Bachelor Thesis

RNA Secondary Structure Prediction

Sophie Schneiderbauer
soschnei@uos.de

Cognitive Science, University Of Osnabrück
First supervisor: Prof. Dr. Volker Sperschneider
Second supervisor: Prof. Dr. Oliver Vornberger

15th September 2008

Contents

1 Introduction **3**

2 Biological Basics **3**
2.1 What is RNA and where is it used 3
2.1.1 The structure of DNA 4
2.1.2 The structure of RNA 5
2.1.3 The structure of proteins 6
2.1.4 Gene expression 6
2.2 Primary, secondary and tertiary structures 7
2.3 Importance of RNA secondary structure prediction 9
2.4 Different secondary structure elements 10
2.4.1 A stem 10
2.4.2 A hairpin loop 10
2.4.3 An internal loop 11
2.4.4 A bulge loop 11
2.4.5 A multiloop 11
2.5 Pseudoknots 12

3 Prediction Methods **13**
3.1 Prediction based on graph theory 13
3.1.1 The Nussinov algorithm 14
3.1.2 The Zuker algorithm 16
3.2 Prediction based on information theory 21
3.3 Prediction based on genetic algorithms 21

4 The Program **22**
4.1 Implementation 22
4.1.1 Manual for users 22
4.1.2 The structure 23
4.1.3 Nussinov algorithm in the program 25
4.1.4 Zuker algorithm in the program 26
4.1.5 Pseudoknots in the program 27
4.1.6 The graphical representations 28
4.2 Parameter value estimation 30
4.2.1 The accuracy measure 30
4.2.2 The database used 31
4.3 Results and value variations 31

5 Compared To Other Programs.. **36**

6 Critical Evaluation **37**

1 Introduction

For my Bachelor Thesis in the domain of Bioinformatics, I decided to work in the field of secondary structure prediction of RNA. I have made a program which developes good predictions for a given RNA sequence, by using a dynamic algorithm. The user of the program can choose between three different methods to make the prediction: Either he or she can use the Nussinov-algorithm, the Zuker-algorithm or the Zuker-algorithm with one or two pseudoknots. Later on, I will provide more specific information about these different methods.
First of all, I am going to give a rough sketch of the background in the area of genetic biology. I am going to explain what RNA is, where it is used and why it is important to predict its secondary structure. Therwith will be explained the differences between the different structures and in more detail the existing forms of secondary structure elements.
In the next section an overview of several prediction methods will be given and the methods I have used in my program will be explained precisely.
Furthermore, an outline of the program and its structure will be given. I am going to illustrate how it can be used and how the algorithms and the graphical representations have been implemented.
Afterwards, I am going to explain briefly how I varied the values for the parameters, how the accuracy has been measured and what the results have been.
At the end of this thesis, the program will be compared to other, already existing programs and finally I am going to evaluate the program critically.

2 Biological Basics

2.1 What is RNA and where is it used

Proteins and nucleic acids, like RNA(ribonucleic acid) and DNA(deoxyribonucleic acid), play an important role in reproducing and maintaining life: Proteins are important because they control processes like energy metabolism, intercellular communication and biosyntheses. They are synthesized using the genetic information which is stored in the DNA [1].
DNA is used for storing genetic information and instructions. This information is used to maintain an organism and to create a new organism by reproducing the DNA [1]. It is stored on a number of DNA molecules, which altogether are called the organism's genome. The number varies within all living organisms. Humans, for example, have a set of 46 DNA molecules.
RNA molecules are used for the synthesis of proteins, they act as messengers [1].

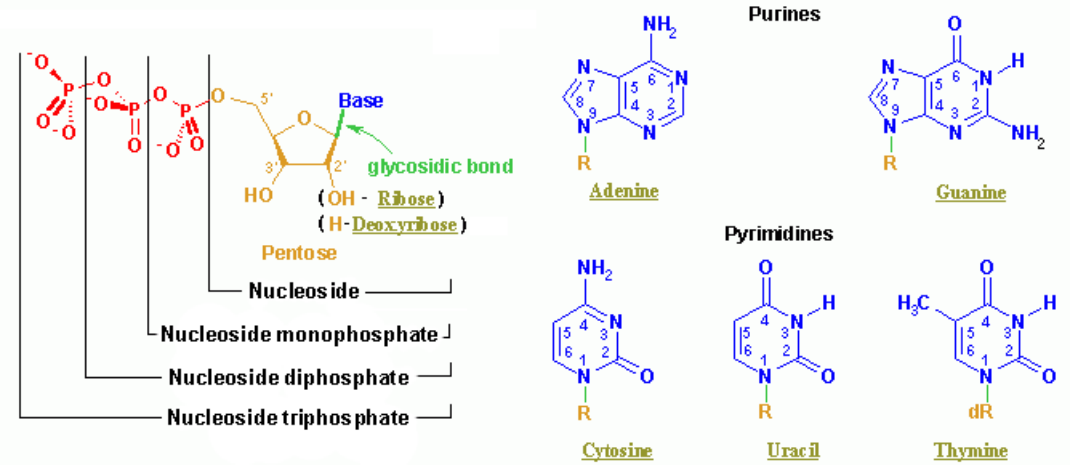

Figure 1: Nucleotides [3]

To understand this whole process in detail, it is important to know the structure of DNA, RNA and proteins:
Both, DNA and RNA are composed of subunits, the so-called nucleotides or bases. There are only four different types of nucleotides in a molecule, but DNA and RNA do have other sets of nucleotides [1].
Nucleotides consist of a nitrogen-containing base, a five-carbon sugar ring and a phosphate group (see Figure 1). The nucleotides are linked together by phosphordiester linkages through the hydroxil group on the sugar on one nucleotide and the phosphate on the next one. As a result one can observe a strand with the so-called 5'-end, where a free phosphate group can be found, and the 3'-end with a free hydroxil group.
The nitrogenous base has the structure of a planar ring and is either a purine or a pyrimidine.

2.1.1 The structure of DNA

Deoxyribose is the sugar that is used in DNA nucleotides and hence they are called deoxyribonucleotides [1]. The purine bases are adenosine (A) and guanine (G) and the pyrimidine bases are cytosine (C) and thymine (T) [1].
The chemically simple DNA molecules are very long and the linear order of their bases encodes the genetic information [1].
DNA can be often be found in a three-dimensional structure of a simple, regular double helix. It is formed by two complementary DNA chains, which interact via base-pairing [1].
The base-pairing has been discovered by James Watson and Francis Crick and therfore is

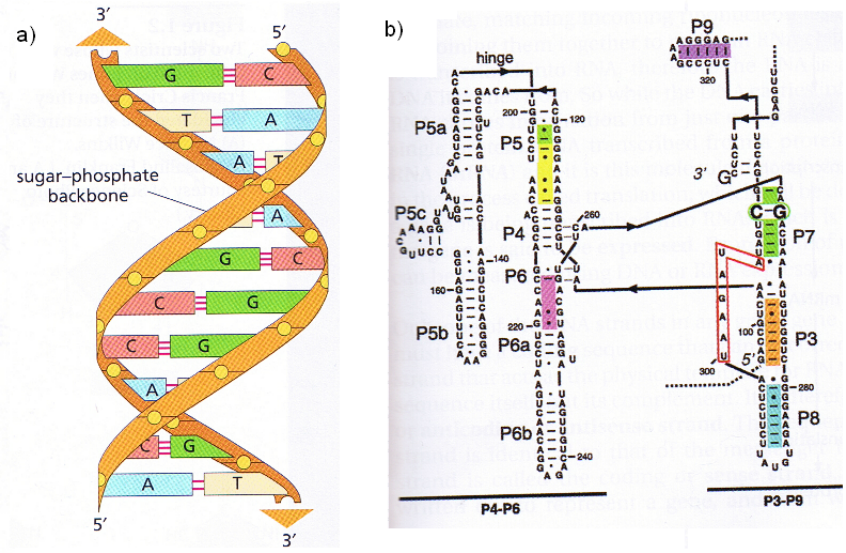

Figure 2: a) DNA double helix [1] b) RNA structure [1]

called Watson-Crick base-pairing: A specific pyrimidine pairs only with a specific purine. In fact A only pairs with T and G only pairs with C [1].
Hence each chain of a DNA double helix is complementary to the other chain and both are running in opposite directions (see Figure 2a).
The pairings are realized by noncovalent hydrogen bonds between the two bases. And due to the fact that they are weaker than covalent bonds, the two strands of the helix can be separated easily. This is important because the double helix has to be separated in certain processes, like for example during the DNA replication. This process takes place before the process of cell division: The DNA has to be copied in such a way that every new cell contains all the genetic material. Each of the separated strands then serves as a template and the new complementary strand can be synthesized easily by finding the matching nucleotide and putting all the nucleotides together. This process is done by the enzyme DNA polymerase [1].
One important feature which should be mentioned here is the fact that during the DNA replication errors do occur with the probability of 1 to 10^9 [1]. These errors are also called mutations and it has been shown that they are vital for evolution, because they make it possible for organisms to change and adopt to a change of situations.

2.1.2 The structure of RNA

In RNA nucleotides the sugar which is used is ribose and therefore they are also called ribonucleotides [1]. The purine bases are also adenosine (A) and guanine (G), but the

pyrimidine bases are cytosine (C) and uracil (U) [2].
RNA molecules are much smaller than DNA molecules, but they are also linear polymers [1]. Moreover they do not seem to have a regular three-dimensional structure and are mostly single stranded [1]. This makes them more flexible than DNA and they can also act also as enzymes [1].
Due to these irregular bindings within the strands, the primary structure, the sequence of the bases on its own is not sufficient to determine the function of the RNA [2]. The molecules often contain a very stable three dimensional structure with unpaired regions, which are very flexible. The wobble base-pairs make up an important factor for this flexibility [1]. Beside the Watson-Crick base-pairs, A:U and G:C, is the wobble base-pair , G:U, one of the most common base-pairs in RNA molecules [2]. But aczually any of the bases can build a hydrogen bond with any other base [2]. An example of the irregular structure formed by RNA can be observed in Figure 2b.
Another difference between DNA and RNA is that double-stranded RNA builds up A-helices while double-stranded DNA builds up B-helices [2]. The difference becomes apparent observing the size of the major grooves of the helices: The major groove of the A-helix is rather narrow and deeper. This is due to ribose needing more space than deoxyribose [2].
One should also keep in mind that from a thermodynamical point of view, RNA should always be regarded as a distribution of several structures [2].

2.1.3 The structure of proteins

While the components of the nucleic acids are nucleotides, the components of the proteins are 20 different amino acids [4]. The diversity of chemical characteristics of the amino acids makes proteins more chemically complex than nucleic acids [1].

2.1.4 Gene expression

At this point I am going to describe how we get from the DNA in the cells to the proteins. This process is often called the central dogma of molecular biology (Crick 1970) and is basically identical in all organisms [1] [4]:
The first stage is called transcription. This is the part of the process in which DNA is transcribed into messengerRNA (mRNA) [1]. It is very similar to the process of DNA replication. A single DNA strand acts as template again, but the enzyme RNA polymerase matches ribonucleotides instead of deoxyribonucleotides to the strand. In this case T and U both match A like base-pairs. Hence the result is a RNA chain, the so-called mRNA [1]. Another difference is that not the whole strand is duplicated, but only the part of the DNA strand which is carrying the protein-coding information. Therefore

RNA is a smaller molecule [1]. The gene is then expressed, it is said.
It is also important to mention that the genomic DNA sequence contains more information than the protein. This is due to the fact that it also has to be known where the transcription should begin and when a gene should be transcribed at all [1].
The next stage is called translation. This is the process in which mRNA gets translated into amino acids which form the resulting protein. The translation is done according to the genetic code [1]. This code refers to the rules that regulate which RNA sequence shall be translated into which amino acid sequence.
Every amino acid is encoded by three following bases [1]. The different sets of base-triplets are called codons and because there are more codons (4^3) than amino acids, most of the amino acids can be expressed by several codons. There are also three stop codons that do not generate any amino acid, but mark the end of a protein-coding sequence [1]. Signalling the start is more complicated, but very often the codon AUG starts the translation. For more information the reader is invited to have a closer look at the genetic code shown in Figure 3b.
As the codons consist of three bases there are three different readings of a sequence, depending on the starting base. During the translation detailed control signals determine the appropriate reading [1]. Whenever predictions are made these signals are often missing. But since normally proteins consist of at least 100 amino acids, incorrect reading frames can easily be detected when the resulting protein is too short [1].
Translation is carried out by large multimolecular complexes made of ribosomal RNA (rRNA) and proteins. These complexes are called ribosomes and they consist of two main subunits [1]. One of the subunits binds to the mRNA and the other one to small transfer RNA (tRNA) molecules. When these tRNA molecules are bound to the ribosomes, they transfer the amino acid to the already existing protein chain [1]. Having a three-base anticodon, the tRNA binds to a specific codon in the mRNA . Humans, for instance, have 48 different anticodons in their tRNA [1]. Furthermore tRNA is a very good example of flexibility and of a molecule of RNA whose three-dimensional structure is being the decisive factor for its function [1].
Both stages can be observed in Figure 3a.

2.2 Primary, secondary and tertiary structures

In this section I am going to explain the differences between primary, secondary and tertiary structures. This is important because DNA, RNA and also proteins do have all three structures and in the program the secondary and tertiary structure of RNA are used.
The primary structure of a molecule describes only the one-dimensional sequence of it components. As I mentioned before this is a crucial structure for DNA molecules, because it encodes the genetic information [1]. The primary structure of RNA is almost identical

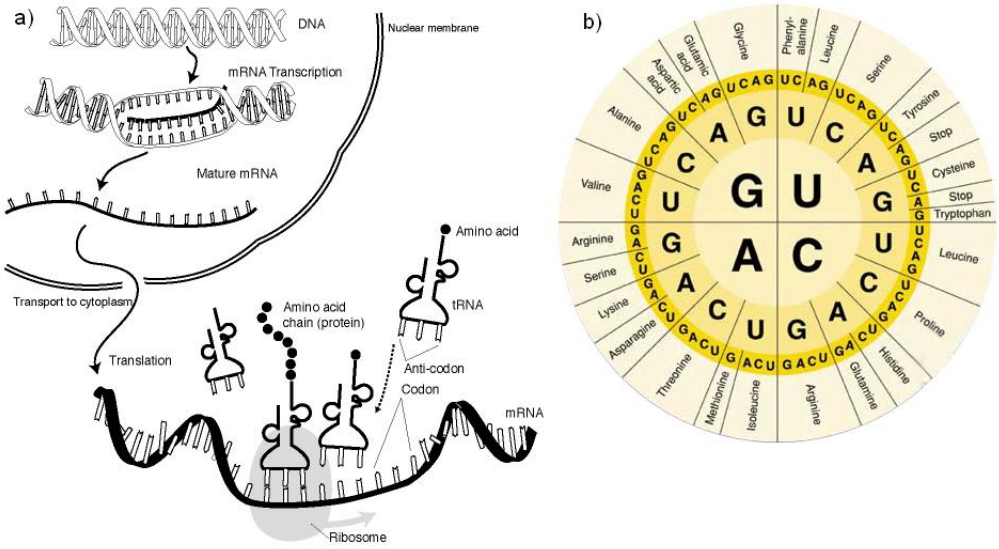

Figure 3: a) gene expression [5] b) genetic code [6]

to the primary structure of DNA, besides the components being A,C,G and U instead of T [2]. The primary structure of proteins is the sequence of the amino acids. The twenty different amino acids have different characteristics. They can for instance be hydrophobic, polar, acid or basic [4].
The secondary structure of molecules is more complex than the primary structure and can be drawn in two-dimensional space [2]. By definition the local conformation of polymers is called secondary structure [4]. For RNA and DNA the secondary structure is their base-paired structure [7]. In the next but one section I am going to explain the most common secondary structures of RNA in more detail.
α-helices, β-sheets and random-coil formations are the secondary structure elements of proteins [4]. In proteins many regions of secondary structures can be existent as a result of the locality of secondary structures [8].
The tertiary structure is the overall three-dimensional structure of molecules [4]. It is built on the interactions of the lower-order secondary structures [4] [2]. Helices are examples of RNA and DNA tertiary structures. Moreover the so-called pseudoknot is a tertiary structure of RNA and it will be explained in section 2.5. [2]. In proteins the tertiary structure often folds into several domains, whenever the proteins are big enough [4]. The tertiary structure is crucial for the function and the characteristics of a protein [4]. It should be also mentioned that proteins have a quaternary structure, which describes the result of spatial relationships between several protein domains [4] [8].
In Figure 4 the different structures of a protein can be compared.

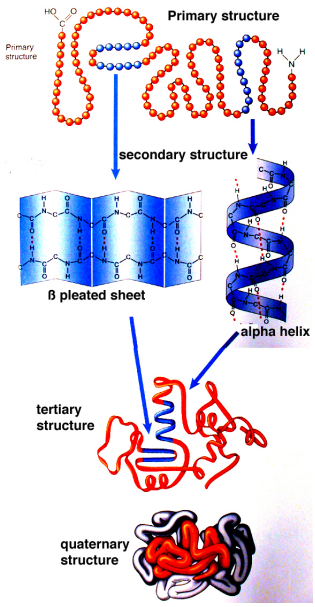

Figure 4: The different structures of a protein [9]

2.3 Importance of RNA secondary structure prediction

The three-dimensional structure of a RNA sequence has a functional role for many molecules and since knowing the sequence is not sufficient to determine the structure, as mentioned above, it is important to be able to predict the most probable structures [1]. The different structures are important for catalytic, regulatory or structural roles within the cells and according to Elena Rivas and Sean R. Eddy the prediction of RNA secondary structure seems to be an achievable goal, because the Watson-Crick pairs are a relatively simple and stereotyped interaction [7].
Due to the argument that secondary structure can be predicted relatively easy and it is harder to predict tertiary structure, it is assumed that secondary structure is formed before tertiary structure. Therefore the secondary structure is normally predicted before it is tried to predetermine the tertiary structure [2]. Furthermore the three-dimensional structure of a RNA sequence should always be treated as a distribution of multiple structures [2]. And according to this thermodynamical viewpoint the structure can change easily and RNA is more chemically unstable than DNA [2].
Another important aspect of the prediction of RNA secondary structure is that there are many sequences whose structures have not yet been experimentally determined and for which there are no homologues in the databases from which the structure could be derived [10]. Hence it is a good idea to predict the structure.
Moreover it has been shown that RNA secondary structure prediction has applications to the design of nucleic acid probes [11]. It is also used by molecular biologists to help

predict conserved structural elements in non-coding regions of gene transcripts [11]. Finally there is also an application in predicting structures that are conserved during evolution [11].

2.4 Different secondary structure elements

Because the secondary structure of RNA is important in order to understand the procedural method of the algorithms used in my program, I am going to explain the known structures in this chapter.
In addition the structure elements will be specified in a formal notion as well, since the algorithms I am going to describe below also use the notion of the different RNA secondary structures: One way to identify the numerous structures is by its closing base pair [12]. The closing base pair is the one with the largest bond and if one wants to get to another accessible base pair within this one, there will not be any.
The formal definition is: R is an RNA structure over a sequence S. A base k is accessible from a closing base pair $(i, j) \in R$ if there are no other base pairs $(i', j') \in R$ such that $i < i' < k < j' < j$ [12]. An interior base pair is a base pair $(k, l) \in R$ which is accessible from a closing base pair (i, j) [12].

2.4.1 A stem

A stem is just a simple base pair and if several base pairs follow each other a helix can be formed. In addition to theabove-mentioned A-helix there exist also other helical structures which can be formed by RNA stems [2]. The elongation of a helix by one base pair is the fastest process in building a RNA structure, next to simple stacking [2]. In the formal definition a stem is a loop with a closing base pair $(i, j) \in R$ and one interior base pair $(i + 1, j - 1) \in R$ (see Figure 5.1.a) [12].

2.4.2 A hairpin loop

A hairpin loop describes the structure of a sequence that folds back on itself, usually a stem or a double helix and thereby forming an unpaired loop. Such a loop is called a hairpin loop and is formed relatively quick [2]. The time needed to grow the loop is at its minimum in the range of only a few microseconds and is growing with the length of the unpaired loop [2].
The thermodynamical stability of the loop depends on the sequence of the loop, on the type of the closing base-pair and on the size of the loop [2]. A hairpin loop needs at least four unpaired bases and often loops of five unpaired bases are the most stable ones [2].

Very stable tetraloop hairpins can be found in rRNA and even bigger hairpin loops can for instance be found in tRNA: The anticodon loops consist of seven bases [2].
In the formal definition a hairpin loop is a loop with closing base pair $(i, j) \in R$ and no interior base pairs (see Figure 5.1.b) [12].

2.4.3 An internal loop

Internal loops have unpaired bases on both strands in a double-stranded region. The thermodynamical stability of the loops depends on the types and the number of the unpaired bases [2]. If the number of the unpaired bases in both strands are of equal size, the internal loop is called symmetric and if in addition the size is one, it is called mismatch [2].
Nevertheless the loop can be very inflexible due to stacking and/or hydrogen bonds [2].
In the formal definition an internal loop is a loop with closing base pair $(i, j) \in R$ and one interior base pair $(i', j') \in R$ with $(i' - i) + (j - j') > 2$ (see Figure 5.1.d) [12].

2.4.4 A bulge loop

Bulge loops have unpaired bases on only one strand in a double-stranded region, whereas the other strand only has paired bases [2]. The size of the bulge loop is at least the size of one unpaired base, but in principle there is no upper limit [2]. They have the ability to bend a stem and thereby influence the three-dimensional structure [2].
In the formal definition an interial loop is called a bulge loop, if $i' = i + 1$ or $j' = j - 1$ (see Figure 5.1.c) [12].

2.4.5 A multiloop

Loops which connect more than two helices are called multiloops. In between the helices unpaired bases can be found. Together with the closing base pair, the unpaired bases are decisive for the stacking of the helices and thereby they form the three-dimensional structure [2].
Very often it can be observed that four helices are connected within a multiloop, for instance in tRNA, but also more or less helices can be connected [2].
In the formal definition a k-multiloop is a loop with interior base pairs $(i_1, j_1) \ldots (i_k, j_k) \in R$ and a closing base pair $(i, j) \in R$ (see Figure 5.1.e) [12].

In Figure 5.2 a typical RNA secondary structure can be observed, containing all structural elements described above: stems (S), hairpin loops (H), internal loops (I),

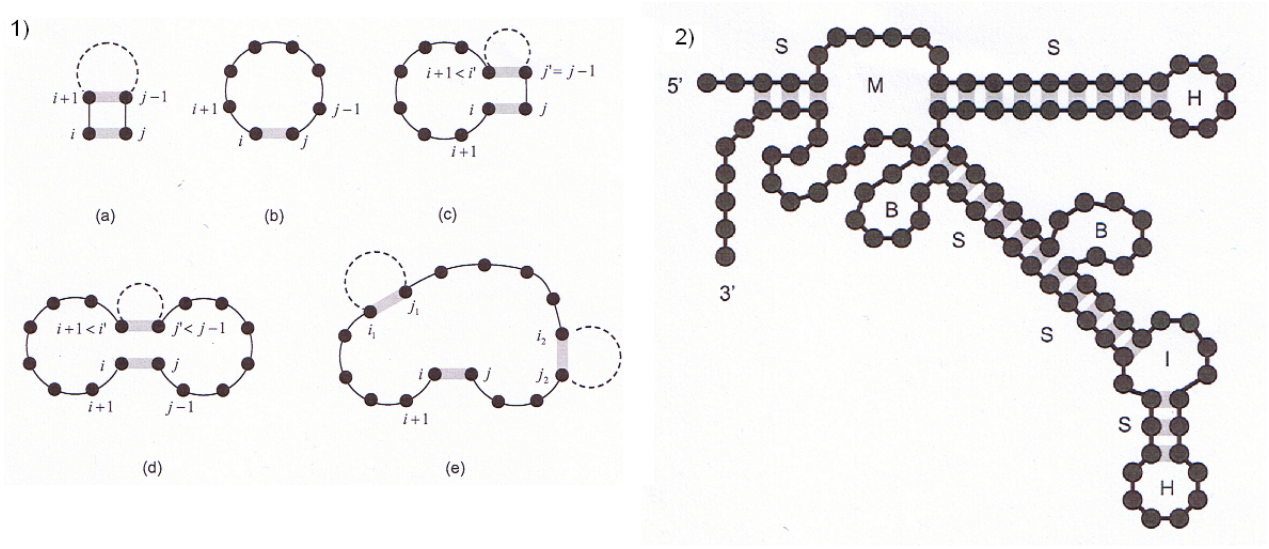

Figure 5: In both figures covalent bonds are indicated by solid lines and hydrogen bonds by gray areas 1)The structure elements are defined by their closing base pair (i, j) [12] 2) Typical secondary structure [13]

bulge loops (B) and multiloops(M).

2.5 Pseudoknots

A pseudoknot is a tertiary structural element of RNA. It is formed by base-pairing between an already existing secondary structure loop and a free ending [2]. Nucleotides within a hairpin loop form base pairs with nucleotides outside the stem [7]. Hence base pairs occur that overlap each other in their sequence position [15]. The principle of pseudoknots is shown in Figure 6.
In a formal definition one can say that a pseudoknot is a structure violating the nesting convention, which is obeyed by all non-pseudoknotted structures [7], like the ones described above. The nesting convention indicates that for any two base pairs $(i, j) \in R$ and $(k, l) \in R$ with $i < j\ , k < l\ and\ i < k$, it follows either $i < k < l < j$ or $i < j < k < l$ [7]. Hence in a pseudoknotted structure there exist at least two base pairs $(i, j) \in R$ and $(k, l) \in R$ with $i < j\ , k < l\ and\ i < k$ and $i < k < j < l$.
The prediction of pseudoknots is very hard, in fact it has been shown that the prediction of a pseudoknotted structure with minimum free energy, that means finding the most stable structure, is a NP-complete problem [14]. It has been proven that the pseudoknot prediction problem can be reduced to the well-known NP-complete 3SAT problem and hence it is NP-complete as well [14].

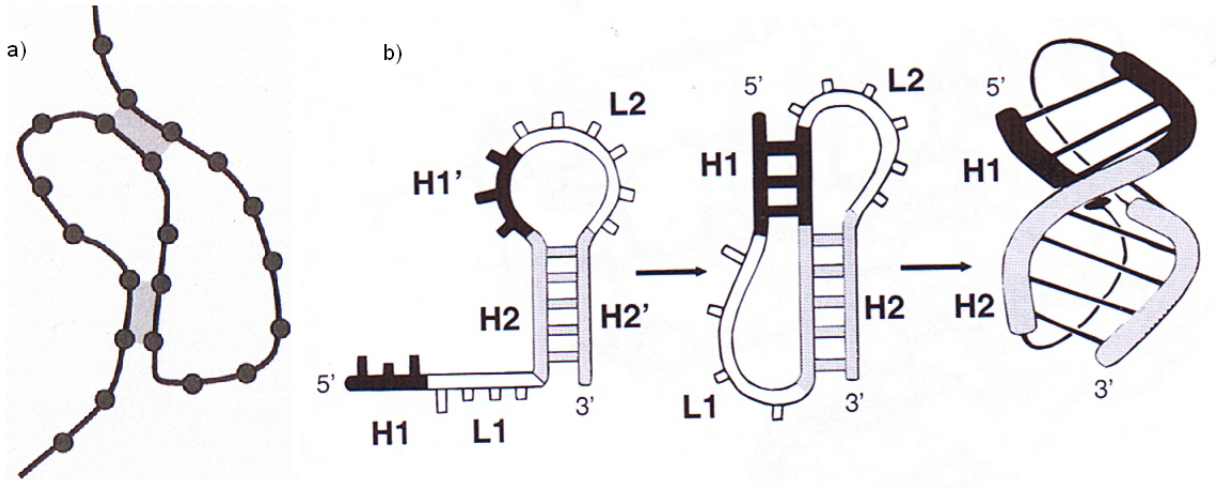

Figure 6: Pseudoknot structures: a) [13] b) [2]

NP-complete or NP-hard is *"the complexity class of decision problems that are intrinsically harder than those that can be solved by a nondeterministic Turing machine in polynomial time."* [16]. Hence it becomes impossible to simply find the optimal solution if the number of bases in the sequence is really high.
Nonetheless the tertiary structure of some RNA molecules is crucial for their biological function [2]. An example for this is human telomerase [15].

3 Prediction Methods

In this part an overview of several prediction methods will be given, whereat the two algorithms which are used in the program will be explained in all detail: The Nussinov algorithm and the Zuker algorithm. They both belong to the class of methods making notion of graph theory to predict a pseudoknot-free folding of a sequence of bases [2]. Besides the approaches using graph theory for making predictions there are also approaches based on information theory and on genetic algorithms [2].

3.1 Prediction based on graph theory

Graph theory describes the use of a dynamic programming algorithm together with backtracking [2]. This approach is used for predicting the best thermodynamical structure for single-stranded RNA [2].

3.1.1 The Nussinov algorithm

The Nussinov algorithm (Nussinov et al, 1987) [7] wants to maximize the possible number of base pairs of a given sequence [2]. The underlying assumption is that the more base pairs there are in a structure the more stable the structure is and the more likely it is [2].
The algorithm takes advantage of the fact that the optimization problem can be solved by breaking it down into smaller subproblems and solving them [2] [12]. The recursive solution to the problem is to calculate the best structure for a subsequence $S[i \dots j]$ *with* $1 \leq i \leq n$ *and* $i < j \leq n$ ($n = length\ of\ S$) from the best structure for even smaller subsequences, which has been calculated before [12].
The result of the maximum number of pairs is stored in a two-dimensional matrix β at $\beta(i, j)$ [12]. There are four different cases which can occur during the calculation of $\beta(i, j)$ [12] :

1. An unpaired base i together with the best structure for the smaller subsequence $S[i + 1 \dots j]$ gives the best result (see Figure 7.1.a) [12].

   In this case the maximum number of base pairs $\beta(i, j)$ is the same like $\beta(i + 1, j)$ [12].

2. An unpaired base j together with the best structure for the smaller subsequence $S[i \dots j - 1]$ gives the best result (see Figure 7.1.b) [12].

   In this case the maximum number of base pairs $\beta(i, j)$ is the same like $\beta(i, j - 1)$ [12].

3. A base pair (i, j) together with the best structure for the smaller subsequence $S[i + 1 \dots j - 1]$ gives the best result (see Figure 7.1.c) [12].

   In this case the new base pair (i, j) is added with the score $\delta(i, j)$ to $\beta(i + 1, j - 1)$ receiving the maximum number of base pairs $\beta(i, j)$ [12].

4. The combination of the best two structures for the smaller subsequences $S[i \dots k]$ and $S[k + 1 \dots j]$ gives the best result (see Figure 7.1.d) [12].

   In this case k has to be found such that $\beta(i, k) + \beta(k + 1, j)$ is maximal and then the maximum number of base pairs $\beta(i, j)$ is $\beta(i, k) + \beta(k + 1, j)$ [12].

The case $\beta(i, i)$ as well as the case $\beta(i, i - 1)$ have to be made impossible. Therefore the maximum number of their base pairs is set to zero, because the algorithm tries to maximize β [12]. In the following the equations of all cases are shown in summary:

$$\beta(i, i) = 0 \qquad \text{for } i = 1, \dots, n$$

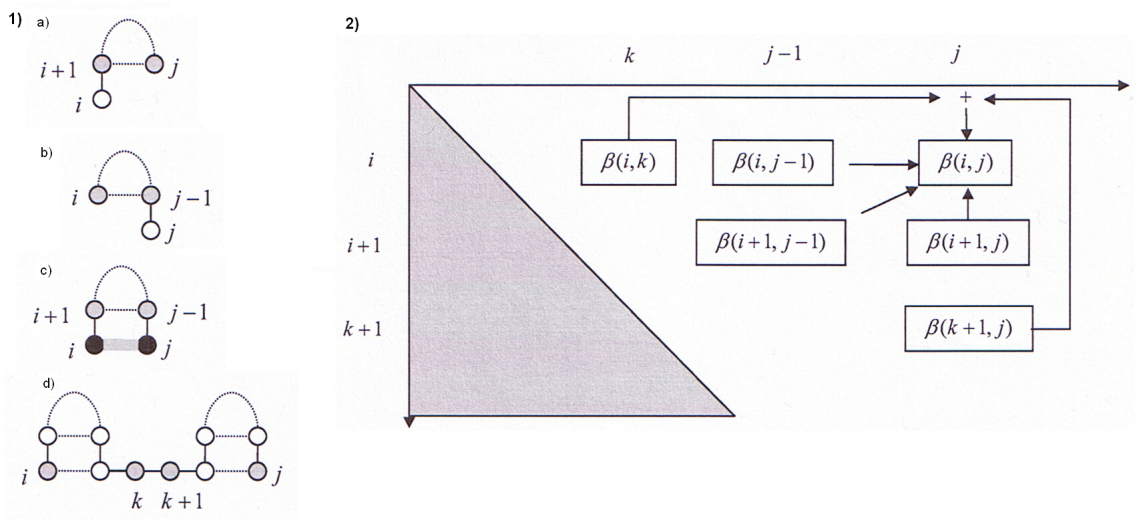

Figure 7: 1) The four different cases [12] 2) Filling of the matrix [12]

$$\beta(i, i-1) = 0 \text{ for } i = 2, \ldots, n$$

$$\beta(i,j) = \max \begin{cases} \beta(i+1, j) \\ \beta(i, j-1) \\ \beta(i+1, j-1) + \delta(i,j) \\ \max_{i<k<j}\{\beta(i,k) + \beta(k+1, j)\} \end{cases}$$

The definition of the score δ can be a constant or a value of the free energy. Furthermore if $\delta(i,j)$ is defined correctly it can allow only admissible base pairs and exclude incorrect ones[12] :

$$\delta(i,j) = \begin{cases} 1 & \text{if } (i,j) = \text{(A,U) or (C,G)} \\ 0 & \text{else} \end{cases}$$

The matrix β is then filled, beginning with the initialization of the diagonal and then computing either the upper or the lower half and beginning with smaller sequences [12]:

for $l = 1$ to n **do**
  for $i = 1$ to $n + 1$ - l **do**
    $j = i + l$
    compute $\beta(i,j)$
  end for
end for

This principle is also visualised in Figure 7.2 .
After the matrix has been filled the solution can be received via backtracking [2]. Beginning at the maximum number of base pairs for the whole sequence, which is stored in $\beta(1, n)$, one just needs to trace back from which of the four possible cells the maximum has been calculated. Then the subsequences are known and for them again backtracing is done. Like this the base pairs of the optimal folding can be revovered [2]. The overall time complexity of the recursion is $O(n^3)$ and the complexity of space is $O(n^2)$ [12].

3.1.2 The Zuker algorithm

But structures with a maximum number of base pairs rather are biologically and biochemically unrealistic [18]. Moreover the submaximum number of base pairs will not be found with the Nussinov algorithm [2] and the different types of possible loops will not be taken into account, either.
The Zuker algorithm (Zuker et al. 1981 [17]) is also a dynamical programming algorithm and it works on the basis of identifying the globally minimal energy structure for a sequence [17]. This algorithm has been a fusion of the mathematical approach to predict RNA secondary structures, like the Nussinov algorithm and of the approach using thermodynamics, like for instance by Pipas and McMahon [17].
The Zuker algorithm is more sophisticated than the Nussinov algorithm due to the fact that for every structural element an individual energy is calculated which then contributes to the overall energy of the structure [12]. Therefore an energy function is defined for each of the loops explained in 2.4, with the closing bond (i, j):

1. the energy of a stem is $eS(i, j, i + 1, j - 1)$
2. the energy of a hairpin loop is $eH(i, j)$
3. the energy of an internal or a bulge loop is $eL(i, j, i', j')$
4. the energy of a k-multiloop is $eM(i, j, i_1, j_1, \ldots, i_k, j_k) = a + bk + ck'$, with $(i_1, j_1) \ldots (i_k, j_k)$ being the interior base pairs and a, b, c = constants and k' = number of unpaired bases in the loop

Then every secondary structure elements contributes its energy value $E^R_{i,j} \in \{eS, eH, eL, eM\}$ to the overall free energy $E(R)$ of an RNA structure R [12]:

$$E(R) = \sum_{(i,j) \in R} E^R_{i,j}$$

During the calculations three different functions have to be optimised, which will be handled by filling of three matrices as well [12].

- $\epsilon(i)$ stores the minimum free energy of a structure on subsequence $S[1 \dots i]$, the whole sequence being $S[1 \dots n]$
- $\pi(i, j)$ stores the minimal free energy of a structure on subsequence $S[i \dots j]$ with (i, j) being a base pair
- $\mu(i, j)$ stores the minimal free energy of a structure on subsequence $S[i \dots j]$ that is part of a multiloop

Beginning with the description of $\pi(i, j)$, four different cases can occur [12] :

1. A stem is closed by the base pair (i, j)

   In this case the energy $E^R_{i,j}$ is the sum of the energy of the stem $eS(i, j, i + 1, j - 1)$ and the minimum energy of a structure on smaller subsequence $S[i + 1 \dots j - 1]$ with closing pair $(i + 1,\ j - 1)$ (see Figure 8a) [12].

2. A hairpin loop is closed by the base pair (i, j)

   In this case the energy $E^R_{i,j}$ is the sum of the energy of the hairpin loop $eH(i, j)$ (see Figure 8b) [12].

3. An internal loop or a bulge loop is closed by the base pair (i, j)

   In this case the energy $E^R_{i,j}$ is the sum of the energy of the loop $eL(i, j, i', , j')$ and the minimum energy of a structure on smaller subsequence $S[i' \dots j']$ with closing pair (i', j') (see Figure 8c) [12].

4. A k-multiloop is closed by the base pair (i, j) .

   In this case the offset penalty a is added to the energy, because a multiloop is an unstable structure [12]. Furthermore the multiloop is divided into two smaller subsequences $S[i + 1 \dots k - 1]$ and $S[k \dots j - 1]$, such that the energy $E^R_{i,j}$ which is the sum of a, $\mu(i + 1, k - 1)$ and $\mu(k, j - 1)$, is minimal (see Figure 8d) [12].

In addition a base pair (k, l) which is not admissible can be ruled out by setting $\pi(k, l) = \infty$ [12]. Since the Zuker algorithm tries to minimise the energy (k, l) is an impossible choice. According to this principle also the base pairs $(i, i - 1)$ and (i, i) which cannot form in a structure are eliminated by setting $\pi(i, i - 1) = \pi(i, i) = \infty$ [12].

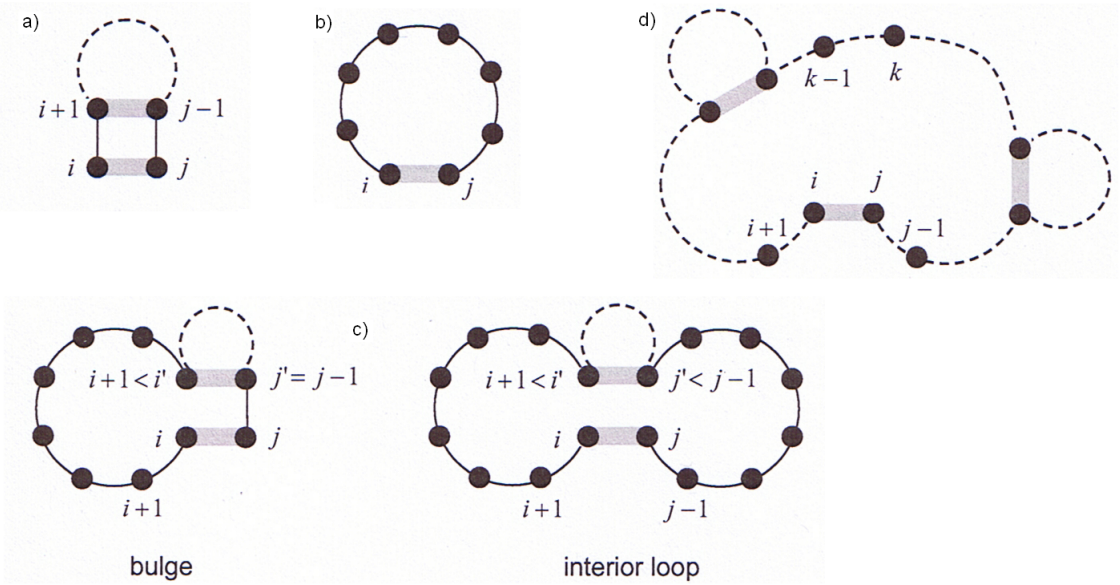

Figure 8: Four different cases in the computation of $\pi(i, j)$ [12]

In summary it holds for all i, j with $1 \leq 1 < j \leq n$ [12] :

$$\pi(i,j) = \min\{E(R) \mid R \textit{ structure for } S[i \ldots j] \wedge (i,j) \in R\}$$

$$= \min \begin{cases} eS(i,j,i+1,j-1) + \pi(i+1,\, j-1) \\ eH(i,j) \\ \min\limits_{i<i'<j'<j \wedge i'-i+j-j'>2} \{eL(i,j,i',,j') + \pi(i',j')\} \\ \min\limits_{i+1<k\leq j-1} \{\mu(i+1,k-1) + \mu(k,j-1) + a\} \end{cases}$$

A trick is used for the computation of the multiloop structure with the minimum energy: The multiloop is recursively cut down into two parts to the interior base pairs [12]. Either one base is moved or another bifurcation is added [12]. This leads to the distinction of the following four cases of $\mu(i, j)$ [12]:

1. One moves to base $i + 1$.

   In this case the energy $E^R_{i,j}$ is the sum of the minimum energy of a structure on smaller subsequence $S[i + 1 \ldots j]$ and the penalty c for one unpaired base (see Figure 9a) [12].

2. One moves to base $j - 1$.

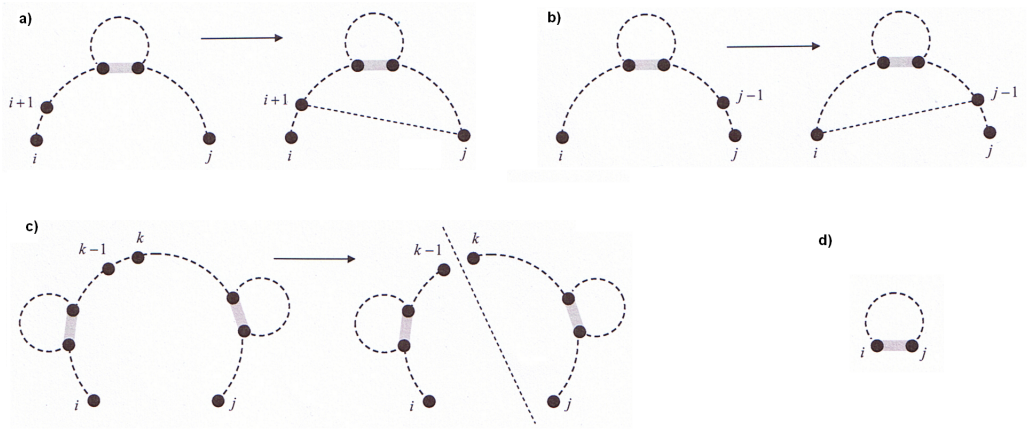

Figure 9: Four different cases in the computation of $\mu(i, j)$ [12]

In this case the energy $E^R_{i,j}$ is the sum of the minimum energy of a structure on smaller subsequence $S[i \dots j - 1]$ and the penalty c for one unpaired base (see Figure 9c) [12].

3. The multiloop is further divided at base k.

   The multiloop is divided such that the energy $E^R_{i,j}$ which is the sum of $\mu(i, k - 1)$ and $\mu(k, j)$, is minimal (see Figure 9c) [12].

4. An interiror base pair (i, j) is found.

   In this case the energy $E^R_{i,j}$ is the sum of the reward b and the energy of the structure of the subsequence $S[i \dots j]$ with (i, j) being a base pair (see Figure 9d) [12].

To ensure at least two interior base pairs it is given that $\mu(i, i) = \infty$ [12].
In summary it holds for all i, j with $1 \leq 1 < j \leq n$ [12] :

$$\mu(i,j) = \min\{E(R) \mid R \textit{ structure for } S[i \dots j] \textit{ that is part of a multiloop}\}$$

$$= \min \begin{cases} \mu(i+1, j) + c \\ \mu(i, j-1) + c \\ \min\limits_{i<k\leq j} \{\mu(i, k-1) + \mu(k, j)\} \\ \pi(i, j) + b \end{cases}$$

Computing the last function $\epsilon(i)$ leads to the distinction of the following two cases [12] :

1. An unpaired base i is simply added to the best structure for smaller subsequence $S[1 \dots i - 1]$ (see Figure 10a) [12].

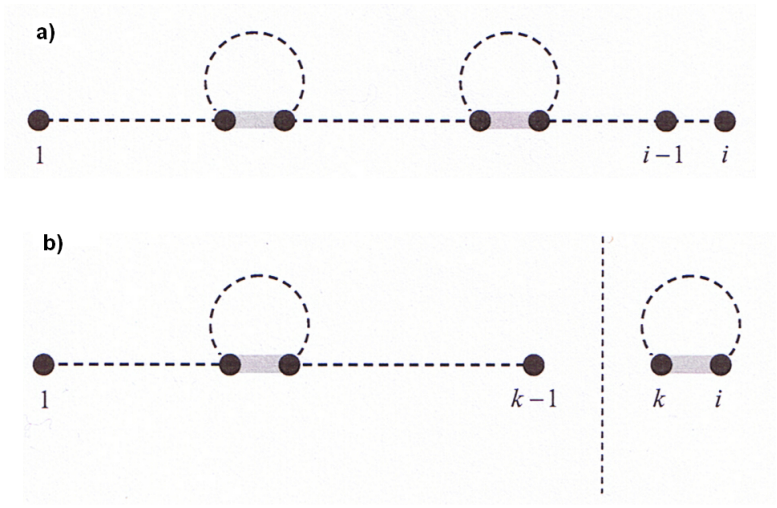

Figure 10: Two different cases in the computation of $\epsilon(i)$ [12]

2. Base i pairs with another base k and the energy is the sum of the energy of smaller subsequences $S[1 \dots k-1]$ and $S[k \dots i]$, whereas the last one is closed by (k, i) (see Figure 10b) [12].

In summary it holds for all$\epsilon(i)$ with $1 \leq i \leq n$ [12]:

$$\begin{aligned} \epsilon(i) &= \min\{E(R) \mid R \textit{ structure for } S[1 \dots i]\} \\ &= \min \begin{cases} \epsilon(i-1j) \\ \min\limits_{1 \leq k \leq i} \{\epsilon(k-1) + \pi(k,i)\} \end{cases} \end{aligned}$$

The minimal free energy is computed via dynamic programming by filling the three matrices $\epsilon(i)$, $\pi(i,j)$ and $\mu(i,j)$ for all $1 \leq i < j \leq n$ [12]. In this point the procedure is the same as it is with the Nussinov algorithm. After the matrices have been filled the minimal free energy is stored in $\epsilon(n)$ and the best structure can be found by backtracing [12]. Furthermore not only the best overall structure for a given sequence $S[1 \dots n]$ has been computed, but also the best structure for each of the possible subsequences of S [17].
The complexity is the same as for the Nussinov algorithm, namely the complexity of time is $O(n^3)$ and the complexity of space is $O(n^2)$ [12].
Altogether it has been shown that the Zuker algorithm is a very powerful framework for RNA secondary structure predictions [11], but it is also important to mention that it still is only an approximation of RNA folding and that there might be more complicated processes and factors in the real RNA folding [12].

3.2 Prediction based on information theory

The approach adapted from information theory is used when no thermodynamical parameters are known [2]. This is not very unusual since one cannot be sure wether the thermodynamical parameters known so far are all totally correct and not all of them are known yet, for instance the parameter values for multiloops [2]. The underlying assumption of this approach is that many RNA sequences of homologues are known and although they are different due to mutations during evolution, it is thought that they have a similar structure, since they function in a similar way [2]. Therefore researchers assume that the mutations or differences in the sequence can be found in non-functional loop regions or that they are compensated by another mutation, so that the bases pair again [2].
Shannon's information theory (Shannon and Weaver 1949) with its entropy measure is used to make good predictions [2]. Entropy is a measure of information. Therefore every possible event has a certain probability and is assigned a certain entropy according to the probabilities [2]

3.3 Prediction based on genetic algorithms

Another remaining problem of the approaches by graph theory is that the number of different thermodynamical parameters grows exponentially with the number of the bases within a sequence [2]. Therefore methods using heuristics are also used for making good predictions. Examples of these methods are genetic algorithms or Monte Carlo methods [2].
A genetic algorithm is an optimization technique which uses biologically inspired techniques such as mutation, selection, and crossover [19]. It operates over a set of individuals over many generations to get better solutions over time [20]. Moreover it is guaranteed that its solutions do get better over time and not worse [20]. But one should keep in mind that for both methods it is not guaranteed that they find an optimal solution and therefore Gerhard Steger suggests to use them only when deterministic or analytical methods have failed [2].
Since genetic algorithms do not find the optimal solution of a specific problem [20], but come up with good results within a relatively small amount of time, they are a good approach to find solutions for NP-complete problems, like the pseuoknot prediction problem, for example.

4 The Program

4.1 Implementation

In this part of the thesis I am going to give an overview of the whole programm I have made to predict secondary RNA structure of a given sequence. First of all a guide how to use the program will be revealed. Thereafter a more technical insight of the structure will be given and it will be explained in more detail how the different classes interact with each other. In addition the implementation of the Nussinov algorithm, the Zuker algorithm and the method to predict a pseudoknotted structure will be illustrated and finally I am going to provide an insight into the different graphical representations which allow to display the RNA secondary structures.

4.1.1 Manual for users

The user of the program is able to enter any RNA structure or load a predefined structure into the specified textfield. Then he or she can decide to let the program run the computations either with the Nussinov algorithm, the Zuker algorithm or with a variation of the Zuker algorithm which is able to predict one pseudoknot. All the three implementations of the different methods are explained below. Furthermore the user can enter the values for the several parameters, if appropriate. There are for instance no parameter values to be changed in the Nussinov algorithm. If the user does not enter any values or just a few, the program uses predefined default values.
Moreover two `JCheckBoxes` can be found which activate to take into account the natural logarithm of the length of the unpaired bases or to make use of energy values.
Besides, a '*help*' button can be found, which, if activated, leads to the opening of an extra window with this manual.
Whenever the user clicks on the '*start*' button the program first reads in the sequence and the values and checks weither they are admissible and then it starts the computations with the method the user has chosen.
As soon as the best structure has been found and the calculations have been finished, the program loads two of the graphical illustrations of the structure into a panel and the user can observe it. He or she can also have a closer look on the structure by clicking on a '*show*' button and then a window pops up with the third graphical illustration. But because this one needs more time to be created the user has to request it separately.
Furthermore the user has the option to save the computed structure in one of the three possible formats. To do so, the user has to click on one of the three appropriate buttons and then a dialog pops up, asking the user where to store the result.

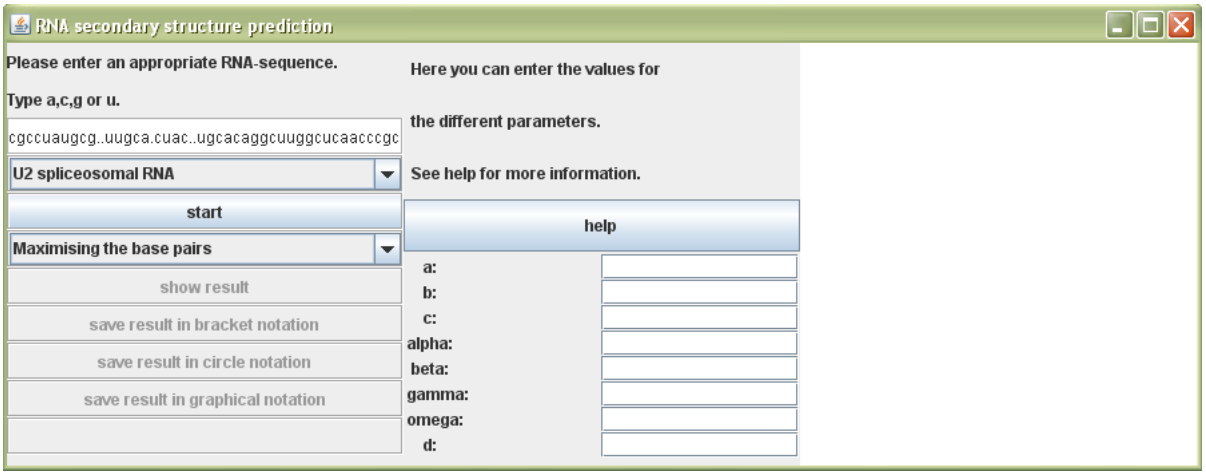

Figure 11: A view of the program

4.1.2 The structure

I have written the code of the program in Java and not in Fortran like the original Zuker algorithm [17]. It is also important to mention that the whole package does not only contain a main class which starts an application, but also one which is loaded whenever a corresponding html file is activated and then the whole program can be used as a `JApplet` within a browser. Furthermore it is also possible to call the program directly. Then the structure of predefined RNA sequences will be calculated and compared to the correct ones (see 4.2.1). Hence I have been able to use this fast call for my experiments and I could easily compare the results, since the sequences have always been the same.
For the exact description of the program I will continue to use the first mode of the program, namely the simple application.
Since the user is able to enter different values for the parameters, the first part of the program is in charge of the input process.
The second part is the computation of the best structure in one of the three modes. Here the structure and its graphical representation can be found according to the parameter values.
The chronological third part is to act according to the user's inputs and either show the result in another notation or save it.
For the GUIs, Swing components were used. All components are built up according to the *Model-View-Controller* distinction [22]:

- The model stores the data.
- The view allows for the graphical representation of the data.

- The controller reacts to commands of the user.

Due to the so-called *look-and-feel*, view and controller are combined in Swing-components [22].
The whole package in which the program is defined is called `zu` and in the subpackage `zu.zuker` the three different classes for the three different calls of the program can be found. In the class `App` the application mode is started, in the class `Applet` the browser mode is started and in the class `Run` the direct call is started.
In the main method of the first and the second mode an object of the class `GUIBuilder` is instantiated. This class resembles the view and the controller and can also be found in the package `zu.zuker`. The third mode does not need a view or a controller because it is only used for the parameter estimation and it does not have to be able to react to the user's input, but only the result is interesting.
In the program the class resembling the model is called `Model` and it is an abstract class stored in the package `zu.model`. The different models are extended from `Model`, one for the Nussinov algorithm, one for the Zuker algorithm and one for the pseudoknot prediction. After having instantiated the `GUIBuilder`, the method `GUIBuilder.getGUI()` is called and this method returns a `JPanel` in which the whole graphical representation of the program can be found.
This `JPanel` contains `JTextFields` for the entering of a sequence and/or the parameter values. Furthermore it also contains `JComboBoxes` where the user can decide for one model and/or load one of four given RNA sequences into the corresponding `JTextField`. Moreover a JPanel can be found in which, whenever a computed structure exists, the graphical notation of this structure is presented. In addition also a `JLabel` exists where the user gets informed about the actual state of the program and if for instance the entered RNA sequence was incorrect, then this is displayed on the `JLabel` as well. Finally also several `JButtons` with their corresponding `ActionListeners` resembling the controller are on the GUI: One '*help*' button displays the manual if activated. One '*start*' button initiates the model with the given sequence and parameter values and starts the whole computation of the structure. If the parameter values are not entered and/or not appropriate, default-values are used. One '*show result*' button, if enabled by an existing result and then activated, constructs a `JDialog` that pops up and displays the structure in a certain notion. The different notions are explained below in section 5.1.6. At last there are also three '*save*' buttons, one for each of the three graphical representations of the resulting structure. By using a `JFileChooser` they select a place where to store the resulting file and by either using a `FileWriter` or a `DOMFileWriter`, which can be found in package zu.zuker, they produce a file containing the result. For a better understanding of these very technical descriptions the whole source code can be asked by writing an email to soschnei(at)uos.de and the reader is invited to take a closer look on all the classes and how they interact with each other.
Every `Model` object is instantiated with a list of the RNA sequence. If further parameters

are needed it is also possible to instantiate the `Model` with the parameters or to set them to the specific values. In addition every `Model` has a method `calculate` in which according to the underlying principle the best structure is calculated. Therefore an object of the class `Structure` is instantiated. This class can be found in the package `zu.structure`. Every `Structure` contains one list with the RNA sequence and another list in which the structure is stored according to the dot-bracket notation (see 5.1.6). Besides, a `Structure` object contains several classes like for instance `paint` or `paint2` for the creation of a `JComponent` which shows one or two of the different graphical notions for the display of the structure. For this purpose exists also the class `Todo` in `zu.structure` which resembles an agenda in which is stored which subparts of the sequence still have to be drawn. These are stored and added into a list and unless this is not empty, the `Structure` gets painted. In addition all classes of the package `zu.graphics` are used to allow the painting of a structure with all its graphical objects, like for instance bases and bonds.
The energy, if existing, is also stored in the `Structure` object.
Whenever the `calculate` method has finished successfully and a new `Structure` has been instantiated, this `Structure` is stored in the `GUIBuilder` and is displayed. Also the buttons for showing or saving the results are enabled. Unless the '*start*' button is clicked again, the whole program remains in this state. Whenever the button is clicked, the whole input process and computation process starts again until the new `Structure` is stored and the old one is deleted.
This process is the same for the application as well as for the applet. But for the third mode the `main` method of the `Run` class is called. Then the corresponding models for the predefined structures will be instantiated. Afterwards the accuracy of the resulting structures will be measured. How this is done is explained below in section 4.2. Then only the overall accuracy is computed and printed out.

4.1.3 Nussinov algorithm in the program

The Nussinov algorithm is implemented in my program in the class `BPModel` in `zu.model`. It extends the abstract class `Model` and therefore has a constructor which is called with a list of characters. This list, named `list` as well, resembles the RNA sequence and is stored as an attribute of the `BPModel`. Furthermore the model has a two-dimensional matrix called `matrix`, which is used for the storage of the maximum number of base pairs of the different subsequences. Moreover, when the model gets constructed, also a list `klammern` for the storage of the different dots or brackets is instantiated. It has the same length as `list` and at the beginning is filled with dots only. Later on the brackets will replace the dots at the corresponding positions.
The method `calculate` computes the maximum number of base pairs for the given sequence. It is composed of the method `forward`, which fills the matrix according to

the principle explained in 3.1.1, and the method `traceback`, which by backtracing determines the pairs of the structure and stores them in `klammern`.
To only allow admissible pairs, the method `delta(i,j)` can be called. This method prooves weither the bases at position i and on position j can form an admissible pair. If they do it returns 1, else it returns 0.
Moreover the method `teile(i,j)` determines the maximum number of base pairs of the sequence between the base i and j and base k with $i < k < j$, such that $\beta(i,k) + \beta(k+1,j)$ is maximal. This refers to the fourth case of the Nussinov algorithm described above in section 3.1.1.
Afterwards a `Structure` object is build with `list` and `klammern`.

4.1.4 Zuker algorithm in the program

The class `ZUModel` in `zu.model` in my program is the implementation of the Zuker algorithm. Since it also extends the abstract class `Model` it has a constructor which is called with a list of characters. This list, named `list` as well, resembles the RNA sequence and is stored as an attribute of the `BPModel`. A further attribute is the list `klammern` which, as in the `BPModel`, stores the structure in dot-bracket notation. It also gets instantiated with the length of the sequence `l` and filled with dots only. In addition the `ZUModel` has two two-dimensional matrices, whereat the matrix `mu` resembles the matrix μ as explained in 3.1.2 and the matrix `tau` resembles π. the matrix ϵ is stored in the one-dimensional array `epsilon`. All the default values for the parameters are also stored in `ZUModel` and I have also introduced the variable `inf` for infinite with a value of 10000000000000000000000.0.
In the `calculate` method the structure with the minimal energy is calculated. At first, the method `initializeMatrizes` is called in which all matrices are filled with `inf`. Here also `epsilon` is instantiated.
Then the method `fillTau` is called. This method fills the matrix `tau` and also calls the method`fillMu` in every step. This is because both matrices rely on each other and cannot be filled one after the other. But since they are filled row by row and the sequences in `mu` are always at least one base shorter than `tau` it is possible to fill both matrices according to the principle explained above.
After both matrices have been filled, the method `fillEpsilon` is called and this method then returns the minimal energy, which is stored at $\epsilon(1,n)$.
Afterwards again the traceback method is called and brackets of the structure with the minimal energy are stored in `klammern` and a `Structure` object is constructed.
There are also several methods like `teile` in `BPModel` which search for the minimal energy of a subsequence (with or without a closing base pair) which is bifurcated by a base k. Additionally, also the traceback methods are numerous, since depending on which matrix or array the minimal energy which is traced back is from, another method

is needed.
Moreover there are also the methods `eS`, `eH` and `eL` which define the energy of the different structure elements, as explained in 3.1.2. The minimal energy of a multiloop which has been explained as eM, is calculated in the matrix `mu`.
The method `delta(i,j)` returns 0 if the bases at position i and j are not admissible and 1 if they are. Besides the Watson-Crick base pairs also the wobble base pair $(g : u)$ is admissible. If a base pair is not admissible, then all of `eS`, `eH` and `eL` return `inf`.
The method `eH` returns the energy of a hairpin loop. It is also assured that the number of unpaired bases is at least four, since less unpaired bases are chemically impossible[17]. The energy is the product of the number of unpaired bases times the parameter `alpha`. Its default value is:
The method `eS` returns the energy of a stem. This constant is stored in the parameter `beta`. Its default value is:
The method `eL` returns the energy of an internal loop or bulge loop. The energy is the product of the number of unpaired bases times the parameter `gamma`. Its default value is:
The constant `a` is the penalty which is charged for each multiloop and it has a value of:
The constant `b` is the reward which is given for each base pair within a multiloop and it has a value of:
The constant `c` is the penalty which is charged for each unpaired base in a multiloop and it has a value of:
The constant `omega` is used within the computation of **epsilon**, since there has to be a high energy for the sequence of the first two bases. If this would not be the case no structures would be build up at all and only one large chain would be the result. But nevertheless it cannot be `inf`. The default value of `omega` is:

4.1.5 Pseudoknots in the program

The underlying principle of the prediction of a pseudoknot in my program is that the whole sequence is split up into four bases that are violating the nesting convention, such that two base pairs $(i, j) \in R$ and $(k, l) \in R$ exist with $i < j\ , k < l\ and\ i < k$ and $i < k < j < l$.
If they are found, then for each of the five resulting subsequences the best secondary structure is computed with the Zuker algorithm and the five resulting energies are added up to the overall energy of the pseudoknotted structure. An inner loop always calculates this overall energy of the structure defined by four bases. An outer loop searches for the four bases which define the structure with the minimal energy. The best structure is stored and after all bases have been passed through it gets displayed.
For the implementation of this idea, I have been able to use the already defined `ZUModel` for the calculations of the best structures of the subsequences. Therefore only some

loops had to be constructed which ran through all the bases finding all of four-bases violating the nesting convention.
The whole implementation can be found in the class `ZUPModel` in `zu.model`.
The parameter `d` is the energy of a substructure containing only one unpaired base and its default value is:

4.1.6 The graphical representations

I have implemented three different representations which graphically show RNA secondary structure:
The first notation which is the simplest one, is the so-called dot-bracket notation. It is used for instance in the program RNAfold (Hofacker et al. 1994) [2].
A dot '·' represents an unpaired base and an opening bracket '(' together with the corresponding closing bracket ')' represents the pairing of the two bases. For the representation of a pseudoknot I have introduced squared brackets: an opening squared bracket '[' together with the corresponding closing squared bracket ']' represents a pseudoknot.
Furthermore the bases are written above each dot or bracket, so that it is clear to which base the symbol corresponds (see Figure 12a).
The principle of representing base pairs by brackets and unpaired bases by dots has also been used in the program during the storage of the structure. Every `Structure` object contains one `List` in which the base characters are stored and one `List` in which the hydrogen bonds are stored in the dot-bracket notation. Hence these two `Lists` are read out as soon as this notation is used.
Whenever the user saves the result in this notation an `OutputFileWriter` is created and the two lines are written into a textfile.

The second notation is the so-called circles plot, which has been used by Nussinov et al. in 1978, for example [2]. It is a notation which is able to show more than just secondary structures [2]. Here the sequence is shown in a circle and every base is displayed as a dot and its corresponding character next to the dot. The gray lines connecting the bases show the covalent bonds, the red and blue lines indicate hydrogen bonding, wherest the blue lines only indicate pseudoknots (see Figure 12b).
The circle's plot notation has been implemented by simply calculating the angles for each base. If for instance the sequence consists of 36 bases, then 360° are divided by 36 and it becomes clear that after 10° wandering down the circle a new base has to be placed. The exact coordinates of each base have been calculated using the sin and cos functions.
Since a `List` exists in the `Structure` object in which the bondings are stored in the dot-bracket notation, only red lines have to be drawn between the bases at which place the matching brackets are stored. Whenever squared brackets occurred the lines have

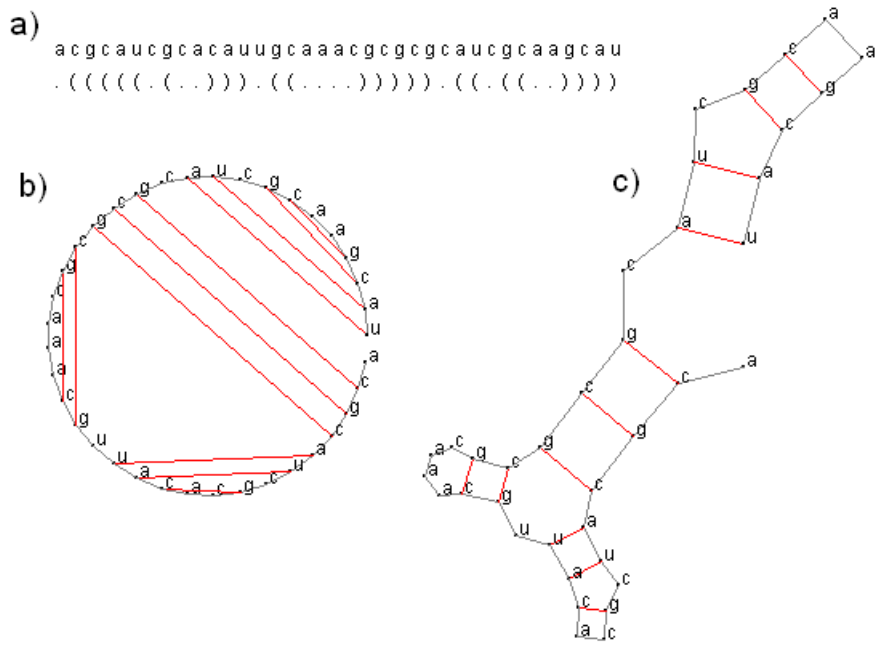

Figure 12: Three different notations of a pseudoknot-free structure.

been drawn in blue colour to indicate a pseudoknot.
Whenever the user wants to save the structure in this notation, SVG (Scalable Vector Graphics) files are created [21]. Their advantage is that the user can zoom into the graphics without any grainy pixels. The graphics can be regarded by most of the common Internet browsers.

The third notation is similar to the so-called squiggles plot (Osterburg and Sommer 1981) [2]. It is the notation which best resembles the real secondary structure but it is not able to show pseudoknots. As in the circle's notation bases are indicated by black dots and their character, covalent bonds are indicated by gray lines, hydrogen bonds by red lines and hydrogen bonds of a pseudoknot by blue lines (see Figure 12c). In fact the notation is able to show a pseudoknot, but biologically it makes no sense.
Stems or helices are shown in a tracklike structure [2] and the closing base pairs of loops have the same distance like the stem base pairs [2]. The loops are indicated by equiangular polygons. As a result a very compact graphic can be observed and the probabililty of an overlapping of the different structural elements is very small [2]. But the loops can get very small, such that the sizes of the loops should not be compared. Furthermore as the sequences get longer the loop sizes can get so small that it is hard to regard them.

The structure is again read out from the `List` in the `Structure` object and then a recursive paint method is called with the whole sequence. Then all bases which belong to the structure element are identified. The bases are unpaired bases within a loop and/or the closing base pairs of other structure elements. Then this loop with all its bases is drawn like in the circle plot notation.
To identify the closing base pairs of other structural elements the opening and closing brackets behind the first opening bracket are added or subtractedand if this number is zero again and a closing bracket is found, this is the correct base to pair. Then the paint method is recursively called with all the bases between the closing base pair, since they again have the next structure element.
To get a correct equiangular polygon further constraints are put on the paint method: (360° - anglestepwidth of old circle) is divided by (number of bases in new circle + 1) to receive the correct angle stepwidth of the new circle. The paint method needs to be called with the centre of the new circle and the angle from which the drawing of the bases is started. The new center's distance to the old center's distance is twice the distance from the old center to the point in the middle of the two closing bases. The angle to the new centre is the same as from the old centre to the point in between the closing base pairs. Furthermore the angle from where the bases are drawn in the new circle is the same as the one between the centers plus half of the old circle's angle step width. Like this it is assured that the new starting point from which the bases are drawn is exactly one of the closing base pairs and that the circle would also end at the other closing base pair.
As for the circles plot, the structures shown with this notation are saved by the creation of SVG files.

4.2 Parameter value estimation

In my experiments I have made use of the fast, direct call of the program. Since the predefined six sequences were always the same I could compare their results easily. I have decided for six sequences whose structures have been available in RNA databases.

4.2.1 The accuracy measure

The accuracy of the resulting sequences compared to the real structures is calculated according to the principle that the simplest way to quantify the identity between two sequences is percentage identity [23]. This identity can be calculated by dividing the number of identical matches by the number of all bases in a sequence [23]
For every base within the sequence it has been compared wether it is also an unpaired base, an opening or a closing base. Whenever the base is the same as in the real

structure it gets awarded. At the end the accuracy is the percentage of the correct bases with respect to the length of the sequence.

4.2.2 The database used

According to Zvelebil et al. a database is a repository of information, having a specific structure so that it is possible to enter or extract data [24]. In general a database has files or tables and accordingly I have been able to extract the correct structure of a specific sequence. Some biologically important databases for secondary structures are Prosite, Prints, Pfam and Interpro [4]. But since these are databases for proteins, the first three sequences were taken from the Rfam homepage [25]. It is a database for different RNAs. All of the different sequences I have used can also be loaded into the program directly.
The first data set I have used is U2 spliceosomal RNA which is a small nuclear RNA (snRNA) component of the spliceosome (involved in pre-mRNA splicing) [26]. To be more specific I worked with the data of X52312.1/235-430.
The second sequence used is U7 small nuclear RNA (U7 snRNP). This is an RNA molecule involved in the splicing of animal histone pre-mRNAs, which are spliced by a different mechanism than nuclear pre-mRNAs and self-splicing introns [27]. More precisely M26278.1/1-53 has been used.
And the last RNA sequence used in the program is non-coding SL2 RNA. It is thought to be involved in trans splicing in lower eukaryotes [28]. AF213679.2/1-109 is the exact number of that RNA sequence
For the sequences of RNA with pseudoknots I used the PseudoBase, which is a special database for Pseudoknots [30].
The first sequence used is ribosomal frameshift signal ORF-2/3 which can be found in beet chlorosis virus (BChV) [31].
As a second sequence I worked with ORF1a/ORF1b (polymerase) ribosomal frameshift site of equine arteritis virus (EAV) [32].
The last sequence I have used is tRNA-like structure 3'end pseudoknot of okra mosaic virus(OMV) [33].

4.3 Results and value variations

First of all the accuracies of the structures developed by the Nussinov algorithm have been calculated:
The first one has been 0.49356223175965663, the second one has been 0.42105263157894735, the third one has been 0.4125 and hence the overall accuracy has been 0.44237162111286804. This result has been taken as a baseline to which I

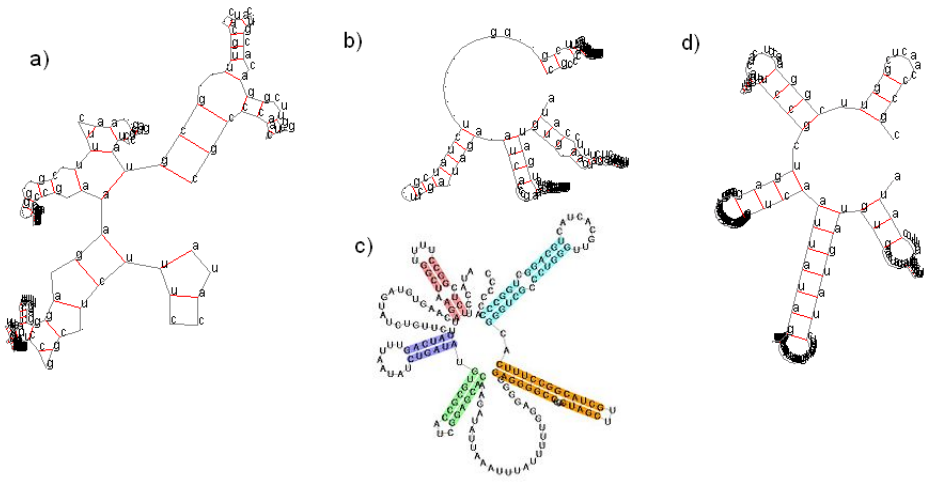

Figure 13: U2 spliceosomal RNA a) result of the program with Nussinov algorithm b) result with default values and Zuker algorithm c) the correct structure [25] d) result with energy values and Zuker algorithm

am going to compare all the other results.
When I first calculated the accuracy of X52312.1/235-430 with the Zuker algorithm and the default values : a=2.0, b=0.2, c=1.0, alpha = 0.4, beta = 0.2 and gamma = 0.3, I got an accuracy of 0.6008583690987125. I used especially these values because I expected them to make sense. For instance that the stem has very little energy compared to a hairpin loop, and so on. The result of the comparison can also be regarded in Figure 13. On the left-hand-side is the result of my program using the Nussinov algorithm, in the middle is the result of my program using the Zuker algorithm and on the right-hand-side is the correct structure. And although there is only an accuracy of 60 percent, the graphs allow to notice that both do have a related structure: very compact and with many arms around a circle.
For the second sequence I achieved an accuracy of 0.7017543859649122 and for the third sequence 0.287. Hence the overall accuracy with the default values has been 0.5300375850212083. Hence the Zuker algorithm with these very simple values achieves an accuracy being nine percent better than the baseline.

Furthermore I also calculated the accuracy using the natural logarithm. This seemed to be a good idea to me, because in the energy values Zuker used, some functions with the natural logarithm are used to compute the energy values [34]. And the idea behind this seems pretty clear: The difference in energy of two loops having one and two unpaired bases should be bigger than the energy difference between two loops having 99 and 100 unpaired bases. Therefore I introduced the possibility to activate the calculation of the energy functions for hairpin loops and internal or bulge loops dependant on the natural

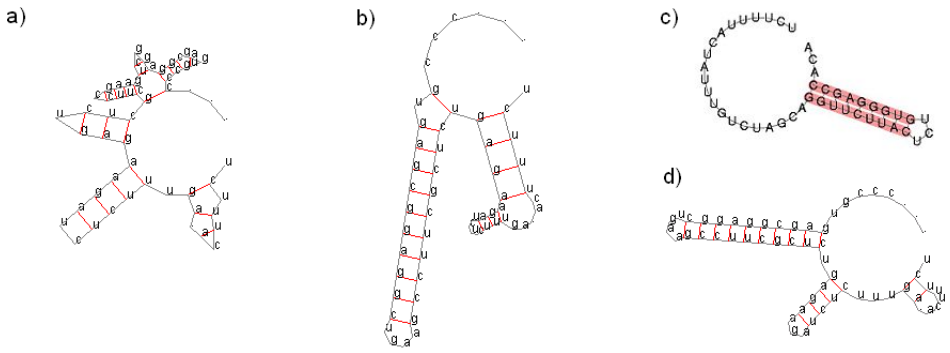

Figure 14: U7 snRNA a) result of the program with Nussinov algorithm b) result with default values and Zuker algorithm c) the correct structure [25] d) result with energy values and Zuker algorithm

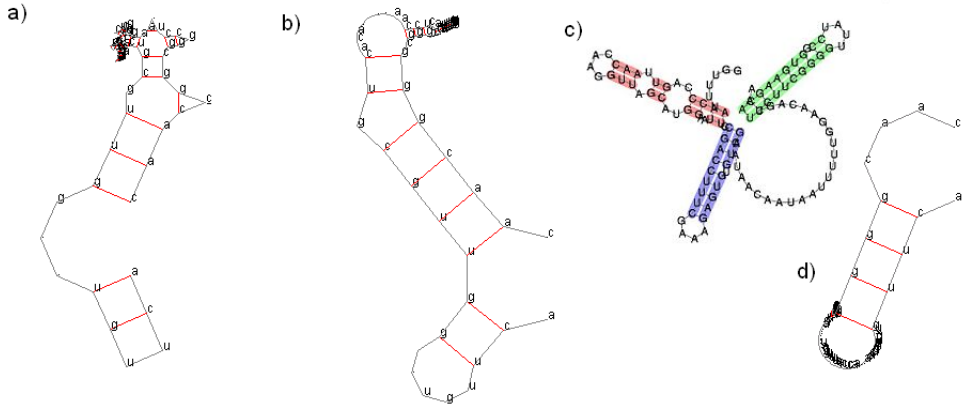

Figure 15: SL2 RNA a) result of the program with Nussinov algorithm b) result with default values and Zuker algorithm c) the correct structure [25] d) result with energy values and Zuker algorithm

a)

	A/U	C/G	G/C	U/A	G/U	U/G
A/U	-0.9	-1.8	-2.3	-1.1	-1.1	-0.8
C/G	-1.7	-2.9	-3.4	-2.3	-2.1	-1.4
G/C	-2.1	-2.0	-2.9	-1.8	-1.9	-1.2
U/A	-0.9	-1.7	-2.1	-0.9	-1.0	-0.5
G/U	-0.5	-1.2	-1.4	-0.8	-0.4	-0.2
U/G	-1.0	-1.9	-2.1	-1.1	-1.5	-0.4

b)

size	internal loop	bulge	hairpin
1	.	3.9	.
2	4.1	3.1	.
3	5.1	3.5	4.1
4	4.9	4.2	4.9
5	5.3	4.8	4.4
10	6.3	5.5	5.3
15	6.7	6.0	5.8
20	7.0	6.3	6.1
25	7.2	6.5	6.3
30	7.4	6.7	6.5

Figure 16: energy values a) for stacking (stems) b) for loops [29]

logarithm of the number of unpaired bases.
But here the results did not improve: the result for the first structure was 0.6051502145922747, for the second it was 0.5263157894736842, for the third it was 0.4 and hence the overall accuracy has been 0.5104886680219862. But after all the performances of the first and the third sequence both improved.

Thereafter I decided to introduce realistic energy values for the energy functions of a stem, a hairpin loop, a bulge loop and an internal loop, like in the original Zuker algorithm [17]. The energy values used can be seen in Figure 16 [29].
Since the values were only given stepwise I also gave the same energy value to a group of sequences. For instance, for all base pairs forming a hairpin loop with six to ten unpaired bases the assigned energy has been the same, namely 5.3. For the same reason I introduced a function based on the natural logarithm for subsequences with more than 30 unpaired bases, so that the resulting values resemble a progression.
With these values the overall accuracy improved clearly to 0.5964538940340838. For the first sequence the accuracy has been 0.575107296137339, for the second it has been 0.7017543859649122 and for the third it has been 0.5125.

Afterwards the only thing left to improve was varying the values for the parameters of the multiloop a, b and c and the parameter *omega*.
I varied the value of a between 1 and 20, the value of b between -2 and 5 and the value of c between 0 and 7, but the best overall accuracy did not change. But when I varied the value of *omega* I could achieve an overall accuracy of 0.6570417011771201, composed of 0.6866952789699571, 0.7719298245614035 and 0.5125.

In the following table all the results are shown:

	U7 spiceosomal RNA	U2 snRNA	SL2 RNA	overall
Nussinov	0.4936	0.4211	0.4125	0.4424
Zuker with default values	0.6009	0.7018	0.287	0.5300
Zuker with ln	0.6052	0.5263	0.4	0.5105
Zuker with energy values	0.5751	0.7018	0.5125	0.5965
Zuker with energy values II	0.6867	0.7719	0.5125	0.6570

Then I decided to work with the Pseudoknot prediction algorithm to achieve good results. For BChV and the energy values as declared above, I achieved an accuracy of 0.6341463414634146. And for EAV 0.6290322580645161 and for OMV 0.42857142857142855.
In the following table I have again listed the results of the pseudoknot predictions calculated with the different modes:

	BChV	EAV	OMV	overall
Zuker with default values	0.6098	0.5968	0.3333	0.5133
Zuker with ln	0.6098	0.4032	0.4286	0.4805
Zuker with energy values	0.6341	0.6290	0.4286	0.5639

In order to improve these results I thought about the idea of also taking into account the possibility that two unpaired bases m and n can form a base pair which reduces the energy. Hence it exist $(m, n) \in R$ and two base pairs $(i, j) \in R$ and $(k, l) \in R$ exist with $i < j\ , k < l\ and\ i < k$ and $i < k < j < l$ and over and above either $i < m < k < n < j < l$ or $i < k < m < j < n < l$.
To realize this idea I would have to implement again an algorithm which calculates the minimal energy of the unpaired bases forming bases between the three subsequences $S_1[i + 1 \dots k - 1]$, $S_2[k + 1 \dots j - 1]$ and $S_3[j + 1 \dots l - 1]$, whereby unpaired bases of S_1 and S_3 can only form bonds with unpaired bases of S_2 and unpaired bases of S_2 can only form a base pair with at most one other base.
The implementation would make the time and space complexity of the algorithm worse and hence I decided not to introduce this idea to the algorithm.
Instead I decided to just have a closer look at the result and wether the pseudoknot which is found by my program is at the correct place or region:
The pseudoknot of the computed structure of BChV has been in another region, but the closing parts are at the same positions.
The following two lines show the result, wherat the first line represents the computed structure and the second line represents the correct one:
. ((. . . . .)) . . . . ((((. . . . .)))) . . . . . [[ ]] .
. . . . . ((((. [[[[.)))) . . . . .]]]] .

The closing part of the pseudoknot for EAV has been in the same region, but the opening part has been 15 bases too early, as can be observed in the following:

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

But pseudoknot of the computed structure of OMV has been correct. The position of the opening base is exactly the same and the position of the closing base differs only in two bases. The result looks like this:
. (((((. . . . .))))) . [. ((((. . . . .)))) [ ] .] .
. . . . . (((. . . [[[[[[ ))) . . .]]]]]] . . .

5 Compared To Other Programs..

After all, I thought it would be interesting to see how well already existing programs can predict secondary RNA structure. Furthermore I wanted to compare my program to the other ones. For being able to compare the results of my program to the results of the other programs I used my accuracy measure for the results of the programs MFold [35] and KnotSeeker [36]. MFold is a web server developed by Michael Zuker and Turner for nucleic acid folding and hybridization prediction [35]. But it is not able to predict pseudoknots. However, KnotSeeker is a program which is able to do so. It has been developed by Jana Sperschneider and Amitava Datta in 2008 and it makes use of heuristics as well as hybrid sequence matching and free energy minimization approach to detect pseudoknots [36].
The results can be observed in the following table:

	U2 spicesomal RNA	U7 snRNA	SL2 RNA	BChV	EAV	OMV
my program	0.6867	0.7719	0.5125	0.6341	0.6290	0.4286
MFold	0.7597	0.8070	0.5625	0.5366	0.6452	0.4048
KnotSeeker				1.0	0.9677	0.4048

It should be also mentioned that KnotSeeker did not detect any pseudoknots for the second and for the third sequence but it did detect a pseudoknot for U2 spicesomal RNA.
As a conclusion, the already existing programs do get better results than mine, but their results are not absolutely correct as well.

6 Critical Evaluation

After having finished my experiments there are a few points I am going to discuss. One of the things that could be improved in the future is to add other methods to show a graphical representation of the resulting structure, since the squiggles plot becomes really small and rather unreadable. But nevertheless it is assured that no overlappings occur and if even more methods were implemented the whole program would become more complex. Which means it gets more complicated to be handled by users.
Another point to think about is to let the program automatically compute the best result of all of the possible procedures, for example with or without pseudoknot. But as a result the computation would last much longer, since all algorithms have to be computed and also it would not be possible to compare the different procedures.
It should also be mentioned that the accuracy would improve if other base pairs $(m, n) \in R$, as explained in section 4.3. , would be taken into account.
Furthermore one should keep in mind that the accuracy measure is really hard. If a base is not totally correct it is not accounted for the accuracy measure. But maybe the structural elements are only shifted by a few positions or the resulting structure has the same functional role as the correct one. But these considerations are not taken into account.
Finally, results can be improved by adding further constraints. This idea has been suggested by Zuker [17]. That this procedure works can be easily observed from the resulting structures (see Figures 13 -15) and the improvements of the accuracies as constraints are added:

- default values
- calculations with the natural logarithm
- specific energy values of the structure elements

Some constraints, like for example that hairpin loops should have at least four unpaired bases or that Watson-Crick base pairs are more stable than the Wobble base pair, have been implemented in the algorithm, but there are loads of other constraints which could be taken into consideration in future: Examples can be found at Michael Zuker's homepage: for instance a bonus for a GGG hairpin loop or the GAIL rule (grossly asymmetric interior loop) [34].
As a conclusion it can be said that a good-working program has been implemented which lets the user compare different methods of RNA secondary structure prediction. Furthermore it has been shown that it is possible to improve the performance and how such improvements can be achieved.

References

[1] Zvelebil M., Baum J.O., "Understanding Bioinformatics", Garland Science, 2008, p.3-20

[2] Steger G., "Bioinformatik- Methoden zur Vorhersage von RNA- und Proteinstrukturen", Birkhäuser Verlag, 2003

[3] Wikipedia, "Nucleotide", *http* : *//en.wikipedia.org/wiki/Nucleotide*, 27.08.2008

[4] Selzer P.M., Marhoefer R.J., Rohwer A., "Angewandte Bioinformatik- eine Einfuehrung", Springer Verlag, 2004, p.33-46

[5] National Human Genome Research Institute, "Gene expression", *http* : *//www.genome.gov/Pages/Hyperion/DIR/VIP/Glossary/Illustration/Images/gene_expres* 27.08.2008

[6] Comeaux J., "Biology",*http* : *//www.biology.lsu.edu/heydrjay/*1201/*Chapter*17/*SCI_Amino _Acid_CIRCLE.jpg*, 27.08.2008

[7] ,Rivas E., Eddy S.R., "A Dynamic Programming Algorithm for RNA Structure Prediction Including Pseudoknots", Academic Press, 1999

[8] Wikipedia, "Protein", *http* : *//en.wikipedia.org/wiki/Protein*, 28.08.2008

[9] Nishiura J., "Biology",*http* : *//academic.brooklyn.cuny.edu/biology/bio*4*fv/page/prot_struct*− 4143.*JPG*,28.08.2008

[10] Zvelebil M., Baum J.O., "Understanding Bioinformatics", Garland Science, 2008, p.461-514

[11] ESI Special Topic, "Fast Breaking Comments by Michael Zuker",*http* : *//www.esi* − *topics.com/fbp/*2004/*august*04 − *MichaelZuker.html*,16.08.2008

[12] ?, "Dynamic Programming", ?, ?, p.74-82

[13] ?, "Protein Structure Prediction", ?, ?, p.19

[14] ?, "NP-Completeness of Core Bioinformatic Problems", ?, ?, p.209-213

[15] Wikipedia, "Pseudoknot", *http* : *//en.wikipedia.org/wiki/Pseudoknot*, 28.08.2008

[16] Atallah, M.J., "Algorithms and Theory of Computation Handbook", CRC Press, 1999, p.19-26

[17] Zuker M., Stiegler P., "Optimal Computer folding of large RNA sequences using thermodynamics and auxiliary information", Nucleic Acids Research 9, 1981, p.133-148

[18] Wikipedia, "Zuker-Algorithmus", *http* : *//de.wikipedia.org/wiki/Zuker* − *Algorithmus*, 16.08.2008

[19] Arkin, R.C., "Behavior-Based Robotics", MIT Press, 2000, p.356

[20] Arkin, R.C., "Behavior-Based Robotics", MIT Press, 2000, p.331-333

[21] SVG Open 2008 , *http* : *//www.svgopen.org/*2008/, 30.08.2008

[22] Thiesing, F., "Objektorientierte Programmierung in Java", GUI 3: Swing und SWT, *http* : *//www.vorlesungen.uos.de/informatik/b*06,24.11.2007

[23] Zvelebil M., Baum J.O., "Understanding Bioinformatics", Garland Science, 2008, p.76

[24] Zvelebil M., Baum J.O., "Understanding Bioinformatics", Garland Science, 2008, p.46-48

[25] RNA database of Sanger Institute, "Rfam", *http* : *//www.sanger.ac.uk/Software/Rfam/*, 01.09.2008

[26] Rfam, "U2 spliceosomal RNA", *http* : *//www.sanger.ac.uk/cgi* − *bin/Rfam/getacc?RF*00004, 01.09.2008

[27] Rfam, "U7 small nuclear RNA", *http* : *//www.sanger.ac.uk/cgi* − *bin/Rfam/getacc?RF*00066, 01.09.2008

[28] Rfam, "SL2 RNA", *http* : *//www.sanger.ac.uk/cgi*−*bin/Rfam/getacc?RF*00199, 01.09.2008

[29] Huson D., Algorithms in Bioinformatics, "8.4.1 Loop-dependant energies", *www* − *ab.informatik.uni* − *tuebingen.de/teaching/ws*06/*albi*1/*script/rna_structure*_06*Dec*2006.*pdf*

[30] RNA database with Pseudoknots, "PseudoBase", *http* : *//wwwbio.leidenuniv.nl/ Batenburg/PKB.html*, 02.09.2008

[31] RNA database with Pseudoknots, "BChV", *http* : *//wwwbio.leidenuniv.nl/ Batenburg/PKBase/PKB*00240.*html*, 02.09.2008

[32] RNA database with Pseudoknots, "EAV", *http* : *//wwwbio.leidenuniv.nl/ Batenburg/PKBase/PKB*00127.*html*, 02.09.2008

[33] RNA database with Pseudoknots, "OMV", *http* : $//wwwbio.leidenuniv.nl/\ Batenburg/PKBase/PKB00016.html$, 02.09.2008

[34] Zuker M., "RNA free energies at 37 degrees", *http* : $//www.bioinfo.rpi.edu/zukerm/cgi-bin/efiles.cgi?T=37\#LOOP$, 16.08.2008

[35] Zuker M., Turner , "Web Server for nucleic acid folding", *http* : $//mfold.bioinfo.rpi.edu/cgi-bin/rna-form1.cgi$, 11.09.2008

[36] Sperschneider J., Datta A., "KnotSeeker: Heuristic pseudoknot detection in long RNA sequences", $http://knotseeker.csse.uwa.edu.au/$, 11.09.2008

Affirmation

I, Sophie Schneiderbauer, hereby confirm that I composed the work "RNA Secondary Structure Prediction" independently and that I did not use any other ressources or auxiliary means than the ones stated.

Berlin, September 15th, 2008

_____-