 3, 4, 37] and dangling ends [7] have been published. However, for DNA we use the same model as for RNA, our base of parameters containing unpublished results obtained by communication with John SantaLucia Jr. [44]. In this section, we present all types of RNA parameters that we use and their format. Other parameters exist, such as coaxial stacking[2] [62, 63, 71], but are not included in our model yet.

[2] Stacking interactions between adjacent helices.

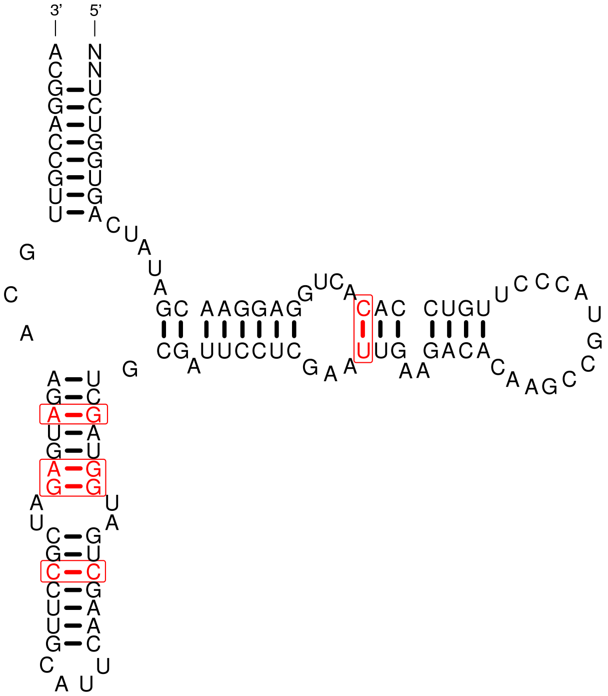


Figure 3.1: Secondary structure of 5S Ribosomal RNA of Staphylococcus aureus [21]. This secondary structure, determined by comparative sequence analysis, contains "odd-pairs", marked by boxes: *(C.U)*, *(A.G)*, *(G.G)* and *(C.C)*.

Watson-Crick base pairs, i.e. *(C.G)* and *(A.U)*, are very important in RNA secondary structures, and thus, they have been studied extensively. The next most common base pairs are wobble pairs, i.e. *(G.U)*. This section presents the thermodynamic parameters determined by Turner's Laboratory [57] and refined by Mathews et al. [33], which considers Watson-Crick pairs and wobble pairs. *NNTM* assigns free energy changes to loops rather than to base pairs [71]. The orientation of the base pairs matters, *(C.G)* being different from *(G.C)*. Thus, six different base pairs are possible: *(C.G)*, *(G.C)*, *(A.U)*, *(U.A)*, *(G.U)* and *(U.G)*. So-called "odd pairs"[3], between *(A.A)*, *(C.C)*, *(G.G)*, *(U.U)*, *(A.C)*, *(A.G)* and *(U.C)*, exist (see Figure 3.2). A database of such pairs in known RNA structures is available [36]; however, currently no or very few parameters are available to predict them [18].

The free energy of a specific secondary structure is based on the NNTM, which simply sums up the contributions of elementary motifs of the structure. Figure 3.2 shows a sequence S and its predicted minimum free energy secondary structure R. Several elementary motifs

[3] Sometimes, these base pairs are called "non-canonical pairs". But wobble pairs are also considered non-canonical pairs. Thus, to not create confusion, we call all possible pairs which are not Watson-Crick, nor wobble, "odd pairs".

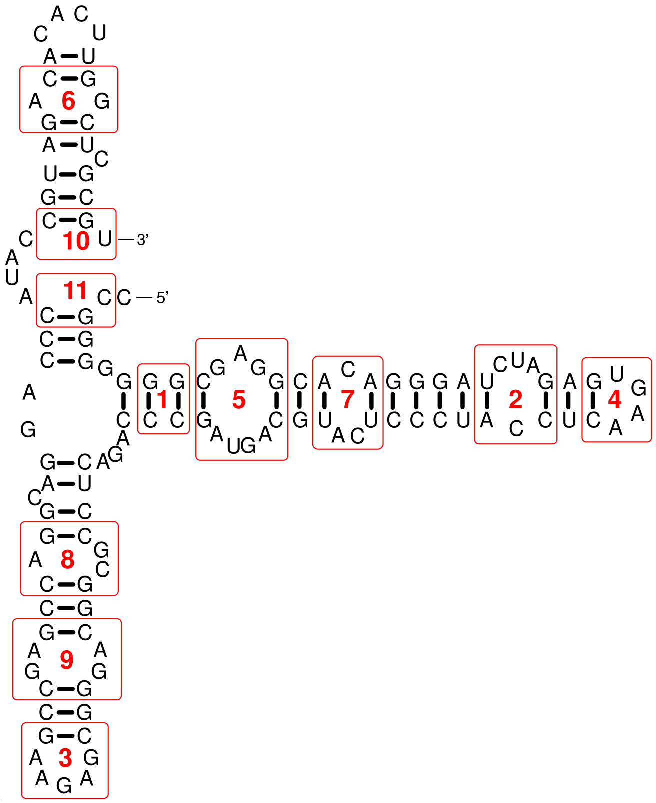


Figure 3.2: Secondary structure of an arbitrarily chosen RNA sequence. It contains all the elementary structures, marked by different labels.

are marked by different labels and will be used as examples, for each particular type of free energy. In the following, we present the thermodynamic parameters for RNA, as determined by Turner's Laboratory [49, 57] and refined by Mathews et al. [33].

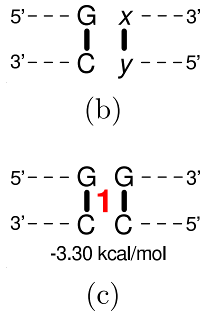
The thermodynamic parameters discussed in this section also contain the standard enthalpy changes for all the situations which will be described in this section. Having the standard enthalpy change and the standard free energy change at $37°C$ for a particular secondary structure, it is straightforward that, using Equation 3.1, we can calculate the standard entropy change, the standard free energy change at any given temperature between $0°C$ and $100°C$ and the melting temperature.

Free energies for stacked loops

The table in Figure 3.3 shows the free energies of stacked loops whose closing pair is *(G.C)*. ΔG-*Stack*(a, b, x, y) denotes the general value for a stacked loop, where $a, b, x, y \in \{A, C, G, U\}$, and *(a.b)*, *(x.y)* form pairs. The table shows ΔG-*Stack(G,C,x,y)*, and the energies for stacked loops are always favourable (negative energies). Note that the values are duplicated in these tables, since $\Delta G\text{-}Stack(a,b,x,y) = \Delta G\text{-}Stack(y,x,b,a)$. Because there are six possible base pairs for each pair, there are $6 \times 6 = 36$ possible stacked loops (with duplicated values). “Odd pairs” energies are denoted by dashes, meaning that no pairing between the corresponding bases is possible, hence the free energy of binding is considered infinite. An example, showing an elementary structure of Figure 3.2, is given in (c).

y x	A	C	G	U
A	-	-	-	-2.40
C	-	-	-3.40	-
G	-	**-3.30**	-	-1.50
U	-2.20	-	-2.50	-

(a)



(b)

(c)

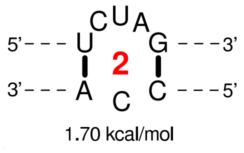
Figure 3.3: (a) Example of a free energy table for stacked loops of the type shown in (b). (c) One particular value from the table, where $x = G$ and $y = C$.

Destabilizing energies by loop size

A free energy penalty (i.e. a positive free energy change) is associated with each hairpin loop, internal loop or bulge, depending on the length of the loop (i.e. the number of free bases between the closing pairs). The table in Figure 3.4 partially shows these parameters. The functions are called $\Delta G\text{-}Length\text{-}Internal(l)$, where l is the length (or size) of the loop, $\Delta G\text{-}Length\text{-}Bulge(l)$ and $\Delta G\text{-}Length\text{-}Hairpin(l)$. Tabulated values exist for $l \leq 30$. For longer loops, a function described in section 3.3 is used. The dashes signify that no loop formation of the corresponding type is possible, and thus, we consider their free energy to be infinite.

Size	Internal	Bulge	Hairpin
1	-	3.80	-
2	-	2.80	-
3	-	3.20	5.70
4	**1.70**	3.60	5.60
5	1.80	4.00	5.60
⋮	⋮	⋮	⋮
30	3.70	6.10	7.70

(a)



(b)

Figure 3.4: (a) Partial snapshot of the table showing the free energy penalties up to the size of the hairpin loop or internal loop. (b) An example of an internal loop of length 4.

Free energies for general hairpin loops

x \ y	A	C	G	U
A	-1.50	-1.50	-1.40	-1.80
C	-1.00	-0.90	-2.90	-0.80
G	**-2.20**	-2.00	-1.60	-1.10
U	-1.70	-1.40	-1.80	-2.00

(a)

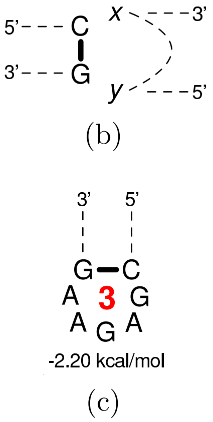


Figure 3.5: (a) A free energy table for hairpin loops of the type shown in (b). An example, where $x = G$ and $y = A$, is shown in (c).

The free energy of a hairpin loop depends on the closing pair and the neighbouring free bases. They are sometimes called *terminal mismatch* free energies for hairpin loops. We call this function ΔG-*Hairpin-n*(a, b, x, y), with *(a.b)*. The table in Figure 3.5 shows such energies for hairpins whose closing pair is *(C.G)*, that is, the function is ΔG-*Hairpin-n*(C, G, x, y). Note that they are negative energies, regardless of the free bases. There are $6 \times 16 = 96$ different hairpins in this case.

Free energies for hairpin loops of length 4

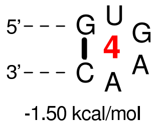
It is believed that hairpins of size 4 are particularly stable. For this reason, bonus values have been determined for such hairpins, as a function of the base pair and all the free bases between them. These bonus values will be added to other thermodynamic parameters for hairpins (see section 3.3). The function is called ΔG-*Hairpin-4*(a, b, c, d, e, f), with *(a.f)*, and a few examples are illustrated in Figure 3.6. Similar bonus values for hairpin loops of size 3 have been determined for DNA [45] (ΔG-*Hairpin-3*(a, b, c, d, e), with *(a.e)*). For hairpins of size 4, $6 \times 4^4 = 1536$ values are possible, but only 30 are included in the current version of the parameters. For the hairpins not included in the table, no bonus is added.

Free energies for general internal loops

For internal loops, *terminal mismatch* free energies have been determined. They are a function of a closing base pair and the neighbouring free bases: ΔG-*Internal-n*(a, b, x, y), with *(a.b)*. The function is applied to both exterior and interior base pairs. Note that for the interior base pair, the order of the variables is reversed. Figure 3.7 shows the table for ΔG-*Internal-n*(C, G, x, y), a general internal loop of this type and an example. There are

Sequence	Energy
GGGGAC	-3.00
GGUGAC	-3.00
⋮	⋮
CGAAGG	-2.50
CUACGG	-2.50
⋮	⋮
CGAGAG	-2.00
⋮	⋮
GUGAAC	**-1.50**
UGGAAA	-1.50

(a)



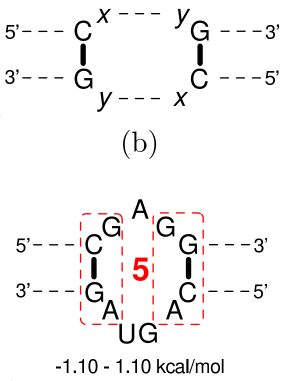
(b)

Figure 3.6: (a) Examples of bonus values for hairpin loops of size 4. (b) An example of such hairpin, along with its associated bonus.

6 × 16 = 96 values in these tables. Particular tabulated values exist for internal loops of size 2, 3 and 4, and will be detailed below.

y x	A	C	G	U
A	-0.00	-0.00	**-1.10**	-0.00
C	-0.00	-0.00	-0.00	-0.00
G	**-1.10**	-0.00	-0.00	-0.00
U	-0.00	-0.00	-0.00	-0.70

(a)



(b)

(c)

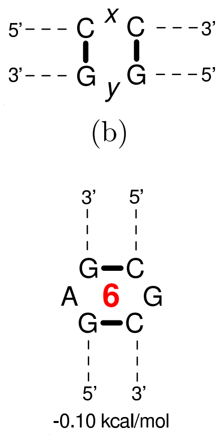
Figure 3.7: (a) Table for internal loop terminal mismatches whose closing pair is *(C,G)*. This type of internal loop is shown in (b). (c) An example of an internal loop, where the two terminal mismatches use values from the table in (a): $x = G$, $y = A$ and $x = A$, $y = G$.

Free energies for symmetric internal loops of size 2

Symmetric internal loops of size 2 (one free base on each side) have been particularly studied, and parameters have been determined for them. The function giving these values is ΔG-*Internal-2*(a, b, m, n, x, y), which depends on the two base pairs *(a.b)*, *(m.n)* and the two free bases x and y. The table in Figure 3.8 shows the parameters for internal loops whose closing pairs are *(C,G)* and *(C,G)*, i.e. ΔG-*Internal-2*(C, G, C, G, x, y). This type and an

x \ y	A	C	G	U
A	0.40	0.30	**-0.10**	0.40
C	-0.40	0.50	0.40	0.00
G	0.40	0.40	-1.70	0.40
U	0.40	0.50	0.40	-0.30

(a)



(b)

(c)

Figure 3.8: (a) Free energies for symmetric internal loops of size 2, of the type shown in (b). (c) shows an example of such an internal loop, where $x = G$ and $y = A$.

example are also shown. There are $6 \times 6 \times 16 = 576$ parameters for this case. Note that these values are duplicated, since $\Delta G\text{-}Internal\text{-}2(a, b, m, n, x, y) = \Delta G\text{-}Internal\text{-}2(n, m, b, a, y, x)$.

Free energies for asymmetric internal loops of size 3

Asymmetric internal loops of size 3 are internal loops which have one free base on one side and two free bases on the other side. The two tables in Figure 3.9 (a) and (d) show these values for internal loops of types illustrated in part (b) and (e), respectively. The general function is called $\Delta G\text{-}Internal\text{-}3(a, b, m, n, x, y, z)$, with *(a.b)* and *(m.n)*. Note that these values are also applicable for the reverse situation, in which the two free bases appear closer to the $5'$ end. The figure shows two particular cases, with example for normal case and the reverse case: (1) $\Delta G\text{-}Internal\text{-}3(A, U, A, U, x, y, C)$. The example (c) uses $x = C$ and $y = A$. (2) $\Delta G\text{-}Internal\text{-}3(C, G, G, C, x, y, G)$. The example (c) uses $x = A$ and $y = C$. There are $6 \times 6 \times 4^3 = 2304$ parameters for this case.

Free energies for symmetric internal loops of size 4

Another special case of internal loops for which the thermodynamic parameters were tabulated are symmetric internal loops of size 4, having two free bases on each side. Figure 3.10 shows the function which gives these values, $\Delta G\text{-}Internal\text{-}4(a, b, m, n, v, w, x, y)$, with *(a.b)* and *(m.n)*. The table shows a partial snapshot of $\Delta G\text{-}Internal\text{-}4(C, G, G, C, v, w, x, y)$. Part (c) shows an example, where $v = A$, $w = A$, $x = G$ and $y = G$. Note that the values in the table are duplicated, and for the given example, both $\Delta G\text{-}Internal\text{-}4(C, G, G, C, A, A, G, G)$ and $\Delta G\text{-}Internal\text{-}4(C, G, G, C, G, G, A, A)$ are valid. There are $6 \times 6 \times 4^4 = 9216$ parameters for symmetric internal loops of size 4.

y x	A	C	G	U
A	3.60	3.20	3.10	5.50
C	**3.70**	4.00	5.50	3.70
G	5.50	5.50	5.50	5.50
U	5.50	3.70	5.50	2.80

(a)

y x	A	C	G	U
A	1.00	**0.60**	0.40	4.00
C	4.00	4.00	4.00	4.00
G	0.80	4.00	2.20	4.00
U	4.00	4.00	4.00	4.00

(d)

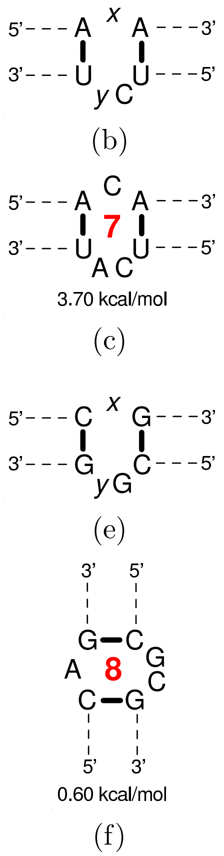


Figure 3.9: (a) Table with free energy parameters for asymmetric internal loops of size 3, for the type shown in (b). (c) shows an example of such an internal loop, where $x = C$ and $y = A$, along with its free energy. (d) Table with free energy parameters for asymmetric internal loops of size 3, for the type shown in (e). (f) shows an example of such an internal loop, where $x = A$ and $y = C$.

Free energies for dangling ends

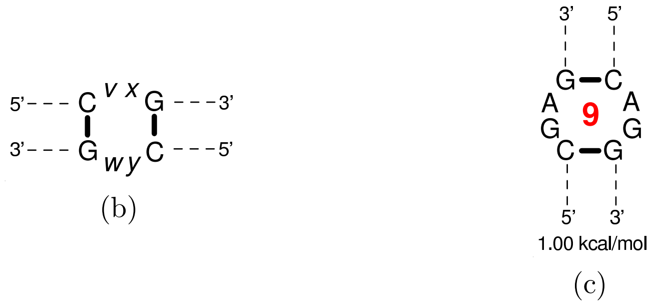
Dangling bases are free bases located in the immediate vicinity of a stem. They may have a contribution to the stability of the structure. Figure 3.12 shows tables for ΔG-*Dangle-3'*(a, b, x), with *(a.b)*, (the free dangling base is close to the $3'$ end), where $a = G$ and $b = C$ (part (a)), and ΔG-*Dangle-5'*(a, b, x), with *(a.b)* (the dangling base is close to the $5'$ end), where $a = C$ and $b = G$ (part (b)). The figure contains two examples, one for the *3'* end: $x = U$, and one for the *5'* end: $x = C$. There are $6 \times 4 \times 2 = 48$ parameters for dangling ends.

Miscellaneous free energy rules

Miscellaneous other parameters are used for multiloops, asymmetric internal loops, special cases of hairpin loops, etc. Table 3.1 gives these parameters, together with their role and a

xy	AA	AC	AG	AU	CA	…	GG	GU	UA	UC	UG	UU
vw												
AA	1.30	1.20	0.30	2.00	1.60	…	**1.00**	-0.40	2.00	1.90	1.10	1.40
AC	1.60	1.50	0.60	2.00	2.00	…	1.40	-1.10	2.00	1.70	0.40	1.80
AG	0.30	0.20	-0.70	2.00	0.60	…	0.00	-0.60	2.00	1.30	0.90	1.30
AU	2.00	2.00	2.00	2.00	2.00	…	2.00	2.00	2.00	2.00	2.00	2.00
CA	1.20	1.10	0.20	2.00	1.50	…	0.90	-1.50	2.00	1.20	0.00	0.30
⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮	⋮	⋮	⋮	⋮	⋮
GG	**1.00**	0.90	0.00	2.00	1.40	…	0.80	-0.70	2.00	1.70	0.90	1.20
GU	1.10	0.00	0.90	2.00	0.40	…	0.90	-2.60	2.00	1.10	-1.10	1.10
UA	2.00	2.00	2.00	2.00	2.00	…	2.00	2.00	2.00	2.00	2.00	2.00
UC	1.90	1.20	1.30	2.00	1.70	…	1.70	-0.40	2.00	1.40	1.10	0.50
UG	-0.40	-1.50	-0.60	2.00	-1.10	…	-0.70	-4.20	2.00	-0.40	-2.60	-0.50
UU	1.40	0.30	1.30	2.00	0.80	…	1.20	-0.50	2.00	0.50	1.10	-0.40

(a)



(b)

(c)

Figure 3.10: (a) Partial snapshot of the table showing the free energy for symmetric internal loops of size 4, of type shown in (b). (c) shows an example, where $v = A$, $w = A$, $x = G$ and $y = G$.

conventional name. Two special types of hairpin loops need to be described here:

1. A *GGG hairpin loop* is a hairpin loop closed by s_i and s_j $(i < j)$ with $s_{i-2} = s_{i-1} = s_i = G$ and $s_j = U$;
2. A *Poly-C hairpin loop* is a hairpin loop where all the free bases are C: $s_{i+1} = \ldots = s_{j-1} = C$.

Briefly, the parameters in Table 3.1 are used in the following situations: (1) is used for long hairpins, internal loops or bulges; (2,3) are used for asymmetric internal loops; (4,5,6) are used for multibranched loops; (7) is a penalty for stems which end in a base pair different from *(C.G)*; (8) is used for *GGG hairpin loops*; (9,10,11) give a bonus for *Poly-C hairpins*; (12) is a penalty added to the secondary structure prediction of a pair of molecules, rather than of one single molecule (see Section 3.5) and (13) decides on the calculation of grossly asymmetric internal loops, with one side of length 1. More details about the cases in which each of these parameters is used are given in Sections 3.3 and 3.5.

x	A	C	G	U
	-1.10	-0.40	-1.30	**-0.60**

(a)

x	A	C	G	U
	-0.20	**-0.30**	-0.00	-0.00

(d)

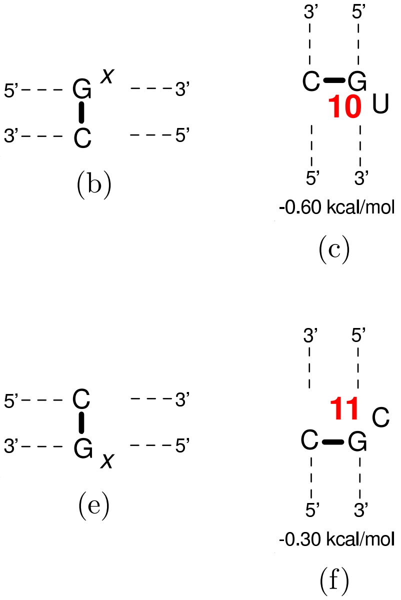


Figure 3.11: (a) Free energies for *3'* dangling ends, of the type shown in (b). (c) shows an example of a *5'* dangling end, where $x = U$. (d) Free energies for *5'* dangling ends, of type shown in (e). (f) shows an example of a *5'* dangling end, where $x = C$.

No	Role	Name	Value
1	Extrapolation for large loops for internal loops, bulges or hairpin loops greater than 30	*Len-Par*	1.079
2	Asymmetric internal loops: the maximum correction	*Asym-Max*	3.00
3	Asymmetric internal loops: the Ninio array	*Asym-Par*	.50 .50 .50 .50
4	Multibranched loops - offset	*Multi-a*	3.40
5	Multibranched loops - helix penalty	*Multi-b*	0.40
6	Multibranched loops - free base penalty	*Multi-c*	0.00
7	Penalty for non-GC terminal	*non-GC-terminal*	0.50
8	Bonus for GGG hairpin	*bonusGGG*	-2.20
9	Poly-C hairpin slope	*C-Hairpin-1*	0.30
10	Poly-C hairpin intercept	*C-Hairpin-2*	1.60
11	Poly-C hairpin of 3	*C-Hairpin-3*	1.40
12	Intermolecular initiation free energy	*Intermol*	4.10
13	GAIL Rule (Grossly Asymmetric Interior Loop Rule) (on=1, off=0)	*Gail-Rule*	1

Table 3.1: Miscellaneous free energy rules.

3.3 Free energy calculation of a secondary structure

This section describes how to calculate the free energy of a secondary structure, using the parameters described in the previous section. We must recall the reader the equations presented here are not our work, but are a detailed revision from the literature on free energy calculation.

General functions

In the following, some general functions, used by calculations for different types of structures, are described.

Studies have shown the helices whose exterior pairs are not *(C.G)* are less stable. The value *Non-GC-terminal* is meant to add a penalty to capture this destabilization. In some references, the nomenclature of *AU terminal penalty* is used. Note that this penalty is also added for wobble pairs. Thus, to avoid confusion, we call it *Non-GC terminal penalty.* The function *Non-GC-Penalty*(a, b) is used, and it is calculated as follows:

$$\textit{Non-GC-Penalty}(a,b) = \begin{cases} 0 & \text{, if } \textit{(a.b)} \text{ is } \textit{(C.G)} \text{ or } \textit{(G.C)} \\ \textit{Non-GC-Terminal}, & \text{otherwise} \end{cases}$$

Penalties corresponding to the size of hairpin loops and internal loops (including bulges), are considered. They are calculated as follows:

$$\Delta G\textit{-Length-H}(l) = \begin{cases} \Delta G\textit{-Length-Hairpin}(\mathrm{l}) & , \quad l \leq 30 \\ \Delta G\textit{-Length-Hairpin}(30) + \textit{Len-Par} \times \textit{log}(l/30), & l > 30 \end{cases}$$

l denotes the length of the hairpin, i.e. the number of free bases. The length penalties for bulges and internal loops can be calculated similarly:

$$\Delta G\textit{-Length-B}(l) = \begin{cases} \Delta G\textit{-Length-Bulge}(\mathrm{l}) & , \quad l \leq 30 \\ \Delta G\textit{-Length-Bulge}(30) + \textit{Len-Par} \times \textit{log}(l/30), & l > 30 \end{cases}$$

$$\Delta G\textit{-Length-I}(l) = \begin{cases} \Delta G\textit{-Length-Internal}(\mathrm{l}) & , \quad l \leq 30 \\ \Delta G\textit{-Length-Internal}(30) + \textit{Len-Par} \times \textit{log}(l/30), & l > 30 \end{cases}$$

Dangling bases can add some stabilization, up to the neighbours involved. A function which we call ΔG*-Dangle* (S, i_1, j_1, i_2, j_2) is used mainly in multi-loops and in multi-domain structures and is calculated as follows (see Figure 3.12):

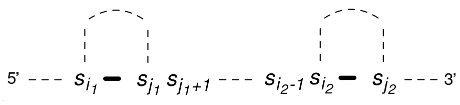

5' --- s_{i_1} — s_{j_1} s_{j_1+1} --- $s_{i_2\text{-}1}$ s_{i_2} — s_{j_2} --- 3'

Figure 3.12: Dangling bases between two branches of a multi-loop or multi-domain structure.

$$\Delta G\textit{-Dangle}(S, i_1, j_1, i_2, j_2) =$$

$$\begin{cases} \Delta G\textit{-Dangle-3'}(s_{j_1}, s_{i_1}, s_{j_1+1}) + \Delta G\textit{-Dangle-5'}(s_{j_2}, s_{i_2}, s_{i_2-1}) & , \quad i_1 + 1 < i_2 - 1 \\ \textit{min}(\Delta G\textit{-Dangle-3'}(s_{j_1}, s_{i_1}, s_{j_1+1}), \Delta G\textit{-Dangle-5'}(s_{j_2}, s_{i_2}, s_{i_2-1})), & i_1 + 1 = i_2 - 1 \\ 0 & , \quad i_1 + 1 > i_2 - 1 \end{cases}$$

Free energy calculation for stacked loops

Given the sequence S, the free energy of a stacked loop $5' - s_i s_{i+1} \dots s_{j-1} s_j - 3'$, with *$(s_i.s_j)$*, *$(s_{i+1}, s_{j-1})$* (Figure 3.13), is given by $\Delta G\text{-}S(S,i,j) = \Delta G\text{-}Stack(s_i, s_j, s_{i+1}, s_{j-1})$.

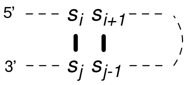


Figure 3.13: A general stacked loop structure.

Free energy calculation for hairpin loops

Procedure 1 describes the calculation of the hairpin loop free energy for sequence S, where the hairpin closing pair is *(s_i,s_j)*. (Figure 3.14).

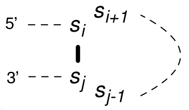


Figure 3.14: A general hairpin loop.

The function is called $\Delta G\text{-}H(S,i,j)$. Hairpins having length shorter than three are not accepted by this model. The free energy of the hairpins of size greater than or equal to three are made of four quantities: ΔG_1, ΔG_2, ΔG_3 and ΔG_4:

- ΔG_1 corresponds to the penalty associated to the hairpin length.
- For hairpins of size 3, ΔG_2 contains the value of the *non-GC* penalty function. For hairpins of size greater than 3, ΔG_2 is the terminal mismatch value, which includes the *non-GC* penalty (and thus, it does not have to be added separately).
- If the hairpin is of size 3 or 4, and a bonus for it was tabulated (ΔG-*Hairpin-3* or ΔG-*Hairpin-4*), ΔG_3 contains this bonus.
- If the hairpin is a special case, such as *GGG hairpin* or *Poly-C hairpin*, then ΔG_4 contains this value. For *Poly-C hairpins*, this is a function of the hairpin length.

Free energy calculation for internal loops

As mentioned before, we consider that bulge loops are a special case of internal loops. The length of an internal loop is given by the number of free bases between the two closing base pairs, which we denote with $(i.j)$ and $(i'.j')$. Let us call l_1 the length of one side of the

Hairpin-Loop-Free-Energy Procedure

input: sequence S, i, j;
output: free energy ΔG;

```
procedure Compute $\Delta G$-H
    $\Delta G_1 := 0$; $\Delta G_2 := 0$; $\Delta G_3 := 0$; $\Delta G_4 := 0$;
    $l := j - i - 1$;
    if $(l < 3)$
        $\Delta G :=$ infinity;
    else
        $\Delta G_1 := \Delta G\text{-}Length\text{-}H(l)$;
        if $(l = 3)$
            $\Delta G_2 := Non\text{-}GC\text{-}penalty(s_i, s_j)$;
            $\Delta G_3 := \Delta G\text{-}Hairpin\text{-}3(s_i, s_{i+1}, s_{i+2}, s_{j-1}, s_j)$;
        else
            $\Delta G_2 := \Delta G\text{-}Hairpin\text{-}n(s_i, s_j, s_{i+1}, s_{j-1})$;
            if $(l = 4)$
                $\Delta G_3 := \Delta G\text{-}Hairpin\text{-}4(s_i, s_{i+1}, s_{i+2}, s_{j-2}, s_{j-1}, s_j)$;
            endif;
        endif;
        if $(s_{i-2} = s_{i-1} = s_i =' G'$ and $s_j =' U')$
            $\Delta G_4 := bonusGGG$;
        else if $(s_{i+1} = \ldots = s_{j-1} =' C')$
            if $(l = 3)$
                $\Delta G_4 := C\text{-}Hairpin\text{-}3$;
            else
                $\Delta G_4 := C\text{-}Hairpin\text{-}2 + C\text{-}Hairpin\text{-}1 \times l$;
            endif;
        endif;
        $\Delta G := \Delta G_1 + \Delta G_2 + \Delta G_3 + \Delta G_4$;
    endif;
    return $\Delta G$;
end procedure $\Delta G$-H.
```

Procedure 1: Outline of the calculation for hairpin loop free energy.

loop, i.e. $l_1 = i' - i - 1$. Then, l_2 will be the length of the other side: $l_2 = j - j' - 1$. The length of the loop will be $l = l_1 + l_2$.

If $l_1 \neq l_2$, then we say the internal loop is asymmetric. Studies have shown that asymmetric internal loops are less stable than symmetric loops. The function $\Delta G\text{-}Asymmetry(l_1, l_2)$ gives this penalty:

$$\Delta G\text{-}Asymmetry(l_1, l_2) = min \begin{cases} Asym\text{-}Max \\ |l_1 - l_2| \times Asym\text{-}Par[min(2, min(l_1, l_2)) - 1] \end{cases}$$

Procedure 2 shows the calculation of the function $\Delta G\text{-}I(S, i, j, i', j')$, which gives the free energy of an internal loop or bulge closed by (s_i, s_j) and $(s_{i'}, s_{j'})$ (see Figure 3.15).

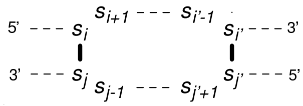


Figure 3.15: Example of a general internal-loop structure.

If it is a bulge of size 1, it is considered as a stacked loop. If the size is greater than 1, than only the *non-GC* penalties for both pairs are added. The function of the length of the bulge is added, and this concludes the bulge case. If it is an internal loop of special case (i.e. of size 2,3 or 4), for which there are tabulated values, then its free energy is given by the corresponding function alone. For any other type of internal loop, four quantities: ΔG_1, ΔG_2, ΔG_3 and ΔG_4 are added:

- ΔG_1 is the penalty dependant of length.
- ΔG_2 and ΔG_3 are the terminal mismatches corresponding to each of the two closing pairs.
- ΔG_4 is the assymetry penalty, calculated as described above.

Free energy calculation for multibranched loops

Consider a multibranched loop with $k + 1$ branches, and whose closing pairs are (s_i, s_j), (s_{i_1}, s_{j_1}), $\ldots$ (s_{i_k}, s_{j_k}) (see figure 3.16).

The multiloops are calculated using the following formula:

$$\begin{aligned} &\Delta G\text{-}M(S, i, j, i_1, j_1, \ldots, i_k, j_k) = \\ &\quad Multi\text{-}a + \\ &\quad Multi\text{-}b \times (k + 1) + \end{aligned}$$

Internal-Loop-Free-Energy Procedure

input: sequence S, i, j, i', j';
output: free energy ΔG;

```
procedure Compute ΔG-I
    $\Delta G_1 := 0$; $\Delta G_2 := 0$; $\Delta G_3 := 0$; $\Delta G_4 := 0$;
    $l_1 := i' - i - 1$; $l_2 := j - j' - 1$; $l = l_1 + l_2$
    if ($l_1 = 0$ or $l_2 = 0$)
        $\Delta G := \Delta G\text{-}Length\text{-}B(l)$;
        if ($l_1 + l_2 = 1$)
            $\Delta G := \Delta G + \Delta G\text{-}Stack(s_i, s_j, s_{i'}, s_{j'})$
        else
            $\Delta G := \Delta G + Non\text{-}GC\text{-}Penalty(s_i, s_j) + Non\text{-}GC\text{-}Penalty(s_{i'}, s_{j'})$;
        endif;
    else if ($l_1 = 1$ and $l_2 = 1$)
        $\Delta G := \Delta G\text{-}Internal\text{-}2(s_i, s_j, s_{i'}, s_{j'}, s_{i+1}, s_{j-1})$;
    else if ($l_1 = 1$ and $l_2 = 2$)
        $\Delta G := \Delta G\text{-}Internal\text{-}3(s_i, s_j, s_{i'}, s_{j'}, s_{i+1}, s_{j-1}, s_{j'+1})$;
    else if ($l_1 = 2$ and $l_2 = 1$)
        $\Delta G := \Delta G\text{-}Internal\text{-}3(s_{j'}, s_{i'}, s_j, s_i, s_{j-1}, s_{i'-1}, s_{i+1})$;
    else if ($l_1 = 2$ and $l_2 = 2$)
        $\Delta G := \Delta G\text{-}Internal\text{-}4(s_i, s_j, s_{i'}, s_{j'}, s_{i+1}, s_{j-1}, s_{i'-1}, s_{j'+1})$;
    else
        $\Delta G_1 := \Delta G\text{-}Length\text{-}I(l)$;
        if (($l_1 = 1$ or $l_2 = 1$) and $Gail\text{-}Rule = 1$)
            $\Delta G_2 := \Delta G\text{-}Internal\text{-}n(s_i, s_j,' A',' A')$;
            $\Delta G_3 := \Delta G\text{-}Internal\text{-}n(s_{i'}, s_{j'},' A',' A')$;
        else
            $\Delta G_2 := \Delta G\text{-}Internal\text{-}n(s_i, s_j, s_{i+1}, s_{j-1})$;
            $\Delta G_3 := \Delta G\text{-}Internal\text{-}n(s_{j'}, s_{i'}, s_{j'+1}, s_{i'-1})$;
        endif;
        $\Delta G_4 := \Delta G\text{-}Asymmetry\ (l_1, l_2)$;
        $\Delta G := \Delta G_1 + \Delta G_2 + \Delta G_3 + \Delta G_4$;
    endif;
    return $\Delta G$;
end procedure ΔG-I.
```

Procedure 2: Outline of the calculation for internal loop free energy.

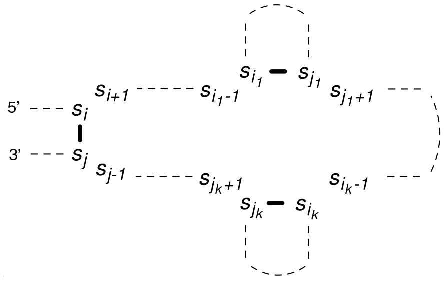


Figure 3.16: Example of a general multi-loop structure, with $k + 1$ branches.

$$\begin{aligned}
&\textit{Multi-c} \times ((i_1 - i - 1) + \sum_{h=1}^{k-1}(i_{h+1} - j_h - 1) + (j - j_k - 1)) + \\
&\textit{Non-GC-penalty}(s_i, s_j) + \sum_{h=1}^{k} \textit{Non-GC-penalty}(s_{i_h}, s_{j_h}) + \\
&\Delta G\textit{-Dangle}(S, j, i, i_1, j_1) + \sum_{h=1}^{k-1} \Delta G\textit{-Dangle}(S, i_h, j_h, i_{h+1}, j_{h+1}) + \\
&\Delta G\textit{-Dangle}(S, i_k, j_k, j, i);
\end{aligned}$$

First, a penalty for starting a new multiloop is added (*Multi-a*). For each branch of the multiloop, including the exterior pair, the value *Multi-b* is added. Also, for each free base, the value *Multi-c* is added. For each closing pair, the *non-GC* penalty is considered, as well as the dangling base contribution, given by the function ΔG*-Dangle* defined above.

Free energy calculation for multi-domain structures

For multi-domain structures, the dangling base energies are considered. The following formula shows the contribution of dangling bases for k domains, where $(s_{i_1}.s_{j_1})$, ... , $(s_{i_k}.s_{j_k})$ are the closing pairs of each domain. The dangling bases located between the domains are calculated in a similar way with the multiloops. If the domain closest to the *5'* end has a dangling base, than its contribution is added. Similar addition is performed if the domain closest to the *3'* end has a dangling base.

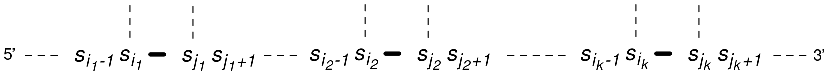


Figure 3.17: Example of a general multidomain structure, with k domains.

$$\Delta G\textit{-D}(S, i_1, j_1, \ldots, i_k, j_k) =$$

$$\sum_{h=1}^{k} \textit{Non-GC-penalty}(s_{i_h}, s_{j_h}) +$$

$$\Delta G\textit{-Dangle-5'}(s_{j_1}, s_{i_1}, s_{i_1-1}) + \sum_{h=1}^{k-1} \Delta G\textit{-Dangle}(S, i_h, j_h, i_{h+1}, j_{h+1}) +$$

$$\Delta G\textit{-Dangle-3'}(s_{j_k}, s_{i_k}, s_{j_k+1});$$

Note that if s_{i_1} is the first base of the sequence, then the dangling energy term $\Delta G\textit{-Dangle-5'}(s_{j_1}, s_{i_1}, s_{i_1-1})$ is replaced by 0. Similarly, if s_{j_k} is the last base of the sequence, the term $\Delta G\textit{-Dangle-3'}(s_{j_k}, s_{i_k}, s_{j_k+1})$ is replaced by 0.

Once the free energy of the elementary structures are calculated, the free energy of a sequence S with a given secondary structure R, $\Delta G(S, R)$, is calculated by the simple addition of the free energies of all elementary structures. The following section gives an example of such calculation.

3.4 Example of free energy calculation

Figure 3.18 contains the same sequence S and secondary structure R as in Figure 3.2. The figure contains the base positions (starting from 1), and 6 different substructures are delimited.

The free energy calculation for each section marked in the figure is calculated as follows:

$$\begin{aligned}
&\Delta G(S,R) = \\
&(1) \quad \Delta G\textit{-D}(S,3,97,102,122) + \\
&(2) \quad \Delta G\textit{-S}(S,3,97) + \Delta G\textit{-S}(S,4,96) + \\
&(3) \quad \Delta G\textit{-M}(S,5,95,7,56,60,92) + \\
&(4) \quad \Delta G\textit{-S}(S,7,56) + \Delta G\textit{-S}(S,8,55) + \Delta G\textit{-S}(S,9,54) + \\
&\qquad \Delta G\textit{-I}(S,10,53,14,48) + \ldots + \Delta G\textit{-H}(S,29,34) + \\
&(5) \quad \Delta G\textit{-S}(S,60,92) + \Delta G\textit{-I}(S,61,91,62,89) + \Delta G\textit{-S}(S,62,89) + \\
&\qquad \Delta G\textit{-I}(S,63,88,66,86) + \Delta G\textit{-S}(S,66,86) + \Delta G\textit{-S}(S,67,85) + \\
&\qquad \Delta G\textit{-I}(S,68,84,71,81) + \Delta G\textit{-S}(S,71,81) + \Delta G\textit{-S}(S,72,80) + \\
&\qquad \Delta G\textit{-H}(S,73,79) + \\
&(6) \quad \Delta G\textit{-S}(S,102,122) + \ldots + \Delta G\textit{-H}(S,109,114) =
\end{aligned}$$

$$\begin{aligned}
&(1) \quad [\textit{Non-GC-Penalty}(G,C) + \textit{Non-GC-Penalty}(C,G) + \\
&\qquad \Delta G\textit{-Dangle-5'}(C,G,C) + \Delta G\textit{-Dangle-3'}(C,G,A) + \\
&\qquad \Delta G\textit{-Dangle-5'}(G,C,C) + \Delta G\textit{-Dangle-3'}(G,C,U)] + \\
&(2) \quad [\Delta G\textit{-Stack}(G,C,G,C)] + [\Delta G\textit{-Stack}(G,C,G,C)] +
\end{aligned}$$

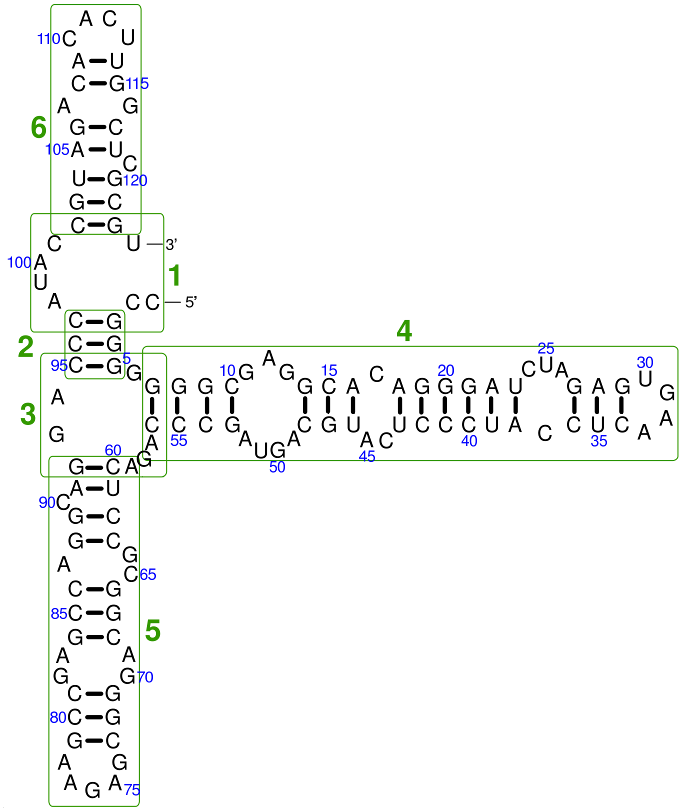


Figure 3.18: Sequence and secondary structure, which will be used to show step by step how the free energy change is calculated.

(3) $[\mathit{Multi\text{-}a} + 3 \cdot \mathit{Multi\text{-}b} + 6 \cdot \mathit{Multi\text{-}c} +$
$\mathit{Non\text{-}GC\text{-}Penalty}(G, C) + \mathit{Non\text{-}GC\text{-}Penalty}(G, C) + \mathit{Non\text{-}GC\text{-}Penalty}(C, G) +$
$\mathit{min}(\Delta G\text{-}\mathit{Dangle\text{-}3'}(G{,}C{,}G){,}\Delta G\text{-}\mathit{Dangle\text{-}5'}(C{,}G{,}G)) +$
$\Delta G\text{-}\mathit{Dangle\text{-}3'}(C{,}G{,}A) + \Delta G\text{-}\mathit{Dangle\text{-}5'}(G{,}C{,}A) +$
$\Delta G\text{-}\mathit{Dangle\text{-}3'}(G{,}C{,}G) + \Delta G\text{-}\mathit{Dangle\text{-}5'}(G{,}C{,}A)] +$

(4) $[\Delta G\text{-}\mathit{Stack}(G, C, G, C)] + [\Delta G\text{-}\mathit{Stack}(G, C, G, C)] + [\Delta G\text{-}\mathit{Stack}(G, C, C, G)] +$
$[\Delta G\text{-}\mathit{Length\text{-}Internal}(7) + \Delta G\text{-}\mathit{Internal\text{-}n}(C, G, G, A) +$
$\Delta G\text{-}\mathit{Internal\text{-}n}(C, G, A, G) +$
$\mathit{min}(\mathit{Asym\text{-}Max},\ |3-4| \times\ \mathit{Asym\text{-}Par}[\mathit{min}(2, \mathit{min}(3, 4)) - 1])] + \ldots +$
$[\Delta G\text{-}\mathit{Length\text{-}Hairpin}(4) + \Delta G\text{-}\mathit{Hairpin\text{-}n}(G, C, U, A) +$
$\Delta G\text{-}\mathit{Hairpin\text{-}4}(G, U, G, A, A, C)] +$

(5) $[\Delta G\text{-}\mathit{Stack}(C, G, U, A)] + [\Delta G\text{-}\mathit{Length\text{-}Bulge}(1) + \Delta G\text{-}\mathit{Stack}(U, A, C, G)] +$
$[\Delta G\text{-}\mathit{Stack}(C, G, C, G)] + [\Delta G\text{-}\mathit{Internal\text{-}3}(C, G, G, C, A, C, G)] +$
$[\Delta G\text{-}\mathit{Stack}(G, C, G, C)] + [\Delta G\text{-}\mathit{Stack}(G, C, C, G)] +$

$[\Delta G\text{-}\mathit{Internal\text{-}4}(C,G,G,C,A,A,G,G)]\ +$
$[\Delta G\text{-}\mathit{Stack}(G,C,G,C)] + [\Delta G\text{-}\mathit{Stack}(G,C,C,G)]\ +$
$[\Delta G\text{-}\mathit{Length\text{-}Hairpin}(5) + \Delta G\text{-}\mathit{Hairpin\text{-}n}(C,G,G,A)]\ +$
(6) $[\Delta G\text{-}\mathit{Stack}(C,G,G,C)] + \ldots + [\Delta G\text{-}\mathit{Length\text{-}Hairpin}(4)\ +$
$\Delta G\text{-}\mathit{Hairpin\text{-}n}(A,U,C,U) + \Delta G\text{-}\mathit{Hairpin\text{-}4}(A,C,A,C,U,U)] =$

(1) $[0 + 0 - 0.30 - 1.70 - 0.30 - 0.60]\ +$
(2) $[-3.30] + [-3.30]\ +$
(3) $[3.4 + 3 \cdot 0.40 + 6 \cdot 0.00 + 0 + 0 + 0 - 1.30 - 0.00 - 1.70 - 0.50 - 1.30 - 0.50]\ +$
(4) $[-3.30] + [-3.30] + [-3.40] + [2.20 - 1.10 - 1.10 + 0.50] + \ldots\ +$
$[5.60 - 1.90 - 1.50]\ +$
(5) $[-2.10] + [3.80 - 2.40] + [-3.30] + [0.60] + [-3.30] + [-3.40]\ +$
$[1.00] + [-3.30] + [-3.40] + [5.60 - 2.20]\ +$
(6) $[-2.40] + \ldots + [5.60 - 0.20 - 0.00] =$

(1) $-2.90\ +$
(2) $-3.30 - 3.30\ +$
(3) $-0.70\ +$
(4) $-3.30 - 3.30 - 3.40 + 0.50 + \ldots + 2.20\ +$
(5) $-2.10 + 1.40 - 3.30 + 0.60 - 3.30 - 3.40 + 1.00 - 3.30 - 3.40 + 3.40\ +$
(6) $-2.40 + \ldots + 5.40 =$

-45.50 kcal/mol

The online version of the *mfold* program [35], which calculates the minimum free energy secondary structure of an RNA sequence, can be used to verify that the free energy values for each elementary structures in this example are consistent with *mfold*.

3.5 Free energy calculation for pairs of molecules

The binding free energy calculation for a pair of RNA or DNA molecules is very similar to the free energy calculation for one single molecule. A penalty for *Intermolecular initiation* is added, i.e. the value *Intermol* from Table 3.1. Figure 3.19 shows a simple example of two short RNA molecules which bind together.

Note that when talking about pairs of RNA or DNA molecules, three situations are possible: (1) both sequences are RNA molecules; (2) both sequences are DNA molecules and (3) one sequence is RNA and one is DNA. Currently, in our algorithms we cover only the first two cases that we use for folding of single molecules, with the same parameters for

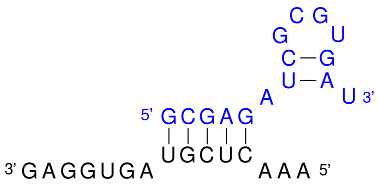


Figure 3.19: Simple example of a pair of RNA molecules and the secondary structure of folding.

RNA and DNA, respectively, Parameters for case 3 exist [52] and they can be incorporated in the same model.

The elementary structures that can be formed are the same as for single molecules, with the difference that stacked loops, hairpin loops, internal loops and multi-loops may have some special cases. Let S_1 and S_2 denote the two input RNA or DNA sequences, and let R denote the secondary structure of their binding. Let $P = p_1 p_2 \ldots p_n$ denote the sequence obtained by concatenating S_1 and S_2, where $n = \mathit{length}(S_1) + \mathit{length}(S_2)$. Let b be the index of the last nucleotide in sequence S_1. b equals $\mathit{length}(S_1)$ (if we start from index 1), and we say that b is the *boundary* between S_1 and S_2.

Figure 3.20 shows the equivalent of Figure 3.12 for a pair of sequences, where the boundary falls between the two pairs.

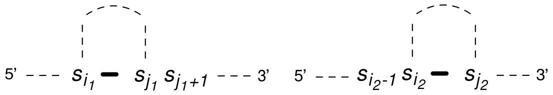

5' - - - s_{i_1} – s_{j_1} s_{j_1+1} - - - 3'
5' - - - s_{i_2-1} s_{i_2} – s_{j_2} - - - 3'

Figure 3.20: The boundary between molecules creates special types of structures.

The dangling energy formula for pairs becomes:

$$\Delta G\text{-}\mathit{Dangle}^{p}(P, i_1, j_1, i_2, j_2, b) =$$
$$\begin{cases} \Delta G\text{-}\mathit{Dangle}\text{-}3'(p_{j_1}, p_{i_1}, p_{j_1+1}) + \\ \quad \Delta G\text{-}\mathit{Dangle}\text{-}5'(p_{j_2}, p_{i_2}, p_{i_2-1}) & , \text{ if } (b > i_1 \text{ and } b+1 < i_2) \\ \mathit{min}(\Delta G\text{-}\mathit{Dangle}\text{-}3'(p_{j_1}, p_{i_1}, p_{j_1+1}), \\ \quad \Delta G\text{-}\mathit{Dangle}\text{-}5'(p_{j_2}, p_{i_2}, p_{i_2-1})) & , \text{ if } (i_1 + 1 = i_2 - 1 \text{ and } b > i_2) \\ \Delta G\text{-}\mathit{Dangle}\text{-}3'(p_{j_1}, p_{i_1}, p_{j_1+1}) & , \text{ if } (b + 1 = i_2) \\ \Delta G\text{-}\mathit{Dangle}\text{-}5'(p_{j_2}, p_{i_2}, p_{i_2-1}) & , \text{ if } (b = i_1) \\ 0 & , \text{ if } (i_1 + 1 > i_2 - 1) \end{cases}$$

Note that if p_{j_1+1} does not exist or is not a free base, the third term will be replaced by 0. The same about the fourth term, if p_{i_2-1} does not exist or is not a free base. $\Delta G\text{-}\mathit{Dangle}^{p}$ will be used for the special types of structures, i.e. when the boundary b breaks the elementary structure and basically transforms it into a multi-domain.

Figure 3.21 shows the special type of stacked loop, the equivalent of Figure 3.13 for pairs, where the boundary breaks the loop.

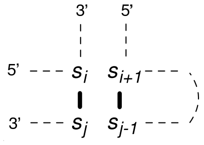


Figure 3.21: Example of a special stacked loop structure in a duplex of molecules.

Thus, the general formula for calculating the stacked loop free energy becomes:

$$\Delta G\text{-}S^p(P,i,j,b) = \begin{cases} \mathit{Intermol} & \text{, if } b = i \text{ or } b+1 = j \\ \Delta G\text{-}S(P,i,j), & \text{otherwise} \end{cases}$$

Similarly, Figure 3.22 shows a special type of hairpin loop, broken by the boundary between the two molecules.

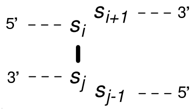


Figure 3.22: Example of a special hairpin loop structure in a duplex of molecules.

If the break between the two molecules is between i and j, then we consider this is a special type of hairpin loop. Thus, the free energy of the hairpin loops for duplexes is calculated as follows:

$$\Delta G\text{-}H^p(P,i,j,b) = \begin{cases} \Delta G\text{-}\mathit{Dangle}^p(P,i,j,i,j,l) + \mathit{Intermol}+ & \\ \quad \mathit{Non\text{-}GC\text{-}Penalty}(p_i,p_j) & , \quad \text{if } i \le b < j \\ \Delta G\text{-}H(P,i,j) & , \quad \text{otherwise} \end{cases}$$

The special type of internal loop is shown in Figure 3.23, the equivalent of Figure 3.15.

The general formula for internal loops for duplexes becomes:

$$\Delta G\text{-}I^p(P,i,j,i',j',b) = \begin{cases} \Delta G\text{-}\mathit{Dangle}^p(P,i,j,i',j',b)+ & \\ \quad \Delta G\text{-}\mathit{Dangle}^p(P,j',i',j,i,b) + \mathit{Intermol}+ & , \quad \text{if } (i \le b < i') \\ \quad \mathit{Non\text{-}GC\text{-}Penalty}(p_i,p_j) + \mathit{Non\text{-}GC\text{-}Penalty}(p_{i'},p_{j'}) & \quad \text{or } (j \le b < j') \\ \Delta G\text{-}I(P,i,j,i',j') & , \quad \text{otherwise} \end{cases}$$

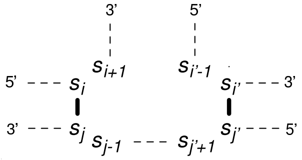


Figure 3.23: Example of a special internal loop structure in a duplex of molecules.

Finally, multi-loops can also be broken by the boundary. Figure 3.24 shows this special type, and the formula to calculate multi-loops for duplexes follows.

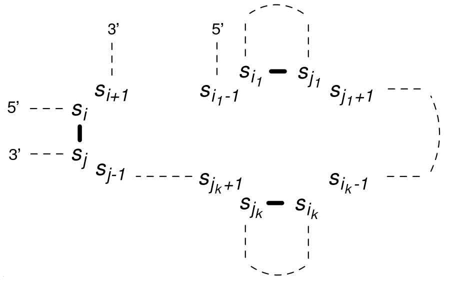


Figure 3.24: Example of a special multi-loop structure in a duplex of molecules.

$$\Delta G\text{-}M^p(P,i,j,i_1,j_1,\ldots,i_k,j_k,b) =$$

$$\begin{cases} \Delta G\text{-}Dangle^p(P,j,i,i_1,j_1,b)+ & ,\quad \text{if } (i \le b < i_1) \\ \quad \sum_{1\le h<k-1} \Delta G\text{-}Dangle^p(P,i_h,j_h,i_{h+1},j_{h+1},b)+ & \quad\text{or } (j_1 \le< i_2)\ldots \\ \quad \Delta G\text{-}Dangle^p(P,i_k,j_k,j,i,b) + Intermol+ & \quad\text{or } (j_{k-1} \le b < i_k) \\ \quad Non\text{-}GC\text{-}Penalty(p_i,p_j)+ & \quad\text{or } (j_k \le b < j) \\ \quad \sum_{1\le h\le k} Non\text{-}GC\text{-}Penalty(p_{i_h},p_{j_h}) & \\ \Delta G\text{-}M(P,i,j,i',j') & ,\quad \text{otherwise} \end{cases}$$

In addition, for self-complementary oligonucleotides, a symmetry correction is added. This is fixed at 0.43 kcal/mol for the standard free energy change at 37°C, and at -1.4 eu for the standard entropy, for both RNA [67] and DNA [37, 45].

3.6 Discussion and limitations of the current model

In section 3.2, we showed there are 36 (stacked loops) + (27+30+28) (size of loop) + 96 (general hairpin loops) + 30 (hairpin loops of size 4) + 96 (general internal loops) + 576 (internal loops of size 2) + 2304 (internal loops of size 3) + 9216 (internal loops of size 4) + 48 (dangling ends) + 16 (miscellaneous energies) = 12,503 parameters, including duplicates. If we exclude duplicates, there are **7589** unique parameters. This seems to be a very big number, and the implementation of any algorithm using the NNTM is complicated. However, many situations are highly approximated, especially the calculation of multi-loop structures. No tertiary interactions are included in the model, no “odd pair” binding, no pseudoknotted structures and no interactions with ions are considered.

The accuracy of secondary structure prediction for biological RNA molecules will be discussed in section 4.4.3. While for some short molecules, the accuracy is up to 100%, for some other, more complicated molecules, the accuracy is as low as 10%. This poor prediction is partly due to the simplified or inexact model, partly to the algorithm, which is simplified for time complexity reasons.

However, researchers have worked for years to create the *NNTM* model and to determine all the current parameters. Whether a better set or model can be determined to better approximate secondary structure formation of natural RNA or single-stranded DNA is an open problem.

Chapter 4

SimFold

This chapter describes the dynamic programming algorithm for secondary structure prediction of an RNA molecule. A thorough understanding of this algorithm is necessary in order to be able to extend it for more complicated tasks, which are the target of this thesis: secondary structure prediction of a pair of molecules, and of a combinatorial set of molecules.

We define the *minimum free energy (MFE) problem of secondary structure prediction of an RNA molecule* as follows: given an RNA sequence S and a thermodynamic model M, find the pseudoknot-free secondary structure R with the minimum free energy change under the model M, in which S can fold. A dynamic programming algorithm for this problem, based on a thermodynamic model M, as described in Chapter 3, has been created by Zuker and Steigler [72], and has been implemented within a number of software packages such as *mfold* [35] and *RNAfold* from the Vienna RNA package [61]. We implemented it as well, under the name of *SimFold*, using the model described in Chapter 3. This implementation, although not new, was necessary for us to fully understand all the different cases of the algorithm and of the model. Moreover, we needed to implement it in such a way, that it can be easily extended to *PairFold* and *CombFold*.

In nature, RNA folding is not simple. Many RNA sequences fold to structures which contain pseudoknots, interactions with ions, other tertiary interactions etc. Moreover, there is no guarantee they will fold into the minimum free energy secondary structure. Approaches to extensions of the MFE problem, for pseudoknot prediction and for prediction of suboptimal structures (i.e. which have a free energy greater than the MFE) exist. In this chapter and throughout this thesis, only the MFE problem will be considered.

A detailed description of the Zuker and Stiegler's [72] algorithm underlying *SimFold* is given in the first section of this chapter. A theoretical analysis of the time and space complexity is discussed in Section 2 and some implementation details are given in Section 3. An empirical evaluation of *SimFold* performance is detailed in Section 4. This shows that *SimFold* is reliable, and correctly implements Zuker and Stiegler's algorithm, being comparable with *mfold* and *RNAfold*.

4.1 Dynamic programming algorithm

The core of the dynamic programming algorithm for RNA secondary structure prediction [72], lies in several recurrence relations, described below. We want to recall the reader that this is not our work, but is the description of Zuker and Stiegler's [72] algorithm[1]

Arrays

The following arrays will be used to calculate the minimum free energy of a sequence and its associated secondary structure:

- $W(j)$ denotes the free energy change of the first j nucleotides of the sequence S: $s_1 s_2 \ldots s_j$. Consequently, $W(n)$ contains the minimum free energy change of the entire sequence S. This array is used to determine multi-domain loops;
- $V(i,j)$ is the minimum free energy of the sequence $s_i \ldots s_j$, assuming that *($s_i.s_j$)* is a base pair;
- $H(i,j)$ is the free energy of the sequence $s_i \ldots s_j$, assuming that *($s_i.s_j$)* closes a hairpin loop;
- $S(i,j)$ is the free energy of the sequence $s_i \ldots s_j$, assuming that *($s_i.s_j$)* closes a stacked loop;
- $VBI(i,j)$ is the free energy of the sequence $s_i \ldots s_j$, assuming that *($s_i.s_j$)* closes an internal loop;
- $VM(i,j)$ is the free energy of the sequence $s_i \ldots s_j$, assuming that *($s_i.s_j$)* closes a multibranched loop;
- $WM(i,j)$ is used to compute the array VM and will be described in detail later in this chapter.

Recurrence relations

The values of the seven aforementioned arrays are computed by recurrence relations, which are interdependent.

The recurrence relation for $W(j)$, as given in [71], is:

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

If the chosen most favourable i is less than j, then we have a multi-domain structure. Otherwise, if i equals j, we have one domain. Recall that for multi-domain structures, the

[1] The notation is partially modified comparing to the original paper, in order to keep this thesis consistent.

dangling energies and the *non-GC-penalties* are added. Thus, a complete version of the recurrence relation for $W(j)$, case $j > 0$ is:

$$W(j) = \min_{1 \leq i < j} \begin{cases} W(j-1) \\ V(i,j) + \textit{Non-GC-penalty}(s_i, s_j) + W(i-1), \\ V(i+1,j) + \textit{Non-GC-penalty}(s_{i+1}, s_j) + \Delta\textit{G-Dangle-3'}(s_j, s_{i+1}, s_i) + W(i-1), \\ V(i,j-1) + \textit{Non-GC-penalty}(s_i, s_{j-1}) + \Delta\textit{G-Dangle-5'}(s_{j-1}, s_i, s_j) + W(i-1), \\ V(i+1,j-1) + \textit{Non-GC-penalty}(s_{i+1}, s_{j-1}) + \Delta\textit{G-Dangle-3'}(s_{j-1}, s_{i+1}, s_i) + \\ \quad \Delta\textit{G-Dangle-5'}(s_{j-1}, s_{i+1}, s_j) + W(i-1) \end{cases}$$

The optimal free energy for $s_i \dots s_j$, $V(i,j)$, is given by the most favourable structure amongst hairpin loop, stacked loop, internal loop and multi-loop. The calculation is performed using the following formula:

$$V(i,j) = \begin{cases} +\infty & , \quad \text{for } i \geq j \\ \min(H(i,j), S(i,j), VBI(i,j), VM(i,j)), & \text{for } i < j \end{cases}$$

To make connection to the equations described in Chapter 3, first note that the following two equations are true:

$$H(i,j) = \Delta\textit{G-H}(S,i,j)$$

$$S(i,j) = \Delta\textit{G-S}(S,i,j) + V(i+1,j-1)$$

The equation for calculating the free energy of an internal loop closed by the external pair *($s_i.s_j$)* must find the optimal internal pair *($s_{i'}.s_{j'}$)*, by searching all possible internal pairs:

$$VBI(i,j) = \min_{i<i'<j'<j} (\Delta\textit{G-I}(S,i,j,i',j') + V(i',j'))$$

The computation of multi-loops requires the computation of another array: WM. $WM(i,j)$ gives the optimal free energy of the sequence $s_i \dots s_j$, assuming that s_i and s_j belong to a multibranched loop (i.e. free bases or a closing pair). WM is calculated as follows:

$$WM(i,i) = \textit{Multi-c}$$

$$WM(i,j) = \min \begin{cases} V(i,j) + \textit{Multi-b} \\ \min_{i \leq h < j}(WM(i,h) + WM(h+1,j)) \end{cases} \text{, for } i < j$$

$WM(i,i)$ corresponds to the situation when s_i is a free base. The first term of $WM(i,j)$ denotes the situation when *(s_i.s_j)* forms an internal base pair, thus defining one of the k branches. The second term appears when s_i and s_j are not paired to each other and the minimum free energy is given by the minimum partition of the sequence into two contiguous subsequences.

Using the WM array, the minimum free energy of a multi-loop is calculated as follows:

$$VM(i,j) = \min_{i<h<j-1}(WM(i+1,h) + WM(h+1,j-1) + \textit{Multi-a})$$

The same as for multi-domains, the equations above for calculating WM and VM are not complete, since they do not capture the dangling energy contributions, nor the *non-GC-penalties*. The following is a complete version of calculating $WM(i,j)$, for $i<j$:

$$WM(i,j) = \min \begin{cases} V(i,j) + \textit{Non-GC-penalty}(s_i,s_j) + \textit{Multi-b}; \\ V(i+1,j) + \textit{Non-GC-penalty}(s_{i+1},s_j) + \Delta \textit{G-Dangle-3'}(s_j,s_{i+1},s_i) + \\ \quad \textit{Multi-b} + \textit{Multi-c}; \\ V(i,j-1) + \textit{Non-GC-penalty}(s_i,s_{j-1}) + \Delta \textit{G-Dangle-5'}(s_{j-1},s_i,s_j) + \\ \quad \textit{Multi-b} + \textit{Multi-c}; \\ V(i+1,j-1) + \textit{Non-GC-penalty}(s_{i+1},s_{j-1}) + \Delta \textit{G-Dangle-3'}(s_{j-1},s_{i+1},s_i) + \\ \quad \Delta \textit{G-Dangle-5'}(s_{j-1},s_{i+1},s_j) + \textit{Multi-b} + 2 \times \textit{Multi-c}; \\ WM(i+1,j) + \textit{Multi-c}; \\ WM(i,j-1) + \textit{Multi-c}; \\ \min_{i \le h < j}(WM(i,h) + WM(h+1,j)). \end{cases}$$

The seven branches correspond to the following situations, respectively:

1. $WM(i,j)$ contains one branch, whose closing pair is *(s_i.s_j)*;

2. One branch, whose closing pair is *(s_{i+1}.s_j)*, and s_i is a free base;

3. One branch, whose closing pair is *(s_i.s_{j-1})*, and s_j is a free base;

4. One branch, whose closing pair is *(s_{i+1}.s_{j-1})*, and s_i, s_j are free bases;

5. $WM(i,j)$ has the same branch(es) as $WM(i+1,j)$ and s_i is a free base;

6. $WM(i,j)$ has the same branch(es) as $WM(i,j-1)$ and s_j is a free base;

7. The best h is chosen, and $WM(i,j)$ has at least two branches: the branch(es) of $WM(i,h)$ and the branch(es) of $WM(h+1,j)$.

The contributions of the dangling bases near the external closing pair of the multi-loop must be captured in the calculation of $VM(i,j)$. At the end, the offset, helix penalty and non-GC-penalty are added:

$$VM(i,j) = \min \begin{cases} WM(i+1, j-1), \\ WM(i+2, j-1) + \Delta G\textit{-Dangle-3'}(s_i, s_j, s_{i+1}) + \textit{Multi-c}, \\ WM(i+1, j-2) + \Delta G\textit{-Dangle-5'}(s_i, s_j, s_{j-1}) + \textit{Multi-c}, \\ WM(i+2, j-2) + \Delta G\textit{-Dangle-3'}(s_i, s_j, s_{i+1}) + \\ \quad \Delta G\textit{-Dangle-5'}(s_i, s_j, s_{j-1}) + 2 \times \textit{Multi-c} \end{cases}$$

$VM(i,j) = VM(i,j) + \textit{Multi-a} + \textit{Multi-b} + \textit{Non-GC-penalty}(s_i, s_j)$.

Note that, since the sequence partitioning into two contiguous subsequences is done in the calculation of WM, there is no need to do it again here. The first branch captures the situation when there is no free base near the *(s_i.s_j)* pair, the second branch - when s_{i+1} is a free base, the third branch - when s_{j-1} is a free base, and the fourth branch - when both of them are free bases.

4.2 Implementation

To be able to backtrack and extract the minimum free energy secondary structure, the arrays W, V and WM will store not only the free energy values, but also some other information. Let W_{ds}, V_{ds} and WM_{ds} denote the data structures associated with each of the arrays W, V and WM. $W_{ds}(j)$ contains the minimum free energy for the sequence $s_0 \ldots s_j$, the number of branches (*num_branches*) of this multi-domain, the closing pair of the last domain (*last_domain*), i.e. with the greatest base indexes and the righmost index of the next domain (*next_domain_i*). $V_{ds}(i,j)$ contains the free energy, the optimal type (HAIRPIN_LOOP, STACKED_LOOP, INTERNAL_LOOP or MULTI_LOOP) for the pair *(s_i.s_j)* and details about where the internal branches for internal loops and multi-loops are located, in case *(s_i.s_j)* closes such an elementary structure. Finally, WM_{ds} contains free energy of the multi-loop fragment and information which help to reconstitute the multi-loop branches completely.

Procedure 3 shows a pseudocode of the dynamic programming algorithm, as implemented in SimFold. The procedure *Compute-WM* calculates the WM_{ds} array, according to the complete equation of WM above. *Compute-V* calculates the optimal structure, using the equation for V, and stores it in the array V_{ds}. After this array is filled in for all i's and j's, the procedure *Compute-W* calculates the multi-domain structures using the complete equation for W, and fills the array W_{ds}. The energy stored in $W(n)$ will be the minimum free energy of folding for sequence S. Using the information stored in the W_{ds} and V_{ds} data

SimFold Procedure

input: RNA or DNA sequence S of length n;
output: minimum free energy ΔG, secondary structure R;

```
procedure SimFold
    for ($j := 1$ to $n$)
        for ($i := j - 1$ down to $1$)
            $WM_{ds}(i, j)$ := Compute-WM(i,j);
        end for;
        for ($i := 1$ to $j - 1$)
            $V_{ds}(i, j)$ := Compute-V(i,j);
        end for;
    end for;
    for ($j := 2$ to $n$)
        $W_{ds}(j)$ := Compute-W(j);
    end for;
    $\Delta G := W_{ds}(n)$.free_energy;
    $i := n$;
    while ($i > 0$ and $W_{ds}(i)$.num_branches $> 0$)
        $(i_d, j_d) := W_{ds}(i)$.last_domain;
        $R$ := SimFold-Backtrack ($i_d$, $j_d$, $V_{ds}$, $W_{ds}$);
        $i := W_{ds}(i)$.next_domain_i;
    end while;
    return ($\Delta G$, $R$);
end procedure SimFold.
```

Procedure 3: Pseudocode for the SimFold algorithm.

structures, now we can backtrack and build the minimum free energy secondary structure R. This is done in the procedure *Backtrack*, which is detailed in Procedure 4.

Procedure *SimFold-Backtrack* is a recursive function which advances one elementary structure in each step, using the information stored in V_{ds} and W_{ds} data structures. It starts from the closing pair of a domain, e.g. *($s_i.s_j$)*, and stops when $j > i$. It checks which type of elementary structure the pair *($s_i.s_j$)* is closing, it saves the partial secondary structure in R, and then, it recursively calls itself on the internal branches of the structure. If it was a hairpin loop, than there is no internal branch, and the function returns.

The *SimFold* program is implemented in C++ and a library that contains the function *SimFold* is provided. The input is the given nucleic acid *sequence* and the MFE secondary structure, in *dot-parenthesis* format, is returned, together with the minimum free energy in kcal/mol. We have also implemented a function which calculates the free energy of a given sequence S, folded in a specific secondary structure R. This function, called *FreeEnergy*(S, R, M), where M is the model under consideration, is important for evaluating the stability of a sequence under some conformation or compare the stability of different conformations. Also, functions to calculate enthalpy, entropy and melting temperature are

SimFold-Backtrack Procedure

input: Indexes i, j, data structures V_{ds}, W_{ds};
output: Partially filled secondary structure R;

```
procedure SimFold-Backtrack
    if ($i > j$)
        return ($\emptyset$);
    else if ($V_{ds}(i,j)$.type = HAIRPIN_LOOP)
        Save-Structure (R);
        return (R);
    else if ($V_{ds}(i,j)$.type = STACKED_LOOP)
        Save-Structure (R);
        return SimFold-Backtrack ($i+1$, $j-1$, $V_{ds}$, $W_{ds}$);
    else if ($V_{ds}(i,j)$.type = INTERNAL_LOOP)
        Save-Structure (R);
        return SimFold-Backtrack ($V_{ds}(i,j).i', V_{ds}(i,j).j', V_{ds}, W_{ds}$);
    else if ($V_{ds}(i,j)$.type = MULTI_LOOP)
        Save-Structure (R);
        for each branch $B$
            return SimFold-Backtrack ($B.i$, $B.j$, $V_{ds}$, $W_{ds}$);
        end for;
    end if;
end procedure SimFold-Backtrack.
```

Procedure 4: Pseudocode for the *SimFold* backtracking algorithm.

provided.

4.3 Time and space complexity - theoretical analysis

Time complexity

The running time to calculate the values in each array can be determined as follows:

- W: $O(n^2)$, because for each j, we minimize over i;
- V: $O(n^2)$, since for each i and j, we minimize over 4 terms;
- H and S: $O(n^2)$;
- VBI: $O(n^4)$, because for each i and j, we have to find the best i' and j';
- WM: $O(n^3)$, since for each i and j, we look for the best h;
- VM: $O(n^2)$: we do a constant number of comparisons for each i and j .

The most expensive task is to calculate the free energy of internal loops. Two solutions have been adopted to reduce time complexity of internal loops from $O(n^4)$ to $O(n^3)$:

1. Hofacker et al. [24] adopted an easy to implement solution: they look at the i'-s and j'-s which are at most c bases distance away from i and j, respectively. Thus, the time complexity becomes proportional to $c^2 \times n^2$, which is considered to be $O(n^3)$. This solution will miss the internal loops having at least one side longer than c, but it seems this case is very unlikely. The number 30 is considered an appropriate value for c.

2. Lyngsø et al. [31] gave a more accurate solution, which would find internal loops of any size in time $O(n^3)$, but is harder to implement, and requires more space. Briefly, instead of using a two-dimensional array $VBI(i, j)$, they use a three-dimensional array $VBI(i, j, l)$, where l denotes the size of the loop, and ranges from 1 to $j-i-1$. Besides the free energy, the entry $VBI(i, j, l)$ will store the best interior pair *($s_{i'}.s_{j'}$)*. They prove that, under some thermodynamic assumptions, if for the exterior pair *($s_i.s_j$)*, the interior pair *($s_{i'}.s_{j'}$)* is better than *($s_{i''}.s_{j''}$)*, than the interior pair *($s_{i'}.s_{j'}$)*, it is also better for the exterior pair *($s_{i-1}.s_{j+1}$)*. Thus, the interior pair *($s_{i'}.s_{j'}$)* stored in $VBI(i, j, l)$, is also the best interior pair for $VBI(i-1, j+1, l+2)$, with only two possible exceptions, which we will not detail further here.

In our implementation of *SimFold*, we used the first method, which has a lower space complexity and is easier to implement.

Space complexity

Note that out of the seven aforementioned arrays, only three of them need to be stored: W, V and WM. H, S, VBI and VM need to be calculated only once. The minimum value, for each i and j, will be stored in the V array. Hence, the space complexity for each of the three arrays W, V and WM is $O(n^2)$, since we store a value (or a data structure of values) for each i and j.

4.4 Performance evaluation

We implemented the algorithm described in this chapter under the name *SimFold*, an implementation of Zuker and Stiegler algorithm, very similar to *mfold* [71] and *RNAfold* [24] from Vienna RNA package.

In this section, first we compare *SimFold* against *RNAfold* and *mfold* and then we run *SimFold* and *RNAfold* on biological RNA structures. We refer to our current best version of *SimFold* as 1.1. For *RNAfold*, we used the latest version, 1.4, which is free software, downloadable from the Vienna RNA package web page [61]. For *mfold*, we also used the latest version, which is 1.3. A web page interface running *mfold* is available [35], but we could not obtain a working version of *mfold* for local runs. Thus, we were able to perform many comparisons between *SimFold* and *RNAfold*, while comparisons between *SimFold* and *mfold* were time-consuming, and hence sparse. *SimFold* and *mfold* can take as input either RNA or DNA, whereas *RNAfold* can only fold RNA. Here, all the tests have been performed on RNA.

We compare the three programs in Subsection 1, then an analysis of *SimFold* and *RNAfold* computation time is performed in Subsection 2. Finally, a study of the *SimFold* accuracy on six sets of biological RNAs is discussed in Subsection 3.

4.4.1 Comparison of secondary structure predictions

For RNA, *RNAfold* uses the set of thermodynamic parameters from SantaLucia Laboratory [44], as opposed to *SimFold* and *mfold*, which use the parameters from Turner Laboratory [57], refined by Mathews et al. [33]. Some of these parameters differ, and we believe this constitutes one main reason for which the results are sometimes different. Another reason is that *RNAfold* does not consider some special types of structures, such as *poly-C hairpins* or *GGG hairpins*. It is unclear whether any of the two sets is better, and a comparison between them is beyond the scope of this thesis. Given that *SimFold* uses the same parameters and exactly the same model as *mfold*[2], the predictions made by *SimFold* are expected to be the same as the predictions made by *mfold* with respect to the minimum free energy secondary structure (*mfold* also predicts suboptimal structures). Still, occasionally, the prediction made by *mfold* has a higher free energy than the prediction made by *SimFold*, because *mfold* does not allow for isolated base pairs. *SimFold* does not incorporate this restriction, allowing everything that the model permits.

To have a good comparison of the three implementations, first we created a set of 100 randomly generated RNA sequences S of length 100 nucleotides. Let ΔG_S and R_S denote the MFE and MFE structure returned by *SimFold*, ΔG_V and R_V denote the MFE and MFE structure returned by *RNAfold* from Vienna package, and ΔG_M and R_M be the MFE and MFE structure returned by *mfold*. The model used by *SimFold* will be referred to as M_S, the model used by *mfold* will be called M_M[3], and the model used by *RNAfold* will be denoted by M_V.

- In 82 cases, the structure prediction of *SimFold* was identical to the structure prediction of *RNAfold*, i.e. $R_S = R_V$;

[2] We believe the model we are using in *SimFold*, which was described in great detail in Chapter 3, is identical to the model used in *mfold*. All our comparisons (counting tens of instances) with the online version of *mfold* confirmed this. However, since we could only access the web version of *mfold*, and since full details about the model used in *mfold* are not publically available, we cannot guarantee that this is true.

[3] We believe $M_S = M_M$.

No	Str. diff.	ΔG_S	ΔG_M	*FreeEnergy(*S, R_S, M_M*)*
1	substantial	-21.80	-21.70	-21.80
2	slight	-24.50	-23.90	-24.50
3	substantial	-32.60	-31.80	-32.60
4	substantial	-17.50	-16.80	-17.50
5	slight	-21.40	-20.60	-21.40
6	substantial	-28.00	-27.90	-28.00
7	substantial	-18.80	-17.90	-18.80

Table 4.1: Differences between *SimFold* and *mfold* on a set of 7 RNA sequences of length 100 nucleotides. The model used in *mfold* on the MFE structure returned by *SimFold* gives the same free energy as *SimFold*, which is lower than the MFE returned by *mfold*. This happens because *mfold* does not allow isolated base pairs in teh minimization algorithm.

- In 7 cases, the structures R_S and R_V were slightly different, but the free energy of R_V, when using model M_S, was equal to the minimum free energy returned by *SimFold*: *FreeEnergy*$(S, R_V, M_S) = \Delta G_S$. This shows that more than one structure gave the same minimum free energy, and *SimFold* chose another one than *RNAfold*. The distances between the two structures range between 2 and 38, where by *distance* between structure R_1 and structure R_2 we mean the number of bases that have a different bonding status in R_1 when compared to R_2. This observation clearly shows that predicting suboptimal structures is very important, since another structure (even with the same MFE) might be much closer to the real one, and the distance between the two predicted structures might be high;

- For all the remaining 11 cases, the free energy *FreeEnergy*(S, R_V, M_S) was greater than ΔG_S, hence *SimFold* considered the structure R_V as being a suboptimal structure. The distance between R_S and R_V in these cases ranged between 2 and 75. At this point, a comparison with the prediction made by *mfold* was performed.
  - In 4 cases, the prediction made by *SimFold* was exactly the same as the prediction made by *mfold*: $R_S = R_M$ and $\Delta G_S = \Delta G_M$. This means that *RNAfold* made the prediction of a suboptimal structure, according to the model M_S or M_M;
  - In the 7 other cases, $\Delta G_M > \Delta G_S$, and the structures R_M and R_S were slightly different in 2 cases and substantially different in 5 cases. Mfold web server [35] contains a program called "Free Energy Determination", which we will refer to as *FreeEnergy(*S^*, R^*, M_M*)*, where S^* and R^* are any RNA sequence and structure. Running this program with the structure R_S predicted by *SimFold*, the free energy obtained equals the MFE returned by *SimFold* and is lower than the MFE returned by *mfold*: *FreeEnergy(*S, R_S, M_M*)* $= \Delta G_S < \Delta G_M$. The equality confirms our supposition that $M_S = M_M$. The inequality happens

Length	% Identical	% Equal FE	% Lower FE
100-500	0.52	0.10 (2-31)	0.38 (2-227)
600-1000	0.18	0.06 (23-319)	0.76 (2-538)
1100-1500	0.12	0.04 (2-3)	0.84 (2-766)
1600-2000	0.00	0.00 (-)	1.00 (2-1021)

Table 4.2: Comparison of structure prediction between *SimFold* and *RNAfold* as a function of sequence length.

because, as we mentioned before, *mfold* does not allow for isolated base pairs. In all these cases, *SimFold* predictions include isolated base pairs. Table 4.1 shows the free energies ΔG_S (column 3), ΔG_M (column 4) and *FreeEnergy*(S, R_S, M_M) (column 5). Table A.1 in Appendix A gives the 7 sequences and the structures predicted by *SimFold*, in dot-parenthesis format.

The second set we created to compare predictions of *SimFold* and *RNAfold* contains 10 sequences of length 200, 10 sequences of length 300, ..., 10 sequences of length 2000. As sequences grow longer, the percentage of identical structures R_S and R_V gets lower. Table 4.4 shows the percentage of the cases where R_S and R_V are identical (second column), the percentage of the cases where *FreeEnergy*(S, R_S, M_S) = *FreeEnergy*(S, R_V, M_S) (third column) and the percentage of the cases where *FreeEnergy*$(S, R_S, M_S) <$ *FreeEnergy*(S, R_V, M_S) (fourth column). Thus, the third column corresponds to alternative structures with the same free energy, and the fourth column corresponds to MFE structure versus suboptimal structure, as predicted by *SimFold* and model M_S. The numbers in the parenthesis represent the minimum and the maximum distance between the two structures.

4.4.2 CPU times

The *RNAfold* program has a very efficient implementation, and is used by other programs, such as RNA Designer from RNAsoft suite [5]. *SimFold* was implemented such that the code could be easily extended to accommodate the necessary differences for *PairFold* and *CombFold*. However, *SimFold* is comparable with *RNAfold* in terms of speed, being only between approximately 10% to 120% slower on sequences shorter than 2000 nucleotides. The computational experiments have been performed on PCs with dual 2GHz Pentium III processors, 512 KB cache and 2GB RAM using Linux 2.4.20.

On the set of length 100 nucleotides, the speed of *SimFold* relative to *RNAfold* is 1.28 on average: it took 0.08 CPU seconds for *RNAfold* and 0.06 CPU seconds for *SimFold* for each sequence. As the length of the sequences increases, the speed of *SimFold* relative to *RNAfold* increases as well. Table 4.3 shows the average CPU time in seconds performed by *SimFold* (second column) and *RNAfold* (third column), as well as the relative time: (Time *SimFold*) / (Time*RNAfold*).

Length	Time *SimFold*	Time *RNAfold*	Relative time
100-500	0.71	0.46	1.48
600-1000	6.14	3.63	1.67
1100-1500	20.71	10.67	1.92
1600-2000	50.07	22.44	2.22

Table 4.3: Comparison of CPU time between *SimFold* and *RNAfold* as a function of sequence length.

Since the *mfold* program predicts suboptimal structures in addition to MFE structures, and since its public availability is restricted, we did not compare it against *SimFold* in terms of speed.

4.4.3 *SimFold* accuracy on biological structures

In order to evaluate the accuracy of *SimFold* prediction, we assembled six sets of RNA sequences with known secondary structures. The first set contains tRNA genes, whose structures where experimentally determined [51] and the other sets contain 5S rRNA, 16S rRNA, 23S rRNA, Group I Intron and Group II Intron sequences determined by comparative sequence analysis [11, 21].

Mathews et al. [33] performed a thorough analysis of *mfold* program (MFE foldings and suboptimal foldings) on biological RNA sequences with known secondary structure. Their sets overlap our sets, still it is unclear whether perfect matches between their sets and ours exist. Konings and Gutell [28] and Fields and Gutell [15] have also analyzed *mfold* program on 16S and 23S rRNA sequences, respectively, which are known to be long and hard to predict.

To measure the accuracy level of *SimFold* prediction, we evaluated four parameters:

1. Q_1 is the level of accuracy (0 for completely wrong prediction and 1 for perfect prediction) calculated by dividing the number of correctly predicted base pairs to the number of base pairs in the real structure. We think the same parameter is used in Mathews et al. [33] measurements of accuracy. Note that with Q_1, the free bases are not taken into consideration at all. Thus, a perfect prediction of the correct base pairs, which also predicts base pairs where the bases are actually free bases, would yield a value of 1;

2. Q_2 also measures the level of accuracy, but by taking the base pairs as well as the free bases into consideration. Q_2 is calculated as follows: first, the *distance* d between the predicted structure R_S and the real structure R_R is calculated, by counting the number of bases whose pairing status is different between R_S and R_R. A similar distance function is used in RNA Designer from the RNAsoft suite of programs [5]. Then $Q_2 = 1 - d/l$, where l is the length of the given sequence S;

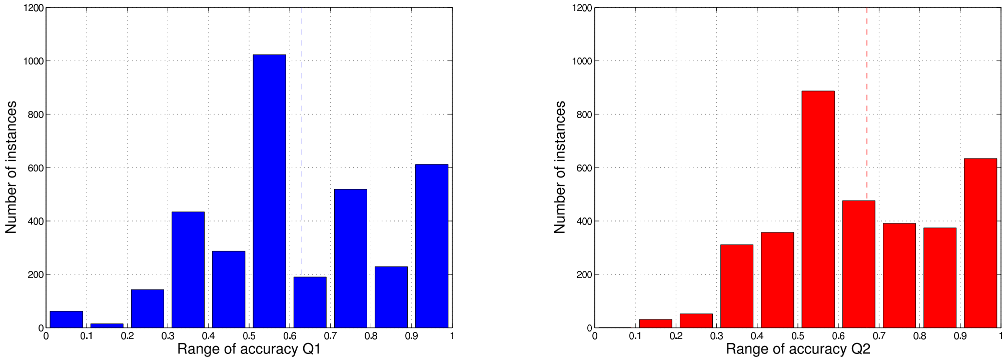


Figure 4.1: Evaluation of *SimFold* on tRNA genes: Distribution of the predicted structures after the level of accuracy, on intervals of 0.1.

3. The percentage of "odd pairs" (or non-canonical pairs) in the real structure R_R is calculated. This parameter is important because, as our model does not consider odd pairs, we do not expect to have a perfect prediction for structures with odd pairs;

4. The percentage of pseudoknots in R_R is even more important, since Zuker and Stiegler algorithm, which is the basis of *SimFold*, can only predict pseudoknot-free secondary structures. The pseudoknot percentage was determined by dividing the number of pairs which are within a pseudoknot by the total number of pairs in R_R. A pair $(i.j)$ is considered to be within a pseudoknot if a pair $(i'.j')$ exists such that $i < i' < j < j'$ or $i' < i < j' < j$.

The tRNA genes set contains a very small fraction of odd pairs and no pseudoknots, and the sequences are shorter than 100 nucleotides. However, other structures that are not considered by our model exist: 7 sequences have hairpins of size 0: `()`, 16 sequences have hairpins of size 1: `(.)` and 38 sequences have hairpins of size 2: `(..)`; also, tertiary interactions known to exist in tRNA are not reflected in our analysis. We performed a thorough analysis on 3514 sequences and calculated Q_1, Q_2, the free energy returned by *SimFold* and the free energy of the real structure R_R calculated with our model: $\mathit{FreeEnergy}(S, R_R, M_S)$.

While some structures have been predicted with more than 0.90 accuracy, others had only 0.20-0.50 accuracy. Figure 4.1 shows the distribution of the predictions on intervals of level of accuracy. The left image shows the distribution when measuring the accuracy with Q_1, and the right image corresponds to Q_2. The vertical dashed line in each image shows the average accuracy of the corresponding parameter.

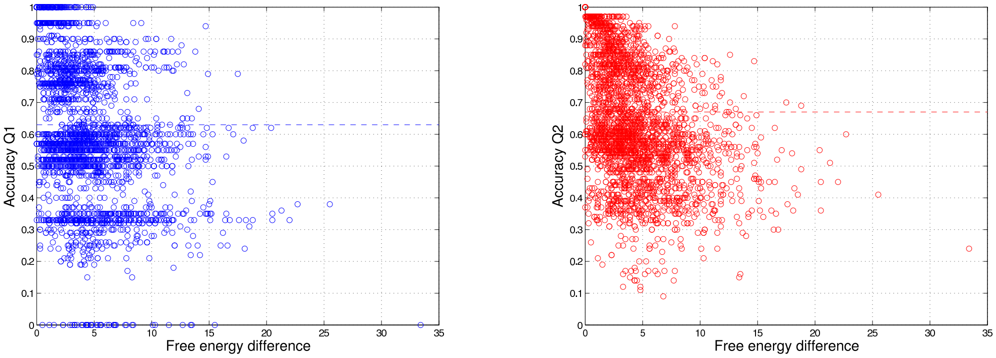


Figure 4.2: Evaluation of *SimFold* on tRNA genes: Correlation between the level of accuracy and the difference between the predicted minimum free energy and the free energy of the real structure, considering our model.

For all the predictions whose accuracy was lower than 1.00, the free energy calculated for the real structure ΔG_R^p was higher than the predicted minimum free energy ΔG_S. Obviously, this is another convincing hint that the calculation of suboptimal structures is important. It is interesting to look at the difference $\Delta G_R^p - \Delta G_S$ and at the level of accuracy. Figure 4.2 shows the correlation between these two values for Q_1 (left) and Q_2 (right). The general trend is that the greater the difference in free energy, the lower the level of accuracy. Still, in many cases, e.g. for accuracy 0.35, the free energy difference is close to 0. Note that for the Q_1 parameter, there are quite a few cases where the level of accuracy is 0.00, and the free energy difference ranges in $(0, 16]$.

Table 4.4 shows the summary of the results predicted by *SimFold* and *RNAfold* on the six data sets. Column 1 shows the data set type and column 2 gives the number of instances for each type that was considered in this study. Column 3 shows the range length for each type and columns 4 and 5 give the average percentage of odd pairs and pseudoknots. Column 6 presents the average accuracy of *SimFold*, measured with Q_1 and Q_2, respectively, and the last column shows the average performance of *RNAfold*.

The table shows that the prediction of *SimFold* is better on short sequences, without or with a very small fraction of odd pairs and pseudoknots. As the percentage of odd pairs or pseudoknots increases, the accuracy goes down. Although for some sequences, the prediction of *SimFold* differs from the prediction of *RNAfold*, note that on the average, their prediction is very close. For the tRNA genes set, *RNAfold* gives better accuracy than *SimFold* in 118 cases, and *SimFold* gives better accuracy in 150 cases. Table A.2 in the Appendix A shows a

RNA type	No. inst.	Len	% Odd	% Pk	*SimFold* Q_1 (Q_2)	*RNAfold* Q_1 (Q_2)
tRNA genes	3514	51-93	0.00	0.00	0.63 (0.67)	0.63 (0.66)
Group I Intron	8	394-1031	0.04	0.10	0.48 (0.52)	0.48 (0.52)
Group II Intron	2	905-2520	0.00	0.45	0.40 (0.43)	0.50 (0.47)
5S rRNA	9	117-124	0.11	0.00	0.57 (0.62)	0.50 (0.56)
16S rRNA	91	697-2147	0.06	0.06	0.42 (0.50)	0.42 (0.49)
23S rRNA	59	953-4381	0.06	0.16	0.41 (0.50)	0.40 (0.49)

Table 4.4: Summary of *SimFold* accuracy on different types of RNA sequences.

set of 30 examples of tRNA gene sequences, together with the parameters discussed above. Complete tables of the measurement of accuracy for the remaining five sets of introns and rRNAs are given in Appendix A.

Comparing Table 4.4 with Table 1 in [33], where Mathews et al. report a thorough analysis of *mfold*, we notice the accuracy measured with Q_1 is from 8% for 16S rRNA to 20% for tRNA lower than reported there. We consider that the reasons for these differences include:

- In the study performed by Mathews et al. [33], they use the known real structures to refine the thermodynamic parameters used. Then, the accuracy evaluation is performed on the same set. It is not clear whether further tests on a different set would give the same level of accuracy;

- The sets that we used seem to overlap with the sets that Mathews et al. used, but not to coincide. Also, we are not sure whether the method of measuring the level of accuracy is identical;

- Finally, our supposition that the model we use is identical with the model *mfold* uses, might not be completely true. However, in the future, *SimFold* and its extensions, *PairFold* and *CombFold*, can incorporate more parameters.

Although it is hard to make a very correct estimate, we believe that *SimFold* is as good as *RNAfold* and *mfold* regarding the prediction of minimum free energy and structure. The purpose of implementing *SimFold* was not to create a better prediction program, but to be able to extend Zuker and Stiegler's [72] basic algorithm of secondary structure prediction towards prediction of pairs and combinatorial sets of molecules, which will be presented in great detail in the next two chapters.

Chapter 5

PairFold

The algorithm for predicting the secondary structure of a pair of RNA or DNA molecules, which we call *PairFold*, is described in this chapter. We define the *minimum free energy (MFE) problem of secondary structure prediction of a pair of RNA molecules* as follows: given two RNA sequences S_1 and S_2 and a thermodynamic model M, find the pseudoknot-free secondary structure R with the smallest free energy change under the model M, in which S_1 and S_2 can fold. Note that this problem includes interactions between the two molecules, as well as interactions within each molecule.

The same as in case of secondary structure prediction of single molecules, for pair of molecules, the problems of predicting pseudoknotted secondary structures, other interactions or suboptimal structures (i.e. which have a free energy greater than the MFE) are important and challenging. In this chapter, only the MFE problem will be considered.

PairFold algorithm is a simple extension of Zuker and Steigler [72] algorithm, described in Chapter 4. As discussed in Section 3.5, consider that we concatenate the two input sequences, S_1 and S_2, into sequence P and that we denote the linkage location with b (b equals the last position in S_1). With some modifications, which will be described in this chapter, we can apply the *SimFold* algorithm on S_1S_2. Briefly, the elementary structures are the same as for a single molecule, with the exception that they can be "broken" by the molecular link b between them. In these situations, the free energy is calculated in a way similar to the calculation of multi-domain structures, and an *Intermolecular initiation penalty*, i.e. *Intermol* from Table 3.1 is added.

5.1 Dynamic programming algorithm

The arrays to calculate minimum free energies for different secondary structures are called $W^p(j)$, $V^p(i,j)$, $H^p(i,j)$, $S^p(i,j)$, $VBI^p(i,j)$, $VM^p(i,j)$, $VM'(i,j)$, $WM^p(i,j)$ and $WM'(i,j)$. They correspond to the arrays with the same names, but without the subscript ^{p} from Section 4.1. Note the new arrays $VM'(i,j)$ and $WM'(i,j)$. The same as for *SimFold*, the computation of multi-loops is done by the computation of the partial multi-loop structures $WM^p(i,j)$. If b is inside a multi-loop structure (i.e. b the intermolecular linkage is between two branches, yielding a special multi-loop), then the multi-loop will be calculated as a multi-domain structure. Otherwise, it will be a regular multi-loop. However, at the moment of the calculation of $WM^p(i,j)$, since we are calculating partial structures, we do not know whether the remaining part of the multi-loop contains the b. Thus, the calculation of the two arrays: $WM^p(i,j)$ and $WM'(i,j)$ for the regular and special partial multi-loops, respectively, is needed. Consequently, there will be two arrays for calculating complete multi-loops too, one for regular multi-loops, one for special multi-loops.

The complete recurrence relation for $W^p(j)$ when $j > 0$ follows. This is an extension of the equation for $W(j)$ from Section 4.1. Depending on the position of b, dangling ends are added or not. This will be shown in the terms T_1, T_2 and T_3, described below:

$$W^p(j) = \min_{1 \leq i \leq j}$$
$$\begin{cases} V^p(i,j) + \textit{Non-GC-penalty}(p_i, p_j) + W^p(i-1), \\ V^p(i+1,j) + \textit{Non-GC-penalty}(p_{i+1}, p_j) + W^p(i-1) + T_1, \\ V^p(i,j-1) + \textit{Non-GC-penalty}(p_i, p_{j-1}) + W^p(i-1) + T_2, \\ V^p(i+1,j-1) + \textit{Non-GC-penalty}(p_{i+1}, p_{j-1}) + W^p(i-1) + T_3 \end{cases}$$

$$T_1 = \begin{cases} 0 & , \text{ if } b = i \\ \Delta G\textit{-Dangle-3'}(p_j, p_{i+1}, p_i), & \text{otherwise} \end{cases}$$

$$T_2 = \begin{cases} 0 & , \text{ if } b = j-1 \\ \Delta G\textit{-Dangle-5'}(p_{j-1}, p_i, p_j), & \text{otherwise} \end{cases}$$

$$T_3 = \begin{cases} \Delta G\textit{-Dangle-5'}(p_{j-1}, p_{i+1}, p_j) & , \text{ if } b = i \\ \Delta G\textit{-Dangle-3'}(p_{j-1}, p_{i+1}, p_i) & , \text{ if } b = j-1 \\ \Delta G\textit{-Dangle-3'}(p_{j-1}, p_{i+1}, p_i) + \Delta G\textit{-Dangle-5'}(p_{j-1}, p_{i+1}, p_j), & \text{otherwise} \end{cases}$$

The optimal free energy for $p_i \dots p_j$, $V^p(i,j)$, is given by the most favourable structure amongst the regular or special types of hairpin loop, stacked loop, internal loop and multi-loop. The calculation is performed using the following formula:

$$V^p(i,j) = \begin{cases} +\infty & , \text{ for } i \geq j \\ \min(H^p(i,j), S^p(i,j), VBI^p(i,j), VM^p(i,j)), & \text{for } i < j \end{cases}$$

Integrating the model for duplexes, described in Section 3.5, each individual type of elementary structure will be calculated as follows:

$$H^p(i,j) = \Delta G\textit{-}H^p(P,i,j,b)$$

$$S^p(i,j) = \Delta G\textit{-}S^p(P,i,j,b) + V^p(i+1,j-1)$$

$$VBI^p(i,j) = \min_{i<i'<j'<j}(\Delta G\text{-}I^p(P,i,j,i',j',b) + V^p(i',j'))$$

$WM'(i,j)$ is calculated similarly to $WM(i,j)$, for all situations when the boundary is not inside the multi-loop:

$$WM'(i,i) = \textit{Multi-c}$$

$$WM'(i,j) = \min \begin{cases} V^p(i,j) + \textit{Non-GC-penalty}(p_i,p_j) + \textit{Multi-b} & \\ V^p(i+1,j) + \textit{Non-GC-penalty}(p_{i+1},p_j) + & \\ \quad \Delta G\text{-}\textit{Dangle-3'}(p_j,p_{i+1},p_i) + \textit{Multi-b} + \textit{Multi-c}, & \text{if } b \neq i \\ V^p(i,j-1) + \textit{Non-GC-penalty}(p_i,p_{j-1}) + & \\ \quad \Delta G\text{-}\textit{Dangle-5'}(p_{j-1},p_i,p_j) + \textit{Multi-b} + \textit{Multi-c}, & \text{if } b \neq j-1 \\ V^p(i+1,j-1) + \textit{Non-GC-penalty}(p_{i+1},p_{j-1}) + & \\ \quad \Delta G\text{-}\textit{Dangle-3'}(p_{j-1},p_{i+1},p_i) + & \\ \quad \Delta G\text{-}\textit{Dangle-5'}(p_{j-1,i+1},p_j) + & \\ \quad \textit{Multi-b} + 2 \times \textit{Multi-c} & , \text{ if } b \neq i \text{ and } b \neq j-1 \\ WM'(i+1,j) + \textit{Multi-c} & , \text{ if } b \neq i \\ WM'(i,j-1) + \textit{Multi-c} & , \text{ if } b \neq j-1 \\ \min_{i \leq h < j}(WM'(i,h) + WM'(h+1,j)) & , \text{ if } b \text{ is not within the multi-loop} \end{cases}$$

The condition of the last case, "if b is not within the multi-loop", means that b is not separating any two branches of the multi-loop, hence this is a regular multi-loop. If the branches are closed by $(i_1,j_1),(i_2,j_2)\ldots(i_k,j_k)$, then b is within the multi-loop if and only if $(i \leq b < i_1)$ or $j_1 \leq b < i_2$ or $\ldots j_{k-1} \leq b < i_k$ or $j_k \leq b < j$. Note that we are not interested whether b is situated somewhere on a branch of the multi-loop. This situation will be handled by that specific elementary structure.

If the boundary b is inside the multi-loop structure, then the array $WM^p(i,j)$ will contain the information and free energy needed. The following formula shows the same seven branches, but the multi-loop penalties *Multi-a*, *Multi-b* and *Multi-c* are not needed any more, being replaced by the intermolecular initiation. The terms T_1 - T_7 represent additional terms, which depend on the position of b and are detailed below.

$$WM^p(i,j) = \min$$

$$\begin{cases} V^p(i,j) + \textit{Non-GC-penalty}(p_i, p_j), \\ V^p(i+1,j) + \textit{Non-GC-penalty}(p_{i+1}, p_j) + T_1, \\ V^p(i,j-1) + \textit{Non-GC-penalty}(p_i, p_{j-1}) + T_2, \\ V^p(i+1,j-1) + \textit{Non-GC-penalty}(p_{i+1}, p_{j-1}) + T_3 + T_4, \\ WM^p(i+1,j) + T_5, \\ WM^p(i,j-1) + T_6, \\ \min_{i \le h < j}(WM^p(i,h) + WM^p(h+1,j) + T_7) \end{cases}$$

$$T_1 = \begin{cases} \textit{Intermol} & , \text{ if } b = i \\ \Delta G\textit{-Dangle-3'}(p_j, p_{i+1}, p_i), & \text{if } b \neq i \end{cases}$$

$$T_2 = \begin{cases} \textit{Intermol} & , \text{ if } b = j-1 \\ \Delta G\textit{-Dangle-5'}(p_{j-1}, p_i, p_j), & \text{if } b \neq j-1 \end{cases}$$

$$T_3 = \begin{cases} \textit{Intermol} & , \text{ if } b = i \\ \Delta G\textit{-Dangle-3'}(p_{j-1}, p_{i+1}, p_i), & \text{if } b \neq i \end{cases}$$

$$T_4 = \begin{cases} \textit{Intermol} & , \text{ if } b = j-1 \\ \Delta G\textit{-Dangle-5'}(p_{j-1}, p_{i+1}, p_j), & \text{if } b \neq j-1 \end{cases}$$

$$T_5 = \begin{cases} \textit{Intermol}, & \text{if } i+1 < b \le i^* \text{ or } j^* < b \le j \\ 0 & , \text{ otherwise} \end{cases}$$

where i^* is the downstream base of the first branch of $WM^p(i+1,j)$, and j^* is the upstream base of the last branch of $WM^p(i+1,j)$.

$$T_6 = \begin{cases} \textit{Intermol}, & \text{if } i < b \le i^* \text{ or } j^* < b \le j-1 \\ 0 & , \text{ otherwise} \end{cases}$$

where i^* is the downstream base of the first branch of $WM^p(i+1,j)$, and j^* is the upstream base of the last branch of $WM^p(i+1,j)$.

$$T_7 = \begin{cases} \textit{Intermol}, & \text{if } b = h \\ 0 & , \text{ otherwise} \end{cases}$$

WM' will help creating the array VM', while WM^p will create the array VM^p. For each i and j, the minimum of $VM'(i,j)$ and $VM^p(i,j)$ will become $VM^p(i,j)$. The calculation of VM' and VM^p follows:

$$
VM'(i,j) = \min
\begin{cases}
WM'(i+1,j-1) & , \text{ if } b \notin \{i, j-1\} \\
WM'(i+2,j-1) + \Delta G\text{-}\mathit{Dangle\text{-}3'}(p_i,p_j,p_{i+1}) + \\
\quad \mathit{Multi\text{-}c} & , \text{ if } b \notin \{i, i+1, j-1\} \\
WM'(i+1,j-2) + \Delta G\text{-}\mathit{Dangle\text{-}5'}(p_i,p_j,p_{j-1}) + \\
\quad \mathit{Multi\text{-}c} & , \text{ if } b \notin \{i, j-1, j-2\} \\
WM'(i+2,j-2) + \Delta G\text{-}\mathit{Dangle\text{-}3'}(p_i,p_j,p_{i+1}) + \\
\quad \Delta G\text{-}\mathit{Dangle\text{-}5'}(p_i,p_j,p_{j-1}) + 2 \times \mathit{Multi\text{-}c} & , \text{ if } b \notin \{i, i+1, j-1, j-2\}
\end{cases}
$$

The multi-loop specific penalties are finally added: $VM'(i,j) = VM'(i,j) + \mathit{Multi\text{-}a} + \mathit{Multi\text{-}b} + \mathit{Non\text{-}GC\text{-}penalty}(p_i,p_j)$.

$$
VM^p(i,j) = \min
\begin{cases}
WM^p(i+1,j-1) + \mathit{Non\text{-}GC\text{-}Penalty}(p_i,p_j) + T_1, \\
WM^p(i+2,j-1) + \mathit{Non\text{-}GC\text{-}Penalty}(p_i,p_j) + T_2, \\
WM^p(i+1,j-2) + \mathit{Non\text{-}GC\text{-}Penalty}(p_i,p_j) + T_3, \\
WM^p(i+2,j-2) + \mathit{Non\text{-}GC\text{-}Penalty}(p_i,p_j) + T_4
\end{cases}
$$

$$
T_1 = \begin{cases}
\mathit{Intermol}, & \text{if } b \in \{i, j-1\} \\
0 & , \text{ otherwise}
\end{cases}
$$

$$
T_2 = \begin{cases}
\mathit{Intermol} & , \text{ if } b = i \\
\Delta G\text{-}\mathit{Dangle\text{-}3'}(p_i,p_j,p_{i+1}) + \mathit{Intermol}, & \text{if } b \in \{i+1, j-1\} \\
\Delta G\text{-}\mathit{Dangle\text{-}3'}(p_i,p_j,p_{i+1}) & , \text{ otherwise}
\end{cases}
$$

$$
T_3 = \begin{cases}
\mathit{Intermol} & , \text{ if } b = j-1 \\
\Delta G\text{-}\mathit{Dangle\text{-}5'}(p_i,p_j,p_{j-1}) + \mathit{Intermol}, & \text{if } b \in \{i, j-2\} \\
\Delta G\text{-}\mathit{Dangle\text{-}5'}(p_i,p_j,p_{j-1}) & , \text{ otherwise}
\end{cases}
$$

$$
T_4 = \begin{cases}
\Delta G\text{-}\mathit{Dangle\text{-}5'}(p_i,p_j,p_{j-1}) + \mathit{Intermol} & , \text{ if } b = i \\
\Delta G\text{-}\mathit{Dangle\text{-}3'}(p_i,p_j,p_{i+1}) + \mathit{Intermol} & , \text{ if } b = j-1 \\
\Delta G\text{-}\mathit{Dangle\text{-}3'}(p_i,p_j,p_{i+1}) + \Delta G\text{-}\mathit{Dangle\text{-}5'}(p_i,p_j,p_{j-1}) +, \\
\quad \mathit{Intermol} & , \text{ if } b \in \{i+1, j-2\} \\
\Delta G\text{-}\mathit{Dangle\text{-}3'}(p_i,p_j,p_{i+1}) + \Delta G\text{-}\mathit{Dangle\text{-}5'}(p_i,p_j,p_{j-1}) & , \text{ otherwise}
\end{cases}
$$

Note that no other additions to VM^p are necessary at this point.

PairFold Procedure

input: RNA or DNA sequences S_1 and S_2 of length n_1 and n_2;
output: minimum free energy ΔG, secondary structure R;

```
procedure PairFold
    $S = S_1 S_2$;
    $n = n_1 + n_2$;
    for ($j = 1$ to $n$)
        for ($i = j - 1$ down to 1)
            $WM'_{ds}$(i,j) = Compute-WM'(i,j);
            $WM^p_{ds}$(i,j) = Compute-WM$^p$(i,j);
        end for;
        for ($i := 1$ to $j - 1$)
            $V^p_{ds}(i, j)$ := Compute-V$^p$(i,j);
        end for;
    end for;
    for ($j := 2$ to $n$)
        $W^p_{ds}(j)$ := Compute-W$^p$(j);
    end for;
    $\Delta G := W^p_{ds}(n)$.free_energy;
    $i := n$;
    while ($i > 0$ and $W^p_{ds}(i)$.num_branches $> 0$)
        $(i_d, j_d) := W^p_{ds}(i)$.last_domain;
        $R$ := PairFold-Backtrack ($i_d$, $j_d$, $V^p_{ds}$, $W^p_{ds}$);
        $i := W^p_{ds}(i)$.next_domain_i;
    end while;
    return ($\Delta G$, $R$);
end procedure PairFold.
```

Procedure 5: Pseudocode for the PairFold algorithm.

5.2 Implementation

The implementation of *PairFold* is very similar to *SimFold* implementation. The main differences are: (1) the calculation of two arrays for WM^p and WM', (2) the checks for b in each particular situation. Procedure 5 shows a pseudocode of the implementation, which highlights point (1). Point (2) was described in detail in the previous section.

The same as in *SimFold* implementation, we implemented a function that calculates the free energy of a duplex, when the structure is known: *FreeEnergy(S_1, S_2, R)*. In the same way, once the MFE secondary structure is found, we can compute the enthalpy of the folded molecule, using the enthalpy thermodynamic parameters. Then, the calculation of the entropy and the melting temperature, when the reactants concentrations are given, is straightforward using the equations explained in Section 3.1.

The MFE secondary structure, MFE, enthalpy and entropy changes, and the melting temperature, for the sequences (S_1, S_2), (S_1, S_1) and (S_2, S_2) are provided in the online

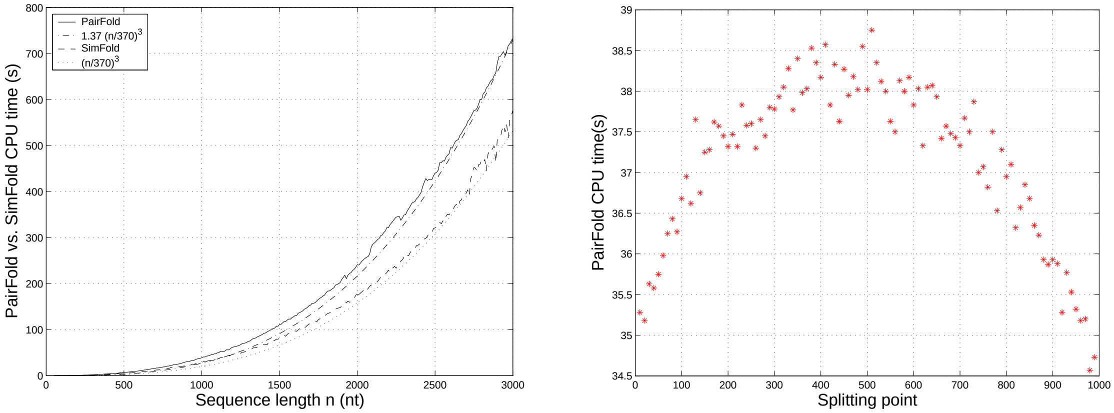


Figure 5.1: Comparison between *PairFold* and *SimFold* speed (left); *PairFold* speed on a sequence with different splitting points.

version of *PairFold* [5, 41]. The secondary structure of (S_2, S_1) does not have to be calculated separately, because the answer will be the same as for (S_1, S_2). Section 5.4 gives an informal proof that, given the thermodynamic model we are using, *PairFold* is symmetric.

5.3 Time and space complexity

Being a very simple extension of the *SimFold* algorithm, the time and space complexity are of the same theoretical order as for *SimFold*: $O(n^3)$ for time and $O(n^2)$ for space, where n is the sum of the two input sequences lengths.

In practice, the time will be longer, due to the additional checking for the location of b and due to supplementary calculations for multi-loops. The space required to store the necessary arrays will be larger too, since instead of storing three arrays, as in the case of *SimFold*: W, V and WM, we now store four arrays: W^p, V^p, WM^p and WM'.

The left graph in Figure 5.1 shows the speed of *PairFold* compared to the speed of *SimFold*. 296 sequences have been randomly generated, of length 50, 60, 70, ... 3000 nucleotides and *SimFold* has been run on them. Then, each sequence has been split in half (yielding $b = n/2$) and *PairFold* has been run on each such pair. The dashed line shows the CPU time in seconds, that *SimFold* needed to predict the MFE secondary structures. The solid line shows the speed of *PairFold*. The dotted and dash-dot lines have been manually fit to match the *SimFold* and *PairFold* lines, respectively. The dotted line is the function $(n/370)^3$, where n is the sequence length (or the sum of the two sequences lengths, in case of *PairFold*), and the dash-dot line is the function $1.3(n/370)^3$. These manual fits show that both *SimFold* and *PairFold* run in time proportional to $O(n^3)$, and *PairFold* is about 1.3 times slower than *SimFold*.

The right graph in Figure 5.1 shows the speed of *PairFold* on a sequence of length

1000 nucleotides, which has been split at different points: $b = \{10, 20, \ldots, 990\}$. The plot shows that *PairFold* is slightly slower when the two sequences are roughly equal to each other, and slightly faster when one sequence is short and the other is long. Note that the difference is not big, and this can be explained by the fact that the two arrays that help multi-loop calculations: WM^p and WM', are fully computed, no matter where the boundary b is.

5.4 PairFold symmetry

In this section we show that the minimum free energy ΔG_{12} for the pair of molecules (S_1, S_2) under the symmetric model M equals the minimum free energy ΔG_{21} of the pair (S_2, S_1) under the same model.

First, we show that given S_1, S_2 and a secondary structure R_{12} with free energy ΔG_{12}, in which the pair (S_1, S_s) could fold, the secondary structure R_{21} which contains the same base pairing as R_{12}, but for the pair (S_2, S_1), has the same free energy: $\Delta G_{21} = \Delta G_{12}$.

In Section 3.3 we showed that $\Delta G_{12} = \sum_e \Delta G_e$, where e is an elementary structure, and the sum goes over all elementary structures in the structure R_{12}. Then, R_{21} will contain the "mirrored" elementary structures. For example, a multi-domain structure in R_{12} will be a special multi-loop structure in R_{21}, but their free energies will be the same. Given that the Nearest Neighbour Thermodynamic Model is symmetric, all the "mirrored" elementary structures will have the same free energy as the original elementary structures. Thus, $\Delta G_{21} = \sum_e \Delta G_e$, hence $\Delta G_{12} = \Delta G_{21}$.

Now suppose that ΔG_{12} and ΔG_{21} differ and $\Delta G_{12} < \Delta G_{21}$. If the MFE structures R_{12} and R_{21} are "mirrored", then from the proof above, $\Delta G_{12} = \Delta G_{21}$. Suppose R_{12} and R_{21} are different. Let R_{12}^m denote the "mirrored" structure of R_{12}, i.e. the structure containing the same base pairs as R_{12}, but for (S_2, S_1), Then, from the proof above, $\Delta G_{12} = \Delta G_{12}^m$. But $\Delta G_{12} < \Delta G_{21}$, so $\Delta G_{12}^m < \Delta G_{21}$. This means that there exists a secondary structure, namely R_{12}^m, for (S_2, S_1), whose corresponding free energy ΔG_{12}^m is lower than the minimum free energy ΔG_{21}. This is a contradiction, which proves that the hypothesis we made is false.

We showed that if R_{12} and R_{21} are identical, then $\Delta G_{12} = \Delta G_{21}$. The reverse implication is not always true, because it is possible for two different secondary structures to have the same free energy.

When several MFE secondary structures exist, our algorithms will select the first one (hence always the same one), out of the set of possible secondary structures. For this reason, we think it might be possible for *PairFold* to select different secondary structures for (S_1, S_1) and (S_2, S_1) if they exist. Further analysis is needed to support this supposition.

5.5 Applications and performance evaluation

This section presents several applications of *PairFold* and gives a thorough analysis of its performance on different biological sets. First, we show its performance on short DNA and RNA duplexes, giving also other parameters, such as enthalpy, entropy and melting temperature. Then, we show a very important and promising application of *PairFold*, on ribozyme-target hybridization and on a ribozyme combinatorial library. *PairFold* can also be used for primer/probe design and testing. Finally, we show that it can be useful for including thermodynamic models into DNA word design.

5.5.1 Prediction of DNA or RNA duplex formation

Free energy, enthalpy, entropy and melting temperature of DNA or RNA duplexes which are complementary or have few mismatches have been experimentally determined. The information obtained was mainly used for finding thermodynamic parameters for stacking energies and simple mismatches, and for testing of prediction programs. Knowing these parameters is useful for any secondary structure prediction and for primer design [37].

Peyret et al. [37] have used a set of 51 DNA duplexes which contain A·A, C·C, G·G and T·T mismatches, to find thermodynamic parameters for these mismatches, by linear regression. Then, they compared the predicted free energy, enthalpy, entropy and melting temperature with the ones determined experimentally. In a similar way, Xia et al. [67] have used a set of 119 perfectly complementary RNA duplexes, to determine or test stacking parameters. In this subsection we run *PairFold* on these duplexes and compare our predictions with the experimental measurements and predictions reported in [37] and [67].

NA	Prediction	ΔH° (avg) (kcal/mol)	ΔS° (avg) (eu)	ΔG° (avg) (kcal/mol)	T_m (avg) (°C)
DNA	*PairFold* prediction	-59.06	-168.62	-6.76	41.76
	Experimental values [37]	-59.79	-170.94	-6.78	41.75
	Peyret et al. prediction [37]	-59.57	-170.32	-6.74	41.50
RNA	*PairFold* prediction	-58.13	-161.27	-8.12	48.59
	Experimental values [67]	-58.80	-163.32	-8.15	48.31
	Xia et al. prediction [67]	-57.78	-160.08	-8.13	48.49

Table 5.1: Comparison between *PairFold* predictions, experimental data and other predictions [37, 67], on short DNA and RNA duplexes. The values represent average enthalpies, entropies, free energies and melting temperatures.

Table 5.1 shows average enthalpies, entropies, free energies and melting temperatures of *PairFold* prediction (first line), the experimental values reported in [37] (second line), and the predictions reported in [37] (third line), on DNA duplexes. The same type of measurements have been performed for the RNA duplexes reported by Xia et al. [67], and their experimental and predicted values.

The results show that our predictions are nearly as good as the predictions reported in [37] and [67], both of them being close to the experimental values. Tables with all values for each duplex are provided in Tables B.1 and B.2 from Appendix B. The small differences in predictions are explained by the following reasons:

- The predictions performed by Peyret et al. and Xia et al. assume that the secondary structures are known (mismatches for the underlined nucleotides in Table B.1 and perfect matches for the remaining DNA bases, and perfect matches for RNA), and the free energy and enthalpy are calculated for these specific structures. On the contrary, *PairFold* runs the free-energy minimization algorithm. For one duplex out of the 51 DNA duplexes, the predicted structure is substantially different than the expected structure. Also, for 15 out of the 109 RNA duplexes, some of the ends are predicted to be dangling ends rather than base pairs. Details are given in Appendix B.
- For the DNA duplexes, the thermodynamic parameters used by Peyret et al. and *PairFold* are from SantaLucia [45], but some of them might be different versions;
- For the RNA duplexes, we use Turner parameters [58, 59], which we think are somewhat different from the parameters used by Xia et al. [67]. However, the overall prediction accuracy is fairly close.

5.5.2 Predicting ribozyme-mRNA target hybridization

Many experiments on ribozyme-target duplexes have been performed in the last years. *PairFold* can be used to predict secondary structure of a ribozyme-target duplex. Although currently not 100% accurate, it can give some useful information about potential interactions in the duplex. It can also help in ribozyme design, where a pool of many closely related ribozymes are tried until good ones are found. *PairFold* can help eliminate some of these ribozymes, which are unlikely to perform well.

Table 5.2 shows the performance of *PairFold* on 11 ribozyme - mRNA target duplexes drawn from the literature. The secondary structures of their binding are found experimentally and are reported in the referenced papers. Column 2 indicates the reference and the figure showing the sequences and the secondary structures. Columns 3, 4 and 5 give the length of the two sequences and the percentage of odd pairs existing in the structure. Column 6 shows the number of correctly predicted base pairs out of the total number of base pairs in the real structure. Column 7 gives two accuracy parameters: Q_1 and Q_2, calculated as described in Section 4.4.3. Column 8 gives the minimum free energy, as predicted by *PairFold*, and column 9 shows the free energy of the real structure, calculated with our model. Because our model does not consider odd pairs, their contribution to the total free energy was ignored (they were considered 0). The structures do not have pseudoknots.

For 5 duplexes, both parameters Q_1 and Q_2 gave accuracy greater than 0.90, and for all cases, the accuracy was greater than 0.70. Table B.3 in Appendix B shows the sequences,

No.	Reference	l_1	l_2	%Odd	#bp	Q_1 (Q_2)	ΔG_P	ΔG_R^p
1	[27] Fig.1A	11	32	0.00	14/14	1.00 (1.00)	-26.90	-26.90
2	[60] Fig. 2	16	36	0.00	18/19	0.95 (0.88)	-22.60	-21.10
3	[60] Fig. 3	17	38	0.06	17/18	0.94 (0.96)	-20.00	-19.00
4	[43] Fig. 1b	21	92	0.22	33/46	0.72 (0.74)	-58.70	-42.50
5	[53] Fig. 1a	14	59	0.00	23/23	1.00 (0.97)	-34.20	-33.70
6	[53] Fig. 4a	14	59	0.00	19/23	0.83 (0.79)	-35.90	-34.70
7	[47] Fig. 1c	14	65	0.00	19/19	1.00 (0.85)	-21.10	-18.80
8	[47] Fig. 1d	14	55	0.00	15/18	0.83 (0.83)	-16.30	-13.50
9	[47] Fig. 1e	34	120	0.00	40/43	0.93 (0.84)	-58.20	-53.10
10	[25] Fig. 1 left	25	46	0.06	28/31	0.90 (0.92)	-45.80	-43.40
11	[25] Fig. 1 right	70	100	0.03	63/69	0.91 (0.85)	-142.10	-135.90

Table 5.2: Measurement of PairFold accuracy on a set of mRNA target - ribozyme pairs.

the predicted and the real secondary structures, and highlights the wrong predictions and the odd pairs.

5.5.3 Analysis of a combinatorial library of hairpin ribozymes

Some ribozymes, such as hammerhead ribozymes and hairpin ribozymes, are short RNA sequences with catalytic abilities, and they have a specific shape: a part folds to itself and another part binds to an RNA target and it cleaves it at the middle. Figure 5.2 (the same structure has been shown earlier in Figure 1.3) shows the typical structure of a hairpin ribozyme [68]. The top sequence represents the RNA target, also called *substrate*, which is cleaved at the site indicated by the arrow. The bottom sequence is a hairpin ribozyme, which performs the cleavage. The part of the ribozyme which folds to itself is typically a fixed sequence (this core sequence can be found in [68]), but the other part of the ribozyme, also called *substrate binding domain*, varies. Yu et al. [68] created a combinatorial library of ribozymes and tested their cleavage ability on a “highly structured viral mRNA, the 26 S subgenomic RNA of Sindbis virus” [68], GenBank accession number: U38305.

The target RNA was originally of length 4.2 kilo-bases (kb). It has been cut in 4 sequences, of length roughly 1.1kb, and the library of ribozymes has been tested on these fragments. In the presence of ions of Magnesium, 15 sites that were cleaved with high efficiency, called “activity-selected sites”, have been identified. Other 23 sites have been predicted to be cleaved, but the cleavage efficiency was low. These are called “sequence-selected sites”. Yu et al. have also performed experiments on oligonucleotide substrates, as opposed to targeting the long sequences. The reported results show that the cleavage efficiencies are much higher on the short substrates, even for those which were not cleaved while being part of the long sequences.

We used *PairFold* to predict the binding interactions between the hairpin ribozymes which cleaved RNA and their target. Table 5.3 shows our predictions, in the cases when

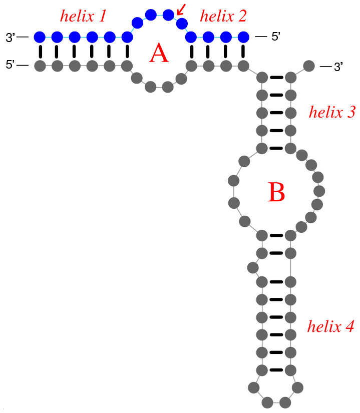


Figure 5.2: Typical structure of a hairpin ribozyme bound to an mRNA target.

the target is an oligo or a long sequence. The first column represents an identification number that we assigned to each case. 'A' stands for "activity-selected site" and 'S' stands for "sequence-selected site". They are followed by a number. The experiments performed by Yu et al. [68] show which was the substrate, but in some cases, one nucleotide of the corresponding ribozyme is unknown, three different bases being possible. In these cases, we performed tests for all three different alternatives, and we chose the best one. The letter at the end of some identification numbers shows which base was chosen in these cases. It is also underlined in the ribozyme substrate binding domain (column 4). Column 2 shows the cleavage site, which is the number of the base in the GenBank nomenclature. Column 3 gives the experimental cleavage efficiencies for the long RNA targets. 3 means strong cleavage efficiency, 2 means moderate, 1 is poor and 0 is undetectable. Columns 5 to 8 give *PairFold* prediction on oligos as target. Column 5 gives the number of correctly predicted base pairs out of the total number of base pairs existing in the real structure. Column 6 gives the two accuracy parameters that we used before: Q_1 and Q_2. Column 7 shows the minimum free energy obtained by *PairFold*, and column 8 gives the standard free energy change of the real structure, using the model underlying *PairFold*. The last column shows the results obtained when running *PairFold* on the ribozymes and the long 1.1kb sequences. If the binding of the ribozyme to the right site was predicted with an accuracy higher than 0.60, then a "+" is indicated. If the secondary structure obtained was very different from the typical ribozyme-substrate duplex secondary structure, then a "-" is indicated. If binding to a site different than the right site has been predicted (i.e. a false positive), the predicted target site is given.

While the predictions between the ribozymes and the oligos show over 0.85 accuracy for most cases and at least 0.70 for all cases, only 4 out of 15 bindings similar to the typical structure are predicted for long sequences. This shows that *PairFold* performs well on short sequences, but as the sequences length increases, the accuracy goes down. For the

Id	Cleav.	Cl.	Ribozyme	#bp	Oligo	ΔG_P	ΔG_R^p	Long
	site	eff.	fragment		$Q_1(Q_2)$			
A1U	8242	2	UGUCUCUGAAGCCU	22/23	0.96 (0.86)	-32.90	-29.30	+
A2	8581	3	UUUUACCGAAGCCG	22/23	0.96 (0.89)	-25.90	-25.10	-
A3	8658	2	GCACGGCGAAGUAU	23/23	1.00 (0.89)	-31.30	-27.80	-
A4A	8663	3	CUAAAGAGAAGUUC	23/23	1.00 (0.93)	-24.90	-24.20	-
A5	9333	2	CGUGUGAGAAGCGU	23/23	1.00 (0.93)	-29.50	-28.80	9152
A6	9413	2	ACUACGCGAAGCGC	23/23	1.00 (0.93)	-29.90	-29.20	+
A7	9516	2	GAUCCACGAAGUGG	23/23	1.00 (0.89)	-32.00	-28.50	-
A8A	9841	3	ACCUAAAGAAGCAC	23/23	1.00 (0.89)	-31.20	-27.70	-
A9	9861	2	GAAUGUCGAAGCAU	23/23	1.00 (0.89)	-27.90	-24.40	-
A10A	9926	2	GGUAUAAGAAGCUG	23/23	1.00 (0.89)	-31.00	-27.50	-
A11C	10115	2	ACAGUACGAAGCAA	19/23	0.83 (0.79)	-23.20	-21.50	-
A12	10122	2	GGACAUAGAAGUAA	22/23	0.96 (0.79)	-28.30	-24.60	-
A13A	10603	3	UCAUCGAGAAGUAU	23/23	1.00 (0.93)	-27.20	-26.50	+
A14	11160	3	UUUGCACGAAGCAU	22/23	0.96 (0.89)	-25.40	-24.60	-
A15U	11178	3	GAUAUGUGAAGCUG	23/23	1.00 (0.89)	-30.10	-26.60	+
S1	7552	0	AUUUAGAGAAGCCG	23/23	1.00 (0.86)	-27.30	-24.70	-
S2	7562	0	UAUGCUAGAAGUUU	19/23	0.83 (0.70)	-24.80	-21.10	-
S3	7752	1	GGCACUAGAAGCUG	23/23	1.00 (0.86)	-32.30	-30.30	-
S4	8084	0	UAUGCUAGAAGCUU	23/23	1.00 (0.86)	-25.40	-23.40	-
S5	8254	0	UCGGACAGAAGCUG	23/23	1.00 (0.86)	-31.30	-29.60	-
S6	8257	0	UAAUCGAGAAGCCA	22/23	0.96 (0.82)	-27.20	-24.60	-
S7	8286	0	UAUCGCAGAAGCCC	23/23	1.00 (0.89)	-31.50	-28.80	-
S8	8295	0	UCCGAGAGAAGUCG	23/23	1.00 (0.86)	-30.40	-27.90	-
S9	8340	0	CCAGGUAGAAGCCG	23/23	1.00 (0.86)	-32.10	-30.10	-
S10	8399	0	GCAGCAAGAAGCUC	23/23	1.00 (0.86)	-32.20	-30.10	-
S11	8647	0	UAUGGUAGAAGUAC	23/23	1.00 (0.82)	-28.40	-24.80	-
S12	8814	0	UUCUUUAGAAGUAU	22/23	0.96 (0.79)	-24.10	-20.40	-
S13	9061	0	CUUUCAAGAAGUCG	23/23	1.00 (0.86)	-27.00	-25.00	-
S14	9393	0	AACAGGAGAAGUGC	23/23	1.00 (0.86)	-30.30	-27.70	-
S15	9552	0	UCGGUCAGAAGUGA	23/23	1.00 (0.86)	-30.30	-28.70	-
S16	9702	0	UGAUGCAGAAGCUA	22/23	0.96 (0.82)	-28.90	-27.20	-
S17	9893	0	CUGUUCAGAAGUAA	22/23	0.96 (0.82)	-25.30	-23.50	-
S18	9945	0	AACGACAGAAGCGG	23/23	1.00 (0.89)	-32.50	-29.00	-
S19	9948	0	UAGAACAGAAGCAG	23/23	1.00 (0.86)	-26.30	-24.70	-
S20	10188	0	GGAGGGAGAAGCAG	23/23	1.00 (0.86)	-34.40	-31.80	-
S21	10355	0	UCUACUAGAAGUUC	23/23	1.00 (0.82)	-26.00	-23.20	-
S22	10812	0	CGGAUUAGAAGCAA	22/23	0.96 (0.82)	-27.70	-25.60	-
S23	10920	0	ACAUUUAGAAGUUG	23/23	1.00 (0.86)	-23.50	-21.50	-

Table 5.3: Measurement of PairFold accuracy on a library of hairpin ribozymes.

"sequence-selected sites", we note that the accuracy parameter Q_2 is consistently lower than the same parameter for the "activity-selected sites". The predictions with the long targets did not find any false positives.

Note that in Yu et al.'s experiments, cleavage on the long targets was detected only in the presence of Magnesium ions. These might have an important role in stabilizing the hairpin structure and in facilitating the binding to the substrates. Our model does not consider the influence of ions on the structure. Including this in secondary structure prediction models is an open question.

5.5.4 Prediction of primer-target binding

Primers are designed to bind to specific targets such that they will perform some function. They are perfect complements of fragments of a longer RNA sequence. Most probably these fragments form some structure with the rest of the sequence. Thus, primers are designed to bind to the target fragment, so that the target can be amplified using the polymerase enzyme. PairFold can be used at predicting the minimum free energy site binding between the primer and the target fragment.

Yu et al. [68] designed 24 primers in order to identify the cleavage sites of the ribozymes used in their experiments, and in order to read the nucleotides which compose the four 1.1 kb sequences they used. Table 5.4 shows the results we obtained when running *PairFold* on the designed primers and the long target sequences. Columns 1 and 2 show the identification number used in Yu et al. [68] and the primer length. Column 3 shows the target site for which the primers were designed, and which, according to Yu et al., the primers bind perfectly. Column 4 shows the binding site predicted by *PairFold*.

In 22 out of the 24 cases, *PairFold* predicts the sites exactly. For 2 duplexes, shown in bold, the prediction made by *PairFold* is that the primers form an internal loop with two different locations.

5.5.5 PairFold in DNA word design

DNA codes are designed for information storage in DNA computation or for molecular bar-codes in chemical libraries [50, 55, 56]. Designing good words is important for minimizing errors due to unwanted hybridization between non-complementary words. Different restrictions between words in the set, such as: hamming distance, GC content and reverse complement hamming distance, have been used for DNA word design [17, 55, 56]. Including thermodynamic constraints in addition to other constraints has been proposed as future work by Tulpan et al. [56]. Currently, they use *PairFold* in their stochastic search methods for designing DNA codes. Whether or not using such constraints proves to be a good solution is an open problem.

Also, *PairFold* has been recently used by Shortreed et al. [50] for generating a combinatorial library of oligonucleotides for large scale DNA computing projects. The final

Id	Len.	Binding site [68]	Predicted site
PA0	21	7483 - 7503	7484 - 7503
PA1	**17**	**7672 - 7688**	**7518 - 7520** **8227 - 8239**
PA2	121	7852 - 7972	7852 - 7972
PA3	19	8106 - 8124	8106 - 8124
PA4	21	8358 - 8378	8358 - 8378
PA5	18	8532 - 8549	8532 - 8549
PB0	20	8503 - 8522	8503 - 8522
PB1	19	8734 - 8752	8734 - 8752
PB2	20	8963 - 8982	8963 - 8982
PB3	19	9184 - 9202	9184 - 9202
PB4	20	9405 - 9424	9405 - 9424
PB5	20	9590 - 9609	9590 - 9609
PC0	20	9514 - 9533	9514 - 9533
PC1	20	9719 - 9738	9719 - 9738
PC2	20	9941 - 9960	9941 - 9960
PC3	**21**	**10163 - 10183**	**9514 - 9516** **10613 - 10616**
PC4	18	10394 - 10411	10394 - 10411
PC5	21	10629 - 10649	10629 - 10649
PD0	19	10576 - 10594	10577 - 10594
PD1	18	10770 - 10787	10770 - 10787
PD2	20	10975 - 10994	10975 - 10994
PD3	20	11210 - 11229	11210 - 11229
PD4	17	11433 - 11449	11433 - 11449
PD5	26	11638 - 11663	11638 - 11663

Table 5.4: PairFold predictions on a set of designed primers.

purpose of their work is to solve a large-scale satisfiability problem using DNA computing.

Chapter 6

CombFold

This chapter discusses our second extension of the dynamic programming algorithm for nucleic acids secondary structure prediction. This extension is applied to a set of RNA or DNA sequences rather than to a single molecule or a pair of molecules.

Consider the following definitions and notation:

- Let *word* denote an RNA or DNA sequence $w = v_1 v_2 \ldots v_l$, where $v_i \in \{A, C, G, U\}$ for RNA and $v_i \in \{A, C, G, T\}$ for DNA. The orientation of the strand is from 5' to 3', unless otherwise stated. For example, `ACGCUAGGCA` is an RNA word of length 10.

- Let *set* denote a set of g *words* of the same length l, formally $S = \{w_1, w_2, \ldots, w_g \mid \mathit{length}(w_i) = \mathit{length}(w_j), \forall i, j \in \{1, \ldots, g\}, i \neq j\}$. The following set is formed of 4 words of length 5:

  ```
  AUACG
  UAGCG
  GCCGA
  CUGCG
  ```

  The words order in a set does not matter, but for convenience later, we assume that the words in S are ranked by their index.

- Let *Input-Set* denote a sequence of s *sets*, $IS = S_1, S_2, \ldots, S_s$, e.g. the following is an *Input-Set* of 5 sets:

UAGCGA	CAGCGUAAUAU	AUGCG	AUAGCGGUA	AUCG
AUAGAU	AGAUGCGCGGU		GAGCGCAAG	CUGC
	UAGGCUAGCGU			GCGA

  Note that the number of words in each *set* can differ, the same for the length of the words across *sets*.

$$\begin{array}{cccc} w_{11} & w_{21} & \dots & w_{s1} \\ w_{12} & w_{22} & \dots & w_{s2} \\ \vdots & \vdots & \vdots & \vdots \\ w_{1g_1} & w_{2g_2} & \dots & w_{sg_s} \end{array}$$

An *Input-Set* can also be written in terms of *words* rather than *sets*: $IS = \{w_{ij}, 1 \leq i \leq s, 1 \leq j \leq g_i\}$, where g_i is the number of *words* in the *set* i.

Thus, an *Input-Set* IS is characterized by s sets, where each set S_i has g_i words, of length l_i. In what follows, when IS is fixed, we consider all its characteristics: s, w_{ij}, g_i, l_i, $\forall i, j, 1 \leq i \leq s, 1 \leq j \leq g_i$, to be known.

- Let *Combination* denote an RNA/DNA sequence, formed by concatenating one word w_{ij} of each set S_i from IS, starting at S_1 and finishing at S_s. For example $C = w_{11}w_{21}\dots w_{s1}$ is a combination formed by concatenating the first word of each set together. Generally, a *combination* is of the form $C = w_{1b_1}w_{2b_2}\dots w_{sb_s}$, where $1 \leq b_i \leq g_i$. Here, b_i denotes the word rank within the set S_i. A *combination* has the length $n = \sum_{i=1}^{s} l_i$. If we think of a *combination* as a sequence of nucleotides rather than a concatenation of words, we can denote it as $C = c_1c_2\dots c_n$.

- Given an *Input-Set* IS, the set of all possible *combinations* forms the *Combina-torial-Set*: $CS = \{w_{1b_1}w_{2b_2}\dots w_{sb_s} \mid 1 \leq b_i \leq g_i\}$. Note that all *combinations* have the same length: $n = \sum_{i=1,s} l_i$ and that CS has $g_1 \times g_2 \times \dots \times g_s$ elements. If $g_i > 1, \forall i$, then the number of elements in CS is exponential in s.

Considering the aforementioned notations, we define the *optimal MFE combination problem of secondary structure prediction of a combinatorial set of RNA molecules* as follows: given an *RNA Input-Set* IS and a thermodynamic model M, predict which *combination*, out of all elements of the *Combinatorial-Set* CS formed from IS, folds to a pseudoknot-free secondary structure with the lowest minimum free energy.

An extension of the *optimal MFE combination problem* is to find the k best MFE combinations, rather than the optimal one only. We define the *k-suboptimal MFE combinations problem of secondary structure prediction of a combinatorial set of RNA molecules* as follows: given an *RNA Input-Set* IS and a thermodynamic model M, predict which k different *combinations*, out of all elements of the *Combinatorial-Set* CS formed from IS, fold to pseudoknot-free secondary structures with the lowest minimum free energies.

The optimal and k-suboptimal MFE combination problem are discussed in this chapter. Note that in our algorithms, given a combination C, we look at the minimum free energy MFE only. Extensions of these problems would be to find suboptimal structures (i.e. whose free energy is greater than the MFE), or to consider pseudoknots, interactions with ions or other tertiary interactions.

		123...	....i......	.....	......j..	...n
b_j	1	UAGCGA	CAGCGUAAUAU	AUGCG	AUAGCG$\underline{\text{G}}$UA	AUCG
b_i	2	AUAGAU	AGAU$\underline{\text{G}}$CGCGGU		GAGCGCAAG	CUGC
	3		UAGGCUAGCGU			GCGA

Table 6.1: Example of a combinatorial set of short RNA sequences.

The remainder of this chapter will first give a dynamic programming algorithm which runs in polynomial time, for solving the *optimal MFE combination problem.* Then, in Section 2, an algorithm for the *k-suboptimal MFE combinations problem* will be presented. Implementation details of these two algorithms into the program *CombFold* will be given in Section 3. A thorough theoretical and empirical analysis of the *optimal and k-suboptimal MFE combinations problem* in Section 4 show that, although they are complex algorithms, they run in polynomial time. We conclude this chapter by presenting applications to biochemical experiments and DNA computing in Section 5.

6.1 A dynamic programming algorithm for the optimal MFE combination problem

The main idea of the *CombFold* algorithm is based on similar arrays and recurrence relations as *SimFold*, discussed in Section 4.1.

One method to solve this problem is to create all possible *combinations* and then to run *SimFold* or an equivalent program on each of them. However, depending on the characteristics of the *Input-Set*, the number of *combinations* may be very big. If $g_i = g > 1, \forall i$, then there are g^s *combinations*, and *SimFold* complexity is $O(n^3)$, as we show in Section 4.3. This leads to a complexity of $O(g^s \cdot n^3)$, where n is the length of the combinations. More generally, the number of combinations is exponential in the number of sets which have at least two words. We have implemented this exhaustive search approach under the name of *ExhaustS*, which will be discussed in Section 6.4 later in this chapter.

This section describes a dynamic programming algorithm to solve the *basic problem of secondary structure prediction of a combinatorial set of RNA molecules* in polynomial time. The main idea of the algorithm is to extend the Zuker and Stiegler [71] algorithm, in which at each position i, there might be several possible nucleotides, since there might be several *words* in the set of i. In the description that follows, we will use indices i and j for the nucleotide position, i.e. columns in Table 6.1. $s(i)$ and $s(j)$ will be the set in which i and j are positioned, respectively. b_i and b_j will be the index of the word within the set $s(i)$ and $s(j)$, i.e. the rows in Table 6.1. Given a set S, $g(S)$ will return the number of words in S. Hence, b_i can take $g(s(i))$ values. In order to get the base at the column i and row b_i of the *Input-Set IS*, we will use the function: $c_i = \mathit{Nucleotide}(IS, b_i, i)$. Table 6.1 shows the nucleotides c_i and c_j.

Arrays

Instead of two-dimensional arrays, as we showed that *SimFold* needed (Section 4.1), now we deal with four-dimensional arrays, since for the nucleotide at the position i, there might be several possible words, and similarly for the nucleotide at the position j. The names of the arrays for *CombFold* correspond to the same names for *SimFold*, and a c (from *CombFold*) is added as subscript. Note that, having i and fixing the value of b_i, the word index within the set of i, we can easily determine the nucleotide. In what follows, we assume we use the function *Nucleotide* to find the base c_i whenever IS, b_i and i are known, without explicitly stating so. In the following arrays, the values for each b_i, b_j, i and j, will be calculated:

- $W'(j)$ denotes the free energy change of the first j nucleotides of a *combination* $c_1c_2 \ldots c_j$, where the word index within the set of j, b_j, was chosen such that the free energy is minimized. Consequently, $W'(n)$ contains the minimum free energy change over the entire *Combinatorial-Set* corresponding to the *Input-Set* IS;
- $W^c(b_j, j)$ denotes the free energy change of the *combination* in which the index of the set of j is b_j , therefore it is fixed;
- $V^c(b_i, b_j, i, j)$ is the minimum free energy of the *combination* in which b_i and b_j are fixed, assuming that *(c_i.c_j)* is a base pair;
- $H^c(b_i, b_j, i, j)$ is the free energy of the *combination* in which b_i and b_j are fixed, assuming that *(c_i.c_j)* closes a hairpin loop;
- $S^c(b_i, b_j, i, j)$ is the free energy of the *combination* in which b_i and b_j are fixed, assuming that *(c_i.c_j)* closes a stacked loop;
- $VBI^c(b_i, b_j, i, j)$ is the free energy of the *combination* in which b_i and b_j are fixed, assuming that *(c_i.c_j)* closes an internal loop;
- $VM^c(b_i, b_j, i, j)$ is the free energy of the *combination* in which b_i and b_j are fixed, assuming that *(c_i.c_j)* closes a multi-branched loop;
- $WM^c(b_i, b_j, i, j)$ is used to compute the array VM.

Recurrence relations

The same as for *SimFold* (Section 4.1) and *PairFold* (Section 5.1), the aforementioned arrays are constructed by recurrence relations.

$W'(j)$ is calculated by minimizing the values $W^c(b_j, j)$ over all possible b_j:

$$W'(j) = min_{b_j} W^c(b_j, j)$$

The equation for $W^c(b_j, j)$ is an extension of the relation for $W(j)$ from Section 4.1:

$$W^c(b_j, j) = \min_{1\leq i<j}$$

$$\begin{cases}
\min_{b_{j-1}\in X(\{b_j,j\},\{j-1\})} W^c(b_{j-1}, j-1), \\
\min_{b_{i-1},b_i\in X(\{b_j,j\},\{i-1,i\})}(V^c(b_i, b_j, i, j) + \textit{Non-GC-penalty}(c_i, c_j)+ \\
\quad W^c(b_{i-1}, i-1)), \\
\min_{b_{i-1},b_i,b_{i+1}\in X(\{b_j,j\},\{i-1,i,i+1\})}(V^c(b_{i+1}, b_j, i+1, j)+ \\
\quad \textit{Non-GC-penalty}(c_{i+1}, c_j) + \Delta \textit{G-Dangle-3'}(c_j, c_{i+1}, c_i) + W^c(b_{i-1}, i-1)), \\
\min_{b_i,b_{j-1}\in X(\{b_j,j\},\{i,j-1\})}(V^c(b_i, b_{j-1}, i, j-1) + \textit{Non-GC-penalty}(c_i, c_{j-1})+ \\
\quad \Delta \textit{G-Dangle-5'}(c_{j-1}, c_i, c_j) + W^c(b_{i-1}, i-1)), \\
\min_{b_{i-1},b_i,b_{i+1},b_{j-1}\in X(\{b_j,j\},\{i-1,i,i+1,j-1\})}(V^c(b_{i+1}, b_{j-1}, i+1, j-1)+ \\
\quad \textit{Non-GC-penalty}(c_{i+1}, c_{j-1}) + \Delta \textit{G-Dangle-3'}(c_{j-1}, c_{i+1}, c_i)+ \\
\quad \Delta \textit{G-Dangle-5'}(c_{j-1}, c_{i+1}, s_j) + W^c(b_{i-1}, i-1))
\end{cases}$$

where X is a function which returns the range of words for all the needed indexes. For example, in the first line, in *SimFold*, we had to calculate $W(j-1)$. In *CombFold*, the word corresponding to $j-1$ depends on the sets to which j and $j-1$ belong, and on b_j :

$$X(\{b_j, j\}, \{j-1\}) = \begin{cases} \{b_j\} & , \text{ if } (s(j-1) = s(j)) \\ \{1, \ldots, g(s(j-1))\}, & \text{if } (s(j-1) \neq s(j)) \end{cases}$$

For the second line, there are two word indices, b_{i-1} and b_i, that we have to find the ranges for:

$$X(\{b_j, j\}, \{i-1, i\}) =$$

$$\begin{cases}
b_j, b_j & , \text{ if } (s(i-1) = s(i) = s(j)) \\
\{1, \ldots, g(s(i-1))\}, b_j & , \text{ if } (s(i-1) \neq s(i) = s(j)) \\
\{1, \ldots, g(s(i-1))\}, b_{i-1} & , \text{ if } (s(i-1) = s(i) \neq s(j)) \\
\{1, \ldots, g(s(i-1))\}, \{1, \ldots, g(s(i))\}, & \text{if } (s(i-1) \neq s(i) \neq s(j))
\end{cases}$$

In the first two *if* lines of the equation for W^c above, the values of b_{j-1} (first line), and b_{i-1}, b_i (second line), depend on one other index: i, and its corresponding b. However, in a more general case, there are p indexes with known b's, and q indexes with unknown b's, for which we want to find the ranges. Note that the number of *if* lines will be 2^{p+q-1}. Given the thermodynamic model described in Chapter 3, the highest values for p and q are 4 and 4, i.e. in the case of internal loops.

The function X calculates the ranges for the unknown b's, for any number of known and unknown indexes. Procedure 6 gives the pseudocode for X procedure. The input is comprised of two sets: the first set contains the known b's and the known indexes: $\{b_{i_1} \ldots b_{i_p}, i_1 \ldots i_p\}$. They will help to decide the ranges of the unknown b's. The second

```
Compute X Procedure

input: set of $p$ indexes with known $b$'s $\{b_{i_1} \ldots b_{i_p}, i_1 \ldots i_p\}$,
set of $q$ indexes with unknown $b$'s $\{j_1 \ldots j_q\}$;
output: $q$ sets $B_{j_1} \ldots B_{j_q}$ corresponding to $\{j_1 \ldots j_q\}$;

    procedure Compute X
        order the indexes $i$'s and $j$'s;
        identify the sets $S_1 \ldots S_m$ to which $i$'s and $j$'s belong;
        for ($S = S_1$ to $S_m$)
            if (there exists $i_k$ in set $S$)
                foreach ($j_u$ in set $S$)
                    $B_{j_u} = \{b_{i_k}\}$;
                end foreach;
            else
                $j_v \leftarrow$ the smallest $j$ in $S$;
                $B_{j_v} = \{1, \ldots, g(S)\}$;
                foreach ($j_u$ in set $S$, with $j_u \neq j_v$)
                    $B_{j_u} = B_{j_v}$;
                end foreach;
            end if;
        end for;
        return $B_{j_1}, \ldots, B_{j_q}$;
    end procedure X.
```

Procedure 6: Pseudocode for the X procedure. Details are described in the text.

set contains the indexes of the unknown b's, $\{j_1 \ldots j_q\}$. First, we need to order all the values $i_1 \ldots i_p, j_1 \ldots j_q$. This is necessary for the second step, which identifies the sets corresponding to each index. The two extreme situations are: (1) all indexes are in the same set, and thus there will be only one possible configuration for the unknown b's; (2) all indexes are in different sets, hence there will be $g(s(j_1)) \times \ldots \times g(s(j_q))$ possible values for the unknown b's.

Once we identified the sets, for each set S, first we check whether there exists the index of a known b in this set. If this is the case, then all the j's in S will have the corresponding b's equal to the known b. No other option is available for these unknown b's, since the value for the known b is fixed. If no known b exists in S, then all the unknown b's in S will be in the range $\{1, \ldots, g(S)\}$, with the constraint that they will have the same value at a given time, being in the same set. In other words, we can give a value to the b of the smallest index in S, and all the other b's in S will have the same value. The function X will return a set of values for the needed unknown b's.

An example of a particular situation, with the sets $\{b_i, b_j, i, j\}$ and $\{i+1, i+2, j-2, j-1\}$ as input, is presented in Table 6.2, where $s(i) \neq s(i+1) = s(i+2) \neq s(j-2) \neq s(j-1) = s(j)$. In this case, b_{i+1} will take values in the range $\{1, \ldots, g(s(i+1))\}$, b_{i+2} will take the value that b_{i+1} takes, b_{j-2} will be in the range $\{1, \ldots, g(s(j-2))\}$, and b_{j-1}

equals b_j. Hence, for this particular situation, there will be $g(s(i+1)) \times g(s(j-2))$ terms to minimize over.

i	$i+1$	$i+2$	$j-2$	$j-1$	j
b_i	1	b_{i+1}	1	b_j	b_j
	$\vdots$		$\vdots$		
	$g(s(i+1))$		$g(s(j-2))$		

Table 6.2: Example of choices for b values for a particular situation.

Using function X to decide which the possible values for each word, the remaining recurrence relations for *CombFold* are a logical extension of the corresponding recurrence relations for *SimFold*, described in Section 4.1: the equation for V^c is an extension of the relation for V, H^c corresponds to H, S^c with S, VBI^c with VBI, VM^c with VM and WM^c with WM. The relations for V^c and H^c are straightforward:

$$V^c(b_i, b_j, i, j) = \begin{cases} +\infty & , \quad \text{for } i \geq j \\ \min(H^c(b_i, b_j, i, j), S^c(b_i, b_j, i, j), VBI^c(b_i, b_j, i, j) & \\ \quad VM^c(b_i, b_j, i, j)) & , \quad \text{for } i < j \end{cases}$$

$$H^c(b_i, b_j, i, j) = \Delta G\text{-}H^c(IS, b_i, b_j, i, j)$$

Note that, although here we use the same thermodynamic model as for *SimFold*, the procedure for hairpin calculation for *CombFold* needs to be rewritten. Procedure 7 shows that, in the situations where we need to access the nucleotides at other locations than the input (i.e. i and j), a minimization over the possible b's is necessary. Also, note that the special cases of *GGG hairpin* and *poly-C hairpin* are omitted at this version of *CombFold*, since the algorithm gets too complicated. At the end of the procedure, the values that turned out to be the best for the b's tried inside the procedure are saved for later reference.

For the calculation of stacked loops, finding b_{i+1} and b_{j-1} is imposed again by the nearest neighbour model itself.

$$S^c(b_i, b_j, i, j) = \min_{b_{i+1}, b_{j-1} \in X(\{b_i, b_j, i, j\}, \{i+1, j-1\})}$$
$$(\Delta G\text{-}S^c(IS, b_i, b_j, b_{i+1}, b_{j-1}, i, j) + V(b_{i+1}, b_{j-1}, i+1, j-1))$$

where

$$\Delta G\text{-}S^c(IS, b_i, b_j, b_{i+1}, b_{j-1}, i, j) = \Delta\text{-}\mathit{Stack}(c_i, c_j, c_{i+1}, c_{j-1})$$

The internal loop free energy calculation is a minimization over i' and j', i.e. the closing pair of the internal loop. Once i' and j' fixed, we calculate the free energy value for

CombFold Hairpin-Loop-Free-Energy Procedure

input: IS,b_i,b_j,i,j;
output: free energy ΔG;

```
procedure Compute $\Delta G$-$H^c$
    $\Delta G_1 := 0$; $\Delta G_2 := 0$; $\Delta G_3 := 0$;
    $l := j - i - 1$;
    if $(l < 3)$
        $\Delta G :=$ infinity;
    else
        $\Delta G_1 := \Delta G$-Length-$H(l)$;
        if $(l = 3)$
            $\Delta G_2 :=$ Non-GC-penalty($c_i, c_j$);
            $\Delta G_3 := \min_{b_{i+1},b_{i+2},b_{j-1} \in X(\{b_i,b_j,i,j\},\{i+1,i+2,j-1\})}$
                    $\Delta G$-Hairpin-3$(c_i, c_{i+1}, c_{i+2}, c_{j-1}, c_j)$;
        else
            $\Delta G_2 := \min_{b_{i+1},b_{j-1} \in X(\{b_i,b_j,i,j\},\{i+1,j-1\})} \Delta G$-Hairpin-n$(c_i, c_j, c_{i+1}, c_{j-1})$;
            if $(l = 4)$
                $\Delta G_3 := \min_{b_{i+1},b_{i+2},b_{j-2},b_{j-1} \in X(\{b_i,b_j,i,j\},\{i+1,i+2,j-2,j-1\})}$
                        $\Delta G$-Hairpin-4$(c_i, c_{i+1}, c_{i+2}, c_{j-2}, c_{j-1}, c_j)$;
            endif;
        endif;
        $\Delta G := \Delta G_1 + \Delta G_2 + \Delta G_3$;
    endif;
    save best $b$'s;
    return $\Delta G$;
end procedure $\Delta G$-$H^c$.
```

Procedure 7: Outline of the calculation for hairpin loop free energy for *CombFold*.

each possible $b_{i'}$ and $b_{j'}$. The procedure for ΔG-I^c calculation needs some changes too, i.e. minimization over different b for the special cases, and is given in Procedure 8.

$$VBI^c(b_i, b_j, i, j) = \min_{i<i'<j'<j} \Big(\min_{b_{i'},b_{j'} \in X(\{b_i,b_j,i,j\},\{i',j'\})}$$
$$(\Delta G\text{-}I^c(IS, b_i, b_j, b_{i'}, b_{j'}, i, j, i', j') + V(b_{i'}, b_{j'}, i', j'))\Big)$$

The free energy for multi-loops adds the minimization over the necessary b's as well. The equations for WM^c and VM^c follow:

$$WM^c(b_i, b_j, i, j) = \min$$

CombFold Internal-Loop-Free-Energy Procedure

input: IS,b_i,b_j,$b_{i'}$,$b_{j'}$,i,j,i',j';
output: free energy ΔG;

```
procedure Compute $\Delta G$-$I^c$
    $\Delta G_1 := 0$; $\Delta G_2 := 0$; $\Delta G_3 := 0$;
    $l_1 := i' - i - 1$; $l_2 := j - j' - 1$; $l = l_1 + l_2$
    if ($l_1 = 0$ or $l_2 = 0$)
        $\Delta G := \Delta G$-Length-B$(l)$;
        if ($l_1 + l_2 = 1$)
            $\Delta G := \Delta G + \Delta G$-Stack$(c_i, c_j, c_{i'}, c_{j'})$
        else
            $\Delta G := \Delta G$ + Non-GC-Penalty$(c_i, c_j)$ + Non-GC-Penalty$(c_{i'}, c_{j'})$;
        endif;
    else if ($l_1 = 1$ and $l_2 = 1$)
        $\Delta G := \min_{b_{i+1}, b_{j-1} \in X(\{b_i, b_j, i, j\}, \{i+1, j-1\})}$
                $\Delta G$-Internal-2$(c_i, c_j, c_{i'}, c_{j'}, c_{i+1}, c_{j-1})$;
    else if ($l_1 = 1$ and $l_2 = 2$)
        $\Delta G := \min_{b_{i+1}, b_{j-1}, b_{j'+1} \in X(\{b_i, b_j, i, j\}, \{i+1, j-1, j'+1\})}$
                $\Delta G$-Internal-3$(c_i, c_j, c_{i'}, c_{j'}, c_{i+1}, c_{j-1}, c_{j'+1})$;
    else if ($l_1 = 2$ and $l_2 = 1$)
        $\Delta G := \min_{b_{j-1}, b_{i'-1}, b_{i+1} \in X(\{b_i, b_j, i, j\}, \{j-1, i'-1, i+1\})}$
                $\Delta G$-Internal-3$(c_{j'}, c_{i'}, c_j, c_i, c_{j-1}, c_{i'-1}, c_{i+1})$;
    else if ($l_1 = 2$ and $l_2 = 2$)
        $\Delta G := \min_{b_{i+1}, b_{j-1}, b_{i'-1}, b_{j'+1} \in X(\{b_i, b_j, i, j\}, \{i+1, j-1, i'-1, j'+1\})}$
                $\Delta G$-Internal-4$(c_i, c_j, c_{i'}, c_{j'}, c_{i+1}, c_{j-1}, c_{i'-1}, c_{j'+1})$;
    else
        $\Delta G_1 := \Delta G$-Length-I$(l)$;
        if (($l_1 = 1$ or $l_2 = 1$) and Gail-Rule $= 1$)
            $\Delta G_2 := \Delta G$-Internal-n$(c_i, c_j,' A',' A') + \Delta G$-Internal-n$(c_{i'}, c_{j'},' A',' A')$;
        else
            $\Delta G_2 := \min_{b_{i+1}, b_{j-1}, b_{i'-1}, b_{j'+1} \in X(\{b_i, b_j, i, j\}, \{i+1, j-1, i'-1, j'+1\})}$
                $\Delta G$-Internal-n$(c_i, c_j, c_{i+1}, c_{j-1}) + \Delta G$-Internal-n$(c_{j'}, c_{i'}, c_{j'+1}, c_{i'-1})$;
        endif;
        $\Delta G_3 := \Delta G$-Asymmetry $(l_1, l_2)$;
        $\Delta G := \Delta G_1 + \Delta G_2 + \Delta G_3$;
    endif;
    save best $b$'s;
    return $\Delta G$;
end procedure $\Delta G$-$I^c$.
```

Procedure 8: Outline of the calculation for internal loop free energy for *CombFold*.

$$\begin{cases} V^c(b_i, b_j, i, j) + \textit{Non-GC-penalty}(c_i, c_j) + \textit{Multi-b}, \\ \min_{b_{i+1} \in X(\{b_i, b_j, i, j\}, \{i+1\})} (V^c(b_{i+1}, b_j, i+1, j) + \textit{Non-GC-penalty}(c_{i+1}, c_j) + \\ \quad \Delta \textit{G-Dangle-3'}(c_j, c_{i+1}, c_i) + \textit{Multi-b} + \textit{Multi-c}), \\ \min_{b_{j-1} \in X(\{b_i, b_j, i, j\}, \{j-1\})} (V^c(b_i, b_{j-1}, i, j-1) + \textit{Non-GC-penalty}(c_i, c_{j-1}) + \\ \quad \Delta \textit{G-Dangle-5'}(c_{j-1}, c_i, c_j) + \textit{Multi-b} + \textit{Multi-c}), \\ \min_{b_{i+1}, b_{j-1} \in X(\{b_i, b_j, i, j\}, \{i+1, j-1\})} (V^c(b_{i+1}, b_{j-1}, i+1, j-1) + \\ \quad \textit{Non-GC-penalty}(c_{i+1}, c_{j-1}) + \Delta \textit{G-Dangle-3'}(c_{j-1}, c_{i+1}, c_i) + \\ \quad \Delta \textit{G-Dangle-5'}(c_{j-1}, c_{i+1}, c_j) + \textit{Multi-b} + 2 \times \textit{Multi-c}), \\ \min_{b_{i+1} \in X(\{b_i, b_j, i, j\}, \{i+1\})} (WM^c(b_{i+1}, b_j, i+1, j) + \textit{Multi-c}), \\ \min_{b_{j-1} \in X(\{b_i, b_j, i, j\}, \{j-1\})} (WM^c(b_i, b_{j-1}, i, j-1) + \textit{Multi-c}), \\ \min_{i \le h < j, b_h, b_{h+1} \in X(\{b_i, b_j, i, j\}, \{h, h+1\})} (WM^c(b_i, b_h, i, h) + WM^c(b_{h+1}, b_j, h+1, j)) \end{cases}$$

$$VM^c(b_i, b_j, i, j) = \min \begin{cases} \min_{b_{i+1}, b_{j-1} \in X(\{b_i, b_j, i, j\}, \{i+1, j-1\})} WM^c(b_{i+1}, b_{j-1}, i+1, j-1), \\ \min_{b_{i+1}, b_{i+2}, b_{j-1} \in X(\{b_i, b_j, i, j\}, \{i+1, i+2, j-1\})} (WM^c(b_{i+2}, b_{j-1}, i+2, j-1) + \\ \quad \Delta \textit{G-Dangle-3'}(c_i, c_j, c_{i+1}) + \textit{Multi-c}), \\ \min_{b_{i+1}, b_{j-2}, b_{j-1} \in X(\{b_i, b_j, i, j\}, \{i+1, j-2, j-1\})} (WM^c(b_{i+1}, b_{j-2}, i+1, j-2) + \\ \quad \Delta \textit{G-Dangle-5'}(c_i, c_j, c_{j-1}) + \textit{Multi-c}), \\ \min_{b_{i+1}, b_{i+2}, b_{j-2}, b_{j-1} \in X(\{b_i, b_j, i, j\}, \{i+1, i+2, j-2, j-1\})} (WM^c(b_{i+2}, b_{j-2}, i+2, j-2) + \\ \quad \Delta \textit{G-Dangle-3'}(c_i, c_j, c_{i+1}) + \Delta \textit{G-Dangle-5'}(c_i, c_j, c_{j-1}) + 2 \times \textit{Multi-c}) \end{cases}$$

In this section we described the arrays and recurrence equations for predicting the optimal MFE combination of an *Input-Set.* The algorithm, as presented here, will find only the best combination only, but it can be used to find the next k best combinations. An algorithm for finding the k best combinations is described in the following section.

6.2 An algorithm for the k-suboptimal MFE combinations problem

The algorithm for the *optimal MFE combination problem*, just described in the previous section, returns only the combination which has the smallest MFE. The algorithm can be extended to return the k combinations that have the lowest MFE.

Consider the *Input-Set IS* contains s sets S_i, each having g_i words. We will add the superscript "(1)" to the notation of our sets to denote that first we are looking for the optimal combinations. The superscripts for the next combinations will be "(2)" and so on. Thus, $IS^{(1)} = \{S_1^{(1)}, S_2^{(1)}, \ldots, S_s^{(1)}\}$ will have the *Combinatorial-Set* $CS^{(1)}$ associated and will contain the following words:

First, we find the optimal MFE combination using the method described in the previous section. Let the combination $C^{(1)} = \{w_{1C_1}^{(1)} w_{2C_2}^{(1)} \ldots w_{sC_s}^{(1)}\}$ denote the optimal MFE

$S_1^{(1)}$	$S_2^{(1)}$	$\dots$	$S_s^{(1)}$
w_{11}	w_{21}	$\dots$	w_{s1}
w_{12}	w_{22}	$\dots$	w_{s2}
$\vdots$	$\vdots$	$\vdots$	$\vdots$
w_{1g_1}	w_{2g_2}	$\dots$	w_{sg_s}

combination, where C_i denotes the index of the word in the set $S_i^{(1)}$, which belongs to the optimum combination. The *Input-Set* $IS^{(1)}$ contains all the possible combinations of the original set IS. To find the next best combinations, first we partition the set $IS^{(1)}$ into s sets which do not contain $C^{(1)}$:

$$\begin{array}{llllll}
IS^{(2)1} = \{ & S_1^{(1)} - \{w_{1C_1}^{(1)}\}, & S_2^{(1)}, & \dots & , S_s^{(1)} & \} \\
IS^{(2)2} = \{ & \{w_{1C_1}^{(1)}\}, & S_2^{(1)} - \{w_{2C_2}^{(1)}\}, & \dots & , S_s^{(1)} & \} \\
\vdots & & & & & \\
IS^{(2)s} = \{ & \{w_{1C_1}^{(1)}\}, & \{w_{2C_2}^{(1)}\}, & \dots & , S_s^{(1)} - \{w_{sC_s}^{(1)}\} & \}
\end{array}$$

For convenience later, we denote the newly created sets with $S_i^{(2)j}$, where $1 \leq i, j \leq s$, i denotes the set index within the *Input-Set*, as in the previous notations, and j denotes the index of the newly created *Input-Set*:

$$\begin{array}{l}
IS^{(2)1} = \{S_1^{(2)1}, S_2^{(2)1}, \dots, S_s^{(2)1}\} \\
IS^{(2)2} = \{S_1^{(2)2}, S_2^{(2)2}, \dots, S_s^{(2)2}\} \\
\vdots \\
IS^{(2)s} = \{S_1^{(2)s}, S_2^{(2)s}, \dots, S_s^{(2)s}\}
\end{array}$$

The *Input-Sets* $IS^{(2)1}, IS^{(2)2}, \dots, IS^{(2)s}$ have the following properties:

- $C^{(1)} \notin CS^{(2)m}, \forall m, 1 \leq m \leq s$;
- $CS^{(2)m} \cap CS^{(2)m'} = \emptyset, \forall m, m', 1 \leq m, m' \leq s, m \neq m'$;
- $\{C^{(1)}\} \cup CS^{(2)m} \cup CS^{(2)m'} = CS^{(1)}, \forall m, m', 1 \leq m, m' \leq s, m \neq m'$,

where $CS^{(i)j}$ denotes the *Combinatorial-Set* associated to the *Input-Set* $IS^{(i)j}$. In other words, (1) the combination $C^{(1)}$ is not included in any of the new *Input-Sets* created by partitioning, (2) the new input sets do not have any combinations in common and (3) the whole space of combinations in $CS^{(1)}$ is covered by the new input sets plus the optimal combination found. This leads to finding the optimal combinations for each of $IS^{(2)1}, IS^{(2)2}, \dots, IS^{(2)s}$, followed by choosing the one with the smallest MFE. Thus, the standard free energy change

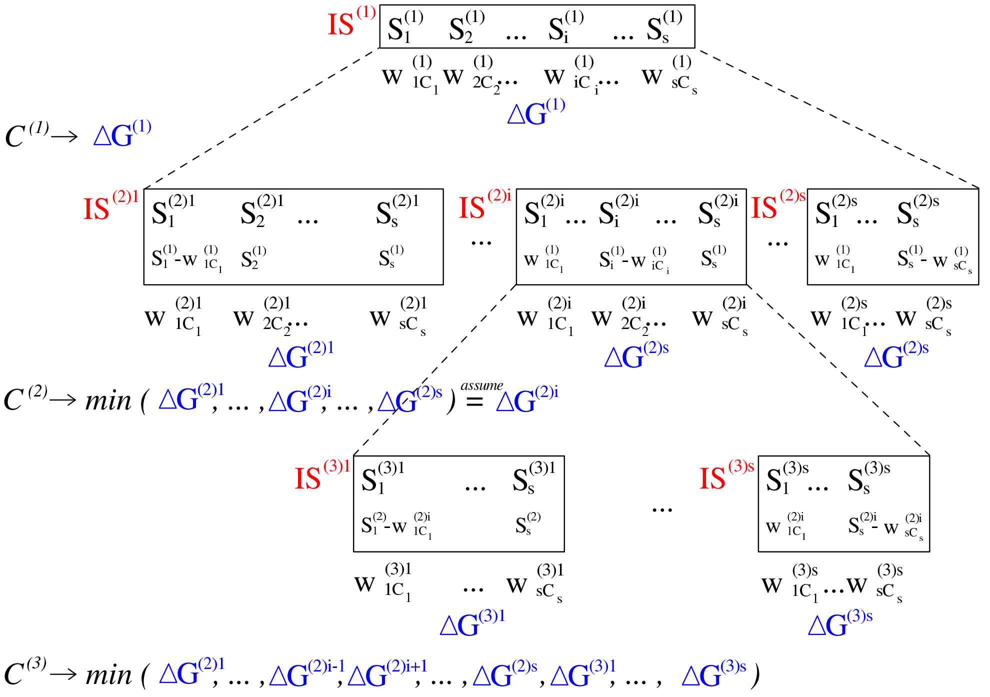

IS(1)
S1(1) S2(1) ... Si(1) ... Ss(1)
w(1)1C1 w(1)2C2 ... w(1)iCi ... w(1)sCs
ΔG(1)
C(1) → ΔG(1)
IS(2)1
S1(2)1 S2(2)1 ... Ss(2)1
S1(1)-w(1)1C1 S2(1) Ss(1)
w(2)11C1 w(2)12C2 ... w(2)1sCs
ΔG(2)1
IS(2)i
S1(2)i ... Si(2)i ... Ss(2)i
w(1)1C1 Si(1)-w(1)iCi Ss(1)
w(2)i1C1 w(2)i2C2 ... w(2)isCs
ΔG(2)s
IS(2)s
S1(2)s ... Ss(2)s
w(1)1C1 Ss(1)- w(1)sCs
w(2)s1C1 ... w(2)ssCs
ΔG(2)s
C(2) → min (ΔG(2)1, ... , ΔG(2)i, ... , ΔG(2)s) assume = ΔG(2)i
IS(3)1
S1(3)1 ... Ss(3)1
S1(2)-w(2)i1C1 Ss(2)
w(3)11C1 ... w(3)1sCs
ΔG(3)1
IS(3)s
S1(3)s ... Ss(3)s
w(2)i1C1 Ss(2)i- w(2)isCs
w(3)s1C1 ... w(3)ssCs
ΔG(3)s
C(3) → min (ΔG(2)1, ... , ΔG(2)i-1, ΔG(2)i+1, ... , ΔG(2)s, ΔG(3)1, ... , ΔG(3)s)

Figure 6.1: The algorithm for finding the k-suboptimal MFE combinations of a combinatorial set.

of the second combination will be $\Delta G^{(2)} = \min(\Delta G^{(2)1}, \Delta G^{(2)2}, \dots \Delta G^{(2)s})$, where $\Delta G^{(2)i}$ is the MFE of the $IS^{(2)i}$. Let i be such that $\Delta G^{(2)} = \Delta G^{(2)i}$ is the smallest MFE and let $C^{(2)} = \{w^{(2)i}_{1C_1} w^{(2)i}_{2C_2} \dots w^{(2)i}_{sC_s}\}$ denote the second best combination. The next step is to partition $IS^{(2)i}$, in the same way we partitioned $IS^{(1)}$. We will obtain the *Input-Sets* $IS^{(3)1}, IS^{(3)2}, \dots IS^{(3)s}$. Now, note that the following are true:

- $C^{(1)}$ and $C^{(2)} \notin CS^{(2)m}$ and $CS^{(3)n}, \forall m, n, i \le m, n \le s, m \neq i$;
- $CS^{(2)m} \cap CS^{(2)m'} \cap CS^{(3)n} \cap CS^{(3)n'} = \emptyset, \forall m, n, i \le m, n \le s, m \neq i, m \neq m', n \neq n'$;
- $\{C^{(1)}, C^{(2)}\} \cup CS^{(2)m} \cup CS^{(2)m'} \cup CS^{(3)n} \cup CS^{(3)n'} = CS^{(1)}, \forall m, n, i \le m, n \le s, m \neq i, m \neq m', n \neq n'$.

Thus, the MFE of the third combination will be:
$\Delta G^{(3)} = \min(\Delta G^{(2)1}, \dots, \Delta G^{(2)i-1}, \Delta G^{(2)i+1}, \dots, \Delta G^{(2)s}, \Delta G^{(3)1}, \dots, \Delta G^{(3)s})$.

Procedure 6.1 shows the steps just described. Recursively continuing in the same way, we can find the best k combinations. However, note that the tree of partitioned *Input-Sets* will grow proportionally with k, more exactly, it will grow by at most $k \cdot s$, which implies increase in run time and space. Section 6.3 will discuss how to implement this extension in an efficient way.

It is important to note that when creating the new *Input-Set*, the ones with a lower index will typically have a bigger solution space (i.e. number of possible combinations) than the ones with a higher index. Thus, if after we found the second combination, $\Delta G^{(2)} = \Delta G^{(2)s}$, the third combination will be found much more quickly than if $\Delta G^{(2)} = \Delta G^{(2)1}$. Also, it is possible that the *Input-Set* which has the next best combinations will be partitioned in less than s partitions (or even no partitions at all), since the other partitions are empty. In this case, only the optimal MFE combinations of the non-empty partitions will be considered. Examples of the running time on some problem instances are discussed in Section 6.4.

6.3 Implementation

In this section we describe implementation details of *CombFold*, as a logical extension of the implementation details described for *SimFold*, in Section 4.2.

Implementation for the optimal MFE combination problem

First, the pseudocode of the algorithm for the optimal MFE combination problem is described. The arrays W', W^c, V^c and WM^c will store information necessary to backtrack and extract the combination with the smallest MFE, and its secondary structure and MFE value. Let W'_{ds}, W^c_{ds}, V^c_{ds} and WM^c_{ds} denote the data structures associated with each of the arrays W', W^c, V^c and WM^c. $W'_{ds}(j)$ will contain information about the free energy of the best combination up to index j, as well as the index b_j within the set. $W^c_{ds}(b_j, j)$ will contain the free energy of the best combination up to index (b_j, j), the number of branches of the multi-domain, the indexes of the last domain, and the right index (including its b) of the next domain. $V^c_{ds}(b_i, b_j, i, j)$ contains the free energy, the optimal type (HAIRPIN_LOOP, STACKED_LOOP, INTERNAL_LOOP or MULTI_LOOP) for the pair *($c_i.c_j$)* and details about where the internal branches for internal loops and multi-loops are located, in case *($c_i.c_j$)* closes such an elementary structure. Finally, $WM^c_{ds}(b_i, b_j, i, j)$ contains the free energy of the multi-loop fragment and other information needed to reconstitute the multi-loop branches completely.

Note that the arrays W^c_{ds} and V^c_{ds}, as opposed to the corresponding arrays W_{ds} and V_{ds} for *SimFold*, will also contain information about the neighbouring indexes, such as $b_{i+1}, b_{j-1}, i+1, j-1$, that have been chosen while constructing the path of the best combination. Using all these dependencies will permit finding the best combination in the backtracking procedure.

Procedure 9 gives a pseudocode of the *CombFold* algorithm for determining the optimal MFE combination. The main idea is the same as in the case of *SimFold*, but now we have to traverse all the words in each set, in addition to the combinations indexed from 1 to n. The function *Compute-WM^c* fills the WM^c_{ds} array with information regarding partial multi-loops, using the equation for WM^c given in Section 6.1. The function *Compute-*

CombFold-optimal Procedure

input: RNA or DNA *Input-Set* IS;
output: minimum free energy ΔG, secondary structure R, combination C;

```
procedure CombFold-optimal
    for ($j := 1$ to $n$)
        for ($b_j := 1$ to $g(s(j))$)
            for ($i := j - 1$ down to 1)
                for ($b_i \in X(\{b_j, j\}, \{i\})$)
                    $WM^c_{ds}(b_i, b_j, i, j) :=$ Compute-WM$^c(b_i, b_j, i, j)$;
                end for;
            end for;
            for ($i := 1$ to $j - 1$)
                for ($b_i \in X(\{b_j, j\}, \{i\})$)
                    $V^c_{ds}(b_i, b_j, i, j) :=$ Compute-V$^c(b_i, b_j, i, j)$;
                end for;
            end for;
        end for;
    end for;
    for ($j := 2$ to $n$)
        for ($b_j := 1$ to $g(s(j))$)
            $W^c_{ds}(b_j, j) :=$ Compute-W$^c(b_j, j)$;
        end for;
        $W'_{ds}(j)$.free_energy := $\min_{b_j}(W^c_{ds}(b_j, j)$.free_energy);
        $W'_{ds}(j).b := b_j$ that minimizes $W^c_{ds}(b_j, j)$.free_energy;
    end for;
    $\Delta G := W'_{ds}(n)$.free_energy;
    $i := n$;
    $b_i := W'_{ds}(i).b$;
    while ($i > 0$ and $W^c_{ds}(b_i, i)$.num_branches $> 0$)
        $(b_{i_d}, b_{j_d}, i_d, j_d) := W^c_{ds}(b_i, i)$.last_domain;
        $(R, C) :=$ CombFold-Backtrack $(b_{i_d}, b_{j_d}, i_d, j_d, V^c_{ds}, W^c_{ds})$;
        $(b_i, i) := W^c_{ds}(b_i, i)$.next_domain_i;
    end while;
    return $\Delta G$, $R$, $C$;
end procedure CombFold-optimal.
```

Procedure 9: Pseudocode for the CombFold algorithm, when only the *optimal MFE combination* is searched for.

V^c minimises over the four possible elementary structures, and also stores dependency information in the array V^c_{ds}. Once this array is filled for all b_i, b_j, i, j, it can be used to find the multi-domains for each potential combination ending in b_j, j. The function *Compute-W*^{c} fills the array W^c_{ds}, which can then be straightforwardly used to create the array W'_{ds}. The value W'_{ds}.free_energy will be the lowest MFE of the given *Input-Set*. The combination C and the MFE structure R can be extracted by backtracking through the V^c_{ds} array. Procedure 10 gives the resursive backtracking procedure, which is very close to the

CombFold-Backtrack Procedure

input: Indexes b_i, b_j, i, j, data structures V^c_{ds}, W^c_{ds};
output: Partially filled secondary structure R, partial combination C;

```
procedure CombFold-Backtrack
    if ($i > j$)
        return ($\emptyset, \emptyset$);
    else if ($V^c_{ds}(b_i, b_j, i, j)$.type = HAIRPIN_LOOP)
        Save-Structure ($R$);
        Save-Combination ($C$);
        return $R, C$;
    else if ($V^c_{ds}(b_i, b_j, i, j)$.type = STACKED_LOOP)
        Save-Structure ($R$);
        Save-Combination ($C$);
        return CombFold-Backtrack ($b_{i+1}, b_{j-1}, i+1, j-1$, $V^c_{ds}$, $W^c_{ds}$);
    else if ($V^c_{ds}(b_i, b_j, i, j)$.type = INTERNAL_LOOP)
        Save-Structure ($R$);
        Save-Combination ($C$);
        return CombFold-Backtrack ($V^c_{ds}(i,j).b_{i'}, V^c_{ds}(i,j).b_{j'}, V^c_{ds}(i,j).i', V^c_{ds}(i,j).j', V^c_{ds}, W^c_{ds}$);
    else if ($V^c_{ds}(b_i, b_j, i, j)$.type = MULTI_LOOP)
        Save-Structure ($R$);
        Save-Combination ($C$);
        for each branch $B$
            return CombFold-Backtrack ($B.b_i, B.b_j, B.i, B.j$, $V^c_{ds}$, $W^c_{ds}$);
        end for;
    end if;
end procedure CombFold-Backtrack.
```

Procedure 10: Pseudocode for the backtracking algorithm for *CombFold-optimal*.

corresponding procedure for *SimFold*, given in Procedure 4 in Chapter 4.

Implementation of the k-suboptimal MFE combination problem

Procedure 11 shows the pseudocode for the *k-suboptimal MFE combination problem* which was described in Section 6.2. The input to the procedure *CombFold-k-suboptimal* is the *Input-Set* and k. The output is comprised of three vectors of size k, for the best k free energies, along with their associated secondary structures and combinations. To find the optimal combination, first the *Combfold-optimal* procedure is called. To find the next $k-1$ combinations, three stacks for candidate free energies and their associated structures and combinations are used. First, they are empty, and a resursive procedure, called *Subopt-recursion*, is called. Its pseudocode is described in Procedure 12.

The input to the *Subopt-recursion* procedure is an *Input-Set IS* to partition, the remaining number of combinations k, the parent combination C_p, which will be used for partitioning of IS, and the three stacks which keep the candidate solutions. The output

CombFold-k-suboptimal Procedure

input: RNA or DNA *Input-Set* IS, number of suboptimal combinations k;
output: vectors $\Delta G_{1:k}$, $R_{1:k}$ and $C_{1:k}$ with the k best free energies,
secondary structures and combinations;

```
procedure CombFold-k-suboptimal
    $(\Delta G_1, R_1, C_1) :=$ CombFold-optimal $(IS)$;
    $\Delta G$-stack $:= \emptyset$;
    $R$-stack $:= \emptyset$;
    $C$-stack $:= \emptyset$;
    $(\Delta G_{2:k}, R_{2:k}, C_{2:k}) :=$ Subopt-recursion $(IS, k-1, C_1, \Delta G\text{-stack}, R\text{-stack}, C\text{-stack})$;
    return $\Delta G_{1:k}, R_{1:k}, C_{1:k}$;
end procedure CombFold-k-suboptimal.
```

Procedure 11: Pseudocode for the *k-suboptimal MFE combinations* problem.

is three vectors containing the best combinations found, and their MFE and structures. The recursive procedure stops when there are no more suboptimal combinations to look for ($k = 0$), in which case it returns empty sets. If $k > 0$, first we partition the input set IS using C_p, as described in Section 6.2. A number of s new *Input-Sets* $IS'_1, \ldots, IS'_s$ will be created, and the optimal MFE combination will be found. The free energies, structures and combinations are added to the corresponding stacks.

The smallest MFE out of all MFEs in ΔG-stack will be the the MFE of the next best combination. The corresponding structure and combination are stored, and then the procedure is called recursively, continuing from the last best combination found.

Note that, in total, at most $s \cdot k$ calls to *CombFold-optimal* will be performed, and at most $(s-1) \cdot k$ elements will exist on each of the three stacks. However, some of the already calculated 4-dimensional arrays V^c_{ds} and WM^c_{ds} can be used for the children. Consider the parent *Input-Set* $IS^{(1)} = \{S^{(1)}_1 \ldots S^{(1)}_s\}$ and consider that V^c_{ds} and WM^c_{ds} are calculated for this *Input-Set*. Consider the child partition $IS^{(2)i} = \{\{w^{(1)}_{1C_1}\}, \ldots, S^{(1)}_i - \{w^{(1)}_{iC_i}\}, S^{(1)}_{i+1}, \ldots, S^{(1)}_s\}$. The V^c_{ds} and WM^c_{ds} values for $S^{(1)}_{i+1}, \ldots, S^{(1)}_s$ will be the same, hence they can be reused for the children. This is an important save on running time and space, which is implemented in the current version of *CombFold*.

In order to reuse parts of the already calculated data structures, the data structures corresponding to all nodes of the tree must be kept in memory. This implies need for much space, which increases with k. A trick for saving some of this space, which is not implemented yet, would be to keep only the k lowest MFE values in the stack ΔG-stack. There is no need to keep the next values, since they will never become one of the best k combinations. The nodes in the tree corresponding to the values $k + 1$ and beyond can be released from memory.

Subopt-recursion Procedure

input: RNA or DNA *Input-Set* IS, number of suboptimal combinations k,
the parent combination C_p, stacks ΔG-stack, R-stack, C-stack;
output: vectors $\Delta G_{1:k}, R_{1:k}, C_{1:k}$ with the k best free energies,
secondary structures and combinations;

```
procedure Subopt-recursion
    if ($k = 0$)
        return ($\emptyset, \emptyset, \emptyset$);
    end if;
    $(IS'_1, \ldots, IS'_s)$ = Partition-input-set $(IS, C_p)$;
    for ($i := 1$ to $s$)
        $(\Delta G'_i, R'_i, C'_i)$ := CombFold-optimal $(IS'_i)$;
    end for;
    add $\Delta G'_{1:s}$ to $\Delta G$-stack;
    add $R'_{1:s}$ to $R$-stack;
    add $C'_{1:s}$ to $C$-stack;
    $\Delta G_1 := \min_i \Delta G\text{-stack}_i$;
    pos := get position of $\Delta G_1$ in $\Delta G$-stack;
    $R_1 := R\text{-stack}_{pos}$;
    $C_1 := C\text{-stack}_{pos}$;
    remove $\Delta G_1$ from $\Delta G$-stack;
    remove $R_1$ from $R$-stack;
    remove $C_1$ from $C$-stack;
    $(\Delta G_{2:k}, R_{2:k}, C_{2:k})$ := Subopt-recursion $(IS'_{pos}, k-1, C_1, \Delta G\text{-stack}, R\text{-stack}, C\text{-stack})$;
    return $\Delta G_{1:k}, R_{1:k}, C_{1:k}$;
end procedure Subopt-recursion.
```

Procedure 12: Pseudocode for the recursive function to find the next best combination in the *k-suboptimal MFE combinations* problem.

6.4 Time and space complexity

Theoretical analysis

Extending the $O(n^3)$ algorithm for secondary structure prediction of single nucleic acid molecules, the *optimal MFE combination* algorithm traverses the *Input-Set* in the same way, but for each position i and j, several possibilities might exist. We consider that the number of words g_i in each set S_i is limited by a constant bound g_{max}, and we measure the complexity in terms of the combinations length: $n = l_1 + l_2 + \ldots + l_s$. Also, we consider that the ranges returned by the X function is bounded by a constant and will be omitted from the theoretical analysis. In practice, the number of words in each set, the number of sets, the length of the words in each set, as well as the nucleotides composing the set, all have an impact on the run time. First we give an analysis of the theoretical complexity, and later in this section we will analyse the *CombFold* implementation on several specific

Input-Sets.

The theoretical time complexity of calculating each array described in Section 6.1 in the worst case follows:

- W': $O(g_{max} \cdot n)$, because for each j calculated in W^c, we minimize over all possible words of j: g_{max}, and $j = 1..n$;

- W^c: $O(g_{max}^4 \cdot n^2)$, because for each $j, 1 \leq j \leq n$ there are at most g_{max} possibilities, and we minimize over i. The maximum number of options for the b's of i and i, j's neighbours is 4 (third line in the equations for W^c) but b_{i-1}, b_i and b_{i+1} can only be in different words if the length of the word if $l(s(i))$ is 1. If this is true, then $g(s(i))$ is maximum 4 (because there are 4 different nucleotides), no matter what the values of g_{max} is. We consider this case as being constant;

- V^c: $O(g_{max}^2 \cdot n^2)$, because for each i and j, we minimize over a constant number of terms, and for each i and j there are at most g_{max} possibilities;

- S^c: $O(g_{max}^4 \cdot n^2)$, because for each i, j and their corresponding b_i and b_j, we minimize over potential different values for b_{i+1} and b_{j-1};

- H^c: $O(g_{max}^4 \cdot n^2)$, because for each i, j and their corresponding b_i and b_j, the term which has the greatest complexity has minimization over 4 terms, but 2 of them happen only if the word length is 1, so they are reduced to constant times;

- VBI^c: $O(g_{max}^8 \cdot n^4)$, but the same as for the previous algorithms, we assume the internal loops do not have more than c bases on each side between the branches (i. e. $c = 30$), and thus the complexity for internal loops becomes $O(g_{max}^8 \cdot n^2)$. The power of 8 comes from the most general case of internal loops;

- WM^c: $O(g_{max}^4 \cdot n^3)$, because the most costly branch of the WM^c calculation for each i and j is to find the best h for multi-loop partitioning. Each of i, j and h are in at most g_{max} words;

- VM^c: $O(g_{max}^4 \cdot n^2)$, because for each $i, j = 1..n$, which go over at most g_{max} possible words, we do a constant number of comparisons.

Thus, if we consider both g_{max} and n in our analysis, the worst case time complexity is $O(g_{max}^4 \cdot n^3 + g_{max}^8 \cdot n^2)$, because depending on the input, any of the two terms may be greater. In practice, g_{max} is often considered a constant, which leads to complexity proportional to n^3. The arrays W', W^c, V^c and WM^c need to be stored in memory. The space complexity is $O(g_{max}^2 \cdot n^2)$, or $O(n^2)$ if we consider g_{max} a constant.

The worst theoretical time complexity of the k-suboptimal MFE combinations problem is $O(s \cdot k \cdot g_{max}^4 \cdot n^3 + s \cdot k \cdot g_{max}^8 \cdot n^2)$ and the worst space complexity is $O(s \cdot k \cdot b_{max}^2 \cdot n^2)$. However, in practice, some of the *Input-Sets* after partitioning become empty. Also, as described earlier in Section 6.3, parts of the arrays and space can be reused.

Empirical analysis

We compared the running time performance of *CombFold* with that of *ExhaustS*, a simple (exponential time) exhaustive search algorithm, which creates all possible combinations and runs *SimFold* on each of them. For *Input-Sets* with a small number of combinations, it is expected that *CombFold* takes more time and space than *ExhaustS*, because of the increased complexity. However, although the space is not a problem for *ExhaustS*, the running time quickly grows and becomes impractical.

Figure 6.2 gives the run time performance of *CombFold* with $k = 1, 2, 3, 10$ and *ExhaustS* on randomly generated *Input-Sets* of different characteristics. All the tests have been performed on machines with CPU Pentium III 733 MHz, memory cache 256 KB and RAM memory 1GB, running Linux 2.4.20. All graphs show the CPU time in seconds, presented on a log scale, versus variation of different characteristics of the *Input-Sets.* To simplify the analysis, we chose $g_1 = \ldots = g_s = g$ and $l_1 = \ldots = l_s = l$, and we took variations of s, g and l. Having all set sizes equal and all set lengths equal, the number of combinations will be g^s, and the length of the combinations will be $l \cdot s$.

The graph in (a) shows a comparison between the running time of *CombFold* with $k = 1, 2, 3, 10$ and *ExhaustS*, on a set of 19 instances having g and l fixed at 2 and 10, respectively. The number of sets s varies from 1 to 19, yielding $2^1 = 1$ combination of length 10 to $2^{19} \approx 0.5 \cdot 10^6$ combinations of length 190.*CombFold* with $k = 1$ becomes faster than *ExhaustS* at $s = 8$, with $k = 2$ and 3 becomes faster at $s = 10$, and *CombFold* with $k = 10$ becomes faster at $s = 12$. Note that the slope of the curves suggest that *CombFold* grows polynomially, while *ExhaustS* grows exponentially.

The graph in (b) shows a similar situation as in graph (a), but when g is fixed at 3 rather then 2. $l = 10$ and s takes values in the range 1 to 12, leading to $3^1 = 3$ combinations of length 10 to $3^{12} \approx 0.5 \cdot 10^6$ combinations of length 120. The number of combinations being bigger for the same s, *CombFold* with $k = 1$ outperforms *ExhaustS* when $s = 6$, with $k = 2$ and 3 when $s = 7$, and with $k = 10$ when $s = 8$.

Graph (c) shows a comparison when s and l are fixed to 6 and 10 respectively, but g varies from 1 to 13. These yield $1^6 = 1$ to $13^6 \approx 4.8 \cdot 10^6$ combinations of length 60. Note that in this case *ExhaustS* grows polynomially, as opposed to cases (a) and (b). This happens because g is increasing and g is the base of the formula which gives the number of combinations: g^s. However, *ExhaustS* grows more quickly than *CombFold.* Indeed, the graph shows that *CombFold* with $k = 1$ becomes faster than the *ExhaustS* when $g = 3$, with $k = 2$ and 3 when $g = 4$ and with $k = 10$ when $g = 5$.

Graph (d) gives the comparison when s and g are fixed to 8 and 2, respectively, leading to a fixed number of $2^8 = 256$ combinations. However, the length of the words vary from 10 to 100, yielding combinations of length 80 to 800. Again, *ExhaustS* grows more quickly, but still polynomially, only the length of the combinations being changed. *ExhaustS* is outperformed by *CombFold*($k = 1$) at $l = 10$ and by *CombFold*($k = 2$) at $l = 50$. On the instances we tested, *ExhaustS* outperforms *CombFold* with $k = 10$, and becomes roughly

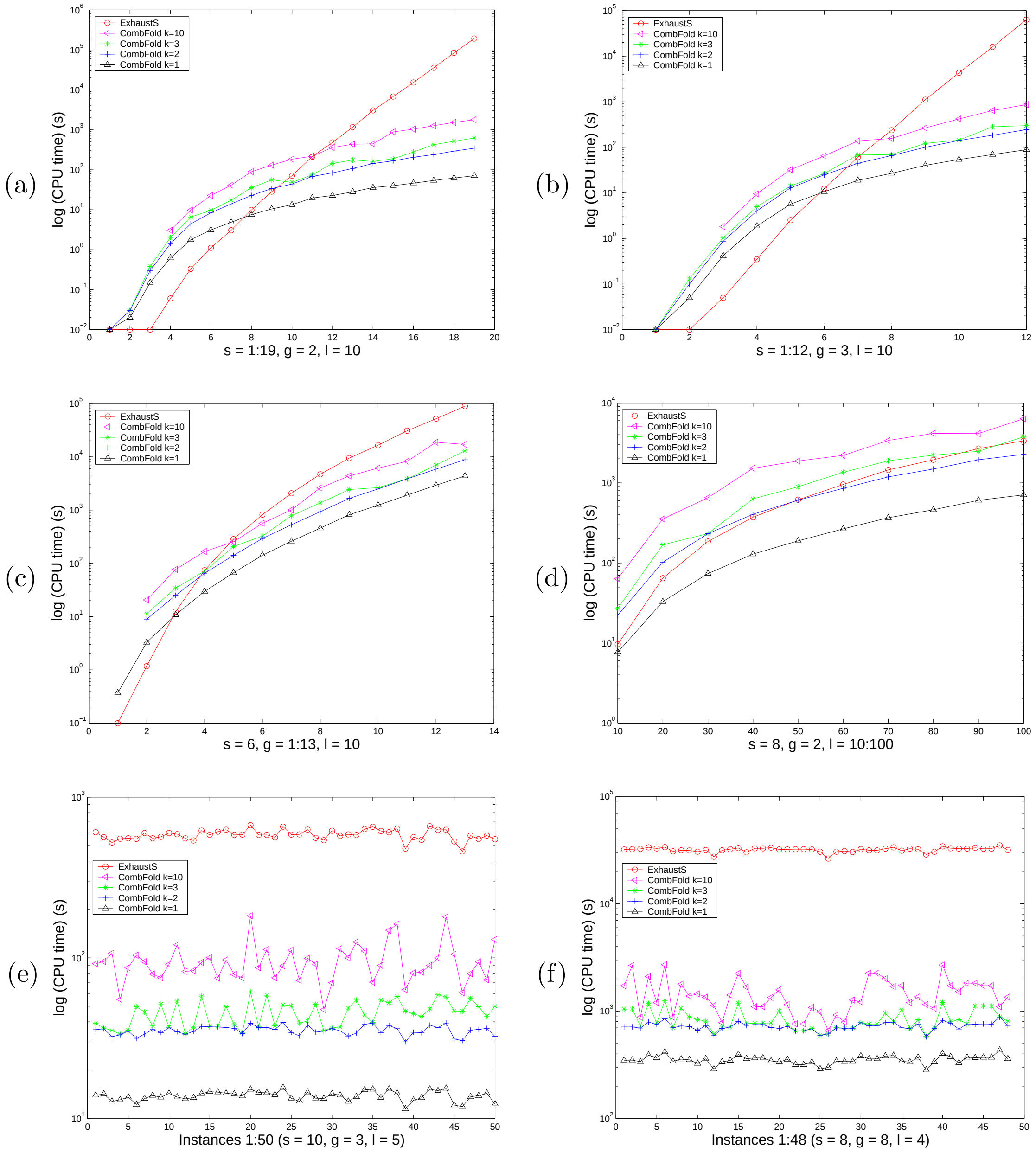


Figure 6.2: Performance of *CombFold* with $k = 1, 2, 3, 10$ and *ExhaustS*, on sets with different characteristics: (a) 19 instances with s ranging from 1 to 19, and the same $g = 2$ and $l = 10$; (b) 12 instances with s ranging from 1 to 12, and the same $g = 3$ and $l = 10$; (c) 13 instances with g ranging from 1 to 13, and the same $s = 6$ and $l = 10$; (d) 10 instances with l ranging from 10 to 100, and the same $s = 8$ and $g = 2$; (e) 50 instances with the same characteristics: $s = 10, g = 3, l = 5$; (f) 48 instances with the same characteristics: $s = 8, g = 8, l = 4$.

the same speed as *CombFold* with $k = 3$ when $l = 100$.

On all these four graphs, we note that *CombFold* with $k = 1$ and 2, and *ExhaustS* are nicely curved, while *CombFold* with $k = 3$ and 10 have "hills" and "valleys". To see how the curves look like, we created two sets of 50 instances of *Input-Sets* with exactly the same characteristics: graph (e) with $s = 10, g = 3, l = 5$ and graph (f) with $s = 8, g = 8, l = 4$. The results comfirm the explanation we gave earlier in Section 6.2: When $k = 1$, *CombFold* fills all the arrays, a small variation happening due to the distribution of the nucleotides in the words. When $k = 2$, the arrays for s more sets are always calculated, no matter what the optimal combination is. However, depending on which the second best combination is, the size of the next *Input-Sets* that partition the solutions space can differ substantially. This influence propagates on to the next best combinations, such that when $k = 10$, the differences in time between different instances can vary substantially. Also, note that for some instances, the time for $k = 3$, and even for $k = 10$, is very close or equal to the time for $k = 2$. This means that the second best combination was part of a very small *Input-Set*, which was partitioned in fewer (or even 0) non-empy *Input-Sets*. The graphs also show the run time of the exponential algorithm. For graph (e) there are $3^{10} \approx 60,000$ combinations of length 50, and *ExhaustS* is more than one order of magnitude slower than *CombFold* with $k = 1$, and 5-6 times slower than *CombFold* with $k = 10$. For graph (f), where the number of combinations is $8^8 \approx 16.8 \cdot 10^6$ of length 32, the exponential algorithm is substantially slower, being about two orders of magnitude slower than *CombFold*($k = 1$), and more than one order of magnitude slower than *CombFold*($k = 10$).

6.5 Applications and experimental results

CombFold can be used for at least two important applications: (1) directed mutagenesis and SELEX experiments [38] and (2) for testing whether DNA words used in DNA computations fold without secondary structures [8, 9, 10].

6.5.1 Directed mutagenesis and SELEX experiments

Directed mutagenesis and SELEX experiments are biochemical experiments where variations of an RNA or DNA sequence are used to perform analysis on a library of sequences and study whether simple mutations of them have some better properties than others. The input sequences, which we call *IUPAC-Input*, are typically regular expressions, composed of characters (DNA or RNA nucleotides) and wild cards, which can be replaced by several different characters. A conventional coding for the wild cards is to use "IUPAC" format (International Union of Pure and Applied Chemistry), which contains the following codes:

```
R = G A (purine)          B = G T C
Y = T C (pyrimidine)      D = G A T
K = G T (keto)            H = A C T
M = A C (amino)           V = G C A
S = G C                   N = A G C T (any)
```

W = A T

Thus, if the input sequence for a SELEX experiment is AURCAAUGCSNAUGCSNAUGCAC, then we can convert it into an *Input-Set* in straightforward way: for each different wild card, we create a different set:

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9
IUPAC-Input	AU	R	CAAUGC	S	N	AUGC	S	N	AUGCAC
Input-Set	AU	G	CAAUGC	G	A	AUGC	G	A	AUGCAC
	A		C	G		C	G		
				C			C		
				U			U		

Thus, we can very easily transform the *IUPAC-Input* format into an *Input-Set* and use *CombFold* to predict which nucleotides replacing the wild cards would determine the sequences to fold into the most stable configurations. Collaboration with researchers doing practical directed mutagenesis or SELEX experiments would be very valuable for testing and improving our tool.

6.5.2 DNA computing and tag - anti-tag libraries

In search-and-prune DNA or RNA computations [8, 9, 14] or in tag - anti-tag libraries [10], sets of short DNA or RNA words are used to create many long DNA strands. These sets are carefully designed using computational or information-theoretic methods [56, 55], such that the resulting long strands behave well in computation. The main property for "good behaviour" of these long strands is that they do not form secondary structure (in other words, they are *structure free*), thus begin available to hibridyze with probes.

We have collected five *Input-Sets* from the DNA computing literature and we used *CombFold* to test whether they are indeed structure free:

1. Braich et al. [8] used an *Input-Set* of 6 sets of 2 15-mer DNA words each, to solve a 6-variable 11-clause 3-SAT problem with a gel-based DNA computer. Note that there are $2^6 = 64$ combinations of length $15 \cdot 6 = 90$. In our tests, we call this set *IS-Braich-2001*;

2. Braich et al. [9] used an *Input-Set* of 20 sets of 2 15-mer DNA words each, to solve a 20 variable 24-clause 3-SAT problem with a DNA computer. There are $2^{20} \approx 10^6$ combinations, of length 300. In our tests, we call this set *IS-Braich-2002.*

3. Brenner et al. [10] used an *Input-Set* of 8 sets of 8 4-mer DNA words each, to clone DNA molecules onto microbead surfaces. There are $8^8 \approx 16.7 \cdot 10^6$ combinations of length $4 \cdot 8 = 32$ generated by this set. We call this set *IS-Brenner-2000*;

No	*Input-Set* (source)	NA	optimal MFE (kcal/mol)	CPU time CombFold (s)	CPU time ExhaustS (s)
1	*IS-Braich-2001*[8]	DNA	-0.21	3.58	1.13
2	*IS-Braich-2002*[9]	DNA	0	63.68	≈ 5 days
3	*IS-Brenner-2000*[10]	DNA	-3.19	187.05	15,520.08
4	*IS-Faulhammer-2000*[14]	RNA	-2.90	18.97	157.03
5	*IS-Frutos-1997-r2*[17]	DNA	-12.26	414.65	16.68

Table 6.3: Output of CombFold on some published combinatorial sets for structure freeness.

No	Sequence - structure

1	`TATTCTCACCCATAAACACTATCAACATCACCTTTACCTCAATAAATCTTTAAATACCCCTCCATTT`
	`...................................................(((((...........`
	`CTCCATATTTTCTTCCATCACAT`
	`.....))))).............`

3	`CAAAAATCCTTTTTACCAAAAATCCTTTTTAC`
	`.((((.....))))...((((.....))))..`

4	`CTCTTACTCAATTCTTCTACCATATCAACATCTTAATAACATCCTCCACTTCACACTTAATTAAAAT`
	`............................................................(((((((`
	`CTTCCCTCTTTACACCTTACTTTCCATATACAAGTACATTCTCCCTACTCCTTCATAATCTTATATT`
	`.................(((((.........))))).......................((((((.`
	`CTCAATATAATCACATACTTCTCCAACATTCCTTATCCCACACACATTTTAAATTTCACAA`
	`....)))))).................................)))))))).........`

5	`AACGTACGTCCTGCAATTCGATGCAGGACGTT`
	`.....((((((((((......)))))))))).`

Table 6.4: The optimal MFE combinations (sequences and structures), as predicted by *CombFold*, for the *Input-Sets* in Table 6.5.2.

4. Faulhammer et al. [14] used an *Input-Set* of 19 sets. 10 of these have 2 15 mer words and other 9 sets having 1 word each of length 5 are used as spacers between the 10 2-word sets. They used this set for solving a variant of a "Knight problem" which finds in which configurations of knights placed on a 3×3 chess board no knight attacks any other knight. We call this set *IS-Faulhammer-2000*;

5. Frutos et al. [17] proposed a set of 108 8-mer DNA words to be used on DNA computing on surfaces. Several such sets of 108 words can be separated by different 4-mer spacers, in the configuration ((spacer) (108-word set) (spacer))^r, where r denotes the number of times the three sets are repeated (note that the spacers are different for each repeat, and the 108-word set is always the same). We tested the set for $r = 2$ ($108^2 = 11,664$ 16-mer combinations) and we call it *IS-Frutos-1997-r2*.

Table 6.5.2 shows the results of our tests on the five described sets. Column 3 shows the nucleic acid type (DNA or RNA) that was used by the authors. Our runs were performed on the same type. Column 4 shows the optimal MFE obtained for each of the sets. Columns

4 and 5 show the CPU time in seconds that *CombFold* with k=1 and *ExhaustS*, respectively, spent to solve each problem instance. The parameter used for the temperature was set to 37°C. The tests were performed on machines with CPU Pentium III 1GHz, 256 KB cache and 1GB RAM.

Our predictions show that the set *IS-Braich-2001* has a slightly negative free energy and the set *IS-Braich-2002* is indeed structure free. Note that, while for the smaller set *ExhaustS* is faster, for the larger set, it is substantially slower than *CombFold* ($\approx$ 5 days versus $\approx$ 1 minute).

The remaining three sets are predicted to have structure for the combinations indicated in Table 6.4. For the set from Brenner et al., the CPU time of *ExhaustS* is substantially slower than the time spent by *CombFold*. For the last two sequences, the times are not substantially different.

In conclusion, the predictions obtained with *CombFold* show that all of the sets except *IS-Frutos-1997-r2* have free energy close to 0, indicating the sets are well designed.

Chapter 7

Conclusions and Future Work

In this thesis we have introduced algorithms for secondary structure prediction of pairs of nucleic acid molecules and for finding the best combinations out of a combinatorial set of strands. First, a thorough description of the model used by these algorithms was provided in Chapter 3. Second, understanding the free energy minimization algorithm for secondary structure prediction of single nucleic acid molecules was necessary in order to follow the extensions that we propose. We gave a detailed description about this algorithm and our implementation, *SimFold*, and we analyse it in comparison to other implementations of the same algorithm and on biological data, in Chapter 4. Then, in Chapters 5 and 6, we described in detail our algorithms: *PairFold* and *CombFold*, and we analysed their complexity, performance and accuracy on biological data. We showed concrete examples on which our algorithms can provide useful information.

Our work can be continued in several directions:

- First of all, the free energy minimization algorithm for secondary structure predictions is based on a simplified model and set of thermodynamic parameters. We have shown that the prediction accuracy goes down with the sequence length. Improving the free energy model to give better prediction accuracy would be, in our opinion, the first and most important step for future work;

- It has been shown that predicting suboptimal structures for single molecules improved the overall accuracy of the free energy minimization algorithm [33, 66]. Incorporating suboptimal folding into our algorithms would provide more insight on nucleic acid folding; Also, partition function calculation for base pairing probabilities [34], as well as algorithms for pseudoknotted structures [40] could be incorporated into our algorithms;

- Another important way to extend our work is to combine *PairFold* and *CombFold* in one application, call *PairCombFold*, to predict the most stable duplexes of two combinatorial sets of strands. This would be useful for at least three reasons:

  - predicting interactions between a pair of molecules, when at least one molecule is part of a library obtained by mutagenesis. Examples include libaries of ribozymes, where fragments of the ribozyme sequences (and sometimes the targets also) are combinatorial sets [68]. Predicting which of the randomized pairs bind more strongly to each other can provide insight on ribozyme behaviour;
  - predicting binding site of a probe with a combinatorial set of molecules used in DNA computing [8, 9, 10]. We showed that *CombFold* can be used to test whether sets of DNA words concatenate without secondary structures. The next step for problems such as 3-SAT solvers using DNA computing is to make sure that the probes bind to the right target. A combination of *PairFold* and *CombFold* would predict which is the most stable binding and which are are next k ones;
  - better designing probes to bind with targets. The OligoWalk program [32] finds the best probe to bind with a target, but only considers probes that are complementary to windows of the target. With *PairCombfold*, one could consider the whole combinatorial set of probes.

- We have performed analysis of *PairFold* on biological pairs of molecules. However, the literature of the last ten years includes many more such examples, especially on ribozyme - mRNA target complexes, on which *PairFold* can be tested. Moreover, creating a database of RNA pairs of molecules with known structures and using it to improve the prediction would be a very interesting future step;

- We have shown that our algorithm *CombFold* has polynomial time and space complexity. However, in practice, it can become impractical for large sets of inputs. Optimizing the algorithm and its implementation further would make it more efficient. A few optimization tricks have been proposed by Cohen and Skiena [12] and have not been applied here yet. Other optimizations, for more efficient data structures, can be useful;

- We reported *CombFold* run times on several types of instances. However, predicting the *CombFold* run time on a given *Input-Set* is not trivial, especially when the number of words in each set and the length of the words across sets vary. Predicting this time for *CombFold* with $k = 1$ and $k = 2$ would be valuable.

In conclusion, we contributed with two important algorithms and practical tools to the scientific community of computational and molecular biology. Further collaboration with researchers doing work related to what we propose will be crucial in finding benefit in their applicability and in improving their accuracy.

Bibliography

[1] H. T. Allawi and J. SantaLucia Jr., *Thermodyamics and NMR of Internal G-T Mismatches in DNA*, Biochemistry (1997) 36, 10581-10594.

[2] H. T. Allawi and J. SantaLucia Jr., *Thermodynamics of internal C·T mismatches in DNA*, Nucl. Acids. Res. (1998), Vol. 26, No. 11.

[3] H. T. Allawi and J. SantaLucia Jr., *Nearest-Neighbor Thermodynamics of Internal A·C Mismatches in DNA: Sequence Dependence and pH Effects*, Biochemistry (1998), 37, 9435-9444.

[4] H. T. Allawi and J. SantaLucia Jr., *Nearest Neighbor Thermodynamic Parameters for Internal G·A Mismatches in DNA*, Biochemistry (1998), 37, 2170-2179

[5] M. Andronescu, R. Aguirre-Hernandez, A. Condon, H. Hoos, *RNAsoft: a suite of RNA secondary structure prediction and design software tools*, Nucl. Acids. Res. (2003) 31: 3416-3422.

[6] M. Andronescu, D. Dees, L. Slaybaugh, Y. Zhao, B. Cohen, A. Condon, and S. Skiena, *Algorithms for testing that sets of DNA words concatenate without secondary structure*, Proc. 8th International Workshop on DNA Based Computers; LNCS (2003) 2568: 182-195. A journal version will appear in Natural Computing (in press).

Fifth edition, McGraw-Hill Inc (1988).

[7] S. Bommarito, N. Peyret and J. SantaLucia Jr., *Thermodynamic parameters for DNA sequences with dangling ends*, Nucl. Acids. Res. (2000) 28: 1929-1934.

[8] R. S. Braich, C. Johnson, P. W. K. Rothemund D. Hwuang, N. Chelyapov and L. M. Adleman, *Solution of a satisfiability problem on a gel-based DNA computer*, Proc. of 6th Intl. Conf. on DNA Computation; Springer-Verlag LNCS (2001), 2045: 27-41.

[9] R. S. Braich, N. Chelyapov, C. Johnson, P. W. Rothemund and L. Adleman, *Solution of a 20-variable 3-SAT problem on a DNA computer*, Science (2002); 296 (5567): 499-502.

[10] S. Brenner, S. R. Williams, E. H. Vermaas, T. Storck, K. Moon, C. McCollum, J. I. Mao, S. Luo S, J. J. Kirchner, S. Eletr, R. B. DuBridge, T. Burcham and G. Albrecht, *In vitro cloning of complex mixtures of DNA on microbeads: physical separation of differentially expressed cDNAs*, Proc. Natl. Acad. Sci. U S A. (2000); 97 (4): 1665-1670.

[11] J. J. Cannone, S. Subramanian, M. N. Schnare, J. R. Collett, L. M. D'Souza, Y. Du, B. Feng, N. Lin, L. V. Madabusi, K. M. Muller, N. Pande, Z. Shang, N. Yu and R. R. Gutell, *The Comparative RNA Web (CRW) Site: An Online Database of Comparative Sequence and Structure Information for Ribosomal, Intron, and other RNAs.*, BioMed Central Bioinformatics (2002), 3:2. [Correction: BioMed Central Bioinformatics. 3:15.]

[12] B. Cohen and S. Skiena, *Designing RNA sequences: natural and artificial selection*, Proc. 6th Int. Conf. Computational Molecular Biology (RECOMB) (2002), pp. 109-116.

[13] R. M. Dirks and N. A. Pierce, *A partition function algorithm for nucleic acid secondary structure including pseudoknots*, J. Comput. Chem. (2003); 24 (13): 1664-1677.

[14] D. Faulhammer, A. R. Cukras, R. J. Lipton and L. F. Landweber, *Molecular computation: RNA solutions to chess problems*, Proc. Natl. Acad. Sci. (2000); 97 (4): 1385-1389.

[15] D. S. Fields and R. R. Gutell, *An analysis of large rRNA sequences folded by a thermodynamic method*, Folding & Design (1996) 1: 419-430.

[16] S. M. Freier, R. Kierzek, J. A. Jaeger, N. Sugimoto, M. H. Caruthers, T. Neilson and D. H. Turner, *Improved free-energy parameters for predictions of RNA duplex stability*, Proc. Natl. Acad. Sci. U S A. (1986), 83 (24): 9373-9377.

[17] A. G. Frutos, Q. Liu, A. J. Thiel, A. M. Sanner, A. E. Condon, L. M. Smith and R. M. Corn, *Demonstration of a word design strategy for DNA computing on surfaces*, Nucl. Acids. Res. (1997); 25 (23): 4748-4757.

[18] M. Gouy, *Secondary structure prediction of RNA, Nucleic Acid and protein sequence analysis*, Bishop MJ & Rawlings CJ, IRL Press, Washington (1987).

[19] A. J. F. Griffiths, W. M. Gelbart, R. C. Lewontin and J. H. Miller, *Modern genetic analysis: integrating genes and genomes*, 2nd edition, W. H. Freeman and Company, 2002.

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

[21] Gutell Lab Comparative RNA Web Site, `http://www.rna.icmb.utexas.edu`.

[22] R. R. Gutell, J. C. Lee and J. J. Cannone, *The accuracy of ribosomal RNA comparative structure models*, Current Opinion in Structural Biology 2002, 12:301-310.

[23] I. L. Hofacker, M. Fekete, P. F. Stadler, *Secondary Structure Prediction for Aligned RNA Sequences*, J.Mol.Biol. (2002) 319, 1059-1066.

[24] I. L. Hofacker, W. Fontana, P. F. Stadler, L. S. Bonhoeffer, M. Tacker and P. Schuster, *Fast Folding and Comparison of RNA Secondary Structures*, Chemical Monthly (1994) 125: 167-188.

[25] R. Hormes, M. Homann, I. Oelze, P. Marschall, M. Tabler, F. Eckstein and G. Sczakiel, *The subcellular localization and length of hammerhead ribozymes determine efficacy in human cells*, Nucl. Acids. Res. (1997) 25: 769-775.

[26] HyTher - Hybridization Thermodynamics Web Page, `http://ozone2.chem.wayne.edu/Hyther/hythermenu.html`.

[27] Y. Kasai, H. Shizuku, Y. Takagi, M. Warashina and K. Taira, *Measurements of weak interactions between truncated substrates and a hammerhead ribozyme by competitive kinetic analyses: implications for the design of new and efficient ribozymes with high sequence specificity*, Nucl. Acids. Res. (2002) 30: 2383-2389.

[28] D. A. Konings and R. R. Gutell, *A comparison of thermodynamic foldings with comparatively derived structures of 16S and 16S-like rRNAs*, RNA (1995), 1(6):559-74

[29] F. Li, G. D. Stormo, *Selection of optimal DNA oligos for gene expression arrays*, Bioinformatics (2001); 17 (11): 1067-76.

[30] R. B. Lyngsø and C. N. Pedersen, *RNA pseudoknot prediction in energy-based models*, J Comput Biol. (2000); 7 (3-4): 409-27.

[31] R. B. Lyngsø, M. Zuker, C. N. Pedersen, *Fast evaluation of internal loops in RNA secondary structure prediction*, Bioinformatics (1999), 15 (6): 440-5.

[32] D. H. Mathews, M. E. Burkard, S. M. Freier, J. R. Wyatt and D. H. Turner, *Predicting oligonucleotide affinity to nucleic acid targets*, RNA (1999), 5: 1458-1469.

[33] D. H. Mathews, J. Sabina, M Zuker and D. H. Turner, *Expanded Sequence Dependence of Thermodynamic Parameters Improves Prediction of RNA Secondary Structure*, J. Mol. Biol. (1999) 288, 911-940.

[34] J. S. McCaskill, *The equilibrium partition function and base pair binding probabilities for RNA secondary structure*, Biopolymers (1990), Vol 29, 1105-1119.

[35] Mfold Web Server, `http://www.bioinfo.rpi.edu/applications/mfold`.

[36] U. Nagaswamy, M. Larios-Sanz, J. Hury, S. Collins, Z. Zhang and G. E. Fox, *NCIR: a database of non-canonical interactions found in known RNA structures*, Nucl. Acids. Res. (2002) 30: 395-397.

[37] N. Peyret, A. P. Seneviratne, H. T. Allawi and J. SantaLucia Jr., *Nearest-Neighbor Thermodynamics and NMR of DNA Sequences with Internal A·A, C·C, G·G and T·T Mismatches*, Biochemistry (1999), 38: 3468-3477.

[38] J. V. Ponomarenko, G. V. Orlova, A. S. Frolov, M. S. Gelfand and M. P. Ponomarenko, *SELEX_DB: a database on in vitro selected oligomers adapted for recognizing natural sites and for analyzing both SNPs and site-directed mutagenesis data*, Nucl. Acids. Res. (2002); 30 (1): 195-199.

[39] J. H. Reif, T. H. LaBean and N. C. Seeman, *Challenges and Applications for Self-Assembled DNA Nanostructures*, Proc. 6th International Workshop on DNA Based Computers; LNCS (2001) 2054: 173-198.

[40] E. Rivas and S. R. Eddy, *A dynamic programming algorithm for RNA structure prediction including pseudoknots*, J. Mol. Biol. (1999) 285, 2053-2068.

[41] RNAsoft - Software for RNA/DNA secondary structure prediction and design, `http://www.RNAsoft.ca`.

[42] RNAstructure Web Site, `http://128.151.176.70/RNAstructure.html`.

[43] P. B. Rupert and A. R. Ferre-d'Amare, *Crystal structure of a hairpin ribozyme - inhibitor complex with implications for catalysis*, Nature (2001) 410.

[44] SantaLucia Laboratory, `http://ozone.chem.wayne.edu/`.

[45] J. SantaLucia Jr., *A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics*, Proc. Natl. Acad. Sci. USA, Vol. 95, pp 1460-1465; Biochemistry (1998).

[46] J. SantaLucia Jr. and D. H. Turner, *Measuring the Thermodynamics of RNA Secondary Structure Formation*, John Wiley & Sons, Inc. Biopoly (1997) 44: 309-319.

[47] C. Schmidt, R. Welz and S. Mller, *RNA double cleavage by a hairpin-derived twin ribozyme*, Nucl. Acids. Res. (2000) 28: 886-894.

[48] R. Schroeder, R. Grossberger, A. Pichler and C. Waldsich, *RNA folding in vivo*, Current Opinion in Structural Biology (2002), 12:296-300.

[49] M. J. Serra, D. H. Turner and S. M. Freier, *Predicting thermodynamic properties of RNA*, Meth. Enzymol. (1995) 259, 243-261.

[50] M. R.Shortreed, S. B. Chang, M. Andronescu, D. C. Tulpan, H. Hoos, A. Condon and L. M. Smith, *An algorithm for designing structure-free concatenated DNA word sets*, Proc. 9th International Workshop on DNA Based Computers (2003).

[51] M. Sprinzl, C. Horn, M. Brown, A. Ioudovitch and S. Steinberg, *Compilation of tRNA sequences and sequences of tRNA genes*, Nucl. Acids. Res. (1998) 26: 148-153.

[52] N. Sugimoto, S. Nakano, M. Katoh, A. Matsumura, H. Nakamuta, T. Ohmichi, M. Yoneyama and M. Sasaki, *Thermodynamic parameters to predict stability of RNA/DNA hybrid duplexes* Biochemistry (1995); 34 (35): 11211-11216.

[53] J. Tang and R. R. Breaker, *Mechanism for allosteric inhibition of an ATP-sensitive ribozyme*, Nucl. Acids. Res. (1998) 26: 4214-4221.

[54] P. A. Tipler, *Physics For Scientists and Engineers*, Third Edition, Worth Publishers (1991).

[55] D. C. Tulpan and H. H. Hoos, *Hybrid Randomised Neighbourhoods Improve Stochastic Local Search for DNA Code Design*, Canadian Conference on AI (2003), 418-433.

[56] D. C. Tulpan, H. H. Hoos and Anne Condon, *Stochastic Local Search Algorithms for DNA Word Design*, Proc. 8th International Workshop on DNA Based Computers (2002): 229-241.

[57] Turner Group Web Page, `http://128.151.176.70/`.

[58] D. H. Turner, N. Sugimoto, J. A. Jaeger, C. E. Longfellow, S. M. Freier and R. Kierzek, *Improved parameters for prediction of RNA structure*, Cold Spring Harb. Symp. Quant. Biol. (1987); 52: 123-33.

[59] D. H. Turner and N. Sugimoto, *RNA structure prediction*, Annu. Rev. Biophys. Biophys. Chem. (1988); 17: 167-92

[60] N. K. Vaish, A. R. Kore and F. Eckstein, *Recent developments in the hammerhead ribozyme field*, Nucl. Acids. Res. (1998) 26: 5237-5242.

[61] Vienna RNA Package, RNA Secondary Structure Prediction and Comparison `http://www.tbi.univie.ac.at/ ivo/RNA/`

[62] A. E. Walter amd D. H. Turner, *Sequence dependence of stability for coaxial stacking of RNA-Helixes with Watson-Crick base paired interfaces*, Biochemistry (1994), 33, 12715-12719.

[63] A. E. Walter, D. H. Turner, J. Kim, M. H. Lyttle, P. Muller, D. H. Mathews and M. Zuker, *Coaxial stacking of helixes enhances binding of oligoribonucleotides and improves predictions of RNA folding*, Proc. Natl. Acad. Sci. U S A. (1994); 91 (20): 9218-9222.

[64] J. G. Wetmur, *DNA Probes: Applications of the Principles of Nucleic Acid Hybridization*, Critical Reviews in Biochemistry and Molecular Biology (1991), 26 (3/4): 227-259.

[65] M. Wu, J. A. McDowell, D. H. Turner, *A Periodic Table of Tandem Mismatches in RNA* Biochemistry (1995); 34 (10); 3204-3211.

[66] S. Wuchty, W. Fontana, I. L. Hofacker and P. Schuster, *Complete suboptimal folding of RNA and the stability of secondary structures*, Biopolymers (1999); 49 (2): 145-65.

[67] T. Xia, J. SantaLucia Jr., M. E. Burkard, R. Kierzek, S. J. Schroeder, X. Jiao, C. Cox, and D. H. Turner, *Thermodynamic Parameters for an Expanded Nearest-Neighbor Model for Formation of RNA Duplexes with Watson-Crick Base Pairs*, Biochemistry (1998), 37: 14719-14735.

[68] Q. Yu, D. B. Pecchia, S. L. Kingsley, J. E. Heckman and H.M. Burke, *Cleavage of highly structured viral RNA molecules by combinatorial libraries of hairpin ribozymes*, Journal of Biological Chemistry (1998), 273 (36): 23524-23533.

[69] B. Yurke, A. J. Turberfield, A. P. Mills Jr, F. C. Simmel and J.L. Neumann, *A DNA-fuelled molecular machine made of DNA*, Nature (2000); 406 (6796): 605-608.

[70] M. Zuker, *Mfold web server for nucleic acid folding and hybridization prediction*, Nucl. Acids. Res. (2003) 31: 3406-3415.

[71] A. M. Zuker, B. D.H. Mathews and C. D.H. Turner, *Algorithms and Thermodynamics for RNA Secondary Structure Prediction: a Practical Guide.*

[72] M. Zuker and P. Stiegler, *Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information*, Nucl. Acids. Res. (1981) 9: 133-148.

Appendix A

SimFold analysis

Table A.1 shows 7 artificially created sequences, on which *SimFold* and *mfold* predictions differ, as well as the MFE structures predicted by *SimFold*. Details are discussed in Section 4.4.1.

No	Sequence and MFE structure returned by *SimFold*
1	CUCGUCUUGAAACGACAGAUCCAUAAGGUCACUCAGUGAUGAACCUGGGGACUACUUGAUGAGCGCUAUACCUAUAAUCAGCGAGCCCUGGAGUCGAUCC ............((((...((((...((((.(((((........))))))))).......(..((((............))))..)..))))))))....
2	UCGGCCGAGUUACUCGAAGGGAUUGUUGGGACCCAUGCCGAACAAUAUAUCGUCGAAAAAAGGGUCUAGUUAGCGCGGAACACUUGCCGGCCCGAUUGUU (((((((......((((....((((((.((.......)).))))))......))))...(((..(((.(....)..)))...)))..))).)))).....
3	GUGCCUGGCUGGGUCUUCAGCCCUUAGGUAUGAUCUUAGGGAAAAUGGUAAAGGUUCCUCGUCAGUGACAUAUGUCGCUCGACGCCAGCUUCUGGACGGU ((((((((((((....))))))...)))))).......(((((..(.....)..)))))((((((((((....)))))).))))((((...)))).....
4	AUCGACCUGCAAUGCAAUUAAUGCGGAUGGUUUACUACCUGAUCGGUCACGCUCGCUUGGUGAUGACAUGACCCCUAUCCCCCGACUAUAAGGUAUCUAA ....((((.....(((.....)))((((((....(.....)...(((((.((((((...)))).).).))))).))))))..........))))......
5	AUGCCUACAUGUAACCUUGCUAGGUGCCACUUUUUGGUAUAAACAGCUCUUUGACGUACUACAUUACCCCUAGAGAGAGGGACGCGGACAGCUUCGCCGC ........((((......(((..((((((.....))))))....)))......))))..........((((......))))..((((.(......)))))
6	CAGCAGGCAUCGUUUCUGUCGGCGUAUACCUGCGAAAAUUUCACAUGGCGUGGGGCGCAACACCGAGCACGGCUACGAAGCCCAAAGCGGCCCUUGUUCG ..(((((.((((((......)))).)).))))).............((((.(((.(((..(.....)...((((....))))....))).))).))))..
7	GCUAAGUAGAGUUCAGCAGAGGUUAGUUACAGAGCUAAGUGAUGUGGCGUAUGCAGUUAUGGAGCGAUAGUCAAGAACGUGCAGGAUAACGAAUUUAGUU (((((((...(((..(((..(.((((((....)))))).)..))))))...(((((((...((.(....)))...))).)))).........))))))).

Table A.1: 7 artificially created sequences, on which *SimFold* and *mfold* predictions differ.

Table A.2 presents comparative analysis between real structures, *SimFold* prediction and *RNAfold* prediction, on the first 30 RNA sequences in the tRNA genes database [51]. The entries are in the alphabetical order of the identification number used in the database (second column). Columns 1 and 2 show the organism and identification number and column 3 shows the length of the sequence. Column 4 gives the minimum free energy, measured in kcal/mol, returned by *SimFold*. Column 5 gives the free energy of the real structure, using the model used in *SimFold*. Columns 6 and 8 report the number of pairs predicted by *SimFold* and *RNAfold*, respectively, out of the total number of pairs in the real structure. Columns 7 and 9 give the level of accuracy performed by *SimFold* and *RNAfold*, measured with two parameters. More information is given in Section 4.4.3.

Organism	Number	Len	ΔG_S	ΔG_R^p	*SimFold*		*RNAfold*	
					#bp	Q_1 (Q_2)	#bp	Q_1 (Q_2)
PHAGE T5	DA0260	75	-25.90	-22.50	17/22	0.77 (0.83)	17/22	0.77 (0.83)
HALORUBRUM DISTRI.	DA0310	72	-29.90	-27.80	12/21	0.57 (0.56)	12/21	0.57 (0.56)
HALORUBRUM LACUSP.	DA0320	72	-31.60	-28.50	12/21	0.57 (0.53)	12/21	0.57 (0.53)
HALORUBRUM SACCHA.	DA0330	72	-28.90	-27.80	20/21	0.95 (0.94)	12/21	0.57 (0.58)
ARCHAEGLOBUS FULG.	DA0340	72	-37.80	-34.50	7/21	0.33 (0.33)	7/21	0.33 (0.33)
ARCHAEGLOBUS FULG.	DA0341	72	-39.80	-34.50	7/21	0.33 (0.33)	7/21	0.33 (0.33)

ARCHAEGLOBUS FULG.	DA0342	74	-35.60	-33.60	11/21	0.52 (0.59)	15/21	0.71 (0.73)
HALORUBRUM SODOME.	DA0350	72	-29.90	-27.80	12/21	0.57 (0.56)	12/21	0.57 (0.56)
HALORUBRUM VACUOL.	DA0360	72	-29.90	-27.80	12/21	0.57 (0.56)	12/21	0.57 (0.56)
NATRONOBAC. GREGO.	DA0370	72	-32.60	-30.60	15/21	0.71 (0.72)	12/21	0.57 (0.56)
HALOBACTERIUM CUT.	DA0380	72	-33.70	-30.60	12/21	0.57 (0.53)	12/21	0.57 (0.53)
NATRONOBAC. PHARA.	DA0390	72	-33.70	-30.60	12/21	0.57 (0.53)	12/21	0.57 (0.53)
HALOBACTERIUM HAL.	DA0420	72	-35.00	-30.60	12/21	0.57 (0.53)	12/21	0.57 (0.53)
METHANOBAC.FORMI.	DA0580	74	-34.60	-33.80	20/21	0.95 (0.97)	20/21	0.95 (0.97)
METHANOBAC.THERM.	DA0620	74	-34.60	-33.80	20/21	0.95 (0.97)	20/21	0.95 (0.97)
METHANOCOCCUS JAN.	DA0650	73	-37.80	-34.70	7/21	0.33 (0.34)	7/21	0.33 (0.34)
METHANOCOCCUS JAN.	DA0651	74	-34.20	-32.30	21/21	1.00 (0.97)	21/21	1.00 (0.97)
METHANOCOC.VANI.	DA0660	73	-32.80	-32.80	21/21	1.00 (1.00)	21/21	1.00 (1.00)
METHANOTHRIX SOEH.	DA0670	73	-32.60	-31.40	16/21	0.76 (0.74)	16/21	0.76 (0.74)
METHANOTHERM. FER.	DA0680	74	-37.30	-31.40	12/21	0.57 (0.57)	12/21	0.57 (0.57)
METHANOSPIR. HUNG.	DA0780	73	-33.40	-32.10	16/21	0.76 (0.81)	20/21	0.95 (0.95)
THERMOCOCCUS CELER	DA0940	77	-40.40	-34.30	12/21	0.57 (0.53)	12/21	0.57 (0.53)
THERMOPROT. TENAX	DA0980	72	-34.30	-33.50	7/21	0.33 (0.43)	12/21	0.57 (0.58)
THERMOPROT. TENAX	DA0981	72	-36.20	-33.50	7/21	0.33 (0.40)	7/21	0.33 (0.40)
BARTONELLA ELIZAB.	DA1110	76	-37.00	-28.80	7/20	0.35 (0.39)	7/20	0.35 (0.39)
BARTONELLA QUINT.	DA1130	76	-37.00	-28.80	7/20	0.35 (0.39)	7/20	0.35 (0.39)
MYCOPLASMA CAPRIC.	DA1140	76	-35.40	-29.30	7/21	0.33 (0.33)	7/21	0.33 (0.33)
MYCOPLASMA GEN.	DA1150	76	-27.90	-26.60	21/21	1.00 (0.95)	21/21	1.00 (0.95)
ACETOBACTER ACETI	DA1160	76	-33.60	-28.00	16/20	0.80 (0.75)	16/20	0.80 (0.75)
ACETOBACTER EUROP.	DA1170	76	-37.00	-28.80	7/20	0.35 (0.39)	7/20	0.35 (0.39)

Table A.2: Analysis of SimFold accuracy on a small subset of the tRNA genes database [51].

Tables A.3-A.7, show comparative analysis between real structures, *SimFold* prediction and *RNAfold* prediction on five RNA sets from Gutell database [21]: Group I Intron, Group II Intron, 5S rRNA, 16S rRNA and 23S rRNA. Columns 1 and 2 show the Organism and Accession Number from the database, and column 3 gives the length of the given sequence. Columns 4 and 5 show the percentage of odd pairs (non-canonical pairs) and pseudoknots in the real structures. Columns 6 and 8 report the number of pairs predicted by *SimFold* and *RNAfold*, respectively, out of the total number of pairs in the real structure. Columns 7 and 9 give the level of accuracy performed by *SimFold* and *RNAfold*, measured with two parameters. More information is given in Section 4.4.3.

Organism	Accession	Len	%	%	*SimFold*		*RNAfold*	
	Number		Odd	Pk	#bp	Q_1 (Q_2)	#bp	Q_1 (Q_2)
Acanthamoeba griffini	S81337	526	0.02	0.10	74/132	0.56 (0.58)	74/132	0.56 (0.57)
Acanthamoeba griffini	U02540	556	0.01	0.09	68/131	0.52 (0.53)	68/131	0.52 (0.53)
Bangia fuscopurpurea	AF342745	1031	0.01	0.10	20/133	0.15 (0.32)	20/133	0.15 (0.32)
Hildenbrandia rubra		543	0.01	0.09	48/138	0.35 (0.43)	57/138	0.41 (0.48)
Metarhizium anisopliae var. anisopliae	AF197120	394	0.08	0.11	74/120	0.62 (0.65)	74/120	0.62 (0.64)
Metarhizium anisopliae var. anisopliae	AF197122	456	0.09	0.11	25/115	0.22 (0.36)	25/115	0.22 (0.35)
Porphyra leucosticta	AF342746	605	0.02	0.11	73/121	0.60 (0.54)	73/121	0.60 (0.54)
Tetrahymena thermophila	V01416 J01235	506	0.06	0.09	107/137	0.78 (0.76)	107/137	0.78 (0.76)

Table A.3: Analysis of SimFold accuracy on Group I Intron sequences from Gutell database [21].

Organism	Accession	Len	%	%	*SimFold*		*RNAfold*	
	Number		Odd	Pk	#bp	Q_1 (Q_2)	#bp	Q_1 (Q_2)
Saccharomyces cerevisiae	V00694	905	0.00	0.44	102/264	0.39 (0.46)	130/264	0.49 (0.52)
Saccharomyces cerevisiae (D273-10B)	AJ011856	2520	0.00	0.46	83/209	0.40 (0.41)	107/209	0.51 (0.42)

Table A.4: Analysis of SimFold accuracy on Group II Intron sequences from Gutell database [21].

Organism	Accession Number	Len	% Odd	% Pk	*SimFold*		*RNAfold*	
					#bp	Q_1 (Q_2)	#bp	Q_1 (Q_2)
Agrobacterium tumefaciens	X02627	120	0.10	0.00	12/39	0.31 (0.37)	12/39	0.31 (0.37)
Deinococcus radiodurans	AE002087	124	0.12	0.00	27/40	0.68 (0.74)	27/40	0.68 (0.74)
Escherichia coli	V00336	120	0.07	0.00	10/40	0.25 (0.39)	10/40	0.25 (0.39)
Geobacillus stearothermophilus	AJ251080	117	0.13	0.00	25/38	0.66 (0.67)	25/38	0.66 (0.67)
Geobacillus stearothermophilus	M24839	119	0.24	0.00	8/38	0.21 (0.35)	8/38	0.21 (0.35)
Geobacillus stearothermophilus	M25591	117	0.13	0.00	27/38	0.71 (0.71)	27/38	0.71 (0.71)
Haloarcula marismortui	AF034620	122	0.11	0.00	29/38	0.76 (0.80)	29/38	0.76 (0.80)
Saccharomyces cerevisiae	X67579	118	0.05	0.00	33/37	0.89 (0.85)	28/37	0.76 (0.70)
Thermus aquaticus	X01590	123	0.07	0.00	25/40	0.62 (0.66)	8/40	0.20 (0.33)

Table A.5: Analysis of SimFold accuracy on 5S rRNA sequences from Gutell database [21].

Organism	Accession Number	Len	% Odd	% Pk	*SimFold*		*RNAfold*	
					#bp	Q_1 (Q_2)	#bp	Q_1 (Q_2)
Agrobacterium tumefaciens	M11223	1489	0.05	0.06	239/452	0.53 (0.58)	242/452	0.54 (0.58)
Anabaena sp.	X59559	1489	0.05	0.07	208/456	0.46 (0.53)	238/456	0.52 (0.58)
Antilocapra americana	M55540	958	0.07	0.10	102/267	0.38 (0.51)	102/267	0.38 (0.51)
Arthrobacter globiformis	M23411	1531	0.05	0.06	212/470	0.45 (0.53)	164/470	0.35 (0.45)
Bacillus subtilis	K00637	1552	0.06	0.06	246/478	0.51 (0.58)	246/478	0.51 (0.58)
Bacillus subtilis A	AL009126	1553	0.05	0.06	245/476	0.51 (0.58)	245/476	0.51 (0.58)
Bordetella bronchiseptica	U04948	1532	0.05	0.07	241/473	0.51 (0.57)	283/473	0.60 (0.63)
Borrelia burgdorferi	AE001147	1538	0.04	0.07	260/476	0.55 (0.58)	252/476	0.53 (0.57)
Borrelia burgdorferi	M88329	1537	0.05	0.06	256/477	0.54 (0.57)	248/477	0.52 (0.56)
Bradyrhizobium japonicum	Z35330	1490	0.05	0.07	212/457	0.46 (0.54)	210/457	0.46 (0.53)
Caenorhabditis elegans	X54252	697	0.12	0.11	40/189	0.21 (0.34)	40/189	0.21 (0.34)
Candida albicans	M60302	1787	0.09	0.06	167/495	0.34 (0.41)	172/495	0.35 (0.41)
Chlamydia trachomatis L2/434/BU	U68443	1554	0.05	0.06	233/482	0.48 (0.54)	236/482	0.49 (0.55)
Chlamydomonas eugametos	AF008237	1257	0.06	0.02	138/363	0.38 (0.46)	138/363	0.38 (0.46)
Chlamydomonas reinhardtii	J01395	1474	0.05	0.06	240/440	0.55 (0.59)	240/440	0.55 (0.59)
Chlamydophila pneumoniae	L06108	1554	0.05	0.06	241/483	0.50 (0.55)	244/483	0.51 (0.55)
Chlamydophila psittaci 6BC	U68447	1552	0.05	0.06	228/480	0.47 (0.54)	231/480	0.48 (0.54)
Comamonas testosteroni	M11224	1536	0.05	0.02	248/465	0.53 (0.59)	220/465	0.47 (0.54)
Cyanophora paradoxa	U30821	1495	0.06	0.06	269/458	0.59 (0.63)	272/458	0.59 (0.63)
Deinococcus radiodurans	AE001871	1502	0.06	0.07	306/462	0.66 (0.68)	309/462	0.67 (0.68)
Drosophila melanogaster	M21017	1995	0.08	0.06	160/498	0.32 (0.41)	166/498	0.33 (0.41)
Drosophila virilis	X05914	784	0.05	0.08	36/233	0.15 (0.26)	36/233	0.15 (0.26)
Escherichia coli	J01695	1542	0.04	0.06	198/477	0.42 (0.48)	207/477	0.43 (0.50)
Escherichia coli K12	AE000460	1542	0.04	0.06	198/477	0.42 (0.48)	207/477	0.43 (0.50)
Escherichia coli O157:H7 EDL933	AE005628	1542	0.05	0.06	196/477	0.41 (0.48)	205/477	0.43 (0.49)
Euglena gracilis	X12890	1491	0.06	0.06	79/453	0.17 (0.31)	169/453	0.37 (0.46)
Fragaria x ananassa	X15590	1804	0.10	0.06	126/496	0.25 (0.35)	125/496	0.25 (0.35)
Frankia sp. 1	M55343	1512	0.05	0.07	203/465	0.44 (0.51)	203/465	0.44 (0.51)
Giardia intestinalis	X52949	1452	0.11	0.07	60/398	0.15 (0.28)	60/398	0.15 (0.28)
Gracilariopsis sp. (England-1)	M33639	1782	0.10	0.06	156/499	0.31 (0.39)	184/499	0.37 (0.42)
Haemophilus influenzae (operons A-F)	U32741	1539	0.05	0.02	201/470	0.43 (0.50)	204/470	0.43 (0.50)
Halobacterium sp.	AE005128	1473	0.05	0.06	283/462	0.61 (0.62)	182/462	0.39 (0.47)
Haloferax volcanii	K00421	1474	0.04	0.06	350/458	0.76 (0.74)	350/458	0.76 (0.74)
Homo sapiens	J01415	954	0.11	0.10	88/266	0.33 (0.49)	88/266	0.33 (0.49)
Homo sapiens	K03432	1870	0.11	0.05	133/524	0.25 (0.34)	133/524	0.25 (0.33)
Lactococcus lactis subsp. lactis	AE006456	1551	0.05	0.07	278/476	0.58 (0.62)	277/476	0.58 (0.63)
Leptospira interrogans	X17547	1508	0.05	0.06	244/463	0.53 (0.58)	270/463	0.58 (0.62)
Mycobacterium leprae	X56657	1548	0.05	0.07	86/476	0.18 (0.32)	83/476	0.17 (0.31)
Mycobacterium tuberculosis	Z83862	1537	0.04	0.07	108/469	0.23 (0.36)	108/469	0.23 (0.36)
Mycoplasma gallisepticum	M22441	1519	0.07	0.07	179/466	0.38 (0.47)	179/466	0.38 (0.47)
Mycoplasma genitalium	U39694	1519	0.06	0.07	216/465	0.46 (0.54)	180/465	0.39 (0.50)
Mycoplasma hyopneumoniae	Y00149	1537	0.06	0.07	301/471	0.64 (0.66)	249/471	0.53 (0.57)
Neisseria gonorrhoeae	X07714	1544	0.05	0.02	241/468	0.51 (0.58)	217/468	0.46 (0.54)
Neisseria meningitidis	AE002364	1544	0.05	0.02	240/468	0.51 (0.56)	240/468	0.51 (0.57)
Pasteurella multocida	AE006192	1542	0.05	0.02	232/469	0.49 (0.54)	224/469	0.48 (0.52)
Physarum polycephalum	X75592	1861	0.09	0.07	72/436	0.17 (0.32)	69/436	0.16 (0.31)
Pirellula marina	X62912	1472	0.08	0.07	136/445	0.31 (0.42)	131/445	0.29 (0.40)
Plasmodium falciparum (A gene)	M19172	2090	0.06	0.05	238/537	0.44 (0.48)	236/537	0.44 (0.47)
Plasmodium falciparum (S gene)	M19173	2145	0.09	0.01	142/538	0.26 (0.37)	134/538	0.25 (0.36)
Plasmodium falciparum (plastid-like)	X57167	1426	0.05	0.06	102/432	0.24 (0.35)	120/432	0.28 (0.38)
Plasmodium vivax (A gene)	U07367	2063	0.07	0.05	218/524	0.42 (0.46)	218/524	0.42 (0.46)
Plasmodium vivax (S gene)	U07368	2147	0.07	0.05	185/547	0.34 (0.40)	185/547	0.34 (0.40)
Proteus vulgaris	X07652	1543	0.04	0.06	231/477	0.48 (0.54)	229/477	0.48 (0.54)

Pseudomonas aeruginosa	AE004501	1536	0.05	0.07	229/472	0.49 (0.56)	223/472	0.47 (0.55)
Psychrobacter pacificensis	AB016054	1536	0.05	0.07	210/473	0.44 (0.53)	197/473	0.42 (0.49)
Pyrococcus abyssi	AJ248283	1512	0.04	0.06	305/474	0.64 (0.65)	300/474	0.63 (0.64)
Pyrococcus horikoshii	AP000001	1500	0.04	0.06	305/473	0.64 (0.66)	300/473	0.63 (0.65)
Rhodococcus erythropolis	AF001265	1519	0.05	0.07	173/466	0.37 (0.48)	173/466	0.37 (0.48)
Rickettsia prowazekii	M21789	1502	0.05	0.07	242/463	0.52 (0.59)	202/463	0.44 (0.51)
Rickettsia prowazekii (str. Madrid E)	AJ235272	1501	0.05	0.07	240/462	0.52 (0.58)	200/462	0.43 (0.50)
Saccharomyces cerevisiae	U53879	1800	0.09	0.06	182/497	0.37 (0.45)	182/497	0.37 (0.44)
Staphylococcus aureus	L36472	1555	0.06	0.06	208/475	0.44 (0.51)	208/475	0.44 (0.51)
Staphylococcus aureus N315	AP003130	1555	0.05	0.06	215/475	0.45 (0.52)	215/475	0.45 (0.52)
Streptococcus pyogenes	AE006473	1549	0.05	0.06	253/478	0.53 (0.59)	259/478	0.54 (0.60)
Streptomyces acidiscabies	D63865	1530	0.05	0.07	260/468	0.56 (0.60)	242/468	0.52 (0.57)
Streptomyces bottropensis	D63868	1531	0.05	0.07	192/468	0.41 (0.50)	189/468	0.40 (0.50)
Streptomyces diastatochromogenes	D63867	1531	0.05	0.07	186/468	0.40 (0.49)	213/468	0.46 (0.53)
Streptomyces eurythermus	D63870	1531	0.05	0.07	184/467	0.39 (0.49)	202/467	0.43 (0.52)
Streptomyces griseus	X61478	1528	0.05	0.07	153/468	0.33 (0.43)	153/468	0.33 (0.43)
Streptomyces neyagawaensis	D63869	1531	0.05	0.07	170/468	0.36 (0.46)	167/468	0.36 (0.45)
Streptomyces nodosus	AF114033	1528	0.05	0.07	199/467	0.43 (0.51)	199/467	0.43 (0.51)
Streptomyces sampsonii	D63871	1531	0.05	0.07	169/468	0.36 (0.46)	169/468	0.36 (0.46)
Streptomyces scabiei	D63862	1530	0.05	0.07	191/474	0.40 (0.49)	225/474	0.47 (0.54)
Streptomyces setonii	D63872	1532	0.05	0.07	176/469	0.38 (0.47)	179/469	0.38 (0.48)
Streptomyces sp.	D63866	1530	0.05	0.07	239/468	0.51 (0.56)	236/468	0.50 (0.56)
Streptomyces tendae	D63873	1530	0.05	0.07	202/468	0.43 (0.52)	202/468	0.43 (0.52)
Stylonychia mytilus	AF164123	1771	0.08	0.06	159/480	0.33 (0.42)	122/480	0.25 (0.36)
Suillus sinuspaulianus	UNP00183	1987	0.05	0.01	162/544	0.30 (0.39)	162/544	0.30 (0.39)
Synechococcus sp. PCC 6301	X03538	1488	0.05	0.06	164/453	0.36 (0.45)	176/453	0.39 (0.48)
Synechocystis sp. PCC 6803	D64000	1489	0.05	0.07	178/456	0.39 (0.48)	177/456	0.39 (0.48)
Thermotoga maritima	AE001703	1559	0.05	0.06	279/482	0.58 (0.62)	275/482	0.57 (0.61)
Thermotoga maritima	M21774	1562	0.05	0.06	280/484	0.58 (0.62)	276/484	0.57 (0.61)
Thermus thermophilus	X07998	1518	0.05	0.06	253/468	0.54 (0.59)	263/468	0.56 (0.60)
Thorea violacea	AF026042	1916	0.09	0.05	208/547	0.38 (0.43)	182/547	0.33 (0.40)
Treponema pallidum (rRNA A)	AE001204	1549	0.05	0.06	214/478	0.45 (0.51)	190/478	0.40 (0.48)
Ureaplasma urealyticum	AE002112	1545	0.07	0.07	188/468	0.40 (0.49)	190/468	0.41 (0.50)
Vairimorpha necatrix	Y00266	1244	0.09	0.07	71/370	0.19 (0.31)	67/370	0.18 (0.30)
Xenopus laevis	M27605	945	0.09	0.07	92/251	0.37 (0.52)	71/251	0.28 (0.46)
Xylella fastidiosa	AE003861	1545	0.06	0.07	182/475	0.38 (0.46)	170/475	0.36 (0.44)
Zea mays	X00794	1962	0.06	0.06	138/455	0.30 (0.40)	127/455	0.28 (0.39)
Zea mays	Z00028	1490	0.06	0.06	212/452	0.47 (0.54)	215/452	0.48 (0.55)

Table A.6: Analysis of SimFold accuracy on 16S rRNA sequences from Gutell database [21].

Organism	Accession	Len	%	%	*SimFold*		*RNAfold*	
	Number		Odd	Pk	#bp	Q_1 (Q_2)	#bp	Q_1 (Q_2)
Acinetobacter calcoaceticus	X87280	2903	0.06	0.21	358/871	0.41 (0.50)	357/871	0.41 (0.50)
Albinaria caerulea	X83390	1035	0.10	0.15	91/268	0.34 (0.42)	82/268	0.31 (0.39)
Albinaria turrita	X71393	1077	0.10	0.00	36/265	0.14 (0.28)	63/265	0.24 (0.34)
Arabidopsis thaliana	X52320	3539	0.07	0.00	458/998	0.46 (0.51)	417/998	0.42 (0.47)
Bacillus subtilis	K00637	2927	0.04	0.21	447/871	0.51 (0.58)	437/871	0.50 (0.56)
Bartonella bacilliformis	L39095	2821	0.04	0.22	441/836	0.53 (0.57)	390/836	0.47 (0.53)
Borrelia burgdorferi	M88330	2926	0.04	0.21	339/879	0.39 (0.47)	329/879	0.37 (0.47)
Burkholderia cepacia	X16368	2878	0.05	0.22	459/858	0.53 (0.58)	428/858	0.50 (0.55)
Burkholderia mallei	Y17183	2882	0.05	0.22	413/863	0.48 (0.55)	367/863	0.43 (0.50)
Burkholderia pseudomallei	Y17184	2882	0.05	0.22	409/862	0.47 (0.54)	367/862	0.43 (0.50)
Cacozeliana lacertina	AF101007	1341	0.07	0.12	102/328	0.31 (0.42)	102/328	0.31 (0.42)
Caenorhabditis elegans	X54252	953	0.08	0.00	52/219	0.24 (0.35)	51/219	0.23 (0.32)
Campylobacter jejuni	AL139074	2907	0.06	0.21	388/872	0.44 (0.53)	331/872	0.38 (0.48)
Chlamydia trachomatis L2/434/BU	U68443	2941	0.06	0.21	348/867	0.40 (0.50)	379/867	0.44 (0.53)
Chlamydomonas eugametos	AF008237	1915	0.05	0.00	145/483	0.30 (0.43)	145/483	0.30 (0.43)
Chlamydomonas reinhardtii	X15727	2902	0.05	0.22	364/847	0.43 (0.51)	364/847	0.43 (0.51)
Chlamydophila psittaci 6BC	U68447	2942	0.06	0.21	349/886	0.39 (0.49)	292/886	0.33 (0.44)
Clostridium botulinum A	X65602	2896	0.05	0.21	437/862	0.51 (0.57)	441/862	0.51 (0.57)
Cyanophora paradoxa	U30821	2926	0.04	0.21	364/863	0.42 (0.51)	333/863	0.39 (0.49)
Erysipelothrix rhusiopathiae (str. 715)	AB019250	2901	0.05	0.22	320/858	0.37 (0.49)	318/858	0.37 (0.48)
Escherichia coli	J01695	2904	0.04	0.21	425/869	0.49 (0.56)	435/869	0.50 (0.56)
Euglena gracilis	X12890	2877	0.05	0.18	305/843	0.36 (0.46)	300/843	0.36 (0.46)
Haemophilus influenzae (operons A-F)	U32742	2897	0.05	0.21	359/866	0.41 (0.50)	323/866	0.37 (0.47)
Haloarcula marismortui	X13738	2925	0.04	0.21	378/852	0.44 (0.53)	379/852	0.44 (0.53)
Haloarcula marismortui rrnA	AF034619	2930	0.05	0.21	443/855	0.52 (0.58)	432/855	0.51 (0.57)
Haloarcula marismortui rrnB	AF034620	2930	0.04	0.21	451/855	0.53 (0.58)	379/855	0.44 (0.52)

Helicobacter pylori	U27270	2968	0.05	0.00	386/822	0.47 (0.55)	374/822	0.45 (0.53)
Katharina tunicata	U09810	1275	0.06	0.14	65/300	0.22 (0.37)	63/300	0.21 (0.36)
Klebsiella pneumoniae	X87284	2903	0.05	0.21	360/868	0.41 (0.50)	364/868	0.42 (0.50)
Lactococcus lactis	X68434	2567	0.34	0.58	178/1009	0.18 (0.39)	178/1009	0.18 (0.39)
Leptospira interrogans	X14249	2958	0.06	0.21	384/883	0.43 (0.52)	375/883	0.42 (0.51)
Listeria monocytogenes	X64533	2928	0.05	0.21	369/868	0.43 (0.51)	370/868	0.43 (0.51)
Listeria monocytogenes	X68420	2932	0.04	0.21	448/872	0.51 (0.59)	465/872	0.53 (0.60)
Micrococcus luteus	X06484	3094	0.04	0.00	398/894	0.45 (0.51)	387/894	0.43 (0.50)
Mycobacterium leprae	X56657	3122	0.05	0.00	415/906	0.46 (0.53)	425/906	0.47 (0.53)
Mycoplasma genitalium	U39694	2917	0.04	0.21	305/867	0.35 (0.46)	345/867	0.40 (0.50)
Mycoplasma pneumoniae	X68422 U00089	2905	0.04	0.21	401/866	0.46 (0.54)	399/866	0.46 (0.54)
Neisseria gonorrhoeae	X67293	2890	0.05	0.21	320/865	0.37 (0.47)	349/865	0.40 (0.50)
Neisseria meningitidis	X67300	2890	0.05	0.21	334/865	0.39 (0.48)	334/865	0.39 (0.48)
Oryza sativa	M11585	3541	0.07	0.00	384/1012	0.38 (0.45)	404/1012	0.40 (0.46)
Pecten maximus	X92688	1411	0.07	0.12	65/317	0.21 (0.34)	68/317	0.21 (0.34)
Plasmodium falciparum (A gene)	U21939	3946	0.06	0.00	495/1059	0.47 (0.53)	484/1059	0.46 (0.53)
Plasmodium falciparum (S gene)	U48228	4381	0.07	0.14	442/1147	0.39 (0.45)	416/1147	0.36 (0.42)
Plasmodium falciparum (plastid-like)	X61660	2700	0.05	0.21	202/782	0.26 (0.34)	202/782	0.26 (0.35)
Pseudomonas aeruginosa	Y00432	2893	0.05	0.21	364/867	0.42 (0.50)	378/867	0.44 (0.51)
Rickettsia prowazekii (str. Madrid E)	AJ235270	2763	0.04	0.23	410/819	0.50 (0.57)	455/819	0.56 (0.61)
Ruminobacter amylophilus	X06765	2867	0.06	0.22	330/856	0.39 (0.49)	327/856	0.38 (0.48)
Saccharomyces cerevisiae	U53879	3554	0.06	0.15	488/1066	0.46 (0.51)	479/1066	0.45 (0.50)
Staphylococcus aureus	X68425	2923	0.04	0.21	435/874	0.50 (0.56)	398/874	0.46 (0.53)
Staphylococcus carnosus	X68419	2924	0.04	0.21	463/876	0.53 (0.58)	521/876	0.59 (0.63)
Streptomyces ambofaciens	M27245	3120	0.04	0.00	482/906	0.53 (0.57)	401/906	0.44 (0.51)
Suillus sinuspaulianus	UNP00109	4216	0.07	0.21	212/739	0.29 (0.38)	218/739	0.29 (0.39)
Thermococcus celer	M67497	3029	0.02	0.00	563/879	0.64 (0.64)	562/879	0.64 (0.64)
Thermotoga maritima	M67498	3023	0.03	0.21	533/901	0.59 (0.63)	514/901	0.57 (0.61)
Thermus thermophilus	X12612	2915	0.03	0.00	486/832	0.58 (0.63)	474/832	0.57 (0.61)
Treponema pallidum (rRNA A)	AE001204	2953	0.05	0.21	230/884	0.26 (0.40)	242/884	0.27 (0.41)
Xenopus laevis	M10217	1640	0.09	0.00	125/373	0.34 (0.48)	125/373	0.34 (0.48)
Zea mays	K01868	3514	0.09	0.20	154/745	0.21 (0.33)	133/745	0.18 (0.33)
Zea mays	Z00028	2981	0.07	0.21	302/880	0.34 (0.45)	289/880	0.33 (0.44)

Table A.7: Analysis of SimFold accuracy on 23S rRNA sequences from Gutell database [21].

Appendix B

PairFold Analysis

Table B.1 gives a comparison between the *PairFold* prediction and experimental and predicted values reported by Peyret et al. [37] on a set of DNA duplexes. Column 1 shows the first sequence. The slash at the end of some sequences indicate non-self-complementary sequences. The underlined bases show the mismatches with the second sequence. The second sequence is the perfect complement of the given sequence, with the exception of the underlined base, which is just copied. For example, the first sequence's complement is `5'-CTTTATTTG-3'`. Columns 2 and 3 give the standard enthalpy change (in kcal/mol): the experimental results (column 2), the *PairFold* prediction (column 3), and the prediction reported in [37] (between parentheses in column 3). The same format is adopted in the next columns for standard entropy change, standard free energy change and melting temperature. The line marked with a star shows that the duplex `5'-CAAACAAAG-3' 5'-CTTTCTTTG-3'`, is not predicted to fold into the expected structure, which is `((((.(((( )))).))))`. The predicted structure is `....((((( ....)))))`. This table is refered to in Section 5.5.1.

Sequences	ΔH° (kcal/mol)		ΔS° (eu)		ΔG°_{37} (kcal/mol)		T_m ($^\circ C$)	
	exp	PairFold (P)	exp	PairFold (P)	exp	PairFold (P)	exp	PairFold (P)
CAAAAAAAG/	-36.9	-41.9 (-41.9)	-107.0	-123.9 (-123.8)	-3.71	-3.47 (-3.47)	21.3	21.5 (21.5)
CGATAATCG	-50.8	-42.4 (-42.4)	-148.0	-120.7 (-120.8)	-4.86	-4.96 (-4.96)	32.1	31.9 (31.9)
GGAAATTCC	-51.5	-45.7 (-45.7)	-151.4	-132.2 (-132.2)	-4.59	-4.70 (-4.70)	30.6	30.5 (30.5)
GGACAGTCC	-53.7	-50.7 (-50.7)	-153.2	-144.3 (-144.4)	-6.22	-5.94 (-5.94)	40.2	38.6 (38.6)
GGAGACTCC	-51.6	-53.5 (-53.5)	-145.7	-152.7 (-152.8)	-6.38	-6.14 (-6.14)	41.3	39.7 (39.7)
CATGAAGCTAC/	-65.2	-65.1 (-65.1)	-185.4	-185.3 (-185.4)	-7.70	-7.62 (-7.62)	46.9	46.6 (46.5)
CATGTAACTAC/	-48.0	-54.8 (-54.8)	-133.8	-155.3 (-155.5)	-6.52	-6.62 (-6.62)	42.5	42.5 (42.4)
GATCTATGTAC/	-59.3	-57.9 (-57.9)	-170.6	-166.0 (-165.9)	-6.42	-6.41 (-6.41)	40.9	41.0 (41.0)
GGATGAATAGC/	-69.3	-61.9 (-61.9)	-198.2	-175.0 (-174.9)	-7.81	-7.63 (-7.63)	46.9	47.1 (47.1)
GGATGAGTAGC/	-70.3	-68.6 (-68.6)	-198.3	-193.9 (-194.0)	-8.79	-8.45 (-8.45)	51.4	50.1 (50.1)
CGCAAGAGACGG/	-66.3	-64.5 (-64.5)	-186.6	-179.4 (-179.4)	-8.42	-8.86 (-8.86)	50.4	53.1 (53.1)
GGCAGAGAACGC/	-60.6	-65.6 (-65.6)	-168.6	-183.2 (-183.3)	-8.31	-8.77 (-8.77)	51.1	52.4 (52.3)
GGA(CAG)$_3$AGG/	-74.5	-73.1 (-73.2)	-211.6	-206.9 (207.1)	-8.87	-8.94 (-8.94)	51.0	51.5 (51.0)
*CAAACAAAG/	-55.3	-42.2 (-41.6)	-170.0	-122.2 (-126.4)	-2.57	-4.31 (-2.39)	20.5	27.3 (14.3)
CGATCATCG	-36.6	-39.5 (-39.5)	-104.7	-113.5 (-113.8)	-4.14	-4.29 (-4.24)	24.5	26.5 (26.1)
GGAACTTCC	-44.5	-48.1 (-48.1)	-133.2	-144.6 (-144.4)	-3.14	-3.26 (-3.26)	20.3	22.2 (22.2)
GGACCGTCC	-53.6	-52.0 (-52.1)	-155.4	-150.2 (-150.4)	-5.35	-5.40 (-5.40)	35.1	35.4 (35.4)
GGAGCCACG/	-48.9	-44.2 (-44.2)	-138.9	-123.4 (-123.3)	-5.86	-5.94 (-5.94)	38.2	38.9 (38.9)
GGAGCCTCC	-44.1	-40.7 (-40.7)	-125.9	-115.4 (-115.4)	-5.09	-4.90 (-4.90)	33.0	31.2 (31.2)
CATGTCACTAC/	-54.2	-51.9 (-51.9)	-155.7	-148.2 (-148.4)	-5.92	-5.95 (-5.90)	38.4	38.7 (38.3)
GATCTCTGTAC/	-55.7	-57.6 (-57.6)	-162.1	-168.5 (-168.5)	-5.47	-5.33 (-5.33)	35.9	35.2 (35.2)
GGATCCCTAGC/	-54.7	-62.0 (-62.0)	-152.2	-176.5 (-176.4)	-7.46	-7.25 (-7.25)	47.5	45.1 (45.1)
GGATGCTTAGC/	-55.7	-60.9 (-60.9)	-160.0	-175.4 (-175.3)	-6.05	-6.49 (-6.49)	39.1	41.2 (41.2)
GGATTCCTAGC/	-46.8	-54.0 (-54.0)	-130.1	-152.8 (-152.8)	-6.49	-6.60 (-6.60)	42.5	42.4 (42.4)
GGATTCGTAGC/	-62.0	-59.7 (-59.7)	-179.1	-170.4 (-170.4)	-6.43	-6.85 (-6.85)	40.8	43.2 (43.2)

GTAGCCTCATG/	-70.2	-67.0 (-67.0)	-203.6	-194.6 (-194.5)	-7.09	-6.63 (-6.63)	43.4	41.5 (41.5)
CAAAGAAAG/	-53.5	-49.6 (-52.3)	-158.0	-145.5 (-154.3)	-4.50	-4.46 (-4.46)	30.3	29.6 (30.0)
CGATGATCG	-48.4	-48.9 (-54.9)	-138.8	-140.1 (-159.4)	-5.34	-5.46 (-5.46)	34.9	35.7 (35.8)
GGAAGTTCC	-52.2	-54.6 (-54.2)	-148.9	-156.1 (-154.7)	-6.00	-6.18 (-6.18)	38.9	39.9 (39.9)
GGACGGTCC	-56.7	-58.2 (-58.7)	-160.8	-165.0 (-166.8)	-6.85	-7.02 (-7.02)	43.6	44.4 (44.3)
GGAGGCTCC	-56.9	-60.5 (-59.8)	-156.1	-167.0 (-164.8)	-8.50	-8.70 (-8.70)	53.2	53.3 (53.5)
CATGAGGCTAC/	-76.4	-73.3 (-73.4)	-215.7	-207.6 (-207.8)	-9.53	-8.90 (-8.90)	53.5	51.3 (51.3)
CATGTGACTAC/	-57.0	-61.3 (-67.3)	-160.8	-174.7 (-194.0)	-7.18	-7.12 (-7.12)	45.4	44.5 (43.8)
CCATCGCTACC/	-78.7	-74.1 (-74.0)	-221.0	-207.1 (-206.7)	-10.17	-9.87 (-9.87)	55.8	55.6 (55.6)
CCATTGCTACC/	-75.1	-67.7 (-70.4)	-212.3	-189.2 (-197.8)	-9.30	-9.02 (-9.02)	52.7	53.1 (52.5)
GATCTGTGTAC/	-64.4	-65.6 (-68.3)	-183.4	-187.7 (-196.5)	-7.49	-7.40 (-7.40)	46.0	45.4 (45.0)
GCTAGGTATCC/	-69.6	-72.7 (-72.1)	-194.6	-204.5 (-202.6)	-9.24	-9.26 (-9.26)	53.8	53.1 (53.2)
GCTATGTATCC/	-66.2	-64.3 (-69.8)	-187.9	-183.7 (-199.3)	-7.86	-7.31 (-8.01)	47.6	45.1 (47.7)
CAAATAAAG/	-54.6	-50.2 (-50.6)	-166.0	-150.9 (-152.1)	-3.12	-3.40 (-3.40)	23.1	23.6 (23.6)
CGAGTGTCC/	-55.9	-58.2 (-58.2)	-158.3	-165.0 (-164.9)	-6.83	-7.04 (-7.04)	43.5	44.5 (44.5)
CGATTATCG	-48.6	-51.4 (-52.2)	-140.6	-149.7 (-152.2)	-4.95	-4.98 (-4.98)	32.4	32.9 (32.9)
CGTCTGTCC/	-62.3	-61.6 (-61.6)	-176.0	-173.6 (-173.6)	-7.70	-7.77 (-7.77)	47.4	47.9 (47.9)
CGTGTCTCC/	-60.3	-55.6 (-55.6)	-172.0	-157.9 (-158.1)	-6.90	-6.62 (-6.62)	43.4	42.4 (42.4)
GGAATTTCC	-47.4	-53.3 (-53.3)	-138.4	-157.2 (-157.3)	-4.50	-4.54 (-4.54)	29.5	30.5 (30.5)
GGACTGTCC	-59.4	-59.0 (-59.0)	-168.5	-167.5 (-167.6)	-7.08	-7.04 (-7.04)	44.5	44.4 (44.3)
GGAGTCTCC	-51.1	-52.1 (-52.1)	-146.7	-150.0 (-150.1)	-5.62	-5.58 (-5.58)	36.6	36.5 (36.4)
CATGATGCTAC/	-77.3	-73.1 (-73.1)	-220.8	-209.6 (-209.5)	-8.86	-8.09 (-8.09)	50.3	47.6 (47.6)
CATGTTACTAC/	-61.4	-63.8 (-64.6)	-175.5	-184.3 (-186.9)	-6.99	-6.64 (-6.64)	43.8	41.8 (41.7)
GATCTTTGTAC/	-77.7	-66.2 (-66.6)	-227.7	-193.0 (-194.2)	-7.09	-6.34 (-6.34)	42.8	40.2 (40.1)
GGATGTATAGC/	-72.9	-65.7 (-66.1)	-210.1	-188.1 (-189.3)	-7.70	-7.36 (-7.36)	45.9	45.2 (45.1)
CGCTAGAGTCGG/	-65.5	-71.7 (-72.2)	-184.3	-202.6 (-204.1)	-8.31	-8.86 (-8.86)	50.0	51.4 (51.3)
GGCTGAGATCGC/	-79.7	-77.0 (-77.1)	-226.0	-217.6 (-217.8)	-9.64	-9.51 (-9.51)	53.2	53.3 (53.2)

Table B.1: Measurement of PairFold accuracy on a set of DNA duplexes.

Table B.2 gives a comparison between the *PairFold* prediction and experimental and predicted values reported by Xia et al. [67] on a set of RNA duplexes. Column 1 shows the first sequence. The second sequence is the perfect complement of the first one. The slash at the end of some sequences indicate non-self-complementary sequences. Columns 2 and 3 give the standard enthalpy change (in kcal/mol): the experimental results (column 2), the *PairFold* prediction (column 3), and the prediction reported in [67] (between parentheses in column 3). The same format is adopted in the next columns for standard entropy change (eu), standard free energy change (kcal/mol) and melting temperature (°C). The bases near the star signs show that the structure predicted by *PairFold* is different than expected, i.e. they are predicted as free bases as opposed to paired bases. More details are discussed in Section 5.5.1.

Sequences	ΔH° (kcal/mol)		ΔS° (eu)		ΔG°_{37} (kcal/mol)		T_m ($^\circ C$)	
	exp	PairFold (X)	exp	PairFold (X)	exp	PairFold (X)	exp	PairFold (X)
CCGG	-34.2	-32.4 (-33.8)	-95.6	-90.1 (-95.0)	-4.55	-4.47 (-4.36)	27.2	25.9 (25.3)
CGCG	-33.3	-30.2 (-32.5)	-95.6	-85.5 (-93.2)	-3.66	-3.67 (-3.62)	19.3	17.7 (18.8)
GCGC	-30.5	-36.4 (-36.8)	-83.4	-102.3 (-103.4)	-4.61	-4.67 (-4.68)	26.6	28.7 (29.1)
GGCC	-35.8	-38.6 (-38.0)	-98.1	-106.8 (-105.2)	-5.37	-5.47 (-5.42)	34.3	35.4 (34.9)
ACGCA*/	-45.4	-41.8 (-36.3)	-130.4	-118.3 (-100.5)	-4.97	-5.10 (-5.14)	29.4	29.8 (29.0)
AGCGA/	-46.3	-43.1 (-37.4)	-133.0	-122.2 (-103.7)	-5.05	-5.20 (-5.22)	30.2	30.7 (29.9)
CACAG/	-40.2	-38.8 (-39.1)	-115.4	-110.9 (-111.9)	-4.70	-4.40 (-4.45)	24.5	24.0 (24.1)
GCACG/	-45.3	-42.9 (-43.8)	-126.2	-119.0 (-121.5)	-6.17	-6.00 (-6.04)	37.5	36.3 (36.7)
GCUCG/	-43.4	-43.1 (-44.8)	-120.1	-119.0 (-124.7)	-6.14	-6.20 (-6.12)	37.2	37.7 (37.4)
ACCGGU	-59.8	-52.8 (-49.2)	-164.5	-144.9 (-133.0)	-8.51	-7.87 (-7.94)	53.9	50.5 (51.8)
AGCGCU	-50.1	-51.6 (-50.3)	-135.7	-141.0 (-136.6)	-7.99	-7.87 (-7.94)	52.0	50.8 (51.6)
AGGCCU	-48.2	-53.8 (-51.6)	-128.4	-145.5 (-138.4)	-8.36	-8.67 (-8.68)	55.3	55.3 (55.9)
CACGUG	-50.3	-49.4 (-50.7)	-141.0	-138.4 (-142.4)	-6.59	-6.47 (-6.54)	42.8	42.1 (42.4)
CAGCUG	-51.5	-50.4 (-53.1)	-144.7	-139.1 (-147.8)	-6.68	-7.27 (-7.28)	43.1	47.2 (46.6)
CCAUGG	-56.9	-51.1 (-53.4)	-159.9	-141.0 (-148.8)	-7.30	-7.37 (-7.32)	46.4	47.7 (46.6)
CCGCGG	-60.8	-54.6 (-59.3)	-164.3	-142.9 (-158.6)	-9.84	-10.27 (-10.14)	59.8	65.5 (62.2)
CCUAGG	-54.1	-47.7 (-51.8)	-149.1	-129.4 (-143.0)	-7.80	-7.57 (-7.49)	50.0	49.8 (48.1)
CGCGCG	-54.5	-52.4 (-58.1)	-146.4	-138.4 (-156.8)	-9.12	-9.47 (-9.40)	57.8	61.2 (58.5)
CGGCCG	-54.1	-54.6 (-59.3)	-142.6	-142.9 (-158.6)	-9.90	-10.27 (-10.14)	63.2	65.5 (62.2)

CUGCAG	-55.4	-50.4 (-53.1)	-155.7	-139.1 (-147.8)	-7.11	-7.27 (-7.28)	45.3	47.2 (46.6)
GACGUC	-58.1	-55.0 (-54.7)	-163.5	-154.5 (-153.6)	-7.35	-7.07 (-7.02)	46.2	45.1 (45.1)
GAGAGA/	-62.0	-55.1 (-51.0)	-178.1	-155.7 (-142.7)	-6.95	-6.80 (-6.67)	40.6	41.0 (40.6)
GAGCUC	-62.3	-56.0 (-57.1)	-175.3	-155.2 (-159.0)	-7.98	-7.87 (-7.76)	48.7	49.7 (49.0)
GAGGAG/	-55.7	-54.0 (-55.6)	-152.2	-147.7 (-153.4)	-8.50	-8.20 (-8.03)	50.9	49.6 (48.2)
GCAACG/	-50.6	-49.5 (-50.6)	-140.5	-137.4 (-140.5)	-7.01	-6.90 (-6.97)	42.6	42.1 (42.6)
GCAUCG/	-51.9	-51.7 (-54.2)	-143.9	-143.2 (-151.2)	-7.26	-7.30 (-7.25)	44.1	44.4 (43.9)
GCAUGC	-62.3	-55.1 (-56.4)	-177.2	-153.2 (-157.2)	-7.38	-7.57 (-7.64)	45.7	48.1 (48.3)
GCCGCG/	-59.7	-56.6 (-60.8)	-157.4	-147.7 (-161.4)	-10.88	-10.80 (-10.73)	63.9	65.1 (62.7)
GCCGGC	-62.7	-60.8 (-63.6)	-166.0	-159.7 (-168.8)	-11.20	-11.27 (-11.20)	67.2	68.4 (66.6)
GCGCCG/	-57.9	-56.6 (-60.8)	-151.3	-147.7 (-161.4)	-10.91	-10.80 (-10.73)	65.2	65.1 (62.7)
GCGCGC	-66.0	-58.6 (-62.3)	-178.5	-155.2 (-167.0)	-10.62	-10.47 (-10.46)	62.1	64.7 (63.1)
GCGCGG/	-71.2	-56.6 (-60.8)	-192.7	-147.7 (-161.4)	-11.38	-10.80 (-10.73)	61.9	65.1 (62.7)
GCGGCG/	-58.5	-56.6 (-60.8)	-155.0	-147.7 (-161.4)	-10.40	-10.80 (-10.73)	61.8	65.1 (62.7)
GCGUCG/	-52.4	-53.7 (-56.4)	-140.6	-145.1 (-153.8)	-8.76	-8.70 (-8.64)	53.7	52.8 (51.9)
GCUACG/	-58.0	-48.1 (-51.5)	-162.7	-131.5 (-142.2)	-7.56	-7.30 (-7.34)	45.0	44.9 (44.9)
GCUAGC	-59.1	-51.7 (-54.8)	-165.1	-141.6 (-151.4)	-7.92	-7.77 (-7.81)	49.3	50.1 (49.8)
GGAUCC	-53.7	-56.7 (-57.4)	-149.1	-157.1 (-160.0)	-7.44	-7.97 (-7.80)	47.6	50.1 (48.9)
GGCGCC	-67.8	-60.8 (-63.6)	-182.0	-159.7 (-168.8)	-11.33	-11.27 (-11.20)	65.2	68.4 (66.6)
GGCGCG/	-63.5	-56.6 (-60.8)	-170.1	-147.7 (-161.4)	-10.78	-10.80 (-10.73)	61.6	65.1 (62.7)
GGUACC	-54.9	-52.9 (-53.7)	-153.4	-145.5 (-147.8)	-7.35	-7.77 (-7.81)	46.6	49.8 (49.9)
GUCGAC	-53.6	-55.0 (-54.7)	-150.1	-154.5 (-153.6)	-7.09	-7.07 (-7.02)	45.3	45.1 (45.1)
GUGCAC	-59.6	-55.6 (-55.0)	-167.5	-155.2 (-152.6)	-7.65	-7.47 (-7.60)	47.7	47.4 (48.4)
GUGGUG/	-48.8	-53.6 (-53.5)	-132.7	-147.7 (-147.0)	-7.67	-7.80 (-7.87)	47.4	47.2 (47.6)
GUGUCG/	-50.9	-52.2 (-52.7)	-140.9	-145.1 (-146.6)	-7.18	-7.20 (-7.21)	43.7	43.7 (43.8)
UCAUGA	-41.9	-53.3 (-44.1)	-121.2	-157.1 (-127.4)	-4.31	-4.57 (-4.60)	27.2	30.7 (29.5)
UCCGGA	-51.9	-59.0 (-51.2)	-142.3	-163.6 (-139.0)	-7.79	-8.27 (-8.16)	50.1	51.3 (52.7)
UCGCGA	-48.9	-56.8 (-50.0)	-135.7	-159.1 (-137.2)	-6.85	-7.47 (-7.42)	44.6	47.1 (48.3)
UCUAGA	-36.5	-49.9 (-42.5)	-101.8	-145.5 (-121.6)	-4.95	-4.77 (-4.77)	31.0	31.5 (30.5)
UGAUCA	-44.7	-51.1 (-44.1)	-128.0	-149.4 (-127.4)	-5.05	-4.77 (-4.60)	32.6	31.6 (29.5)
UGCGCA	-51.5	-55.2 (-50.2)	-139.7	-152.0 (-136.2)	-8.22	-8.07 (-8.00)	53.1	51.1 (52.0)
AAGGAGG/	-58.7	-59.5 (-59.7)	-158.6	-161.2 (-162.1)	-9.54	-9.50 (-9.42)	56.2	55.8 (55.1)
ACUGUCA*/	-52.2	-61.2 (-55.5)	-142.9	-171.2 (-152.9)	-7.92	-8.10 (-8.14)	48.2	47.5 (48.7)
AGUCUGA/	-51.5	-62.5 (-56.6)	-141.8	-175.1 (-156.1)	-7.52	-8.20 (-8.22)	45.7	47.8 (49.0)
GACUCAG/	-64.1	-62.5 (-64.1)	-177.5	-171.9 (-177.1)	-9.05	-9.20 (-9.12)	52.0	53.2 (52.4)
GAGUGAG/	-70.5	-62.5 (-64.1)	-196.0	-171.9 (-177.1)	-9.71	-9.20 (-9.12)	53.7	53.2 (52.4)
GUCACUG/	-57.8	-62.3 (-63.0)	-158.6	-171.9 (-173.9)	-8.62	-9.00 (-9.04)	51.1	52.1 (52.2)
AACUAGUU	-54.6	-56.9 (-54.0)	-153.0	-163.6 (-153.6)	-7.16	-6.17 (-6.41)	45.7	39.7 (41.2)
AAUGCAUU	-59.8	-59.8 (-57.1)	-169.7	-173.2 (-164.0)	-7.18	-6.07 (-6.28)	45.0	39.1 (40.1)
ACCUUUGC/	-77.4	-67.9 (-66.9)	-215.3	-185.7 (-182.1)	-10.64	-10.30 (-10.43)	56.3	57.5 (58.4)
ACUAUAGU	-59.2	-57.5 (-57.5)	-168.4	-163.6 (-162.8)	-6.98	-6.77 (-6.98)	44.0	43.0 (44.2)
ACUUAAGU	-47.2	-56.9 (-54.0)	-132.4	-163.6 (-153.6)	-6.16	-6.17 (-6.41)	40.3	39.7 (41.2)
AGAGAGAG/	-73.7	-70.3 (-71.9)	-201.7	-191.2 (-196.9)	-11.12	-11.00 (-10.83)	59.6	60.2 (58.9)
AGAUAUCU	-64.5	-61.3 (-61.2)	-186.8	-175.2 (-175.0)	-6.58	-6.97 (-6.97)	41.4	43.7 (43.7)
AGUUAACU	-52.4	-56.9 (-54.0)	-148.5	-163.6 (-153.6)	-6.36	-6.17 (-6.41)	41.1	39.7 (41.2)
AUACGUAU	-54.4	-56.0 (-56.5)	-154.2	-161.0 (-162.0)	-6.53	-6.07 (-6.28)	42.0	39.2 (40.4)
AUCUAGAU	-59.9	-61.3 (-61.2)	-169.9	-175.2 (-175.0)	-7.20	-6.97 (-6.97)	45.1	43.7 (43.7)
AUGCGCAU	-64.4	-68.8 (-69.0)	-174.8	-189.4 (-189.6)	-10.17	-10.07 (-10.20)	60.3	58.2 (58.7)
AUGUACAU	-55.9	-60.9 (-59.1)	-159.3	-175.2 (-168.6)	-6.49	-6.57 (-6.81)	41.7	41.6 (43.0)
CAAAAAAG/	-53.8	-51.1 (-51.4)	-158.7	-149.9 (-150.5)	-4.61	-4.60 (-4.75)	28.6	28.2 (28.9)
CAUGCAUG	-73.7	-67.6 (-71.8)	-206.3	-187.4 (-200.8)	-9.67	-9.47 (-9.54)	54.9	55.5 (54.5)
CGACGCAG/	-70.5	-71.8 (-77.3)	-187.4	-189.9 (-207.8)	-12.32	-12.90 (-12.83)	67.1	69.5 (66.7)
CUCGCACA*/	-72.6	-73.2 (-73.4)	-195.1	-196.7 (-197.5)	-12.11	-12.20 (-12.13)	64.9	65.2 (64.8)
GAACGUUC	-77.0	-68.2 (-68.3)	-218.3	-191.3 (-191.6)	-9.30	-8.87 (-8.88)	52.3	52.3 (52.5)
GAUAUAUC	-62.0	-59.9 (-64.8)	-180.4	-173.2 (-189.0)	-6.09	-6.17 (-6.14)	39.1	39.6 (39.4)
GAUGCAUC	-72.8	-73.2 (-75.8)	-201.9	-203.5 (-212.0)	-10.12	-10.07 (-10.02)	57.2	56.8 (55.9)
GGCUUCAA*/	-61.6	-69.9 (-67.9)	-165.7	-191.2 (-185.1)	-10.20	-10.60 (-10.54)	59.1	58.3 (58.6)
GUAUAUAC	-63.4	-56.1 (-61.0)	-185.1	-161.6 (-176.8)	-5.94	-5.97 (-6.15)	38.3	38.7 (39.6)
GUCUAGAC	-76.0	-70.3 (-72.7)	-212.5	-193.9 (-201.6)	-10.11	-10.17 (-10.15)	56.2	58.2 (57.5)
GUUCGAAC	-74.2	-68.2 (-68.3)	-211.0	-191.3 (-191.6)	-8.76	-8.87 (-8.88)	50.4	52.3 (52.5)
UAGAUCUA	-60.1	-63.7 (-59.5)	-170.6	-182.3 (-168.8)	-7.25	-7.17 (-7.20)	45.3	44.5 (45.1)
UAUGCAUA	-67.7	-62.8 (-58.9)	-195.0	-180.3 (-167.0)	-7.27	-6.87 (-7.08)	44.4	43.0 (44.4)
UCCUUGCA*/	-70.3	-73.8 (-67.8)	-190.8	-201.5 (-182.5)	-11.09	-11.30 (-11.27)	60.7	60.5 (62.4)
UCUAUAGA	-62.1	-63.7 (-59.5)	-177.7	-182.3 (-168.8)	-6.96	-7.17 (-7.20)	43.5	44.5 (45.1)
UGACCUCA/	-76.1	-75.4 (-70.0)	-205.6	-205.4 (-188.6)	-12.34	-11.70 (-11.51)	64.6	61.9 (62.8)
UUCCGGAA	-67.4	-70.0 (-64.9)	-182.6	-192.6 (-177.0)	-10.79	-10.27 (-10.02)	62.4	58.8 (59.1)
UUGCGCAA	-62.2	-68.4 (-63.9)	-167.6	-188.7 (-174.2)	-10.18	-9.87 (-9.86)	61.2	57.3 (58.6)
UUGGCCAA	-63.7	-70.6 (-65.1)	-169.8	-193.2 (-176.0)	-11.00	-10.67 (-10.60)	65.3	60.6 (62.1)
UUGUACAA	-49.5	-60.5 (-54.0)	-137.8	-174.5 (-153.2)	-6.70	-6.37 (-6.47)	43.6	40.6 (41.5)
CAAAAAAAG/	-59.8	-57.7 (-58.2)	-175.1	-168.3 (-169.5)	-5.47	-5.50 (-5.68)	33.8	33.8 (34.7)
AAGGUUGGAA*/	-75.8	-84.7 (-81.0)	-203.6	-230.9 (-218.9)	-12.69	-13.10 (-13.10)	66.5	64.9 (66.2)

CAUGCG/	-48.6	-48.9 (-52.2)	-134.0	-135.1 (-145.6)	-7.00	-7.00 (-7.01)	42.9	42.8 (42.5)
GAGCUG/	-51.6	-53.2 (-55.1)	-142.2	-145.7 (-152.0)	-7.49	-8.00 (-7.95)	45.5	48.5 (47.9)

GCUGAG/	-55.9	-53.2 (-55.1)	-155.2	-145.7 (-152.0)	-7.72	-8.00 (-7.95)	46.2	48.5 (47.9)
GUGCAG/	-55.9	-53.0 (-54.0)	-155.6	-145.7 (-148.8)	-7.67	-7.80 (-7.87)	46.0	47.3 (47.5)
UAAGGUA/	-51.3	-52.8 (-46.4)	-142.9	-150.9 (-129.8)	-6.95	-6.00 (-6.18)	42.2	36.4 (37.4)
GAGAUCUC	-75.0	-74.1 (-76.5)	-209.2	-205.5 (-213.8)	-10.11	-10.37 (-10.14)	56.5	58.0 (56.4)
GCCAUGGC	-93.9	-79.5 (-83.2)	-254.2	-210.6 (-222.6)	-15.06	-14.17 (-14.16)	71.4	74.1 (72.2)
GCUGCGAC/	-86.2	-78.0 (-81.5)	-233.0	-206.7 (-218.0)	-13.93	-13.90 (-13.89)	67.9	71.5 (70.0)
UCCGCGCA*/	-81.2	-79.3 (-76.3)	-214.6	-209.3 (-199.8)	-14.59	-14.40 (-14.29)	73.2	73.3 (74.3)
CACUG/	-38.8	-38.8 (-39.1)	-114.4	-110.9 (-111.9)	-3.34	-4.40 (-4.45)	16.4	24.0 (24.4)
AGAGAG/	-58.2	-49.4 (-49.0)	-165.7	-138.3 (-137.3)	-6.81	-6.50 (-6.40)	40.7	39.5 (38.9)
AUGCAU	-41.7	-46.6 (-43.5)	-119.2	-136.5 (-126.0)	-4.73	-4.27 (-4.42)	30.1	27.9 (28.1)
CGUACG	-46.6	-44.5 (-48.2)	-133.1	-124.2 (-135.8)	-5.35	-5.97 (-6.01)	34.6	39.1 (39.4)
UGGCCA	-59.9	-57.4 (-51.5)	-164.1	-156.5 (-138.0)	-8.99	-8.87 (-8.74)	55.3	55.3 (56.3)
GCAACGA/	-78.8	-62.8 (-59.3)	-224.0	-174.1 (-162.5)	-9.20	-8.80 (-8.87)	50.2	50.9 (52.3)
AGUAUACU	-53.1	-57.5 (-57.5)	-149.1	-163.6 (-162.8)	-6.80	-6.77 (-6.98)	44.1	43.0 (44.2)
GAGAGAGA/	-91.2	-76.0 (-73.9)	-256.0	-208.6 (-202.3)	-11.80	-11.30 (-11.10)	57.6	59.8 (59.6)
GUGAUCAC	-71.8	-73.7 (-74.3)	-200.9	-205.5 (-207.4)	-9.49	-9.97 (-9.98)	54.4	56.2 (56.2)
A_6U_6	-75.7	-84.9 (-80.2)	-222.5	-253.2 (-236.6)	-6.69	-6.37 (-6.84)	41.0	39.6 (41.4)

Table B.2: Measurement of PairFold accuracy on a set of RNA duplexes.

Table B.3 shows details about the predictions of *PairFold* presented in Table 5.2 from Section 5.5.2. The two sequences in the duplex and their corresponding secondary structures, are separated by a space. For each duplex, the first structure is the predicted secondary structure using *PairFold*, and the second structure is the experimentally determined structure, reported in the referenced paper. The structures are drawn in *dot-brackets* format. The underlined characters in the predicted structures indicate wrong predictions. The underlined characters in the real structures indicate odd pairs.

No.	RNA sequences, predicted structures, real structures
1 pred. real	GCCGUCCCCG CGGGGCUGAUGAGGCCGAAAGGCCGAAACGGC (((((.((((( ))))).......((((....))))...))))) (((((.((((( ))))).......((((....))))...)))))
2 pred. real	UGCAGAUCAUGAGGAU AUCCUUGAUGGCAUGCACUAUGCGCGAUGAUCUGCA .((((((((((((((( ))))))....(((((...)))))...))))))))). ((((((((((.((((( ))))).....((((.....))))...))))))))))
3 pred. real	AAUAAACUCAACGGAGG CCUGCGUUCUGAUGAGUCCGUGAGGACGAAAGUUUACC ..((((((.((((.((( ))).)))).......((((....))))...)))))).. ..((((((.(((((((( )))))))).......((((....))))...))))))..
4 pred. real	GGCCACCUGACAGUCCUCUCC GGAGAGAGAAGUCAACCAGAGAAACACACCAACCCAUUGCACUCCGGGUUGGUGGUAUAUUACCUGGUACGGGGGAAACUUCGUGGUGGCCG (((((((.(((..(((((((( )))))).)).))).(((((..((((.(((((((((..........)))))))))))...))..)))))((((((....))))))))))))). ((((((((((((.(((((((( )))))))))))).((((((((((.((((((((((..........))))))))).).)).)))))))((((((....)))))))))))))).
5 pred. real	GCCGUAGGUUGCCC GGGCGACCCUGAUGAGUUGGGAAGAAACUGUGGCACUUCGGUGCCAGCAACGAAACGGU (((((.(((((((( )))))))).......((((.(......)..((((((....)))))).))))...))))) (((((.(((((((( )))))))).......((((...........((((((....)))))).))))...)))))
6 pred. real	GCCGUAGGUUGCCC GGGCGACCCUGAUGAGUUGGCGGCACUUCGGUGCCGGGAAGAAACUGCAACGAAACGGU (((((.(((((((( ))))))))....((((((..((((((....))))))......)))).)).....))))) (((((.(((((((( )))))))).......((((.((((((....))))))...........))))...)))))
7 pred. real	UCACAGUCCUCUUU GGGAGACGUGGUAUAUUACCUGGUUUCGACCAGAGAAACACACGAAAAAAAAAGAGAGAAGUGAA ((((..((((((((......((((((...((..(((((....)))))..)))).))))......)))))).)).)))). ((((....((((((......((((.........(((((....))))).......))))......))))))....)))).
8 pred. real	AGACAGUCCAGAAA GGGAGUUUCUGAGAAGUCUACCAGAGAAACACACGUUGUGGUAUAUUACCUGGUA ((((..((((((((.....)))))).)).))))(((((.....(((.....))).........))))). ((((....((((((.....))))))....))))(((((.......(((...))).........))))).
9 pred. real	AGACAGUCCAGAAAUCUCCCUCACAGUCCUCUUU GGGAGACGUGGUAUAUUACCUGGUUUCGACCAGAGAAACACACGAAAAAAAAAGAGAGAAGUGAGGGAGAUUUCUGAGA ((((..((((((((((((((((((..((((((((......((((((...((..(((((....)))))..)))).))))......)))))).)).))))))))))))))))).)) ((((....((((((((((((((((....((((((......((((.........(((((....))))).......))))......))))))....))))))))))))))))... AGUCUACCAGAGAAACACACGUUGUGGUAUAUUACCUGGUA .))))(((((.....(((.....))).........))))). .))))(((((.......(((...))).........))))).
10 pred. real	UCAGAAGGCUGUAGACAAAUACUGG CCAGUAUUUGUCCUGAUGAGGCCUCGAGGCCGAAACAGCCUUCUGA ((((((((((((.(((((((((((( ))))))))))))........((((....))))...)))))))))))) ((((((((((((.(((((((((((( ))))))))))))....(((((((....)))))))))))))))))))))

11 pred. real	UUAGGCAUCUCCUAUGGCAGGAAGAAGCGGAGACAGCGACGAAGACCUCCUCAAGGCAGUCAGACUCAUC GGGAACAAAAGCUUAUCUCUGAUGAGGCCUCGAGGCCGAAACU .(((.....(((((((((...........)))).)))))((((....))))...))) ((.(((...... ....))).....((((((....)))))).))) GCCUUGAGGAGGUCUUCGUCGCUGUCUCCGCUUCUUCCUGCCAUAGGAGAUGCCUAA)))))))))))))))))))))))))))).))))))))))))))))))))))))))).)))

Table B.3: Measurement of PairFold accuracy on a set of mRNA target - ribozyme pairs.