Context-free Grammars and RNA Secondary Structure Prediction

James W. J. Anderson

June 16, 2013

Abstract

Prediction of RNA secondary structure from a single sequence, or an alignment of sequences, is a core problem in bioinformatics. Many approaches to RNA secondary structure prediction have been attempted, and probabilistic methods using stochastic context-free grammars (SCFGs) have been one of the more successful tries. In particular, SCFGs can be combined with a molecular evolution model to produce consensus structure predictions which more accurately predict RNA secondary structure than when considering single-sequence prediction. The use of SCFGs in RNA secondary structure prediction, and the potential for further developments make for a truly interesting topic.

In this chapter we discuss the application of SCFGs to RNA secondary structure prediction, from a single sequence, or a single fixed alignment. An introduction to RNA secondary structure prediction is given, some technical issues for SCFGs, such as normal forms and grammar design, are discussed, methods are shown for estimating SCFG parameters. Methods are shown for predicting RNA secondary structures, and some measures are given for analysing SCFG variability. Finally, a brief discussion concerning their predictive quality is had, with some suggestions for further work and web resources given.

Contents

1 Introduction

1.1 RNA Secondary Structure

Ribonucleic acid (RNA) secondary structure prediction is an important problem in molecular biology; almost as soon as RNA started to be sequenced, methods have been established to determine the structure from the sequence of nucleotides. Many models have been developed for RNA secondary structure prediction, and in this chapter one in particular will be examined: SCFGs.

Firstly, we might like to know what defines RNA secondary structure, and why it is important that it is able to be predicted. To look at secondary structure, we start by looking at the structure of deoxyribonucleic acid (DNA), as it is from this RNA forms. DNA is made up of a double helix of nucleotides, adenine (A), thymine (T), cytosine (C) and guanine (G); the helices held together by hydrogen bonds. The nucleotides bond in pairs due to their chemical structure, A bonds with T, C with G.When DNA is copied, only a single strand is copied, using A, C, G, and uracil (U) in place of T, which form the nucleotides of RNA. Single strands of RNA then fold back on themselves to increase thermodynamic stability. It has been found experimentally that AU, CG are base-pairs (echoing the AT, CG pairs found in DNA), but U and G bond also, although with less thermodynamic stability than the AU and CG base-pairs. At this point we introduce 'dot-parenthesis' notation, where a dot represents an unpaired nucleotide, and corresponding parentheses indicated paired nucleotide. The figure below shows an example of a sequence, secondary structure, and graphical representation of the secondary structure, created using the VARNA applet (Darty et al., 2009).

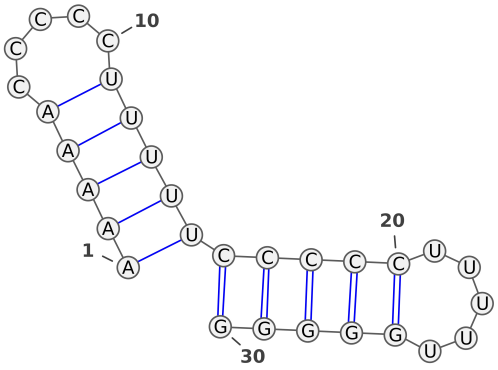


Figure 1: Secondary structure (((((.....)))))((((((.....)))))) of sequence AAAAACCCCCUUUUUCCC-CCUUUUUGGGGG.

Unsurprisingly, there are many biological applications of RNA secondary structure prediction. The most obvious one is in looking at gene expression. RNA can play key roles in gene expression, encouraging or discouraging expression of a given gene, or making post-transcriptional or -translational modifications. Structural flexibility can be important in roles like chaperoning (de Almeida Ribeiro et al., 2011), splice site recognition (Gahura et al., 2011), translation (Loh et al., 2011), and as riboswitches (Vitreschak et al., 2004). Naturally then, by predicting RNA secondary structure, we will have a better idea of the function of the RNA. Similarly, with the advent of next-generation sequencing and transcriptomics, regulatory effects can be looked for by genome-wide prediction of RNA secondary structure (Washietl et al., 2005).

Similarly, we would like to understand RNA folding for RNA sequence design. If there is a specific RNA secondary structure we require in some cellular mechanism, for example in building bio-chemical structures like the DNA box (Andersen et al., 2009), we will need to know what se-

quences could be synthesised in a laboratory to fold into the required RNA structure. Computational applications like Frnakenstein (Lyngs et al., 2012) use RNA folding programs to generate sequences likely to fold into these desired structures. RNA folding methods have also be used in RNA gene finding (Meyer 2010), as structural components can be detected at the genomic level.

The problem changes slightly when homologous RNA sequences are considered. RNA secondary structure is conserved in evolution, so whilst the sequence content might change with mutation and crossover events, in general secondary structure does not, as this would likely change the function of the RNA. So, when considering the secondary structure of our homologous RNAs, it is important that we think about the alignment of the RNA sequences. Equally, when aligning homologous RNA sequences, we must consider the potential secondary structure of the sequences. We would like, therefore, a method, which, given a set of homologous RNA sequences, will given an alignment and a common structure.

Many approaches have been tried in an attempt to produce such an alignment and structure. Early attempts include summing over all possible secondary structures and evaluating them with respect to free-energy functions (Pipas & McMahon, 1975). Since RNA folding helps stability, and less free energy encourages stability, minimising the free energy of the structure was appealing. Another approach formulated the folding of RNA as a loop matching problem, and developed dynamic programming algorithms to find an optimal secondary structure with respect to a simple score function (1 if the two nucleotides can form a base pair, 0 otherwise), hence looking to maximise the number of base-pairs in the structure (Nussinov et al., 1978). Biological and thermodynamical principles have since been used to advanced these energy functions to get more accurate predictions, which have been used to great effect in algorithms such as Mfold (Zuker & Stiegler, 1981, Zuker & Sankoff, 1984, Zuker, 2003), UNAfold (Markham & Zuker, 2008), and RNAfold (Hofacker et al., 1994).

The use of SCFGs in RNA secondary structure prediction was based on the success of Hidden Markov Models (HMMs) in protein and gene modeling (Krogh et al., 1993). It was desired to apply HMMs to annotate RNA secondary structure, but in an HMM, one cannot model global dependencies, which are certainly required for RNA secondary structure. Thus SCFGs, as generalisations of HMMs, were used, initially by Sakakibara et al. (1994). Notably the Pfold algorithm (Knudsen & Hein, 1999, 2003), which used comparative modeling techniques, which was shown to be effective in comparison with other SCFGs (Dowell & Eddy, 2004, Anderson et al., 2012). For reviews on RNA secondary structure prediction, see Gardner & Giegerich (2004), Shapiro et al. (2007), or Xu et al. (2012).

However, one key feature of the basic SCFG method is the inability to predict pseudoknotted structures. A pseudoknot is an RNA secondary structure containing at least two stem-loop structures in which half of one stem is intercalated between the two halves of another stem, for example the structure ((((...[[[...)))))...]]]. RNA structures with pseudoknots in them are highly common (Andronescu et al., 2008). They occur in virtually all classes of RNA, and have important biological function (Chen & Greider, 2005). Pseudoknotted structures are reasonably common in RNA secondary structure, and methods have been taken to adapt the SCFG method to pseudoknotted structures, but we cannot predict them with the straightforward SCFG approach. Consequently, throughout this chapter, I will be referring specifically to non-pseudoknotted structures, unless otherwise stated.

1.2 Data Sources

There are many data sources for RNA sequences with known secondary structures. Two of the main databases are discussed here, but there are a large number of sources available.

RNASTRAND (Andronescu et al., 2008) is an online database of RNA sequences with known secondary structures, which itself takes data from many other sources (Westbrook et al., 2003, Cannone et al., 2002, Andersen et al., 2006, Sprinzl & Vassilenko, 2005, Griffiths-Jones et al., 2005, Brown, 1999, Berman et al., 1992). RNASTRAND is mainly a database of single-sequence entries. At the time of writing, it contains over 4500 structures, of many different types of RNA, such as tmRNA, 16S ribosomal RNA, tRNA, and ribonuclease P RNA. RNASTRAND describes itself as containing simple yet powerful ways of analysing, searching and updating the database. It is the

power of the search which proves particularly useful: being able to search for structural motifs like stems or pseudoknots, and easily filter searched RNA makes RNASTRAND easy to use to obtain a required RNA data set.

On the other hand, Rfam (Griffiths-Jones et al., 2005) is an online database more focused on alignments of RNA families. Particularly noteworthy are the 'seed' alignments, manually curated alignments of RNA sequences designed to be particularly accurate with respect to structure conservation across sequences. So, if you need some particularly trusted alignments for a data set, this is likely where you will find these. At writing, Rfam contains over 2000 families of RNA sequences, each family with a wiki-type documentation to allow for updating of information concerning a given family. Rfam offers searching by family type, and many families actually contain three-dimensional structures too. Both Rfam and RNASTRAND allow users to upload new sequences and new structures to keep data banks dynamic.

1.3 Combinatorics of RNA Secondary Structures

So why is RNA secondary structure prediction a difficult problem? Well, firstly, we need a model for how the RNA folds, which is not trivial in itself. However, even when we have our model, the combinatorics of RNA secondary structures start to become overwhelming, and consequently prediction methods have typically resorted to a function minimisation or maximisation approach by recursing over structures. Here, we will outline the combinatorics of (non-pseudoknotted) RNA secondary structures so that when considering SCFGs, we have better intuition as to why the method has been so successful.

The typical strategy for counting RNA secondary structures is to create a recursion based on the length of the structure concerned. Here we consider all structures which can be represented in dot-parenthesis format, with the paired nucleotides having at least two unpaired nucleotides between them. Note particularly that these do not allow for pseudoknots. However, if we wanted to place additional constraints upon what we considered a valid structure, the counting procedure would still be the same.

For a sequence of length one, the only possible structure is a single, unpaired nucleotide, represented by a dot. We only allow base-pairs exist with at least two nucleotides in between, so for a sequences of lengths two and three, again there is only one allowed structure, all unpaired bases. For a sequence of length 4, we could have either .... or (..) as our structures. This then yields to a general recursion for the number of secondary structures S_n for a sequence of length n, by considering whether the first base is unpaired or paired to some other base k:

$$S_{n+1} = S_n + \sum_{k=0}^{n-2} S_k S_{n-k-1} \quad S_1 = S_2 = S_3 = 1 \tag{1}$$

For larger secondary structures, we can use a large sample approximation to get a asymptotic result for the number of secondary structures

$$S_n \approx 1.104 \times n^{3/2} \times 2.618^n \text{ for large } n \tag{2}$$

It can clearly been seen from this that the number of RNA secondary structures grows exponentially as n becomes large, and in fact it becomes computationally unfeasible to evaluate models on all individual possible structures for longer nucleotide sequences. However, in a similar way to the method used to enumerate RNA secondary structures, dynamic programming, particularly with SCFGs, can be used to recurse over all valid RNA secondary structures in prediction algorithms.

2 Main Body

In this chapter we looks to explore RNA secondary structure prediction, mainly in the framework of SCFGs. Firstly, single-sequence prediction is considered, then prediction from a single fixed alignment. Measures of variability of SCFGs is discussed, and a brief introduction to the thermodynamic

approach of RNA secondary structure prediction given. Note the alignment part of the comparative prediction method is not discussed, as it is beyond the scope of what is done here.

2.1 Context Free Grammars and SCFGs

So why is it that SCFGs are used to model RNA secondary structure? As mentioned in the introduction, using SCFGs was inspired by the use of Hidden Markov Models (HMMs) in protein and gene modeling. The strength of using HMMs was that they could model the coding and non-coding regions of DNA, and something similar was desired for RNA, for example with base-pairing and unpaired regions. However, secondary structures in RNA contain global dependencies, and so it was necessary to use a more complex model than an HMM. SCFGs, simply a generalisation of HMMs, can describe a number of long-range interactions, including several in RNA secondary structure.

To explore the use of SCFGs in RNA secondary structure prediction, we begin by defining context-free grammars and stochastic context-free grammars (Chomsky, 1959). A grammar G is a 4-tuple (N, V, P, S) consisting of the following components:

- A finite set N of non-terminal symbols, throughout denoted 'non-terminals',
- A finite set V of terminal symbols that is disjoint from N, throughout denoted 'terminals',
- A finite set P of production rules,
- A distinguished symbol $S \in N$ that is the start symbol

Furthermore, a grammar is said to be context-free if

- Production rules of the form $A \to aA$ and $A \to a$ for $A \in N$ and $a \in V$ are allowed (requiring solely this condition would give us a regular grammar),
- The left-hand side of each production rule consists of at most a single non-terminal.

Lastly, we can define our context-free grammars to be stochastic: a grammar is said to be stochastic if each there is a probability distribution on the implementation of production rules for each $n \in N$.

Given this definition, it is not immediately obvious why SCFGs might be used for RNA secondary structure prediction. To see this, we will work with an example. Consider the following SCFG:

- Non-terminals $N = \{S\}$
- Terminals $V = \{(,)\}$
- Production rules with probabilities $P = \{(S \to (S), 0.5), (S \to SS, 0.3), (S \to (), 0.2)\}$
- Start symbol S

This grammar might typically be represented in the following shorthand:

$$\begin{array}{lllllll} S & \to & (S) & | & SS & | & . \\ & & 0.5 & | & 0.3 & | & 0.2 \end{array} \tag{3}$$

To produce strings of symbols, we start with the start variable S, then apply a production rule, chosen accordingly to its probability, to change S to, for instance, (S). We then continue applying production rules until we are left with only terminals. For example, to generate the sequence $()()$ we will need to proceed through a sequence containing at least two variables at some point, e.g. as in the derivation $S \to SS \to (S)S \to ()S \to ()(S) \to ()()$. The descendant sequences to each of the two occurrences of S in SS each has to be a balanced sequence of parentheses. However, due to the context-free limitation, once the initial S has been split into SS there is no possibility of coordinating the descendant sequences of each S. Where HMMs allow the modeling of sequential

dependencies the next state only depends on its predecessor CFGs allow the modeling of hierarchical dependencies, i.e. dependencies that have a tree-like structure.

The derivation tree of the string ()() by the grammar can be seen in the figure below. A derivation tree is a tree with internal nodes labeled by variables, leaves labeled by terminal symbols, and where the children of an internal node are the symbols- variables and terminal symbols- of the right hand side of the production rule used to replace the variable in the derivation. The final sequence of the derivation is read off the leaves from left to right. Any derivation in a CFG can be represented by a derivation tree, and as there is an equivalence between trees and sequences with balanced parentheses. The parenthesis example captures exactly the capabilities and limitations of CFGs. In this way, SCFGs generate strings which correspond to RNA secondary structures.

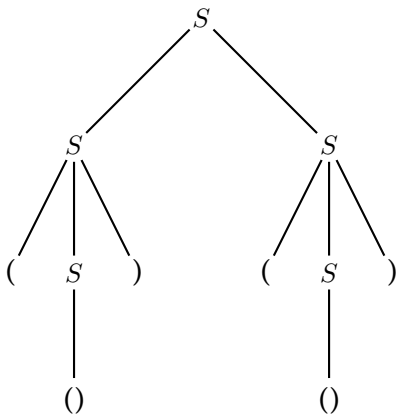


Figure 2: Derivation of the string ()(). The grammar has 2 uses of rule $S->(S)$, 1 use of rule $S->SS$

At this point, it is worth discussing a technical issue- the difference between sequence-generating grammars and structure-generating grammars. We define a sequence-generating grammar to be one where the terminals are the nucleotides A, C, G, and U (and possibly the gap character '-' if considering alignments). The advantage of sequence-generating grammars is that we can have different probability distributions over nucleotides in certain regions of RNA structures (Knudsen & Hein, 1999), for example making stem base-pairs more C-G rich. The disadvantage is that many more parameters are required, and these cannot always be accurately estimated from data. Often sequence generating grammars are abbreviated, giving rules denoted as $S \rightarrow s$, where s denoted a single unpaired nucleotide, so this is in fact four production rules expressed as one.

Structure generating grammars, on the other hand, generate strings of dots and parentheses, and we have already seen an example of this type of grammar. They are simply sequence generating grammars with the nucleotide probabilities constant. This characterisation, though, offers additional mathematical flexibility, as will be seen later. To calculate the probability of, for example, non-terminal S generating a single unpaired A, we take the probability S generates a dot, and multiply it by the nucleotide probability for an A. In this way, the probability of generating a string of nucleotides becomes the product of the grammar probabilities and the nucleotide probabilities. Throughout this chapter, I will mainly use examples from structure-generating grammars, for simplicity, but results will apply equally to both types.

Now that we have concepts of how SCFGs generate strings of nucleotides, and how these correspond to RNA secondary structures, it is worth introducing some measures of success. There have been many metrics on RNA secondary structures designed (Moulton et al., 2000, Shapiro & Zhang, 1990, Freyhult et al., 2005, Hamada et al., 2010), but the ones commonly used are sensitivity, positive predictive value (PPV), and F-score- all of which are defined over predicting base-pairs correctly.

Before considering sensitivity, PPV, and F-score, we must define some intermediaries. Firstly,

we define true positives (TP) to be the number of base pairs which are in the correct structure and also the predicted structure. We define false positives (FP) to be the number of base-pairs which are in the predicted structure, but not in the correct structure. Lastly, we define false negatives (FN) to be the number of base-pairs which are in the correct structure but not in the predicted structure. This then gives us values for sensitivity, PPV, and F-score:

$$\begin{aligned} \text{Sensitivity} &= \frac{\text{TP}}{\text{TP} + \text{FN}} \\ \text{PPV} &= \frac{\text{TP}}{\text{TP} + \text{FP}} \\ \text{F-score} &= \frac{2 * \text{TP}}{2 * \text{TP} + \text{FN} + \text{FP}} \end{aligned}$$

Note that sensitivity, PPV, and F-score are all between 0 and 1. For sensitivity, a 0 shows none of the base-pairs predicted are there, and a 1 indicates all base-pairs in the true structure have been predicted. For PPV, a 0 shows that all base-pairs have been predicted incorrectly, whilst a 1 indicates all predicted base-pairs are correct (by convention, if no base-pairs are predicted we use a PPV of 1).

These measures are hardly ideal measures for RNA secondary structure prediction accuracy. Firstly, note that these measures were originally developed with binary classifiers in mind, where predictions made were independent. However, with base-pairs, if one prediction is wrong, it can affect the correctness of other predicted pairs, by blocking them, for instance. One you start to go wrong with a prediction, the sensitivity and PPV suffer quickly. Secondly, though, note that these measures have little to do with biological function, and do not reflect the functional landscape. An RNA which has mis-folded by, say, missing one particular base-pair, may keep the same function, but if a different base-pair is missing, the function may completely change. Of course, it is not possible to create a measure with this taken account for, but additional realism would with accuracy measures would greatly benefit RNA secondary structure prediction method design.

2.2 Normal Forms

This section forms a more technical section discussing CFG normal forms, null variables, null productions, and pseudoknotted structures. Readers more interested in the practical side of RNA secondary structure prediction using SCFGs may therefore want to skip this section, at least on first reading.

2.2.1 Chomsky Normal Form

The definition of context-free grammars allow arbitrary right hand sides of production rules- for example, we can have the rule $S \to x$, where x is any collection of non-terminals and terminals. Just as there were several possible ways we could restrict regular grammars without changing expressibility (left or right, extended or not), there are several ways the right hand side of the productions of a grammar can be restricted without losing expressibility. These are known as normal forms, and requiring grammars to be on normal form simplifies algorithms as only the restricted set of production types need to be considered.

We will use the Chomsky normal form (CNF) (Chomsky, 1959). Many of the algorithms used are originally designed to work on grammars in CNF, so it is important to understand how to change the grammar being used to CNF. A grammar is said to be in CNF if all of its production rules are of the form:

- $A \to BC$ for $A \in N$ $B, C \in N \backslash \{S\}$
- $A \to a$ for $A \in N$, $a \in V$
- $S \to \epsilon$, where S is the start symbol and ϵ the empty string

Any context-free grammar can be converted to a CNF grammar. For right hand sides of production rules with a sequence of more than two symbols we can break the sequence into the constituent symbols by introducing auxiliary variables. If a symbol in a right-hand side of length two is a terminal symbol we can replace it with an auxiliary variable that can only be replaced with this terminal symbol, and if it is the special start variable S we can replace it with an auxiliary variable with exactly the same production rules as S. However, when the right hand side is shorter than two symbols but the production rule is not in concordance with the CNF restrictions, we need to proceed with a bit more caution.

An empty sequence right hand side, or null production, is only allowed for the special start variable S. To eliminate all other null productions we first identify all nullable variables, i.e. variables that can in one or more steps be replaced with the empty sequence. A variable U is nullable iff

- $U \to \epsilon \in \mathrm{P}$
- $U \to V_1 \dots V_k$ and $V_1, \dots, V_k$ are all nullable

Based on this recursive rule we can formulate Algorithm 1 to efficiently determine the set of nullable variables. Once these have been identified, for each nullable variable, with the exception of S, we remove the null production for this variable, and for every production where it occurs on the right hand side we make two copies of this production: one where the variable still occurs and one where it has been removed from the right hand side sequence. When the right hand side consists of just a single variable remember that right hand sides with just a single symbol are required to be a terminal symbol for grammars in CNF the transformation is even less complicated. All we need to do is copy all the productions of the right hand side variable to the left hand side variable.

Algorithm 1 Nullable variables

```
N = ∅
repeat
  for U → x ∈ P do
    if x does not contain terminal symbols or variables not in N then
      Add U to N
until no further variables are added to N in this iteration
```

This all appear simple enough, so why the warning to proceed with caution? As long as only the language of a grammar, i.e. the set of finite sequences it can generate, is of concern, the last two types of transformations do not require special attention. However, in bioinformatics mostly parameterised, in particular stochastic, context-free grammars are used.

Manipulation of probabilities can easily be included in the first set of transformations mentioned: breaking a long right hand side sequence into pairs, replacing a terminal symbol in a pair with a variable, and adding a copy of S for right hand side use. The only part not strictly trivial would be tying of the probabilities between the productions of S and the copy of S when inferring probabilities from data.

The manipulations eliminating null productions and replacement productions, on the other hand, require more complicated manipulation of probabilities. For an already parameterised SCFG the new probabilities, after elimination of replacement productions, can be described by equations linear in (algebraic) variables describing the probability of eventually replacing variable U with variable V by a finite series of replacement production steps. This, in turn, can be solved to find the probabilities of the transformed grammar. Inferring probabilities from data, however, can become a difficult if not impossible task, if parameters are required to reflect the structure of the original grammar (as will usually be the case). Under maximum likelihood estimation we would need to maximise an equation system similar to the one for parameterised grammars, but with the parameters of the original grammar occurring polynomially, rather than linearly, as variables.

Elimination of null productions is potentially even more complicated when the grammar is stochastic. Even when the grammar is already parameterised, determining probabilities of the transformed grammar may not be straightforward. If the grammar has a production U V W where

both V and W are nullable, the probability that U is eventually replaced by the empty sequence depends on the product of the probabilities that V and W are eventually replaced by the empty sequence. If there are no cycles in these dependencies, the nullability probability can easily be determined for every variable. However, with cyclic dependencies the nullability probabilities may have to be specified by a quadratic equation system, which in general can be hard to solve (Hstad et al, 1993). Because of these complications, when designing a SCFG the best advise is, if at all possible, to avoid introducing replacement productions with cyclic structure and in general null productions.

Algorithm 2 Eliminating replacement productions

repeat
  for $U \rightarrow V \in$ P **do**
    for $V \rightarrow x \in$ P **do**
      Add $U \rightarrow x$ to P
until no updates where required in this iteration
Eliminate all $U \rightarrow V$ productions from P

2.2.2 Double Emission Normal Form

However, while CNF is a convenient form for SCFGs in terms of algorithmics, it does not help particularly for RNA secondary structure prediction. To see why, consider the following sequence-generating grammar in CNF:

$$\begin{array}{lcl} S & \rightarrow & SS \mid AC \mid BC \\ A & \rightarrow & BS \\ B & \rightarrow & s \\ C & \rightarrow & d \end{array}$$

This grammar can then produce strings of s and d. However, there is no way of knowing which d are intended to be paired together. Equally, the grammar can produce strings which do not correspond to RNA secondary structures, like *sd-* the d must be paired to something to create a valid structure.

Consequently, when considering RNA secondary structure prediction, double emission normal form (Anderson et al., 2012) is much more convenient to use. We restrict rules to be of the following form

$$\begin{array}{lcl} U & \rightarrow & VW \\ U & \rightarrow & (V) \\ U & \rightarrow & . \end{array} \tag{4}$$

for non-terminals U,V,W. Double emission normal form maintains all of the flexibility of CNF, as, similarly, any RNA grammar (that is, one that does not produce parentheses that do not match) in CNF can be expressed in double emission normal form, and vice versa, with the exception of being able to generate the empty string. Given the similarity of the two forms, null and replacement productions can be dealt with in a similar fashion.

Furthermore, note that the rules $U \rightarrow \epsilon$ and $U \rightarrow V$ are not allowed. This was originally to prevent cyclical productions from existing, that is having both of the rules $U \rightarrow V$ and $V \rightarrow U$, as many of the algorithms for SCFGs will become stuck in infinite loops if both of these rules exist. Similarly, cyclical productions could come about if we have the rules $U \rightarrow VW$, $V \rightarrow UW$, and $W \rightarrow \epsilon$. However, when if we are designing a grammar by hand, we are unlikely to put in cyclical productions, so these rules could be allowed, if we wanted the extra flexibility. However, double emission normal form in practice allows enough flexibility to predict all types of RNA secondary structures seen.

2.2.3 Pseudoknots

As discussed previously, all grammars and normal forms constructed have not been able to predict pseudoknotted structures. The basic structure of a pseudoknot can be represented as $\{w|w$ has the form $[i(j]i)j\}$. These can, in fact, never be constructed with SCFGs since the form of a pseudoknot is not context-free by the Pumping Lemma (Brown & Wilson, 1996). Hence one cannot use SCFGs in their pure form to predict pseudoknots. Various approaches have been tried to address this problem.

Intersections of SCFGs can be used to produce pseudoknotted structures (Brown & Wilson, 1996, Lefebvre, 1996). Brown and Wilson (1996) use an intersection of two SCFGs, where the structure was classified as a pseudoknot if it can be identified as having one helix which has high probability under one statistical model and simultaneously having high probability under the other model. Lefebvre (1996) uses m-tape context-free grammars, a similar extension in which multiple rows of grammars can evolve and combinations of these grammars taken to represent secondary structure. Although the method uses SCFGs to predict pseudoknots, the intersecting approach is a heuristic approach, and guaranteed optimality of the resulting parse is unfortunately sacrificed, leading to less accurate predictions.

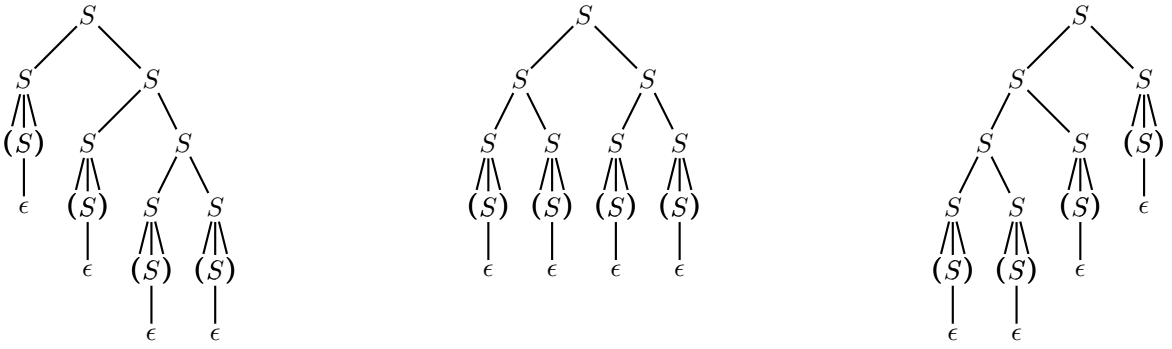
Alternatively, one can leave the framework of SCFGs, whilst still using grammars (Rivas & Eddy, 2000). A formal grammar structure was developed which had a one-to-one correspondence with a pseudoknot folding algorithm (Rivas & Eddy, 1999), which used minimum free-energy structures to find optimal secondary structure. Although this method included context-free grammars, it extended the grammar by using special non-terminal symbols which dictated specific rearrangments of substrings, taking the method out of a SCFG. Similarly, another method of pseudoknot prediction from SCFGs is to use the posterior probabilities found by SCFGs, but decode them in a different manner, such as iterative helix production. There is scope for further research to be done here.

2.3 Ambiguity, Completeness, and Grammar Design

2.3.1 Ambiguity and Completeness

Grammar ambiguity is concerned with SCFGs producing more than one derivation for a given structure. We say a CFG is ambiguous if the grammar has two or more different derivations for a given string. With an unambiguous grammar, there is a one-to-one correspondence between derivations and RNA secondary structures. Consequently, statements we make about the probability distribution on derivations correspond to statements about the probability distribution on RNA secondary structures. However, with ambiguous grammars, we cannot make that interpretation of the probability distribution on derivations.

For example, recall the example grammar used previously. Three different derivation trees for the sequence $()()()()$ are shown as an example in the figure below. These correspond to the leftmost derivations $S \Rightarrow SS \Rightarrow ()S \Rightarrow ()SS \Rightarrow ()()S \Rightarrow ()()SS \Rightarrow ()()()S \Rightarrow ()()()()$, $S \Rightarrow SS \Rightarrow SSS \Rightarrow ()SS \Rightarrow ()()S \Rightarrow ()()SS \Rightarrow ()()()S \Rightarrow ()()()()$, and $S \Rightarrow SS \Rightarrow SSS \Rightarrow SSSS \Rightarrow ()SSS \Rightarrow ()()SS \Rightarrow ()()()S \Rightarrow ()()()()$. Observe that though the trees are similar and utilises each of the three production rules the same number of times, they are still distinctly different.



Given that it is hard to parse an ambiguous grammar's probability distribution, efforts have been made to avoid ambiguity in RNA secondary structure prediction grammars (Reeder et al., 2005, Branrand et al., 2007), despite it being an undecidable problem (Hopcroft & Ullman, 1979). However, it seems that, practically, CFG ambiguity is less of a problem than it might look from a theoretical standpoint. Small ambiguous grammars have been found to perform poorly (Dowell & Eddy, 2004), but this was perhaps due to fundamental limitations in the grammar design than the ambiguity. However, many larger ambiguous grammars have been found that perform as well, or better, than the best unambiguous grammars (Anderson et al., 2012). Even though potentially sub-optimal predictions were chosen, the predictions were still good. However, ambiguity is still something that should be considered in SCFG design.

Completeness can be seen as the complement to ambiguity, considering instead whether or not a grammar can form enough structures, as opposed to too many. For RNA secondary structure, we say a grammar is complete if it has a derivation for all possible structures which have no hairpins shorter than length two, that is, it can generate all structures that we consider valid. Again, we might consider that completeness is a desirable property for a grammar to have, but again there are examples of incomplete grammars still performing well (Anderson et al., 2012). However, the structures which these grammars could not generate were rarely seen in biology, suggesting that the set of structures they could generate was perhaps a good approximation to the set of structures we consider valid.

Note a complete, unambiguous grammar cannot be simply modied without compromising one of the properties. Adding production rules, if they are ever used, will create ambiguity by providing additional derivations. Equally, removing production rules will create incompleteness, unless the rule is never used in a derivation, as the original grammar is assumed unambiguous. Moreover, grammars that are unambiguous and complete are vastly outnumbered by grammars that are not, so designing grammars which are both unambiguous and complete can be practically difficult.

2.3.2 Lightweight Grammar Design

The last thing to discuss before we move onto considering algorithms for SCFGs is the design of the grammar. Up to this point, we have been dealing with example grammars only, for simplicity. However, when using a SCFGs to predict RNA secondary structure optimally, we will need more than this. Several attempts have been made to find out which set of non-terminals and production rules will create the best CFG for RNA secondary structure prediction. Firstly, we focus on the design of lightweight SCFGs, that is grammars with relatively few non-terminals, production rules, and therefore parameters.

Initially the main CFG used was the Knudsen and Hein grammar (Knudsen & Hein, 1999, 2003), hereafter denoted KH99. KH99 was a remarkable simple grammar:

$$\begin{array}{lcll} S & \rightarrow & L & LS \\ & & .131 & .869 \\ L & \rightarrow & s & dF\hat{d} \\ & & .895 & .105 \\ F & \rightarrow & dF\hat{d} & LS \\ & & .788 & .212 \end{array} \tag{5}$$

with d, $\hat{d}$, indicating paired nucleotides, and s indicating an unpaired nucleotide. The grammar generated stem and loop regions, and, although originally designed to be paired with an evolutionary model, performs well on single sequences too (Anderson et al., 2012). Particularly, it has only 3 production rule parameters to be estimated from data, which means it is less likely to suffer from overfitting.

While KH99 was effective, it seems to have been chosen relatively arbitrarily, in that there is little discussion about what motivated the choice of production rules. This problem was addressed by Dowell and Eddy, where nine different lightweight (i.e. a small number of parameters) SCFGs were evaluated for single sequence structure on a benchmark set of RNA secondary structures (Dowell & Eddy, 2004). Lightweight grammars were considered here for their computational advantages and

simplicity, since if one would like to extend SCFGs to model multiple sequences, a combinatorical explosion of parameters can occur if parameterisation is not kept small. The grammars tested were all hand designed, that is that people sat down and tried to figure out which grammars would best produce secondary structure.

Dowell and Eddy (2004) considered two types of grammar: Stacking and Simple. Stacking grammars have production rules such as $P(b,\hat{b})aP(a,\hat{a})\hat{a}$ for nucleotides $a,\hat{a},b,\hat{b}$ which gives a larger set of production rules in the grammar. Simple grammars like KH99 do not have these production rules. One might expect the stacking grammars to have better predictive accuracy due to the added strength of the model, but it is quite possible that as there are many more parameters, which must be estimated from training data, that this is not the case. They found that several of the grammar performed comparably on their test data set, and the stacking grammars generally outperformed the non-stacking grammars, but KH99 performed the best.

This search for good grammars was further extended by used of an evolutionary algorithm to enable a computational search (Anderson et al., 2012). A population of small, very simple grammars was used for an initial population, then grammars were mutated, for example by adding a production rule or non-terminal, and bred, for example by combining production rules. Grammars were then selected for, with a fitness based on the ability to correctly predict RNA secondary structures. This process of mutation, breeding and selection was then iterated for a fixed number of generations, and the grammars tested on an independent data set, to avoid over-fitting. Furthermore, a brute force search of all small grammars, that is with at most two non-terminals, was done, but most performed poorly, perhaps as expected.

Many grammars, however, were found from this evolutionary search which performed as well, or slightly better, than KH99. In particular, a visual inspection of the grammars found yielded no suggestions as to why they were so good at secondary structure prediction- they seemed relatively random in choices of non-terminals and production rules. For example, the largest, most complicated, and best grammar is shown below.

$$
\begin{aligned}
A &\rightarrow DE|AB|BA|AH|.|(F)|(H)\\
B &\rightarrow .\\
C &\rightarrow (H)\\
D &\rightarrow BB|AC\\
E &\rightarrow .|(H)\\
F &\rightarrow FB|CF|.\\
G &\rightarrow GH|(H)|(C)\\
H &\rightarrow FA|AF|HH|(B)|(H)
\end{aligned}
$$

Overall this gives an interpretation of grammars as similar to machine learning approaches, in style- simply structures that learn a probability distribution from data and regurgitate it. There seem to be many grammars possible that perform similarly for the RNA secondary structure prediction problem, and it is the method of SCFG prediction, via either inside-outside or CYK, as opposed to grammar design, which seems to be the methodological bottleneck.

2.3.3 Heavyweight Grammar Design

We can, of course, look to design more complicated grammars to more closely mimic RNA secondary structure. In particular, one might try and mimic the thermodynamic model in SCFGs, for example to use the advantages of comparative modeling.

Most approaches using heavyweight SCFGs have been to try to reproduce the thermodynamic model (Nebel & Scheid, 2011, Rivas et al., 2012). We can create SCFGs that have production rules which mirrors the energy rules, and can calculate the full partition function, to aid sub-optimal folding, or simply to incorporate the evolutionary model. However, given results (Nebel & Scheid, 2011) produced by these grammars, they seem no better than the smaller, lightweight SCFGs used in Pfold, or the thermodynamic methods themselves. This gives further credence to the idea that it is not the SCFG design which is important, but the method of decoding, given the probability distribution.

2.4 Algorithms for SCFGs, Parameter Estimation, Structure Prediction

2.4.1 Algorithms for SCFGs

We now move on to some of the main algorithms used for SCFG prediction. Throughout, we assume that SCFGs are in double emission normal form. There are two main sets of algorithms which are used for RNA secondary structure prediction with SCFGs. I will first introduce the algorithms, and then discuss how each one can be applied.

The Cocke-Younger-Kasimi (CYK) algorithm (Younger, 1967) was originally used to determine whether or not a context-free grammar could derive a given string. However, it is easily adapted to be useful for SCFGs, and determines the maximum probability of a derivation of the SCFG for a given string. It is a dynamic programming algorithm, so backtracking through the dynamic programming table will give the SCFG derivation which produced this maximum probability. This derivation can then be used to find the structure with the maximum probability derivation.

For each non-terminal U, string s, and string indices i and j, we define

$$C(U,i,j) = \begin{cases} \mathbb{P}\left[U \rightarrow .\right]\ \mathbb{P}[s[i] = .] & \text{if i = j} \\ \max\left\{ \max\limits_{U \rightarrow VW} \max\limits_{i \leq k < j} \mathbb{P}[U \rightarrow VW]\ C(V,i,k)\ C(W,k+1,j), \right. & \\ \left. \max\limits_{U \rightarrow (V)} \mathbb{P}[U \rightarrow (V)]\ C(V,i+1,j-1)\ \mathbb{P}[s[i],s[j] = ()] \right\} & \text{if i < j} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$C(U,i,j)$ is then the maximum probability of U generating the subsequence $s[i]...s[j]$, and in particular, $C(S,1,|s|)$ gives the maximum probability of the SCFG generating the sequence. To compute this table, start with subsequences of length one, then length two, and so on, until you have computed the CYK table for the entire string. Pseudocode for the algorithm can be found in Algorithm 3. As can be seen in the pseudocode, we can find $C(S,1,|s|)$ in time $O(|P||s|^3)$ and space $O(|V||s|^2)$, as we compute the $C(U,i,j)$ values in order of sub-sequences of s of increasing length.

Algorithm 3 CYK algorithm

for $i = 1$ **to** $|s|$ **do**
  for $U \in$ V **do**
    $C(U,i,i) = \mathbb{P}\left[U \rightarrow .\right]\ \mathbb{P}[s[i] = .]$
for $l = 1$ **to** $|s| - 1$ **do**
  for $i = 1$ **to** $|s| - l$ **do**
    for $U \in$ V **do**

$$C(U,i,i+l) = \max\left\{ \max_{U \rightarrow VW} \max_{i \leq k < l} \mathbb{P}[U \rightarrow VW]\ C(V,i,k)\ C(W,k+1,l), \right.$$
$$\left. \max_{U \rightarrow (V)} \mathbb{P}[U \rightarrow (V)]\ I(V,i+1,l-1)\ \mathbb{P}[s[i],s[l] = ()] \right\}$$

The inside and outside algorithms (Lari and Young, 1990) are similar to the CYK algorithm, but instead of calculating the maximum probability of a derivation, they calculate the total probability of generating subsequences. First we define the inside table, for each non-terminal U, string s, and

string indices i and j:

$$I(U,i,j) = \begin{cases} \mathbb{P}\left[U \rightarrow .\right]\ \mathbb{P}[s[i] = .] & \text{if i = j} \\ \sum\limits_{U \rightarrow VW} \sum\limits_{i \leq k < j} \mathbb{P}[U \rightarrow VW]\ I(V,i,k)\ I(W,k+1,j) & \\ + \sum\limits_{U \rightarrow (V)} \mathbb{P}[U \rightarrow (V)]\ I(V,i+1,j-1)\ \mathbb{P}[s[i],s[j] = ()] & \text{if i < j} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

and similarly, the outside table:

$$O(U,i,j) = \begin{cases} 1 & \text{if U = S, i = 1, j = |s|} \\ \sum\limits_{V \rightarrow UW} \sum\limits_{k > j} \mathbb{P}[V \rightarrow UW]\ O(V,i,k)\ I(W,j+1,k) & \\ + \sum\limits_{V \rightarrow WU} \sum\limits_{k < i} \mathbb{P}[V \rightarrow WU]\ O(V,k,j)\ I(W,k,i-1) & \\ + \sum\limits_{V \rightarrow (U)} \mathbb{P}[V \rightarrow (U)]\ O(V,i-1,j+1)\ \mathbb{P}[s[i],s[j] = ()] & \text{if i < j} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$I(U,i,j)$ then gives the total probability of U generating the subsequence $s[i]...s[j]$, and $O(U,i,j)$ the total probability of generating everything except the subsequence $s[i]...s[j]$. Hopefully, considering the derivation trees required to produce the subsequence $s[i]...s[j]$, it is intuitive why the tables are named 'inside' and 'outside'. The psuedocode for the inside algorithm is analogous to the CYK algorithm pseudocode. The outside is computed in a similar manner, but instead starting from subsequences of length $|s|$, and working down to subsequences of length one. Again, the inside and outside algorithms are computed in time $O(|P||s|^3)$ and space $O(|V||s|^2)$.

2.4.2 Parameter Estimation

For parameter estimation, we require a set of RNA sequences with known secondary structures, that is, in the language of machine learning, we use supervised training. Interestingly, if you try to train the SCFG in an unsupervised manner, that is just using RNA sequences, the SCFG will choose parameters which enable it to produce long strings at lost 'cost'. To illustrate this, consider the start variable for KH99. The probabilities for the production rules $S \rightarrow LS$ and $S \rightarrow L$ given are 0.869 and 0.131 respectively (Knudsen and Hein, 1999). However, consider also a SCFG with rules $S \rightarrow LS$, $S \rightarrow L$ and $L \rightarrow .$, with rule probabilities 0.99,0.01, and 1. This latter SCFG will generate strings solely of unpaired bases with very high probability- much higher than KH99 does- and this is, in fact, the sort of SCFG you get when using unsupervised training, as you are looking to maximise the probability of the data. For this reason, we need known secondary structures for parameter estimation.

We need also the nucleotide probabilities for SCFG parameter estimation. When using SCFGs over alignments, an evolutionary model is used, discussed later in this chapter. However, then considering only single sequences, we simply take the frequency at which nucleotides appear in paired and unpaired positions and use these frequencies as probabilities. For example, if we see 100 AU base-pairs out of 500 total base-pairs, then the probability of seeing an AU base-pair would be 1/5.

Now that we have our the nucleotide probabilities and our SCFG algorithms, we are interested in determining the production rule probabilities which will be useful in producing the most accurate structure predictions. We can do this parameter estimation using either the CYK algorithm, or by using the inside and outside algorithms.

For parameter estimation with the CYK algorithm, we first consider unambiguous grammars. Recall that if an unambiguous grammar can derive a string, then it will only have one way of doing

so. So here, we use the CYK algorithm instead to generate the known structure, initially using uniform rule probabilities, and an indicator function in the algorithm to determine whether or not the algorithm is generating the known structure. Thus, by keeping track of the rule frequencies used in the derivation to generate this known structure, we will be able to count the total number of rules used over the whole training data set. Again, we just take the frequencies of rules observed. Taking KH99 as an example, if we saw 200 $S \rightarrow LS$ rules used, and 100 $S \rightarrow L$ rules used, our rule probabilities would be 2/3 and 1/3 respectively.

For ambiguous grammars, we no longer have a single derivation for each structure. So, if we want to estimate parameters on the data set, we must choose at random a single derivation. This is done in the backtracking step of the CYK algorithm: if there is not a single production rule which generates a given subsequence, one is chosen at random. Of course, implementing this does not require knowing whether or not a SCFG is ambiguous a priori, since picking randomly will still be correct for unambiguous grammars. Again, by keeping track of production rules, you can determine production rule probabilities.

For the inside and outside algorithms, we used inside-outside training (Lari and Young, 1990). This is an implementation of expectation-maximisation training (Demster et al. 1977). First we randomly initialise the rule probabilities, then use these to calculate the expected number of rule uses. From the expected number of rule uses, we re-estimate the rule probabilities. This process is then iterated until convergence of rule probabilities. The inside and outside algorithms can be used to calculate the expected rule frequencies. If we let $\mathbb{P}_{old}$ be the old SCFG probabilities, then the new probabilities, $\mathbb{P}_{new}$ are estimated as follows:

$$\begin{aligned}
\mathbb{P}_{new}[U \rightarrow VW] &= \frac{\mathbb{E}[\#\ U \rightarrow VW \text{ is used}]}{\mathbb{E}[\#\ U \text{ is used}]} \\
&= \frac{\sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \sum_{k=1}^{j-1} \mathbb{P}_{old}[U \rightarrow VW] I(V,i,k) I(W,k+1,j) O(U,i,j)}{\sum_{i=1}^{m} \sum_{j=1}^{m} I(U,i,j) O(U,i,j)} \\
\mathbb{P}_{new}[U \rightarrow .] &= \frac{\mathbb{E}[\#\ U \rightarrow . \text{ is used}]}{\mathbb{E}[\#\ U \text{ is used}]} \\
&= \frac{\sum_{i=1}^{m} \mathbb{P}_{old}[U \rightarrow .] \mathbb{P}[s[i] = .] O(U,i,i)}{\sum_{i=1}^{m} \sum_{j=1}^{m} I(U,i,j) O(U,i,j)} \\
\mathbb{P}_{new}[U \rightarrow (V)] &= \frac{\mathbb{E}[\#\ U \rightarrow (V) \text{ is used}]}{\mathbb{E}[\#\ U \text{ is used}]} \\
&= \frac{\sum_{i=1}^{m-2} \sum_{j=i+2}^{m} \mathbb{P}_{old}[U \rightarrow (V)] \mathbb{P}[s[i], s[j] = ()] I(V,i+1,j-1) O(U,i,j)}{\sum_{i=1}^{m} \sum_{j=1}^{m} I(U,i,j) O(U,i,j)}
\end{aligned}$$

There are good reasons why one might want to use one variety of training over the other. The CYK training method is significantly quicker: not only does the algorithm take less time (practically, not in complexity) than inside-outside, but there is only a single iteration required. Thus, for very large SCFGs, or if many are being trained, this may be of benefit. On the other hand, inside-outside training deals with ambiguous SCFGs much better, considering all derivations they can make instead of a random one. Similarly, inside-outside training is more focused on getting the overall probability distribution correct, as opposed to the simply a maximum-likelihood style estimator the CYK algorithm uses.

Similarly, we might wonder how stable the parameter estimation procedure is, that is, if we choose slightly different parameters, might the SCFG predict entirely different structures? This

was in particular an important question when considering designing grammars with an automated approach (Anderson et al., 2012). When the data set for parameter estimation was made to be a random subset of the whole data set, the stability of parameterisation could be tested. In fact, with only a handful of sequences, the parameters of small SCFGs varied very little, but with slightly larger SCFGs, the variation was more- not surprising, given there are more production rules to estimate. With a data set of around 100 sequences, though, the variation was very small. This idea has been further confirmed by looking at disturbances in the sampling probabilities of a heavyweight SCFG, and it's corresponding prediction accuracy, and finding that the structures produced do not vary particularly (Scheid & Nebel, 2012). Thus, we can be reasonably confident when estimating SCFG probabilities that we will have reached a reasonable parameterisation.

2.4.3 Secondary Structure Prediction

Now we have our SCFG with a set of parameters found from known structures, we want to begin predicting structures for other sequences. Again, the CYK, inside, and outside algorithms are used in a similar way to the parameter estimation.

Prediction with the CYK algorithm is straightforward. Firstly, we fill in the CYK table using the algorithm above. The value $C(S, 1, |s|)$ gives the maximum probability of the SCFG generating the sequence, and it is the derivation corresponding to this probability which we use for the predicted structure. This derivation, as with the parameter estimation, is found by backtracking through the CYK table. We could also use a modified version CYK algorithm to determine the n-most probable structures (Chappelier & Rajman, 1998).

Prediction with the inside and outside algorithms is a little more difficult. From the inside and outside tables, one can derive the base-pair probability matrix $B(i, j)$, the probability of i pairing with j:

$$\mathbb{B}(i,j) = \frac{\sum_{U \to (V)} O(U,i,j)\mathbb{P}[U \to (V)]\mathbb{P}[s[i], s[j] = ()]I(V, i+1, j-1)}{I(S,1,|s|)} \tag{9}$$

that is, by summing over all rules of type $U \to VW$.

Once we have the base-pair probability matrix, we have a probability distribution over secondary structures, and we can do posterior decoding to obtain our predicted structure. At this point we must decide what we are looking to maximise. Typically, expected positions correctly predicted or expected base-pairs correctly predicted are chosen as the criterion to maximise over. For example, if we are looking to maximise the expected number of positions correctly predicted, we consider

$$E(i,j) = \max \begin{cases} \frac{1}{2}\left(1 - \sum_{1 \le k \le |s|} B(i,k)\right) + E(i+1,j) & (1) \\ B(i,j) + E(i+1,j-1) & (2) \\ \max_{i+1<k<j} [B(i,k) + E(i+1,k-1) + E(k+1,j)] & (3) \end{cases} \tag{10}$$

$E(i, j)$ is the maximum expected number of positions correctly predicted for the subsequence $i...j$. Case (1) considers this maximum having been achieved from i being unpaired, case (2) considers this maximum having been achieved from i being paired with j, and case (3) considers this maximum having been achieved from i being paired with some k in between i and j. $E(1, |s|)$ is then the maximum expected number of positions correctly predicted for the entire sequence, and the structure which produces this maximum, found by backtracking through the $E(i, j)$ table, is the structure predicted.

However, there is no fundamental reason why the posterior base-pair probability matrix should be decoded in this way- we are simply maximising over a function. We could define alternative functions to maximise over to simulate, for instance sub-optimal folding, or to include more biophysical constraints. One simple idea would be to implement iterative helix production, as has been done for the thermodynamic model (Harmanci et al., 2011), to simulate the quicker forming on

helices observed biochemically (Craig et al., 1971). This is an avenue for further research for SCFG methods.

There are some key differences between the CYK and inside-outside methods of prediction (Anderson et al., 2012). The CYK algorithm, predicting structures with the highest probability, predicts structures with a larger proportion of base-pairs. This is in some sense unsurprising when considering the double emission normal form, as forming a base-pair uses only a single rule for creating two positions. Hence the CYK algorithm can predict the structure 'quicker', which typically comes with a higher probability, and so the sensitivity of CYK predictions tends to be relatively higher. Inside-outside, on the other hand, will consider probability contributions from all derivations. Therefore, structures predicted by the inside-outside algorithm tend to have relatively higher PPV, only predicting base-pairs where the majority of derivations (weighted by probability mass) 'agree' on a base-pair being present. In particular, when considering predictions in the context of comparative modeling, this difference can be more pronounced.

2.4.4 Example

At this point many algorithms have been presented, as well as methods of secondary structure prediction for single sequences. Here, I provide an example of implementation of the algorithms and calculations throughout.

Here we will use a sequences from Rfam (Griffiths-Jones et al., 2005). In particular, we will consider RF00390, an upstream pseudoknot domain found in viruses like the tobacco mosaic virus (Felden et al., 1996). The first sequence is as follows:

```
UAAGUUCUCGAUCUUUAAAAUCG
```

In particular, this sequence has a secondary structure which is pseudoknotted,

```
.[[[....((((]]]....))))
```

which means the SCFG method will not be able to predict it correctly. However, this will give an opportunity to show some of the strengths and weaknesses of the method, and demonstrate the accuracy metrics.

Firstly, we will need to convert KH99 to double emission normal form so that we can use the algorithms developed above:

S	$\rightarrow$	.	(F)	LS
		.117	.014	.869
L	$\rightarrow$	.	(F)	
		.895	.105	
F	$\rightarrow$	(F)	LS	
		.788	.212	

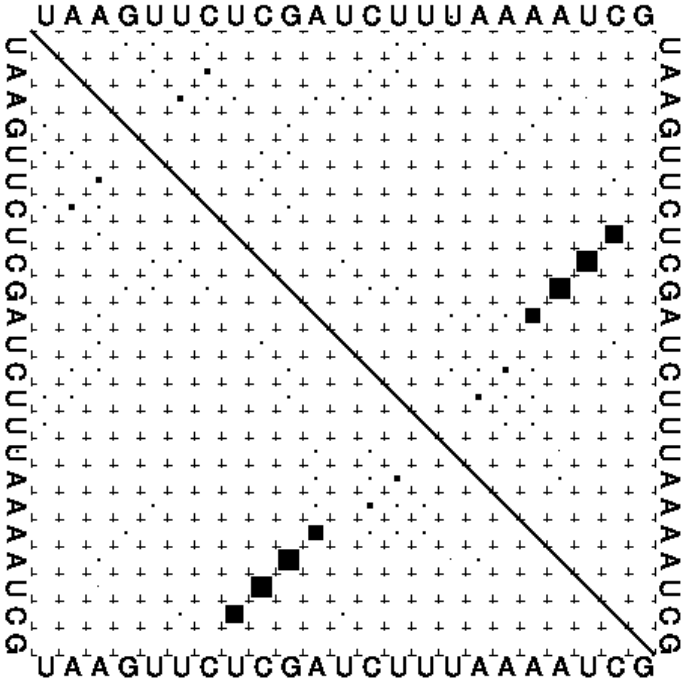
with parameters for the SCFG are taken from the Pfold model. Next, we will need unpaired and paired probabilities. For these, we simply take the frequencies of nucleotides observed in Knudsen and Hein (1999). We have the probability of (A,C,G,U) being unpaired as (0.268, 0.214, 0.267, 0.251), and the probability of (AU,UA,CG,GC,GU,UG) being paired as (0.180, 0.180, 0.272, 0.272, 0.048, 0.048), allowing only canonical base-pairs for simplicity.

Firstly, we consider the CYK algorithm. Now we have the SCFG and nucleotide probabilities, we can calculate the full CYK table for each non-terminal. The value of C(S,1,—s—) here is 9.35e-18, and, backtracking through the CYK table, we then get the structure:

```
CYK Predicted Structure:        ........((((.......))))
True Structure:                 .[[[....((((]]]....))))
```

Inside-outside is very similar. The figure below shows the base-pair probability matrix:

Note that there is practically no probability mass associated with the pseudoknotted base pairs AAG, CUU. Since the SCFG cannot produce both sets of base-pairs, it must 'choose' between the



two sets, and we can see the choice is for the 'regular' pairs. Looking at the base-pair probability matrix produced by inside-outside, we might expect the same structure produced by CYK, and this is in fact what the MEA produces:

```
MEA Predicted Structure:        ........((((.......))))
True Structure:                 .[[[....((((]]]....))))
```

It is not uncommon that CYK and MEA produce the same secondary structure prediction. Furthermore, when they do produce different structures, they often share the same 'core' structure, that is that CYK will produce the same structure as inside-outside, but with a few more base-pairs predicted, and vice versa. This can be seen by considering the normal form and production rules: producing a base-pair will produce two non-terminals. By the SCFG predicting many base-pairs, it will use fewer production rules, and so, in general, a higher probability structure will result. This is the type of structure that CYK will choose. Inside-outside cares more about probabilities over all derivations, and so it more selective about the base-pairs it chooses.

Finally, we can calculate the accuracy measures of sensitivity, PPV, and F-score. We have TP=4, FP=0, FN = 3. So the sensitivity is 4/7=0.571, the PPV is 4/4 = 1, and the F-score, the harmonic mean of sensitivity and PPV, is 0.727. For smaller sequences, this is approximately what we might expect, in terms of accuracy.

2.5 Comparative Modelling

The main strength of SCFG prediction methods, at least, over other prediction methods, is their ease to combine with an evolutionary model. As RNA secondary structure has function, we might expect it to be conserved. Therefore, by considering an alignment of homologous sequences, the extra information provided by the homology should be expected to contribute to the model's ability of structure prediction.

We consider evolution in the setting of having one, single alignment. There are methods of dealing with multiple alignments and averaging, but they are beyond the scope of this chapter. Here, we being by discussing the typical evolutionary model (Knudsen and Hein, 1999, 2003) used in RNA secondary structure prediction, then the application to SCFGs, and finally by discussing some more technical issues, such as considering gaps in alignments and tree estimation.

2.5.1 Evolutionary Model and Column Probabilities

Recall that for single sequences we had that the probability of non-terminal U generating an unpaired A was $\mathbb{P}[U \to .] * \mathbb{P}[\text{unpaired } A]$. However, when we are considering an alignment of RNA sequences, we now have columns of nucleotides instead of single nucleotides. If we have a column, say, A and U, the probability of U generating the unpaired column of A and U is now $\mathbb{P}[U \to .] * \mathbb{P}[\text{unpaired column } A, U]$. So the SCFG rule probabilities stay the same, and the same algorithms can be used for prediction, but we just need an expression for the column probabilities.

As with many evolutionary models, we will need simplifying assumptions. We assume that RNA sequences evolved from a common ancestor, and that lineages are independent. Furthermore, we require evolutionary processes in different unpaired or paired positions of the sequence to be independent, so the probability for the whole alignment is the product of the probabilities of individual patterns. We also need the evolutionary processes to be the same within all paired and the same (although possibly distinct) within all unpaired positions. Lastly, we want the model to be time-reversible, so that evolution looks similar forward and backward in time.

A very simple model for column probabilities might be to say that, for example, the probability of an unpaired column of A, A, and U is

$$\mathbb{P}[\text{unpaired column A,A,U}] = \mathbb{P}[\text{unpaired A}]^2 \times \mathbb{P}[\text{unpaired U}] \tag{11}$$

Why might we want this model? Well, if we consider that secondary structure is likely to be conserved. If there is an A in position 1, and a G in position 6, then it is very unlikely that position 1 and 6 pair, even if all the other sequences look like 1 and 6 should pair. The reason for this is that is 1 and 6 are an A and G in one sequence, that sequence will not fold to have a base-pair in positions 1 and 6. Consequently, we might think that this first sequence has a different function, due to its different secondary structure- but this contradicts our assumption that structure is conserved. Thus, requiring base-pairs in all positions where we pose a base-pair is necessary for our model.

However, what this model does assume is that all of the homologous RNA sequences are independent. Our evolutionary assumptions, though, assume that they evolved from a common ancestor, so we would like to include this co-variation in the structure prediction model.

To do this, first we must define a model for nucleotide evolution. We follow the Pfold evolutionary model (Knudsen and Hein, 1999, 2003), which assumes that nucleotides evolution follows a general reversible model (Tavar, 1986), that is according to a continuous-time Markov chain. But the evolutionary model used for RNA secondary structures is slightly different to the classic evolutionary models like the Jukes-Cantor model (Jukes & Cantor, 1969), in that we want paired columns to evolve together, to maintain base-pair complementarity.

We consider first unpaired bases. To model the evolution of an unpaired nucleotide, we need evolutionary rates, that is our Q-matrix in the continuous time matrix. Thus, for unpaired bases, the probability that, say, A evolves to G in time t is

$$\mathbb{P}[\text{A evolves to G in time } t] = (\exp(tQ))_{1,3} \tag{12}$$

where Q is the 4x4 Q-matrix of the continuous time Markov chain, and we are taking the 1,3 entry to correspond to the first nucleotide A, and the third nucleotide, G. The calculation for paired columns is similar, but instead we have a 16x16 Q-matrix to correspond to all possible base-pairs. The probability that an AU base-pair evolves to a CG base-pair is

$$\mathbb{P}[\text{AU evolves to CG in time } t] = (\exp(tQ))_{4,7} \tag{13}$$

where similarly, we are taking the 4,7 entry to correspond to the fourth base-pair AU, and the seventh base-pair CG.

2.5.2 Implementation with SCFGs

Now we have our nucleotide substitution model, we want to calculate the column probabilities. To do so, we follow the Felsenstein pruning algorithm (Felsenstein, 1981). For unpaired columns, we define a vector $p_i = (p_{i_1}, ...p_{i_4})$ for each node in the phylogenetic tree, where p_{i_1} is the probability that the sequence at node i is an A, p_{i_2} is the probability that the sequence at node i is a C, and so on. We begin by setting the leaf probabilities, that is, for the nucleotides we observe. So, if for sequence 1 we observe an A, the probability vector at that node is (1,0,0,0). For a node n_3 with two children n_1 and n_2 with branch lengths t_1 and t_2 respectively, we have that the probability that n_3 is an A is

$$\mathbb{P}[n_3 = \text{A}] = \sum_{x_1 \in \{1,2,3,4\}} \sum_{x_2 \in \{1,2,3,4\}} \exp(-t_1 Q)_{1,x_1} \mathbb{P}[n_1 = x_1] \exp(-t_2 Q)_{1,x_2} \mathbb{P}[n_2 = x_2] \tag{14}$$

where x_1 and x_2 represent the nucleotides of n_1 and n_2 respectively. To get the probability of a column, we also need a prior on nucleotides $(\pi_A, \pi_C, \pi_G, \pi_U)$. This is usually set to the frequencies at which nucleotides are observed in the data set. Now we can finally calculate the column probability:

$$\mathbb{P}[\text{column unpaired}] = \sum_{x_1 \in \{A,C,G,U\}} \pi_{x_1} \mathbb{P}[n_a = x_1] \tag{15}$$

where n_a is the ancestor node. Column probabilities for those columns which form base-pairs are calculated in a similar manner. We define a vector $p_i = (p_{i_1}, ..., p_{i_16})$ for each node in the phylogenetic tree, where p_{i_1} is the probability that the column pair at node i is an AA, p_{i_2} is the probability that the sequence at node i is a AC, and so on. We begin by setting the leaf probabilities, that is, for the nucleotides we observe. So, if for sequence 1 we observe an AU, the probability vector at that node is (0,0,0,1,0,0,....,0). For a node n_3 with two children n_1 and n_2 with branch lengths t_1 and t_2 respectively, we have that the probability that n_3 is an AU is

$$\mathbb{P}[n_3 = \text{AU}] = \sum_{x_1 \in \{1,...,16\}} \sum_{x_2 \in \{1,...,16\}} \exp(-t_1 Q)_{4,x_1} \mathbb{P}[n_1 = x_1] \exp(-t_2 Q)_{4,x_2} \mathbb{P}[n_2 = x_2] \tag{16}$$

where x_1 and x_2 again represent the nucleotides of n_1 and n_2 respectively. Then finally, with our prior on base-pairs $(pi_A A, pi_A C, pi_A G, pi_A U, ..., pi_U U)$, we have the paired-column probability:

$$\mathbb{P}[\text{column paired}] = \sum_{x_1 \in \{AA,AC,AG,AU,...,UU\}} \pi_{x_1} \mathbb{P}[n_a = x_1] \tag{17}$$

If we permit only canonical base-pairs, that is AU, CG, and GU pairs, it is easy to see there are many fewer terms in this sum.

Once one has these probabilities for paired and unpaired columns, one can calculate the CYK and inside-outside algorithms in the normal manner (see Extended Example for details), and structure prediction done in the normal manner. This is in some sense one of the big advantages of the SCFG method- one can decouple the evolutionary model from the SCFG model- the total probability is simply the product of the evolutionary probabilities and the SCFG probabilities.

The main implementation of this evolutionary approach is in Pfold (Knudsen and Hein, 1999, 2003), which has later been parallelised into PPfold (Sksd et al., 2012). The evolutionary model produces considerable better predictions than for single sequences. This is not only due to additional structural constraints seen through many sequences, but if the predictions from the simple evolutionary model described at the beginning of this section are not as good as those with the nucleotide substitution model.

2.5.3 Estimating Rates

To estimate mutation rates, we follow the process in Knudsen and Hein, (1999). Columns with over 85% sequence identity were taken, so as to suggest only single mutations. Counts were recorded of differences in base-pairs and single nucleotides over the sequences. Gaps were not counted. The rates could then be calculated as a function of the frequencies of nucleotide and base-pair mutations observed, and the evolution time between sequences. In practice, though, since this estimation procedure requires a reasonably big data set, and the mutation rates are made available in the manuscript, it is easier to use the parameters found there. The full rate matrices are available to download from the Pfold website (see Web Resources section).

2.5.4 Insertion-Deletion Events

At this point, little has been mentioned with respect to how to model insertion-deletion events. One approach is to model the gap as an unknown nucleotide, but requiring it to be unpaired, as we cannot have a base-pair involving a nucleotide that is not present. To implement this, we simply put (1,1,1,1) in the leaf-node probability vector. This also has the advantage that the phylogenetic probabilities are the same with or without completely gapped columns, so columns with many gaps can be pruned from the alignment without affecting the prediction quality. This is the approach used by Pfold and PPfold, which eliminate columns with greater than 75

Alternatively, one can treat the gap as a fifth nucleotide. One might expect insertion-deletion events to require a different evolutionary model than substitution events, something which is not considered by treating gaps as unknown nucleotides. Similarly, frequency of gaps will depend on evolutionary distances and number of sequences in the alignment. This approach has been successfully used in RNA gene finding (Rivas & Eddy, 2001).

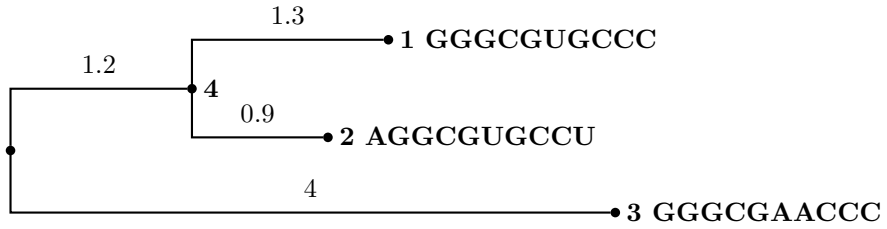
Finally, one can change the evolutionary model, for example by combining the above model with an HMM to better take account of insertion-deletion events (Fang et al., 2008). The HMM considers evolution in structured and unstructured regions, and results suggest that this yields better accuracy than the SCFG approach with a simple evolutionary model. Again, this is an area where more research could be done, particularly with correlated evolution models.

2.5.5 Example

Now that we have developed the full comparative model, an example of its use is provided. Consider an alignment of the following RNAs, assumed to have a consensus secondary structure (((....))):

```
>RNA_1
GGGCGUGCCC
>RNA_2
AGGCGUGCCU
>RNA_3
GGGCGAACCC
```

Furthermore, we can use a predicted evolutionary tree. The figure below shows an examples evolutionary tree, with each node having two leaf children, and the corresponding branch lengths.



Here we will only focus on calculating the phylogenetic probabilities for columns, since the other algorithms for SCFG prediction have been demonstrated prior to this. First, we look at the phylogenetic probability that column 1 (G,A,G) is unpaired. We will assume a prior on unpaired nucleotides of (0.268, 0.214, 0.267, 0.251), the overall frequencies seen in Knudsen & Hein (1999), and use the rate matrix for stems in Knudsen and Hein (1999). For each node, we can now calculate the associated probability vector:

Node	Likelihood of (A, C, G, U)
1	$(0, 0, 1, 0)$
2	$(1, 0, 0, 0)$
3	$(0, 0, 1, 0)$
4	$(0.162, 0.011, 0.105, 0.016)$
Ancestor	$(0.021, 0.010, 0.019, 0.011)$

and then the probability of the column being unpaired is simply the dot product of the prior on unpaired nucleotides, and ancestor nucleotide likelihood:

$$0.00563 + 0.00214 + 0.00507 + 0.00271 = 0.01555$$

Similarly, we consider the phylogenetic probability of the columns 1 (G,A,G) and 10 (C,U,C) being paired. For simplicity, we will assume that only canonical base-pairs (AU, UA, CG, GC, GU, UG) are allowed, and that the prior on base-pairs is (0.180, 0.180, 0.272, 0.272, 0.048, 0.048), the normalised stem frequencies from Knudsen and Hein (1999). For each node, we can now calculate the associated probability vector:

Node	Likelihood of (AU, UA, GC, CG, UG, GU)
1	$(0, 0, 1, 0, 0, 0)$
2	$(1, 0, 0, 0, 0, 0)$
3	$(0, 0, 1, 0, 0, 0)$
4	$(0.121, 0.011, 0.079, 0.007, 0.077, 0.095)$
Ancestor	$(0.019, 0.003, 0.040, 0.003, 0.003, 0.029)$

and then the probability of the two columns being paired

$$0.00342 + 0.00054 + 0.01088 + 0.00082 + 0.00014 + 0.00139 = 0.01719$$

When we have, for each column, the probability of the column being unpaired, and, for each pair of columns, the probability of the pair of columns being paired, we can proceed to structure prediction via CYK or inside-outside.

As with the single-sequence structure prediction problem, once we have the probability distribution over RNA secondary structures, we can decode this distribution in a number of ways. Pfold uses an MEA approach, but other approaches have been implemented as well. A Bayesian sampling approach, which samples helices according to the distribution from SCFGs has been implemented (Doose & Metzler, 2012), which also allows the prediction of pseudoknots, since helices can be nested. Alternatively, an approach which simultaneously aligns and folds (Bradley et al., 2008) using an evolutionary model and SCFG allows us to also take into account the uncertainty in the alignment. However, there is significant room for developments here in novel ways to decode the posterior distribution to gain better predictive accuracy.

2.6 Comparative Modelling

The main method which can be calculated in this way is information entropy. For a probability distribution P with events X, we define the information entropy $H(P)$:

$$H(\mathcal{P}) = -\sum_{x \in \mathcal{X}} \mathcal{P}(x) \log_2(\mathcal{P}(x)) \tag{18}$$

Informally, the information entropy (or Shannon entropy) measures the spread of probability mass over the probability space. We can see from the above equation that if there is only a single event, with probability one, then the information entropy is 0. Similarly, if we have a uniform distribution, that is n events with probability $1/n$ each, we have an entropy of $\log_2(n)$. So 0 and $\log_2(n)$ provide the minimum entropy and maximum entropy respectively. When the base of the logarithm is 2, the entropy is measured in bits.

Entropy has been calculated previously for RNA secondary structure space. In particular, there are many approaches for calculating thermodynamic entropy for the thermodynamic model (Mathews et al., 1999, Markham & Zuker, 2008), considering positional thermodynamic entropy (Hofacker et al., 1994) and thermodynamic entropy in response to base-pair mutation (Kiryu & Asai, 2012).

Firstly, we can improve maximum entropy bound of log_2(number of structures), since we know the number of structures of length n. Recall the asymptotic approximation for the number of structures of length n: Recall the asymptotic approximation for the number of structures of length n:

$$S_n \approx 1.104 \times n^{3/2} \times 2.618^n$$

So our maximum entropy bound is approximately:

$$H_{\max}(n) \approx 0.142 - \frac{3}{2}\log_2(n) + 1.388n \tag{19}$$

Now, to move on to calculating the information entropy for single sequences (Wang et al., 2011). Firstly, the information entropy can be written neatly as a function of expected rule frequencies, summing over the set of production rules P (Nedefhof & Satta, 2004).

$$H(G) = -\sum_{r \in P} log_2(\mathbb{P}[r])\mathbb{E}[\text{uses of } r] \tag{20}$$

This is advantageous as expected rule frequencies can be computed using the inside-outside algorithm in cubic time. Once we have computed inside and outside values, $I(U,i,j)O(U,i,j)$ gives the expectation of $s[i..j]$ being derived from an occurrence of U and

$$O(U,i,j)\sum_{k=i}^{j-1} \mathbb{P}[U \to VW]I(V,i,k)I(W,k+1,j) \tag{21}$$

the expectation of this happening by an initial application of the $U \to VW$ production. Summing over all subsequences of s we can find the expected number of uses of $U \to VW$. Similarly we can find the expected number of uses of terminal production rules $U \to .$ from $O(U,i,i)P[U \to .]$ for i with $s[i]$ unpaired, and in the same manner the paired rule types.

However, this information entropy is only useful to calculate for unambiguous SCFGs. We want to be able to make statements about RNA secondary structure space, and it is only for unambiguous SCFGs that there is a one-to-one correspondence between SCFG derivations and RNA secondary structures. With ambiguous SCFGs, we can still calculate the information entropy, but instead we are calculate the spread of mass over SCFG derivation space, not over RNA secondary structure space.

Information entropy can also be calculate with phylo-SCFGs, that is SCFGs which generate alignment columns, instead of single nucleotides (Sukosd et al., 2012). We have

$$\begin{aligned} H(G) &= -\sum_{d\in\Phi} \mathbb{P}[d]\log_2(\mathbb{P}[d]) \\ &= \log_2(T) - \frac{1}{T}\sum_{d\in\Phi}\mathbb{P}[d]\log_2(\mathbb{P}_G[d]) - \frac{1}{T}\sum_{d\in\Phi}\mathbb{P}[d]\log_2(\mathbb{P}_T[d]) \end{aligned}$$

where Φ is the set of all derivations, T represents the total probability of the alignment, and $\mathbb{P}_G$ and $\mathbb{P}_T$ represent the SCFG probabilities and evolutionary probabilities respectively. We can furthermore calculate the individual terms

$$\begin{aligned} \sum_{d\in\Phi}\mathbb{P}[d]\log_2(\mathbb{P}_G[d]) &= \sum_{r\in P}\log_2(\mathbb{P}[r])\mathbb{E}[\text{uses of } r] \\ \sum_{d\in\Phi}\mathbb{P}[d]\log_2(\mathbb{P}_T[d]) &= \sum_{i\neq j}\log_2(P_T(i,j|\widehat{i,j})\sum_{d\in\Phi}\mathbb{1}^d(i,j)\mathbb{P}[d] + \sum_i \log_2(P_T(i|\dot{i}))\sum_{d\in\Phi}\mathbb{1}^s(i)\mathbb{P}[d] \end{aligned}$$

with $\mathbb{1}^d$ the indicator function for a given derivation d. Note that once we have calculated the inside-outside algorithms, we can calculate the expected rule frequencies in quadratic time, and so the entropy is calculated in quadratic time. This has been implemented in PPfold 3.0.

Once we have calculated information entropy, we are interested in knowing what it can tell us about the space of RNA secondary structures. In some sense, information entropy can be viewed as signal. When the model has a lower entropy, the model has a less spread out probability distribution, and so we might expect the SCFG model is more 'confident' in the prediction. Similarly, when the probability mass is spread out, many secondary structures might look likely, and so we might expect the SCFG model is less 'confident' in making the prediction.

Similarly, we would not expect information entropy would correlate with predictive accuracy. A model might not be very confident about the prediction it makes, but it might still be correct, and similarly, the model might allocate a lot of probability to one structure, but it could be totally wrong. The limitation of information entropy to explain RNA secondary structure space is fundamentally limited by the SCFG model it is part of.

However, there are some useful results that can be found (Sksd et al., 2012). Examples can be found where information entropy can be used to distinguish between alignments with larger information content than others, and this happens to explain the difference in predictive accuracy. Similarly, the amount of information added by the evolutionary model can be quantified against the amount of information in a single sequence for a given alignment. Finally, by considering a large number of random sequences, one can make statements about the relative information content of an alignment. However, with all statistical models, we must view this as only part of the picture, as a tool which can help shed light on how the model views the space.

2.6.1 Reliability Scores

Pfold and PPfold also give reliability scores obtained from posterior probabilities. Once the base-pair probability matrix is obtained via inside-outside, we will predict a structure using a simple MEA algorithm. But how sure are we in our choice of structure? Imagine that we have pairing probabilities of one for each base-pair, then we would be very confident. But if we have pairing probabilities of 0.7 for each base-pair, we may get the same structure from MEA, but be less confident about it. This

reliability score gives a measure of how close the pairing probabilities are, by calculating a function of the distance the pairing probabilities are from one, if positions are predicted paired, and zero if positions are unpaired.

However, as with entropy, this reliability measure is only part of the picture. It can indicate if there are other structures the model might prefer, which are close in probability, which could be useful if you do not, for some biological reason, have confidence in the prediction. On the other hand, it is useful in sequence design programs like Frnakenstein (Lyngsoe et al., 2012), as it can emulate the idea of folding stability. Overall though, it is limited by the SCFG model as a framework, and does not correlate particularly well with predictive accuracy.

2.7 Thermodynamic Methods

At this point, it is worth briefly discussing thermodynamic methods of RNA secondary structure prediction, for comparison. The method is based on energy parameters, which must be gained experimentally (Mathews et al., 1999). In particular, we must consider energy contributions from different combinations of base pairs. For example, if an AU base-pair is next to a GU base-pair, there will be a different energy contribution than if the AU base-pair is next to another AU base-pair. These so-called stacking energies mean that a large number of parameters must be found, over a hundred, which is why many of the thermodynamic models are based on the same energy models.

Once we have all of these energy parameters, again we run into the problem that we cannot evaluate the free-energy of every possible secondary structure. So, what is found, is the structure with the minimum free-energy, through dynamic programming, and this is the structure predicted.

Since, for each secondary structure, we have an associated energy, this naturally leads to a Boltzmann probability distribution over secondary structures. To calculate the probability of a secondary structure S, we have

$$P(S) = \frac{1}{Q} e^{-F(S)/kT} \tag{22}$$

where $F(S)$ is the free-energy of S, T is the temperature, k a constant, and Q a normalising factor. To calculate Q, one needs to calculate the partition function (McCaskill, 1990), and to find the free-energy of a given secondary structure, one simply sums over all the loops and stems, finding the free-energy of the smaller components, using the free-energy parameters determined experimentally. However, if one is only interested in calculating the minimum free-energy structure, this can be done without calculating the full partition function (Hofacker et al., 1999). In particular, it can be done in cubic time, the same complexity as the SCFG method.

The main implementations of the thermodynamic are UNAfold (Markham et al., 2008) and RNAfold, as part of the Vienna package (Hofacker et al., 1994). They are essentially the same method, but with slightly different energy models. They predict the minimum free-energy structure, in the same way that SCFGs can predict the maximum probability structure. Also like SCFG methods, they place a probability distribution on RNA secondary structures, so various different posterior decoding methods have been implemented to obtain slightly different prediction methods. Sfold (Ding et al., 2000) samples from the Boltzmann distribution generated by the free-energy model, and generates a consensus structure from the samples. Kinefold (Xayaphoummine et al., 2005) uses kinetic folding concepts in combination with the free-energy model to try and more closely echo biological folding mechanisms.

The main thermodynamic method has a couple of weaknesses. Firstly, it does not consider non-canonical base-pairs, those are non-AU. -CG, or -GU base-pairs. Part of the reason for this is practical- the number of parameters that would need to be determined from experiments would become significantly larger, and perhaps obtaining these would not be feasible. However, many RNA secondary structures do contain non-canonical base-pairs. Secondly, the method cannot be adapted to an evolutionary model. The thermodynamic methods that do consider alignments (Bernhart et al., 2008) only gain the information concerning additional structure constraints from having multiple sequences with a consensus structure, but this is often considerable. Pseudoknots, though, can be

considered by adapting the energy model, for example in pknots (Rivas & Eddy, 1999). For further reading on the thermodynamic approach, see Janssen et al., (2011).

3 Discussion

As can be seen by the number of approaches attempted at the RNA secondary structure prediction problem, even within the framework of SCFGs, there has not been overwhelming success at the problem. Methods continue to operate at the same kind of sensitivity on data sets, at an average of around 0.55 for single sequences, and 0.75 for alignments. However, this accuracy is highly variable, it may be 1 on one sequence, and 0 on another, without much warning.

Thermodynamic methods and SCFG method continue to compete, with thermodynamic methods having a slight edge on single-sequence structures. The methods do not predict identical structures, or even similar structures (Anderson et al., 2012) when the methods go wrong, and there does not seem to be a way of knowing a priori which method you might want to predict your structure.

Furthermore, although many methods have been developed to measures entropy, or folding pathways (Shapiro et al., 2001), these are all fundamentally limited by the models which produce them. They give us useful statistics concerning what the models think is going on in the RNA folding, and how the RNA might folding differently, but, due to the model accuracy, cannot be completely trusted.

However, it is important that there is continued work on the methods. With developments in next-generation sequencing techniques, particular the fact that methods like RNA-seq are now easily available, and the data being generated in transcriptomics is considerable. RNA folding methods will want to be used in genome-wide studies, considering RNA structure-based effects on gene expression and regulation. Cumulative error from poor prediction will create statistical problems in these studies, and possibly invalidate conclusions. Since RNA is so fundamental in molecular dynamics, we must have a better understanding on the mechanisms for folding, and be able to translate that understanding into modeling principles.

4 Future Trends

RNA is as interesting as ever. Particularly as biological technologies advance, the role of RNA will be come more and more crucial to understand.

Three-dimensional structure prediction techniques are still in their infancy but will no doubt develop as technology and data advance. Particularly, the possibility of extending SCFG prediction to three-dimensional structures will be an interesting task. Similarly, as RNA roles in gene expression become better understood, structural features of RNA may well be used in combination with network inference techniques to better understand gene regulatory networks.

Of course, there are many additional applications of RNA-secondary structure prediction which will benefit from improved methodology: RNA gene finding, annotation, functional prediction, RNA-protein interactions, the list is long. Improved RNA molecular evolution models or alignments may enable prediction quality to improve. However, there has been little investigation into which components of the RNA secondary structure prediction model are important to predictive success.

Given that continued work is important in the field of RNA secondary structure prediction, the question is how to go about it. It seems unlikely that many improvements will come from an improved alignment or evolutionary model, partly because these models are close to being as good as they will get, and partly because the single-sequence structure prediction problem for RNA is still unsolved. The probabilistic and thermodynamic approaches seem to cover all the bases. One might imagine a machine-learning approach could work, but in my opinion, SCFGs are really just that. It seems unlikely to me that a fundamentally different approach will arise, especially given how much work has been done to develop the probabilistic and thermodynamic approaches. Models need to become more biologically realistic, particularly SCFGs. As mentioned previously, SCFGs are little more than a machine-learning approach, and there are many more biological processes going on in RNA folding than a simple sequence to structure map. RNA-specific, co-transcriptional

effects, as well as kinetic effects, could be considered, or perhaps something so simple as being able to incorporate some notion of temperature-dependence might prove useful. The probability distribution generated by SCFGs is can still be used, but perhaps novel methods of decoding could be used to more closely echo biological processes. There is clearly much still to do.

5 Web Resources

There are many web resources available for RNA secondary structure prediction. In particular, most of the methods cited throughout will have made open source programs available, with links in the manuscripts. Similarly, most studies which use data sets will have made them publicly available, with links in the manuscripts. However, a few of the main resources are as follows.

SCFG based approaches:

- Pfold: `http://www.daimi.au.dk/~compbio/pfold/`
- PPfold: `http://www.daimi.au.dk/~compbio/pfold/downloads.html`
- PhyloQFold (pseudoknot sampling SCFG model): `http://evol.bio.lmu.de/_statgen/software/phyloqfold/`
- Evolgram (evolving SCFGs): `http://www.stats.ox.ac.uk/~anderson/Code/evolvingscfgs.html`

Thermodynamic approaches

- UNAfold: `http://mfold.rna.albany.edu/`
- Vienna RNA Package (including RNAfold): `http://www.tbi.univie.ac.at/~ivo/RNA/`
- Sfold (Boltzmann sampling): `http://sfold.wadsworth.org/cgi-bin/index.pl`
- Kinefold (kinetic model): `http://kinefold.curie.fr/`
- Pknots (pseudoknot model): `http://selab.janelia.org/software.html`

Data Sources:

- Rfam: `http://rfam.sanger.ac.uk/`
- RNA STRAND `http://www.rnasoft.ca/strand/`

6 References

Andersen, E., Dong, M., Nielsen, M., Jahn, K., Subramani, R., Mamdouh, W., Golas, M., Sander, B., Stark, H., Oliveira, C., Pedersen, J., Birkedal, V., Besenbacher, F., Gothelf, K., Kjems, J. (2009). Self-assembly of a nanoscale DNA box with a controllable lid. Nature, 459, 73-76.

Andersen ES, Rosenblad MA, Larsen N, Westergaard JC, Burks J, Wower IK, Wower J, Gorodkin J, Samuelsson T, Zwieb C: The tmRDB and SRPDB resources. Nucleic Acids Res 2006, 34(Database issue):163-168.

Anderson, J.W.J., Tataru, P., Staines, J., Hein, J. & Lyngso, R. (2012), Evolving stochastic context-free grammars for RNA secondary structure prediction, BMC Bioinformatics 13(1), 78.

Andronescu, M., Bereg, V., Hoos, H. & Condon, A. (2008), RNA STRAND: The RNA secondary structure and statistical analysis database, BMC Bioinformatics 9(1), 340.

Berman HM, Olson WK, Beveridge DL, Westbrook J, Gelbin A, Demeny T, Hsieh SH, Srinivasan AR, Schneider B: The nucleic acid database. A comprehensive relational database of three-dimensional structures of nucleic acids. Biophys J 1992, 63(3):751-759.

Bernhart, S., Hofacker, I., Will, S., Gruber, A. & Stadler, P. (2008), Rnaalifold: improved consensus structure prediction for rna alignments, BMC Bioinformatics 9(1), 474.

Brabrand, C. Giegerich, R., and Mller, A. (2007). Analyzing ambiguity of context-free grammars. In Proc. 12th International Conference on Implementation and Application of Automata, CIAA 07, volume 4783 of LNCS. Springer-Verlag.

Bradley, R. K., Pachter, L. & Holmes, I. (2008), Specific alignment of structured RNA: stochastic grammars and sequence annealing, Bioinformatics 24(23), 26772683.

Brown J: The Ribonuclease P Database. Nucleic Acids Res 1999, 27:314-314.

Brown, M. & Wilson, C. (1996). RNA pseudoknot modeling using intersections of stochastic context free grammars with applications to database search. Pacific Symposium on Biocomputing 109-125.

Cannone J, Subramanian S, Schnare M, Collett J, D'Souza L, Du Y, Feng B, Lin N, Madabusi L, Muller K, Pande N, Shang Z, Yu N, Gutell R: The comparative RNA web (CRW) site: an online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. BMC Bioinformatics 2002, 3:15.

Chappelier, J. & Rajman, M. (1998), A generalized CYK algorithm for parsing stochastic CFG, Proc. of 5eme conference sur le Traitement Automatique du Langage Naturel. pp. 153157.

Chen, J. & Greider, C. (2005). Functional analysis of the pseudoknot structure in human telomerase RNA. Proceedings of the National Academy of Science of USA, 102, 8080-8085.

Chomsky, N. (1956), Three models for the description of language, Information Theory, IRE Transactions on 2(3), 113124.

Craig, M. E., Crothers, D. M. & Doty, P. (1971), Relaxation kinetics of dimer formation by self complementary oligonucleotides, Journal of Molecular Biology 62(2), 383401.

Darty, K., Denise, A. & Yann, P. (2009). VARNA: Interactive drawing and editing of the RNA secondary structure. Bioinformatics 25:15,. 1974-1975.

de Almeida Ribeiro, E., Beich-Frandsen, M., Konarev, P. V., Shang, W., Veerek, B., Kontaxis, G., Hmmerle, H., Peterlik, H., Svergun, D. I., Blsi, U. & Djinovi-Carugo, K. (2012), Structural flexibility of rna as molecular basis for hfq chaperone function, Nucleic acids research 40(16), 80728084.

Dempster, A.P., Laird, N.M., Rubin, D.B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal of the Royal Statistical Society. Series B (Methodological) 39 (1): 138.

Ding, Y., Chan, C. Y. & Lawrence, C. E. (2005), Rna secondary structure prediction by centroids in a boltzmann weighted ensemble, RNA 11(8), 11571166.

Doose, G. & Metzler, D. (2012), Bayesian sampling of evolutionarily conserved RNA secondary structures with pseudoknots, Bioinformatics 28(17), 22422248.

Dowell, R. & Eddy, S. (2004), Evaluation of several lightweight stochastic context-free grammars for

rna secondary structure prediction, BMC Bioinformatics 5(1), 71.

Fang X.Y., Luo Z.G., Wang Z.H. (2008). Predicting RNA secondary structure using profile stochastic context-free grammars and phylogenic analysis. Journal of Computer Science and Technology 23(4): 582-589.

Felden B., Florentz C., Gieg R., Westhof E. (1996). "A central pseudoknotted three-way junction imposes tRNA-like mimicry and the orientation of three 5' upstream pseudoknots in the 3' terminus of tobacco mosaic virus RNA". RNA 2 (3): 20112.

Felsenstein, J. (1981), Evolutionary trees from dna sequences: A maximum likelihood approach, Journal of Molecular Evolution 17(6), 368376.

Freyhult, E., Gardner, P. & Moulton, V. (2005), A comparison of RNA folding measures, BMC Bioinformatics 6(1), 241.

Gahura, O., Hammann, C., Valentov, A., Pta, F. & Folk, P. (2011), Secondary structure is required for 3 splice site recognition in yeast, Nucleic acids research 39(22), 97599767.

Gardner, P. & Giegerich, R. (2004), A comprehensive comparison of comparative RNA structure prediction approaches, BMC Bioinformatics 5(1), 140.

Griffiths-Jones, S., Moxon, S., Marshall, M., Khanna, A., Eddy, S. R. & Bateman, A. (2005), Rfam: annotating non-coding rnas in complete genomes, Nucleic acids research 33(suppl 1), D121D124.

Hamada, M., Sato, K. & Asai, K. (2010), Prediction of RNA secondary structure by maximizing pseudo-expected accuracy, BMC Bioinformatics 11(1), 586.

Harmanci, A., Sharma, G. & Mathews, D. (2011), Turbofold: Iterative probabilistic estimation of secondary structures for multiple RNA sequences, BMC Bioinformatics 12(1), 108.

J. Hstad, S. Phillips, and S. Safra. A well characterized approximation problem. Information processing letters, 47(6):301305, 1993.

Hofacker I., Fontana W., Stadler P., Bonhoeffer L., Tacker, M., Schuster P. (1994). Fast folding and comparison of RNA secondary structures. Chemical Monthly, 125(2):167188.

Hopcroft, J. & Ullman, J. (1979). Introduction to Automata Theory, Languages, and Computation. Reading, MA: Addison-Wesley.

Janssen, S., Schudoma, C., Steger, G. & Giegerich, R. (2011), Lost in folding space? comparing four variants of the thermodynamic model for RNA secondary structure prediction, BMC Bioinformatics 12(1), 429.

Jukes T.H. & Cantor C.R. (1969). Evolution of Protein Molecules. New York: Academic Press. 21132.

Kiryu H, Asai K (2012). Rchange: Algorithms for computing energy changes of RNA secondary structures in response to base mutations. Bioinformatics 2012, 1:12.

Krogh, A., Brown, M., Mian, I., Sj lander, K. & Haussler, D. (1993). Hidden Markov Models in computational biology: Applications to protein modeling. Journal of Molecular Biology, 235, 1501-1531.

Knudsen, B. & Hein, J. (1999), RNA secondary structure prediction using stochastic context-free grammars and evolutionary history., Bioinformatics 15(6), 446454.

Knudsen, B. & Hein, J. (2003), Pfold: RNA secondary structure prediction using stochastic context-free grammars, Nucleic acids research 31(13), 34233428.

Krogh, A., Brown, M., Mian, I., Sjolander, K. & Haussler, D. (1993). Hidden Markov Models in computational biology: Applications to protein modeling. Journal of Molecular Biology, 235, 1501-1531.

Lari, K. & Young, S. J. (1990). The estimation of stochastic context-free grammars using the inside-outside algorithm. Computer Speech and Language 4(1), 3556.

Lefebvre, F. (1996). A grammar-based unification of several alignment and folding algorithms. ISMB-96 Proceedings,2, 143-154.

Loh, E., Memarpour, F., Vaitkevicius, K., Kallipolitis, B. H., Johansson, J. & Sondn, B. (2011), An unstructured 5-coding region of the prfa mrna is required for efficient translation, Nucleic acids research. 40(4):1818-27.

Lyngs, R. B., Anderson, J.W.J., Badugu, A., Hyland, T., Sizikova, E., and Hein, J. (2012). Frnakenstein: multiple target inverse rna folding. BMC Bioinformatics 13:260.

Markham NR, Zuker M (2008). UNAFold: software for nucleic acid folding and hybridization. Methods in Molecular Biology (Clifton, N.J.), 453:331.

Mathews D, Sabina J, Zuker M, Turne D: Expanded Sequence Dependence of Thermodynamic Parameters Improves Prediction of RNA Secondary Structure. J Mol Biol 1999, 288:911940.

McCaskill, J. S. (1990), The equilibrium partition function and base pair binding probabilities for RNA secondary structure, Biopolymers 29(6-7), 11051119.

Meyer, I. (2007). A practical guide to the art of RNA gene prediction. Briefings in Bioinformatics. 8:6 396-414.

Moulton, V., Zuker, M., Steel, M., Pointon, R. & Penny, D. (2000), Metrics on RNA secondary structures, Journal of Computational Biology 7(1-2), 277292.

Nebel, M. & Scheid, A. (2011), Evaluation of a sophisticated SCFG design for RNA secondary structure prediction. Theory in Biosciences 130:313336.

Nederhof M, Satta G (2004). Kullback-Leibler distance between probabilistic context-free grammars and probabilistic finite automata. In Computer Speech & Language, Association for Computational Linguistics 2004:71.

Nussinov, R., Pieczenik, G., Griggs, J. & Kleitman, D. (1978). Algorithms for loop matchings. SIAM Journal on Applied Mathematics, 35, 68-82.

Pipas, J. & McMahon, J. (1975). Method for predicting RNA secondary structure. Proceedings of the National Academy of Science USA, 72, 2017-2021.

Reeder, J., Steffen, P. & Giegerich, R. (2005), Effective ambiguity checking in biosequence analysis, BMC Bioinformatics 6(1), 153.

Rivas, E. & Eddy, S.R. (1999). A dynamic programming algorithm for RNA structure prediction including pseudoknots. Journal of Molecular Biololgy, 285, 2053-2068.

Rivas, E. & Eddy, S.R. (2000). The language of RNA: a formal grammar that includes pseudoknots. Bioinformatics, 16, 334-340.

Rivas, E. and Eddy, S.R. (2001). Noncoding RNA gene detection using comparative sequence analysis. BMC Bioinformatics, 2, 8.

Rivas, E., Lang, R., and Eddy, S. R. (2012). A Range of Complex Probabilistic Models for RNA Secondary Structure Prediction That Include the Nearest Neighbor Model and More. RNA, 18: 193212.

Sakakibara, Y, Brown, M., Hughey, R., Mian, I., Sjolander, K., Underwood, R. et al. (1994). Stochastic context-free grammars for tRNA modeling. Nucleic Acids Research, 22, 5112-5120.

Scheid, A. & Nebel, M. (2012), Evaluating the effect of disturbed ensemble distributions on SCFG based statistical sampling of RNA secondary structures, BMC Bioinformatics 13(1), 159.

Shapiro, B. A. & Zhang, K. (1990), Comparing multiple RNA secondary structures using tree comparisons, Computer applications in the biosciences : CABIOS 6(4), 309318.

Shapiro, B. A., Bengali, D., Kasprzak, W. & Wu, J. C. (2001), RNA folding pathway functional intermediates: their prediction and analysis, Journal of Molecular Biology 312(1), 2744.

Shapiro, B. A., Yingling, Y. G., Kasprzak, W. & Bindewald, E. (2007), Bridging the gap in RNA structure prediction, Current opinion in structural biology 17(2), 157165.

Sprinzl M, Vassilenko K: Compilation of tRNA sequences and sequences of tRNA genes. Nucleic Acids Res 2005, 33(Database issue):139-140.

Stein P.R. & Waterman M.S. (1978). On some new sequences generalizing the Catalan and Motzkin numbers. Discrete Mathematics, 26:261272.

Sksd, Z., Knudsen, B., Kjems, J. & Pedersen, C.N.S. (2012), PPfold 3.0: fast RNA secondary structure prediction using phylogeny and auxiliary data, Bioinformatics 28(20), 26912692.

Sksd, Z., Knudsen, B., Anderson, J.W.J., Novk, Kjems, J., Pedersen, C,N,S. (2012). Characterising RNA secondary structure space using information entropy. In press.

Tavar S (1986). "Some Probabilistic and Statistical Problems in the Analysis of DNA Sequences". Lectures on Mathematics in the Life Sciences (American Mathematical Society) 17: 5786.

Vitreschak, A. G., Rodionov, D. A., Mironov, A. A. & Gelfand, M. S. (2004), Riboswitches: the oldest mechanism for the regulation of gene expression?, Trends in Genetics 20(1), 4450.

Wang Y., Manzour A., Shareghi P., Shaw T.I., Li Y.W., Malmberg R.L., Cai L. (2011). Stable stem enabled Shannon entropies distinguish non-coding RNAs from random backgrounds. In Computational Advances in Bio and Medical Sciences (ICCABS), 2011 IEEE 1st International Conference on 2011:184.

Washietl, S., Hofacker, I. L., Lukasser, M., Huttenhofer, A. & Stadler, P. F. (2005), Mapping of conserved RNA secondary structures predicts thousands of functional noncoding RNAs in the human genome, Nat Biotech 23(11), 13831390.

Westbrook J, Feng Z, Chen L, Yang H, Berman H: The Protein Data Bank and structural genomics. Nucleic Acids Res 2003, 31:489-491.

Xayaphoummine, A., Bucher, T. & Isambert, H. (2005), Kinefold web server for rna/dna folding path and structure prediction including pseudoknots and knots, Nucleic acids research 33(suppl 2), W605W610.

Xayaphoummine, A., Bucher, T., Thalmann, F. & Isambert, H. (2003), Prediction and statistics of pseudoknots in rna structures using exactly clustered stochastic simulations, Proceedings of the National Academy of Sciences 100(26), 1531015315.

Xu, Z., Almudevar, A. & Mathews, D. H. (2012), Statistical evaluation of improvement in RNA secondary structure prediction, Nucleic Acids Research, Vol. 40, No. 4 e26.

Younger, D. (1967). Recognition and parsing of context-free languages in time n3. Information and Control 10(2), 189208.

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

Zuker, M. & Sankoff, D. (1984). RNA secondary structures and their prediction. Bulletin of Mathematical Biology, 46, 591-621.

Zuker, M. (2003). Mfold web server for nucleic acid folding and hybridization prediction. Nucleic Acids Research, 31, 3406-3415.