The many-state partition sum in DNA and RNA hybridization and its application to microarrays

Matthijs van Dorp

Institute for Theoretical Physics, Utrecht University, Utrecht, and
Mathematics Institute, Utrecht University, Utrecht.

Supervisors:
Prof. G.T. Barkema and Prof. R.H. Bisseling

Revision: July 27, 2009

Contents

Abstract

The assumption that DNA and RNA hybridization can be described by a two-state model, is unjustified for analysis of microarrays. It is expected that microarray analysis would benefit from a model which takes into account many different possible configurations.

In this thesis, we present the tools developed to allow a many-state analysis of microarrays, and we also report many results obtained in various areas. We shall develop an algorithm to calculate many-state partition sums for DNA and RNA using an extended Nearest-neighbor model. Different incarnations of the algorithm are constructed, each of which computes the same final result, but in a separate way. Eventually, five different algorithms will be tested for their performance.

The many-state model is then used to refit the literature parameters for the nearest-neighbor model. An improvement with respect to currently known values is found, even though the many-state model is not expected to be very effective at the short lengths of DNA and RNA strands that are used in the experiments reported in the literature. Two different sets of parameters are fitted and compared, and it is found that only a small subset of possible states is still present in the many-state partition sum, such that our partition sum is similar to the two-state partition sum.

Finally, the many-state model is used in analysis on microarrays. In spite of problems with overfitting, a significant improvement is seen with respect to the two-state model predictions. Parameters discovered for hairpins are similar to the parameters already found in the literature. Additionally, a parameter set is derived for microarray analysis. While the limited availability of data hampers our ability to extract physically relevant parameters, the overall results indeed confirm that the many-state partition sum performs better at microarray analysis than the two-state partition sum. This leads to the conclusion that the many-state partition sum may be expected to improve DNA and RNA hybridization prediction in general.

Documentation of the programs used to obtain the results in this thesis, is available as an appendix to this thesis. The source code of the programs, available under the GNU LGPL license, can be obtained from the author, or alternatively, online from http://www.math.uu.nl/people/bisseling/software.html .

1 Introduction

1.1 Introduction to DNA and RNA

Since the discovery of the DNA structure by Watson and Crick in 1953, a large number of applications have been developed based on our knowledge of DNA. DNA profiles assist in the identification of humans, DNA engineering enables researchers to do extensive research on pathogens using mice instead of humans, and many hereditary diseases have been discovered, giving scientists a lead for developing a cure.

All life that we know, relies on DNA and/or RNA for its existence. As a life form reproduces, it passes on some of its genetic code, which defines its offspring. Parts of DNA may carry information on the physical characteristics of a life form. As the amount of DNA is large, varying combinations of genetic code cause differences between members of the same species. Other parts may carry more specific information, which is the same for (almost) all members of a species. For example, proteins are encoded for in DNA. Humans have thousands of vital proteins, each of which has its specific tasks and abilities. In our cells, the proteins are the primary workforce, a diverse and varied population of specialized molecular machinery.

1.1.1 Proteins

Not all parts of human DNA are relevant at all stages of life. For example, organ transplantation introduces an organ into a body that uses quite different DNA, and that is only approximately equal to the organ that originate from the host's genetic information. Meanwhile, the DNA code for creating such organs lies dormant in the cells of the host. More generally, while there are pieces of the DNA that are frequently used by body cells, these compose only a very small fraction of all DNA.

DNA cannot do much by itself, being only a long thread of nucleotides[1], lying curled up in the nucleus of a cell like a beaded string. Still it is of vital importance to many processes in the cell. To be more precise, the cell actually uses the DNA to produce *proteins*. Many functions in the cell are aided by proteins, three-dimensional structures composed of thousands of *amino acids*, small chemical complexes of which about 20 different types exist. A cell has factories to assemble proteins, which are called *ribosomes*. Some are fixed on a membrane known as the *endoplasmic reticulum* (ER), located next to the nucleus of a cell. *Free ribosomes* also exist, which float freely in the cytoplasm (cell fluid).

Ribosomes translate a strand of mRNA (messenger RNA) into a string of amino acids. Every triplet of nucleotides in the DNA corresponds, via the mRNA, to a certain amino acid. Some are redundant, such that several different triplets cause the same amino acid to be added to the protein under construction. The process of creating proteins using an RNA template is known as *translation*. The string of amino acids is then folded into a three-dimensional structure. This may happen by several mechanisms, including automatic folding immediately following translation. Interestingly, the most powerful distributed computing

[1]Nucleotides are small chemical substances that make up DNA. Their purpose and characteristics will be discussed later on.

cluster in the world at this time is *Folding@home*, which performs calculations on protein folding. In this thesis, however, the focus is solely on a preceding section of the protein assembly line - the messenger RNA, which is transcribed from the DNA and eventually translated to help creating a protein. Therefore, we will start with an introduction to DNA and RNA structure and mechanics.

Figure 1: The double helix structure of DNA.

1.1.2 DNA structure

In this thesis, we try to develop methods for the analysis of microarrays *(to be defined later)*, so first of all let us recall the basic properties of DNA. DNA is an interconnected double helix (a ladder), two strings both circling around a common central axis in nearly perfect antiphase (see Figure 1). The strings consist of sugars and phosphates, and the crossbars of the ladder are formed by hydrogen bonds between small complexes that are attached to the string. This will be investigated in greater detail in the remainder of this section.

Specifically, DNA can be regarded as a string-shaped backbone of phosphate-deoxyribose[2]. The backbone is adorned with *nucleotides*, also called *bases*. A *base* is a small assembly of atoms (carbon, nitrogen, oxygen and hydrogen) that is connected to the sugar part of the backbone, *deoxyribose*. DNA has four distinct bases known as adenine (A), cytosine (C), guanine (G) and thymine (T). DNA consists, in general, of two *complementary* strings, where 'complementary' means that an A base on one string is paired by hydrogen bonds to a T base on the other string; similarly, C and G are complementary. Such strings are more commonly called *strands*.

DNA usually exists only in its helical form, where two fully complementary strands intertwine in a mutual embrace, such that the whole complex is in its preferred state, and all hydrogen bonds are found. When two bases are connected by hydrogen bonds, the pair of bases forming the hydrogen bonds is called a *base pair*. These base pairs form the crossbars of the DNA helical-ladder structure, and are responsible for keeping the two DNA strands together and in their helical configuration.

The process of two DNA strands forming a double helix by establishing hydrogen bonds between nucleotides, thus creating base pairs, is commonly called *hybridization*. The structure of two strands hybridized to each other is called a *hybrid*. The DNA in a cell's nucleus consists of a few long strings comprised of millions of base pairs each, and may be considered a single, very long, double helix. Short pieces of DNA, however, may form a wide variety of structures. When two strings of DNA are only partially complementary, it is still possible for the two strands to form a number of base pairs. However, the pieces of DNA that are not part of a base pair do not form a helix, and such sections behave more like flexible polymers. To facilitate discussions of such cases of hybridization, the notion of *secondary structure* is used.

Secondary structure is the way how DNA twists and bends, but in a rather abstract sense: the true three-dimensional form of the DNA is generally called *tertiary structure*, and the latter is, in general, much harder to determine accurately. The difference is that secondary structure is determined solely by which nucleotides have engaged in base pair formation, establishing hydrogen bonds. Secondary structure does not explicitly involve the spatial organization of DNA, and in this it differs from tertiary structure, which does take into account the precise way of twisting and bending of DNA.

It has been remarked that in its most well-known form, two DNA strands form a double helix. This double helix structure is asymmetric in the sense that although the strands both follow an imaginary cylindrical shape, there is a *minor groove* and a *major groove* separating the two strands, see Figure 2. Essentially, the strands are not in perfect antiphase, an aspect which is also shown in Figure 1. Consequently, when looking at a double helix cylinder from the side, the strands appear as sine-like waves of equal periods, but with a phase difference, which is not equal to π. Additionally, the chemical composition of the backbone is orientation dependent, and DNA is composed of one strand with one direction and a second strand with opposite direction. Thus, the double helix, contrary to intuition, is not rotationally symmetric for any rotation ϕ

[2]Deoxyribose is a type of sugar. The backbone of DNA is composed of alternatingly a phosphate and a sugar residue, hence the name DNA, which stands for *deoxyribonucleic acid*. Similarly, RNA stands for *ribonucleic acid* which, when compared to DNA, has one extra oxygen molecule in each segment.

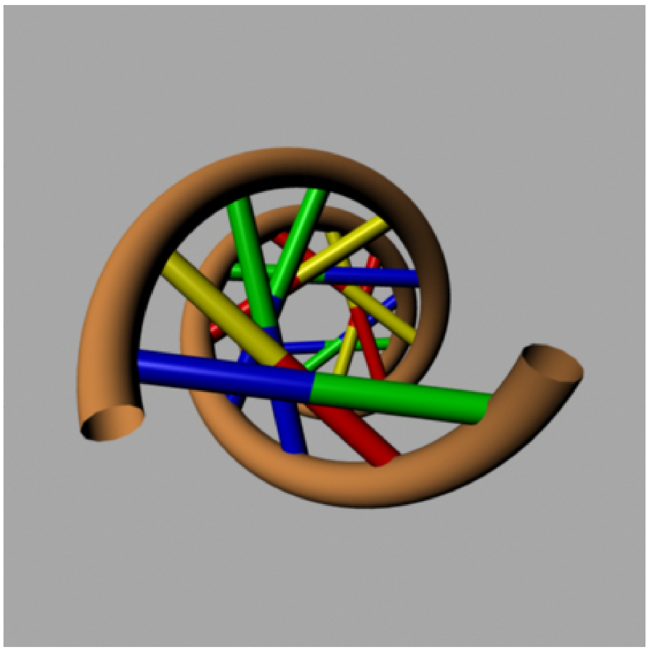
Figure 2: The DNA helix seen from above, with emphasis on the axial asymmetry.

around the axis of the cylinder (except for the trivial case where ϕ is a multiple of 2π).[3]

The only symmetry of a DNA helix, therefore, is of a translational type. Note that this argument also implies that a single strand floating freely still has a front end and a back end, and when we consider a certain nucleotide at some position, we may consider an 'upstream' and a 'downstream' direction, which are by no means interchangeable. The directional dependence thus cuts down the number of possible secondary structures, and is used implicitly in much of the following. Before continuing, we will give a more formal definition and some conventional terms used to denote directionality.

1.1.3 Directionality

As has been mentioned before, the DNA double helix is direction dependent. A strand, therefore, has a direction, and the ends are conventionally denoted 3' and 5' (*three prime* and *five prime*), respectively, due to naming conventions of the sugar ring in the DNA backbone. Consequently, a DNA segment consisting of a backbone with attached to it the bases A and C may be written 5'-A-C-3' - the 5'-3' convention arises because the only way to synthesize DNA is in the

[3]That is to say, the separate strands exchange position. This causes a directional dependence for the backbone, and a 3' and a 5' end. The fact that the strands are not interchangeable, also results in a difference in energy costs for certain sequences, such that if one strand has nucleotides AG hybridizing to nucleotides TC on the other, the energy cost of this hybridization is not equal to that of nucleotides TC on the first strand hybridizing to AG on the second. More on the asymmetry of hybridized DNA will follow later.

5' to 3' direction (one can only extend DNA by appending nucleotides to the 3' end). The complementary strand is 5'-G-T-3', as obviously, the directions of the two strands making up the double helix are mutually opposite. Thus, the above described DNA looks like AC/TG where AC is read in the 5'-3' direction, and TG, the complementary part, is read in the 3'-5' direction. Note that this is essentially different from TG/AC, which corresponds to 5'-T-G-3' combining with 5'-C-A-3', which is the exact mirror image of the previously described structure, and may be briefly denoted as CA/GT. Thus, the directionality, as caused by the asymmetry in the double helix, results in a set of 10 different doublets (sets of two neighboring base pairs, e.g. AC/TG is one of the ten doublets) that are possible in DNA/DNA hybridization.[4]

1.1.4 RNA

Similar to DNA, though not identical, is RNA. While DNA does not usually occur in short strands, RNA is usually relatively short (seldom longer than a few thousand nucleotides) and serves various purposes. Among others, it regulates protein production in the cell. It differs from DNA in several ways. First, while DNA is in almost all cases found as a double-stranded molecule in its well-known helical form, RNA generally occurs single-stranded. As such, it is not a very stable molecule and may be fragmented more easily than ordinary DNA. Secondly, the backbone of RNA is different from DNA. The difference is the presence of a hydroxyl group (OH) at the sugar ring in the RNA backbone, which in the case of DNA is replaced by H (so the oxygen atom is missing). Finally, the base complementary to adenine (A) in DNA is thymine (T), but in RNA thymine does not appear, but is replaced with *uracil* (U). Although the names 'uracil' and 'thymine' do not suggest any similarity, thymine is essentially methylated uracil — uracil, with respect to thymine, is missing a methyl group (replacing CH_3 with just H).

Although RNA is chemically different from DNA in a few aspects, it shares many properties with DNA, and RNA and DNA strands may hybridize to each other.

1.1.5 Protein assembly

One of many uses of RNA is to code for proteins. As stated before, this type of RNA is known as messenger RNA, often abbreviated to mRNA.[5] Messenger RNA is created by a process called *transcription*. When several tags (*markers*) are put on the DNA, it is possible for a copying enzyme, called *RNA polymerase*, to transcribe the DNA into messenger RNA. The messenger RNA then is released into the cell, and carries the copied information to the ribosome.

The information is split in triplets. Each triplet of nucleotides in RNA (such a triplet is called a *codon*) has a special meaning. For example, AUG is a 'start codon' while UAG is a 'stop codon'. These tell the ribosomes where to start

[4]Note that there is still some kind of symmetry, namely, AC/TG is equal to GT/CA, which both correspond to 5'-A-C-3' hybridizing to 5'-G-T-3'. Thus there are not 16 different parameters, but only 10 - the number of independent entries in a symmetric matrix.

[5]The prefix *m* indicates a different function rather than a different physical structure. The object under consideration is RNA, as is tRNA (transfer RNA), rRNA (ribosomal RNA), et cetera. Therefore, throughout this document, the more general word 'RNA' will often be used to refer to mRNA.

and where to stop translating the mRNA into protein. Every triplet between the start and stop codons codes for amino acids,[6] and the ribosome assembles a protein using amino acids, since every amino acid is associated to one or several codons.[7] The amino acids are transported to the ribosomes by other RNA strands, called *tRNA* (transfer RNA).[8] These tRNAs are folded RNA sequences with the amino acid attached to them, that bind (partially) to the mRNA and transfer the carried amino acid to the ribosome, which attaches it to the protein being assembled. At this time, the protein is only a string of amino acids, just as the mRNA is a one-dimensional object. However, self-organization in the string of amino acids causes this polymer-like molecule to fold in a special way. A protein has to be folded in a certain way in order to be able to carry out the task it is designated to perform.

1.1.6 Protein abundance

An organism controls the amount of protein by regulating mRNA concentrations, and in turn, mRNA concentrations are controlled by markers put on the cell DNA. These markers allow (or prevent) gene activity. *Gene expression*, the extent to which a gene is active, is regulated by many factors. Generally speaking, when a cell is in need of some protein, it changes some of the markers on the RNA. Then, the enzyme RNA polymerase copies a part of the DNA of that cell, creating mRNA, setting off the protein-creation chain. This RNA is then used by the ribosomes to produce protein, as discussed in the previous section. RNA strands are continuously being decomposed, such that they are only present in significant amounts when they are being produced by the cell. Thus, the presence of proteins in a cell is related to the presence of mRNA in the cell, and moreover, the presence of mRNA in the cell is an indication of the proteins the cell is currently trying to have created.

Consequently, when we are interested in cellular processes such as protein abundance, the amount of mRNA present gives an indication of the current level and focus of protein production. Many studies and medical therapies require a good estimate of protein abundance for better understanding of the current situation and status of the cell. Thus, since mRNA concentrations are thought to be a very good indicator of protein creation intensity, experiments are done to determine these mRNA concentrations. As there are thousands of proteins, probing for only a single type of mRNA is not an efficient way to give an overview of the various ongoing processes in the cell. Recently, *microarrays* were developed to help finding RNA concentrations. These arrays simultaneously

[6]In fact, this is a simplification, because in fact not all RNA that is copied by the RNA polymerase, is indeed used for creating proteins. Instead, the initial form of newly copied RNA is called pre-mRNA and contains non-coding sequences called *introns* that are not part of the protein code. These are removed by a process called *splicing*, resulting in the final mRNA which consists solely of nucleotides coding for amino acids. It is this final ('mature') mRNA strand that is ultimately used by the ribosomes to create a protein. It is thought that introns are not entirely without a purpose, but discussing this is far outside the scope of this writing, and for our discussion they are irrelevant.

[7]There are 20 different amino acids, and there is a start and a stop codon. Thus, there are 22 meanings to be assigned, while there are $4^3 = 64$ possible configurations of a codon. Consequently, some amino acids have multiple representations in terms of RNA nucleotides. Mathematicians would say that the mapping of the set of RNA triplets into the set of amino acids and start/stop codons is not injective.

[8]This shows how RNA serves different purposes in a cell.

measure the concentration of many thousands of different mRNA sequences.

1.2 Microarrays

Many diseases cause a disruption in gene expression levels, causing shortages or surpluses of certain proteins. These diseases may be diagnosed by measuring mRNA levels in the cell, and comparing measured concentrations with standard concentrations. Measuring these RNA concentrations is not an easy task, but recent improvements in biotechnology have enabled commercial production of *microarrays*, a microarray being millions of short fragments of single-stranded DNA deposited on a chip. This section is devoted to giving an introduction to microarrays.

1.2.1 Description of a microarray

A microarray typically consists of some substrate (made of glass, plastic, or silicon quartz) on which specific (single stranded) DNA fragments are deposited. Different techniques are used to achieve this. The earliest ways of creating such a biochip, was to dip into a solution of all-identical DNA fragments and deposit a droplet of the solution on the substrate. The resulting dot of DNA fragments would be an attractive spot for complementary RNA to bind, so if one would find a way to measure the amount of RNA on a certain dot, one would have a good estimation of the amount of RNA available in some sample that is washed over the dot. Placing several of these dots on a substrate, the biochip known as microarray was developed. Every dot would correspond to a certain piece of RNA, and using techniques to make RNA fluorescent, illumination of the chip with a laser would reveal the concentrations of RNA, measurable as light intensities. Several complications arise, but this is the general principle behind microarrays.

1.2.2 Incarnations

There are several forms of microarrays. Besides the 'dot' form described above, there is another notable technique of measuring RNA concentrations, namely to build short fragments of single-stranded DNA (commonly referred to as *oligonucleotides*) directly on a substrate by means of photolithography. The fragments are usually shorter, but the fragments are a lot more accessible because they are neatly arranged, each on its individual position, rather than an orderless heap of DNA fragments as with the dot method. Several hundreds of identical DNA strands are synthesized in a small area, called a *feature*, whose purpose is to detect (i.e. measure the concentration of) a certain RNA sequence. In this report, we are mainly concerned with the second kind of microarray, with oligonucleotides constructed on the substrate by means of photolithography. Because of the well-defined properties of this system, it is most suitable for developing physical-based theories for estimating the relationship between measured mRNA intensity and actual mRNA concentration.

Several additional modifications are in use to further reduce noise and unwanted effects. For instance, the DNA strands are actually not attached directly to the surface, but instead connected to the surface by a small 'rope', to reduce repulsive effects caused by the substrate. After all, thermal motion of RNA will cause it to be pushed away from the substrate, which restricts its movement, which in turn negatively affects its entropy.

1.2.3 Measurements using microarrays

It has been mentioned before that the main purpose of microarrays is, to measure gene expression levels by means of detecting RNA concentrations, which are closely related to protein concentrations. To be able to detect RNA that has hybridized with the complementary DNA on our chip, we must have some way of ‘seeing’ it. To make RNA visible, it is treated before being washed over our microarray, slightly modifying the U nucleotides present in the RNA. Then, the RNA in solution is applied to the microarray, and many of the RNA fragments (referred to as *targets*) present in the solution will stick to *probes* of DNA on the chip, forming a double-stranded DNA-RNA complex. The remaining RNA is washed off the chip, and fluorescent markers are attached to the modified U nucleotides that are present in the RNA that is part of the double-stranded complex (and thus was not washed away). Finally, a laser beam shines on the different features resulting in an intensity pattern reflecting the number of fluorescent markers present on a certain position. From this result, an estimated concentration of RNA may be derived.

1.2.4 Errors in experiments

There are numerous reasons why the actual use of microarrays is far more complicated than the above description. Here is a list of some known effects:

- The RNA in solution is actually fragmentized first, and various fragments of all sizes are floating around in the solution, causing unpredictable cross-hybridization.
- RNA targets have a significant tendency to bind in the wrong place, causing erroneous measurements of RNA concentrations.
- RNA targets, which are single stranded, may stick to themselves, reducing the amount of RNA available for probe-target hybridization.
- Furthermore, RNA targets might stick to other (accidentally complementary) RNA targets, mutually reducing the apparent concentration.
- Some RNA sequences are richer in uracil (U) nucleotides than others, so they will appear relatively brighter.
- The number of probes is limited, and saturation might occur for very high concentrations.
- The microarray synthesis is complicated, and the photolithography used is slightly unreliable. Only about 10 % of all probes reach length 25 without any errors.

Many of these problems have been circumvented to some extent. For example, the concentration of RNA in the solution is low enough to prevent saturation of probes almost everywhere. The uracil content of RNA sequences is known, so compensation for different brightnesses is possible.

This leaves two harder problems to solve: to compensate for RNA-RNA hybridization, and to solve for RNA incorrectly attached to a DNA probe that was intended for a different RNA strand. The latter has been ‘solved’ by introducing the Perfect Match - Mismatch (PM/MM) concept. Every strand of 25 nucleotides perfectly matching a section of RNA, is neighboured by an almost identical strand, consisting of exactly the same sequence as the perfect match except for position 13 (the middle) where the complementary nucleotide is chosen instead. The reasoning behind this idea is that if something undesirable sticks to a PM, it might stick to the MM as well, while the actual target complementary to the PM will stick to the PM and a lot less to the MM. So, subtracting the MM intensity from the PM intensity, we should find the actual RNA concentration. This technique works for most probes, but there seem to be some subtleties involved yet that remain unsolved, as this does not always seem to work as intended — it is not unusual that MM intensities exceed the corresponding PM intensity.

Finally, target-target hybridization remains a difficult effect to estimate. While self-hybridization (an RNA string sticking to parts of itself) may be estimated, there is little hope for a way to estimate the amount of RNA in solution that is actually sticking in part to other RNA. As this amount will usually be roughly the same fraction for a specific sequence of RNA, repeated experiments with known RNA concentrations might yield information which helps to estimate the error caused in this way, and allow us to correct for this effect.

2 Thermodynamics of RNA and DNA hybridization

2.1 Existing algorithms for hybridization prediction

We shall now leave the subject of microarrays for a while, and instead concentrate on a better understanding of the concept of hybridization. In the following sections, we will work towards an algorithm for computing hybridization free energy. In the past, various algorithms have been devised, for various purposes. Before we discuss an algorithm for the problem stated in the preceding sections, let us give an overview of some known algorithms.

2.1.1 Early calculations on Nucleic Acid hybridization

RNA plays many roles in the cell. One of these examples is transfer RNA, which aids protein assembly by providing amino acids to the ribosome. Transfer RNA is almost completely self-hybridized. And consequently, it is in a more or less constant state - its free energy minimum is very low with respect to its no-hybridization energy, and it is extremely unlikely for such a structure to fall apart. Transfer RNA is made up of around 100 bases. This is quite a bit longer than the Affymetrix DNA probe strands, which are only of length 25, or the RNA targets in Affymetrix experiments, which are 50 nucleotides long on average.

These numbers, however, are nothing when compared to what some RNA scientists must cope with. RNA viruses, for instance, are frequently found to have a length of a few thousand. There are a vast number of different RNA viruses, and understanding their secondary structure is crucial for understanding the way these viruses work — just like tRNA would not be able to perform its duties if it had not folded into the right configuration, a virus depends on its folding for its survival.

Sequences this long can hybridize to themselves in many different ways. Moreover, since the virus is able to survive because of its structure, it is likely that a certain dominant form is present, that is, all viruses of the same type are expected to have the same three-dimensional shape. This dominant form is then expected to contain many more base pairs than any other, significantly different, configuration. Knowing the RNA sequence defining such a virus, the remaining challenge is to reconstruct the folding manner of the RNA strand.

Problems like these prompted scientists to develop computer routines which help predict RNA folding. They were not particularly interested in partition sums or different configurations, as we are, but only in the dominant configuration, the configuration in which a strand of RNA is usually found in nature. Even so, the problem was hard enough. The first attempts were made at the end of the 1970s. Only in the 1980s, when computing power improved and programming became more suited to scientific uses, programs for RNA folding became popular. See for instance, Ref. [15], which notes that while computer programs so far had not been very successful in determining secondary RNA structure correctly, advancements in both programming and computer technology were promising. This article gives an interesting overview of the challenges of writing algorithms in the early days of computers, when sequences of over a thousand nucleotides were troublesome because the order-$N \times N$ matrices could not be stored in the virtual memory.

As the years went by, algorithms improved, and so did processors and memory capacities. By 1989, in an article by Jaeger, Turner and Zuker [16], success rates had improved greatly. Most of the dominant secondary structures could then be found by computer programs. This algorithm also uses knowledge about RNA thermodynamics, such as the sequence dependent nearest-neighbor effect, which is described in greater detail in the upcoming section 2.2.

In the preceding, the primary challenge is to find *the* optimal folding. It takes a lot of effort to find it, but one needs to do the calculation only once. Though intimately related to the folding in our Affymetrix system, there are a few important differences. Our strands are much shorter, but there are scores of different short strands, and the thermodynamic behavior varies between strands. It is to be expected that also *suboptimal configurations*[9] contribute significantly to the associated thermodynamic behavior. After all, while tRNA has probably evolved to be in a certain state with very high probability, the function of mRNA does not lie in the strand folding into whatever configuration, and often, there are several configurations that are close to the energy minimum. Thus, as mentioned, it is not sufficient to consider only the state corresponding to minimal energy, and we should include suboptimal configurations in our calculations.

On the other hand, algorithms developed to calculate thermodynamical characteristics of DNA/RNA hybridization by means of calculating a partition sum, are often similar to the RNA folding problem — both areas of research are computing possible foldings and their energies, albeit on strands of different lengths. This similarity property may be seen in the remainder of this chapter, as we discuss DNA/RNA hybridization algorithms -with folding- in more detail.

2.2 The Nearest-Neighbor model

2.2.1 Introduction to Nearest-Neighbor thermodynamics

Now that we have introduced the necessary biological terms, conventions and definitions, we will advance towards a physics-based model of RNA/DNA hybridization. Therefore, we shall consider the probe-target DNA-RNA hybridization in more detail. We would like to have an accurate theory on the thermodynamics of this binding. Two effects influence the tendency for a strand of RNA to bind to a strand of DNA. The first is an entropic effect: rather than being tied to a spot, RNA would be free-floating. Thus, a penalty must be paid to let the RNA attach to the DNA. This penalty increases the free energy. However, if the RNA is strongly sticking to the DNA, by means of hydrogen bonding, this state is energetically favorable, and might therefore nevertheless be the preferred one. This is determined by the *effective binding energy* or effective hybridization free energy of the RNA strand to the DNA probe. Naturally, the RNA concentration also influences the probability to find RNA hybridized to DNA. Since quite a lot of mRNA strands are floating around, we will describe hybridization probabilities for the DNA probes. We may write the *partition*

[9]The use of the word *suboptimal* reflects that a certain state is not the most probable state. It turns out that frequently, a significant portion of mRNA is not folded in the most probable way. This can be understood by computing the *partition sum*, which may or may not be dominated by a single state. More discussion on partition sums will follow in due time.

sum for a single DNA probe as follows,

$$Z = 1 + c_{\mathrm{RNA}} e^{-\beta \Delta G}, \tag{1}$$

where we introduced the mRNA concentration c_{RNA}, the inverse temperature $\beta = 1/RT$,[10] and the effective binding energy ΔG. Note that the second term is the Boltzmann weight for the hybridized state. The partition sum serves as a normalization when we wish to calculate the fraction of the DNA probes that will find an RNA partner,

$$p_{\mathrm{hybr}} = \frac{c_{\mathrm{RNA}} e^{-\beta \Delta G}}{1 + c_{\mathrm{RNA}} e^{-\beta \Delta G}}, \tag{2}$$

and this result immediately shows why this is expected to be a good approach to the microarray analysis. We have just established a direct relationship between the concentration of RNA and the probability for DNA to have RNA attached to it! Consequently, if we just put a certain number of DNA probes on a microarray, and manage to somehow find out the fraction of DNA probes with RNA on them, and the effective binding energy, we could find the desired concentration c_{RNA}. In a nutshell, this summarizes why we are interested in finding a physics-based theory to be used in microarray analysis.

The amount of RNA on a microarray can be measured experimentally, within reasonable error margins. There are, however, a number of problems that complicate things. As discussed before, DNA probes might attract RNA that is partially complementary. Furthermore, RNA in solution ('free' RNA) might hybridize to other free RNA. These are effects we will not discuss here. What we will focus on, is the following problem: What is this 'effective binding energy', and how can we estimate it accurately? Is it possible to do better than the methods currently in use?

We will see that we can incorporate self-hybridization (RNA sticking to itself and/or DNA sticking to itself), and also count thermodynamic states that do not occupy the energetically most favourable state.[11] This will be done by using a model known as the *Nearest-Neighbor model*. In fact, we are not really extending the nearest-neighbor model itself, but rather we are improving the way it is currently used to predict RNA/DNA hybridization probabilities. Namely, to compute the partition sum more precisely.

2.2.2 Explanation of the Nearest-Neighbor model

The first thing to do is to compute the energy gain when RNA and DNA hybridize. Naively, it is reasonable to assume that we can write this hybridization free energy as a sum over the nucleotides the strand is composed of. After all, C-G and A-T (or A-U) connections may exist, but that is all that can influence the free energy. Perhaps we can find two or three parameters that describe the

[10] This differs from the most commonly used definition of β, $1/k_B T$, by Avogadro's constant. This choice was made because ΔG has units kcal K^{-1} mol^{-1}, and obviously, the quantity in the exponent must be dimensionless. Thus, Avogadro's constant was merged into the definition of β for ease of notation.

[11] There are many different 'bound states' possible, since RNA might stick only partially to a DNA probe in many different ways. This is an entropic effect which alters the effective binding energy.

strength of the connection formed by the hydrogen bonds involved, and thus derive the total hybridization free energy of the DNA/RNA connection.

This idea is basically correct, but it omits one important effect. Because of the helical form of the RNA-DNA duplex, nucleotides feel not only their partner on the other strand, but also their adjacent neighbor on the same strand. This contribution turns out to be essential for an accurate description of duplex formation. Thus, we will not consider just single bonds, but pairs of bonds. The pair of pairs will be called a *base pair doublet* from now on — a doublet which consists of two neighboring nucleotides, which have formed base pairs with another two neighboring nucleotides complementary to the first two nucleotides. As each base pair doublet has its own free energy parameter, we end up with 16 possible nearest-neighbor (NN) pairs.[12] This means that every base pair is used twice, except for the first and the last base pair, which are used only once. The resulting correction factors may be absorbed into the helix initiation parameter, ΔG_{init}.[13] Note that we needed such an initiation parameter, anyways, as a certain (entropic) barrier must be overcome to bring DNA and RNA together and start hybridization. Now considering a duplex where DNA and RNA are complementary and have all their nucleotides connected to each other, it is straightforward to compute the free energy differences between this fully hybridized state and the free state, namely,

$$\Delta G_{\text{NN}} = \Delta G_{\text{init}} + \sum_i \Delta G_{l_i,l_{i+1}}, \tag{3}$$

where $l_i \in \{\text{A}, \text{C}, \text{G}, \text{T}\}$ is the nucleotide at position i, and $\Delta G_{l_i,l_{i+1}}$ is the *stacking free energy* parameter for the pair of nucleotides at $i, i+1$. For example, suppose that $l_i = \text{A}$ and $l_{i+1} = \text{C}$, then ΔG_{AC} is the free energy parameter for the DNA A-C doublet connecting to a RNA G-U doublet. As remarked before,[14] DNA has directional dependence, and thus this parameter is not the same as the C-A parameter.

The ΔG parameter computed in eq. (3), by means of a sum over the nearest-neighbor pairs of the fully hybridized state, may be used in equations (1) and (2). This *two-state hybridization approximation* for the partition sum is the approximation currently used in most computations on nucleic acid hybridization.

However, the nearest-neighbor model is also suited for a more advanced calculation of the partition sum — many of the omitted states may still be included using the nearest-neighbor model. The only reason not to do so, is to keep calculations simple and reduce computation times. It is straightforward, however, to estimate the hybridization free energy for such states using the nearest-neighbor model. In these cases, not all nucleotides on a strand have engaged in a base pair, and the sum in eq. (3) runs only over a subset of all

[12]There are no symmetries in RNA/DNA hybridization, because DNA and RNA backbones are different and the only symmetry in DNA/DNA hybridization as pointed out in 1.1.3 is spoiled. Therefore, we really have 16 different parameters. It is thus NOT true that, for example, AG = GA, or AG = TC or AG = CT, for RNA/DNA duplexes — again, RNA and DNA are chemically different, and the difference goes beyond the difference between thymine (T) and uracil (U).

[13]It is also possible to introduce an additional parameter into the model to distinguish C·G and A·T initiation, but we will not go into further detail yet.

[14]See section 1.1.3.

possible pairs,

$$\Delta G_{\mathrm{NN}} = \Delta G_{\mathrm{init}} + \sum_{\mathrm{pairs}} \Delta G_{l_i,l_{i+1}}. \tag{4}$$

By means of this simple generalization, more states can be incorporated into the partition sum, leading to a many-state partition sum. In practice, however, we have to add a few parameters to the nearest-neighbor model to compensate for effects that show up in this type of hybridization. They will be introduced in section 2.3.2.

In calculations on RNA-RNA hybridization, it was found that duplexes with terminal AU pairs were less stable than those with terminal CG pairs. In some cases, it might therefore be beneficial to distinguish between different initiation energies, and assign a penalty to a duplex for every terminal AU pair it has. The difference arises because C·G base pairs have formed three hydrogen bonds, while A·U (and A·T) pairs have formed only two hydrogen bonds. Consequently, A·U pairs are weaker and a penalty may be introduced to anticipate a lower ΔG for strands with such ends. This model is sometimes referred to as the INN-HB model, an *Improved Nearest-Neighbor* model which takes into account *Hydrogen Bond* effects. For example, Ref. [27] uses a parameter for terminal A·U pairs in their computation of nearest-neighbor parameters for RNA/RNA hybridization.

2.2.3 Limitations to the two-state approximation for the partition sum

We have mentioned before that the two-state approximation may be insufficient for certain fields of research, because its range of validity is limited. In this section, we shall argue why it is desirable to improve this two-state model.

Analysis based on the previous nearest-neighbor model, or incarnations of the model with a slightly different choice (such as the INN-HB model) is seen in various fields of biochemistry. As the approach works best for shorter sequences, mainly short sequences have been used in the derivation of the parameters (Xia, SantaLucia, Sugimoto). However, long strands of *single-stranded*[15] RNA are known to twist and fold, and they may hybridize to themselves. Such effects are negligible when considering short sequences of up to 10 or so base pairs. For longer sequences, however, hairpin formation contributes to the effective binding energy, as RNA folding to itself is effectively removed from the 'pool' of available RNA - only a fraction of RNA is not hybridized, and able to hybridize with DNA. Similarly, DNA might self-hybridize. For microarrays, where DNA strands have length 25 and RNA strands have an average length of 50, these effects likely have a major impact on the actual amount of available RNA, and thus on the estimated effective hybridization free energy.

This is a deviation from the original NN model, which assumes only two states (DNA and RNA are either free, or hybridized). And there is yet another category of states that remain unaccounted for: the states where DNA and RNA are only partially hybridized. Partial hybridization means that not all of the DNA and RNA are connected via hydrogen bonds, and some pieces are not connected. This is a different state than the two mentioned before, but such a state obviously is not preferable at first. Thermodynamical theory dictates

[15]When DNA or RNA is in a double-helix structure, it is called *double-stranded*. When this is not the case, it is considered *single-stranded*.

however, that every possible state may be attained. Even if the weight of such a state in the partition sum is much smaller, if enough such states are possible, all these states together they will alter the partition sum significantly, and the probability for hybridization may be strongly affected.[16]

In other words, it is dangerous to neglect all the partially hybridized states, as is done in the two-state model, even though each state is individually less favorable than the fully hybridized state. We have likely underestimated the part of the partition sum which corresponds to the hybridized state.

Moreover, the two-state model does not allow any self-hybridization, or partial hybridization, and it thus assumes that single-stranded DNA or RNA has only the trivial partition sum $Z = 1$ - there is no state except for the free one. However, it is well-known that DNA and RNA strands may form hairpins, and parameters from hairpins computed from experimental data by Antao and Tinoco [3] show that some hairpins are quite stable. Penalties for hairpin formations are dependent on the nucleotides forming the hairpin loop, with some penalties reported as low as $\Delta G_{\mathrm{hp}} = 1.0$ kcal/mol. This shows that even relatively short sequences are susceptible to hairpin formation. The assumption $Z = 1$ is poor at the very least, and may be expected to introduce a large error in the estimation of the hybridization probability.

2.3 The extended nearest-neighbor model

2.3.1 Extending the two-state approximation for the partition sum

These limitations to the standard NN model call for an improvement. Preferably, we would include all possible configurations that our system can attain. Unfortunately, even by a simple calculation, we may find that the number of configurations is huge beyond measure.

> ***Estimation of the number of configurations***
> Let us derive a simple lower bound for the number of possible configurations. Consider a strand of length $N = 200$. Since for the sake of this counting argument, it is not relevant to consider a complicated sequence, consider a sequence of the form ATATAT … ATATAT. We throw away the bulk of the states by limiting ourselves to the case where any of the 100 AT doublets binds to another AT doublet somewhere on the strand (note that this indeed limits our options severely). For now, let us assume infinite flexibility and stretchability of the DNA. Restricting ourselves to the case where everything is hybridized, we can already find up to $99 \cdot 97 \cdot ... \cdot 1 > 2^{49} \cdot 49!$ distinct ways in which we can pairwise combine the AT doublets. Thus, even for a simple argument using only a small subcategory of states, we see that the number of possible states by its very nature scales as the factorial of the length N or variants thereof. For the similar problem of matrix-chain multiplication, as discussed in Section 3.1.1, it is fairly easy to see that naive approaches require $N!$ operations to be performed. Of course, DNA is not infinitely

[16]The penalty for breaking a base pair does not depend on the length of the strand, but the number of configurations with only one base pair missing, is linearly proportional to the strand length. It follows that the fully hybridized state becomes increasingly less important as the length of the strand increases.

flexible and stretchable, but it is not hard to extend an argument of this type to one that does not suffer from this limitation. In fact, an example will be given later on, when we are discussing an algorithm for computing partition sums.

These millions of possibilities are challenging to compute already, but as a microarray may have half a million different probes, tremendous calculation times would result when we were to find results for a complete microarray. If we are to improve the NN model, we must find some way of dealing with this mountain of possible configurations.

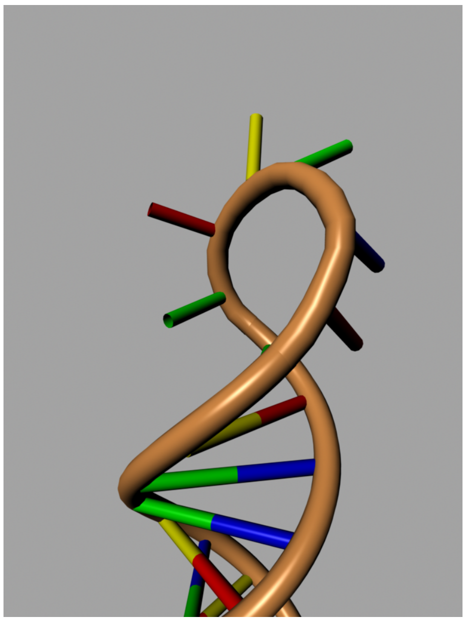

Figure 3: When a strand of DNA bends around and hybridizes to itself, a hairpin is formed.

2.3.2 New parameters in the extended nearest-neighbor model

Though we essentially hope to use the nearest-neighbor model in computing the hybridization free energy, we have to extend it in various ways to allow its full potential to be deployed.

Hairpin parameters. One of the most important effects we hope to cover in our efforts, is that of self-hybridization. In order for a strand to hybridize to itself, it must form a *loop* that is called a *hairpin.* Figure 3 shows an example of a hairpin.

This twisting of RNA or DNA and subsequent re-initiation costs some energy. This energy penalty depends on the strand type (DNA or RNA), the nucleotides forming the hairpin (those that are not connected) and the nucleotides closing the hairpin (the first base pair near the hairpin, which closes the loop).

This gives rise to many dozens of possible variables, related to nucleotide type and content, hairpin length, and more. Because limited experimental data is available, it was decided not to fit all possible hairpin penalties, but instead introduce only two parameters. It is hoped that two parameters will be enough to cover the rough behavior of hairpin formation, and finding many different parameters for hairpins will not be covered here. The two parameters introduced are one for DNA and one for RNA hairpins, respectively.

It is suggested in Ref. [3] that only hairpins of about four nucleotides occur frequently in nature. Furthermore, exploratory fitting indeed confirmed that the best agreement with experimental results is found when hairpins are required to consist of at least 4 nucleotides. Hence, any hairpin smaller than 4 nucleotides is not allowed, and a hairpin with at least 4 nucleotides is assigned the corresponding hairpin parameter as a free energy penalty.

Bulge parameters. In most cases of self-hybridization, there are several stretches of hybridized strand, separated by nucleotides that could not hybridize. After all, nucleotides are matching only by accident in cases of self-hybridization. The strand which has not hybridized, will have to be bent and twisted in order to accommodate the double helix forming on either side. Thus, it is not sufficient to simply sum up the base pair doublet contributions to the free energy, and then add the hairpin penalty - we also have to include a bulge parameter to account for this type of hybridization. Unfortunately, there are many possible bulge parameters, all depending on the scale of the bulge, the nucleotides forming the bulge, et cetera. For simplicity, only three distinct bulge parameters are used, one each for DNA/DNA, RNA/DNA and RNA/RNA bulge formation.

Initiation parameters and terminal A·T or A·U parameters. We have already reported the use of parameters for terminal A·U pairs in section 2.2.2. In the case at hand, we will use three parameters for DNA/DNA, RNA/DNA and RNA/RNA terminal pairs. One could argue that there are essentially four distinct types of initiation for DNA/DNA and eight for RNA/DNA - after all the 3' strand may have either A,C,G or T as its first nucleotide in the case of DNA/DNA hybridization. However, again we do not want to have to fit all parameters, and we restrict ourselves to the few that seem to be most important.

Internal A·U or A·T parameters. We have mentioned a parameter to differentiate between A·T initiation base pairs and C·G initiation base pairs. Both receive the same initiation parameter at first, but in the case of a terminal A·T pair an additional penalty is included. We extend this effect to the bulge case, and use an additional parameter for A·T initiation at a bulge or hairpin. Another parameter might be used for hairpins with a closing A·T base pair. It is possible that this parameter turns out to have a value close to the parameter for a terminal A·T base pair, but there is no clear physical reason to believe the parameters should be identical.

Dangling end parameters. Research has been done towards the effect of loose ends at initiation. For example, Ref. [6] reports that dangling ends appear to stabilize the hybridized strands, increasing the likeliness for hybridization. This effect was also included in our model by introducing a separate initiation parameter for initiation that had either one or two remaining loose ends.

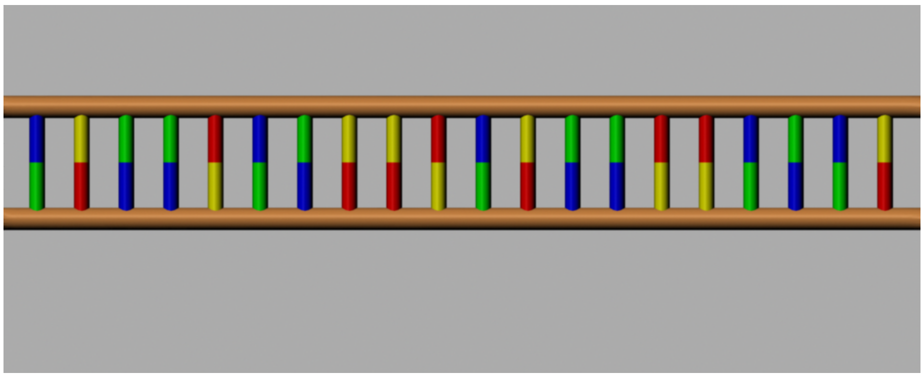

Figure 4: Two-dimensional representation of two strands that are fully hybridized to each other.

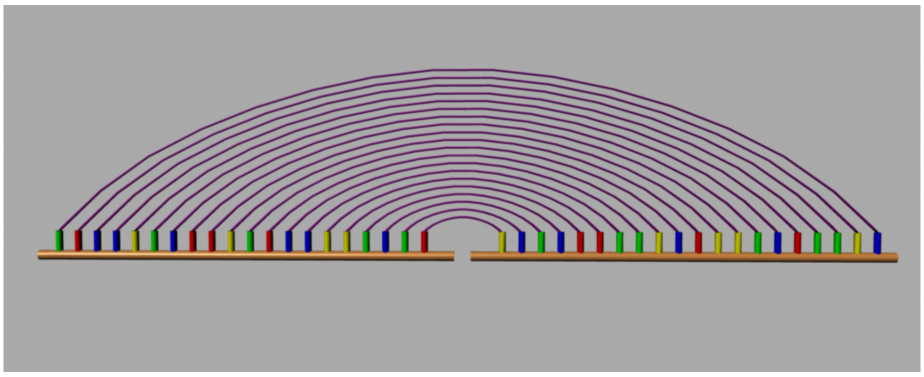

Figure 5: This representation of hybridization has given rise to the term *rainbow diagram*. The above diagram corresponds to two strands that are fully hybridized to each other, as in Figure 4.

2.3.3 Creating a method to implement the extended nearest-neighbor model

To work with our new model, we need to introduce a few new concepts. In the following, we will introduce and use the concept of *rainbow diagrams*. For a partially hybridized duplex of RNA and DNA, suppose we draw two parallel lines representing DNA and RNA, and connect these with lines representing the hydrogen bonds of the base pair (see Figure 4). Now if we 'open up' this diagram by pulling apart one end of the duplex, we arrive at the two strands in line, which may be seen as a single large strand composed of both DNA and RNA, where the crossing lines have become 'rainbows' joining the two strands (as shown in Figure 5). In a similar way, we can account for self-hybridization states (or equivalently, states with hairpins), by 'unfolding' the strand while keeping the connections intact, made visible by bows. Note that we never have to draw intersecting rainbows. This may seem a somewhat odd approach at this time, but the algorithm to estimate the effective hybridization free energy is easier to understand when related to the picture described here.

3 The Rainbow Algorithm

3.1 Introduction

With the concept of rainbow diagrams as introduced at the end of the preceding section, we are now able to begin the development of our algorithms. In the following, we will build towards an algorithm which incorporates as many different states as possible.

3.1.1 Example: Matrix-chain multiplication

Before we treat the Rainbow algorithm, let us first consider another problem, which will turn out to be related to the problem we wish to consider. This example is taken from Ref. [7].

Consider a chain of matrices, $\{A_1, \dots, A_n\}$. If we wish to compute the product $A_1 A_2 \dots A_n$, there are many possible ways to do this. For example, because matrix multiplication is associative, the equation $A_1(A_2A_3) = (A_1A_2)A_3$ holds. If the matrices involved are not square matrices, the computational cost may be different for diffent choices of *parenthesization.*[17] For example, if A_1 is a $p \times q$ matrix, A_2 is a $q \times r$ matrix and A_3 is a $r \times s$ matrix, the computational cost is either $qrs + pqs$ or $pqr + prs$ operations. Depending on the sizes of the matrices, one method might involve fewer multiplications than the other.

An interesting question is, what is the optimal parenthesization for a given matrix chain? Knowing the answer would definitely decrease the time required to complete the calculation with respect to the naive approach of just carrying out the multiplications from left to right. The interested reader is referred to the original in [7], meanwhile, we will not focus on the precise solution of the problem, but primarily on a few interesting properties of this problem, and the techniques required for finding a solution.

To find the solution, we will use the concept of an *optimal product.* An optimal product is a product such that the computational cost involved in computing the product, is minimized. Often, a certain *optimal* parenthesization specifying how to carry out the multiplications, is unique - one solution is better than all others. Of course, it need not be unique, for example a series of square matrices will not have a unique optimal solution.

Suppose now that we have already found the optimal product for *every* subchain of at most $m - 1$ matrices. That means that we suppose that we have found the optimal order for every chain $\{A_i, A_{i+1}, A_{i+2}, \dots, A_j\}$ for $1 \leq i < j \leq n$ with the additional restriction that $j - i + 1 \leq m - 1$, because the chain length is at most $m - 1$. Then it is fairly easy to determine the optimal product for every chain of length m, by the following procedure.

A chain of length m can be divided into two chains by splitting the chain at any of $m - 1$ locations. The computational cost involved in solving the chain of length m easily follows from the product of those two subchains, which can simply be regarded as single matrices. For each of the $m - 1$ locations, a certain total cost will be found, and we can select the choice with the lowest cost as the choice for the optimal product for our chain of length m.

[17]A parenthesization, obviously, is a possible configuration of parentheses, which define the order in which the matrix multiplications are to be carried out.

All that remains is to convince ourselves that the resulting product indeed is the optimal product for the chain, as by our definition, an optimal product is characterized by having the lowest computational cost of all possible parenthesizations. In fact, this is a rather immediate consequence of the method we have described. Any parenthesization can at some point be split in sub-products, this is the point of placing parentheses in the first place. Thus, the *optimal splitting* we have just found, is among the $m-1$ possible splitting locations, and we can construct any parenthesization (and in particular, the optimal product) from parenthesizations on subchains. Recursively applying this argument indeed confirms that we can construct an optimal product for a chain of length m by considering $m-1$ products of chains that are of length at most $m-1$.

Moreover, the number of operations is proportional to the cube of the chain length n. This computation time is possible only because of our observation that an optimal parenthesization is composed of a product of several smaller chains, each of which has an optimal parenthesization. The most naive approach requires $(n-1)!$ options to be checked,[18] while a better approach considering a certain configuration only once, still requires about $n \cdot 2^n$ multiplications, as there are 2^n possible configurations. This means that the number of multiplications still grows exponentially as a function of the chain length, a very undesirable property indeed. Most advanced encryption we know today, is unbreakable primarily because only exponential prime decomposition algorithms are known — in other words, this encryption is based on our inability to efficiently factor integers. In the case of matrix-chain multiplication, however, we can show that polynomial algorithms (in our case, order-n^3 algorithms) are possible.

Now let us consider the computation cost involved. At first sight, the recursive algorithm seems to behave exponentially, as the original approach did. We still have to check $(n-1)!$ places to cut, and in every case we must determine the optimal sub-product. However, there is one important difference. A closer look tells that we are doing a lot of unnecessary work - many of the sub-products are the same! Therefore, we must find a way to store obtained results, so that when we need to find the optimal parenthesization of a sub-product, we only need to look up the result. This leads to order n^3. First of all, there are at most $n^2/2$ subproducts[19] that we would need to compute. And, to optimize a certain sub-product, we need at most n multiplications of optimal sub-products. Thus, storage of results prevents a lot of redundant computations.

Apart from a recursive technique, we might also go bottom-up and just find all $n(n-1)/2$ subproducts. Both techniques are of order n^3. The fact that both recursive and iterative algorithms are possible in this type of problem, will play a major role when we discuss algorithms for DNA and RNA hybridization.

The fact that the optimal configuration must factorize into smaller optimal configurations, and hence, only optimal configurations are important, is crucial in the reduction of the number of configurations we have to consider. We will soon see that our Rainbow algorithm, extending the Nearest-Neighbor model, behaves similarly, and benefits from a very similar observation.

[18]Such an approach would be as follows: There are $n-1$ positions to do a multiplication, then on the resulting chain there are $n-2$ possible positions to choose from, et cetera. This approach is similar to the counting argument given in Section 2.3.1 which also leads to a factorial.

[19]Sub-products are of the form $A_i A_{i+1} \ldots A_j$ for some $1 \leq i < j \leq n$, so there are actually only $\frac{n(n-1)}{2}$ possible choices.

3.1.2 The problem of DNA and RNA hybridization

We have noted before that the number of possible configurations in RNA and DNA hybridization is rather huge. We will construct an algorithm that is based on an observation similar to that in the preceding example for matrix-chain multiplication.

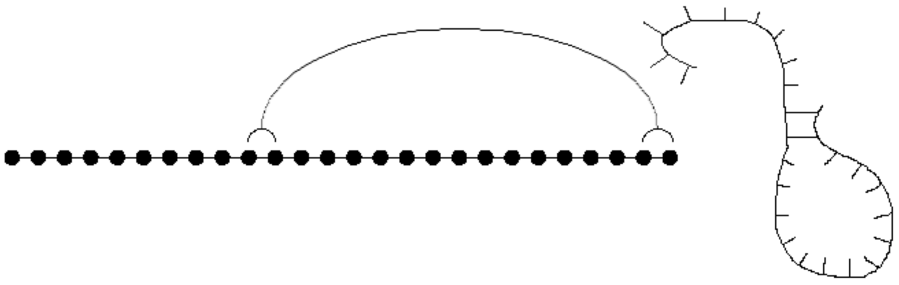

Figure 6: A single rainbow. The nearest-neighbor model requires that we consider not individual base pairs, but doublets of base pairs.

The key property that will allow an order-n^3 algorithm, is that we are going to split the partition sum into two independent parts at any given base pair, see Figure 6.[20] This means that the set of all possible configurations that have a connection at the given point, may be written in terms of the configurations to the left and to the right of that point, which are independent of each other. Note that this is an approximation, and it is known to be false in general. It is expected to be a good approximation for strands that are short enough. What exactly we mean by 'short enough' will be made more precise at a later time.

> ***Factorization of the partition sum***
> Consider a single strand of length n. Consider a pair i, j of nucleotides $1 \le i < j \le n$ such that i and j are complementary (a base pair may form). Consider the set $S_{i,j}$ of all configurations with a base pair between base i and base j. Then, *we assume that no state in $S_{i,j}$ has any base pairs that connect a nucleotide k in the interval (i, j) to a nucleotide k' outside the interval (i, j), for all i, j and for all k, k'.*

By our assumption, base pairs cannot cross our rainbow, and it is impossible for anything in the interval (i, j) to have any interaction with the outside world, see Figure 7. Consequently, we can write the partition sum for all states in the set $S_{i,j}$ as the product of the free energy contribution for the connection, times the partition sum of possible configurations on the left side, times the partition sum of possible configurations on the right side. Any partition sum may now be calculated recursively, using products of smaller partition sums. We shall refer

[20]Obviously, it is also allowed to split at a group of base pairs, and because of the nearest-neighbor model we shall in fact consider doublets of base pairs.

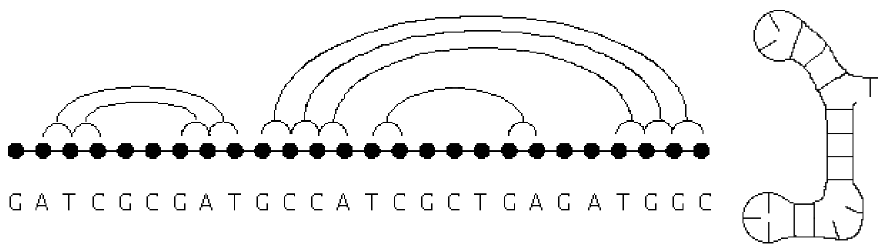

Figure 7: An example of a state of self-hybridization. Note how there are no intersecting rainbows. To keep the illustration simple, $j_0 = 2$ was used as the minimum hairpin size.

to partition sums on intervals as *partial partition sums*, or occasionally simply as *partition sums* when there is no ambiguity.

It is easy to see that even with this restriction, there are a lot of states left. Even under our assumptions, there are still too many states to allow a straightforward sum in polynomial time over all states.

> ***A lower bound for the possible number of allowed configurations***
> Let us find a lower bound for the number of configurations that are allowed under the current restrictions. First, take a strand of 200 nucleotides, of the form $ATATAT\dots ATATAT$. This leaves 100 AT doublets that may be connected by a rainbow, as long as none of the rainbows intersect. Consider now the doublets 1-10 and the doublets 91-100. There are 10^2 different ways of creating a single rainbow (a base pair doublet) that connects one of the first 10 doublets to one of the last 10 doublets.
>
> We do the same for the doublets 11-20 and the doublets 81-90, such that there are $10^2 \cdot 10^2$ ways of forming two bows, one between doublets 1-10 and 91-100, and another one between doublets 11-20 and 81-90. Proceeding in this manner, we find $(10^2)^5$ different configurations. This corresponds to a very small subset of all possible diagrams. Nevertheless, for general n, we find that the number of diagrams for an $ATATAT\dots ATATAT$ strand is bounded from below by $n^{\frac{1}{2}\sqrt{n}}$. This emphasizes the necessity of an algorithm which incorporates our independence assumption, cutting down the number of operations required to order-n^3.

We now recognize that this method of splitting the partition sum in two parts by creating a connection between two (pairs of) nucleotides resembles the problem of placing parentheses, as described in the preceding section. Though there are some differences, there is a major similarity, since in both cases we cut up some chain (strand) into smaller pieces, and in both cases, the computation cost is proportional to only the cube of the length of the chain (strand), because there are only n^2 different subchains (partial partition sums) and each subchain (partial partition sum) is calculated using at most n operations, and involving strictly smaller subchains (partial partition sums).

Additionally, in both cases, we need to store results for each subchain (partial partition sum), so that we can use the stored results in future iterations (we will need to store the optimal parenthesization of the subchain, and the value of the partial partition sum, respectively).

A difference, and major complication, however, is to consider connections formed by two nucleotides rather than one, as demanded by the properties of the nearest-neighbor model. We will have to adjust this algorithm to properly deal with this difference.

These observations give rise to the following rough sketch for an algorithm. Suppose that we have some strand of n DNA and RNA nucleotides, and we want to incorporate all configurations - self-hybridization and partial DNA-RNA hybridization. Then we could start computing the smallest partial partition sums; there should be about n of those. Then we compute the next level, of partial partition sums that are 1 nucleotide 'longer'. This should take at most n^2 products,[21] and in general it will be much less. Doing this for increasing length, we find that indeed we have developed an algorithm of order n^3, as intended. This scaling behavior allows analysis of microarray data, because it leads to quick computations of partition sums.

3.2 Straightforward Implementation

Now, the above loose sketch must be converted to an algorithm, based on which computer implementations may be written. Thus, let us consider a fixed strand of DNA. Denote this strand as $\{s_i\}$, $0 \leq i \leq n-1$, with n the length of the strand, such that s_i is the nucleotide at position i. We will require that a rainbow between two RNA doublets or two DNA doublets has a minimum size, as bending DNA or RNA has limits of its own. In the end, we will adapt our algorithm to reflect this physical property, and others found in DNA/RNA folding, but as the extra parameters are merely a complication, we will start out without them, including only the core elements of the algorithm at first, and then add the special effects later. The first version of our algorithm will thus assume that hairpin formation is not associated with any energy penalty.

In general, it appears that loops in self-hybridization (hairpins) do not form unless the loop consists of at least four nucleotides, so this is about the *minimum rainbow size* one would expect. In the following, let us denote the minimum rainbow size (the number of unconnected nucleotides making up a hairpin) by j_0. We initialize all partial partition sums of lengths smaller than $j_0 + 4$ to 1,[22] and we will then increment the length j by 1 every time, and calculate the values of all partial partition sums of length j. We shall denote the *partial partition sum* of length j starting at position i by $p_{i,j}$.[23] Note that this partition sum, therefore, ends at the position $i+j-1$, not at $i+j$. This partition sum

[21]The number of (partial) partition sums to be found, times the length j of the partition sum. This leads to $(j-1)(n-j)$ operations, which is bounded from above by n^2, in fact even $n^2/4$.

[22]The minimum rainbow size j_0 is usually 4, such that all partial partition sums of lengths up to $j_0 - 1 = 3$ are initialized to 1.

[23]Recall that we have already introduced the concept of a *partial partition sum* in Section 3.1.2.

contains at least all states that are also covered in the partition sum of length $j-1$, which is assumed known as its length is smaller than j, so we will initialize at $p_{i,j} = p_{i,j-1}$. This already covers many of the configurations that are required in $p_{i,j}$. However, at this time we are missing all configurations that make use of the nucleotide at position $i+j-1$. Our algorithm will have to find these configurations and include them in the partial partition sum in an efficient way.

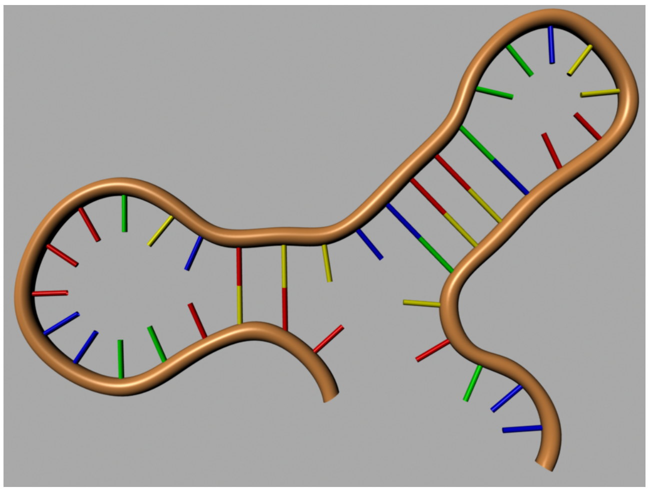

Figure 8: A strand may fold and hybridize to itself. The partition sum should contain a contribution for each possible state.

3.2.1 The single-bond model

To start out with a concise example that features only the most elementary aspects of our algorithm, let us first consider a single-bond model, rather than a double-bond (doublet-based) model such as the nearest neighbor model. In other words, we only consider individual base pairing, and a rainbow may form between A and T, or between C and G. Let us fix j and i, without loss of generality. Any rainbow that may be drawn connecting any two of the first $j-1$ places, is included in the partition sum $p_{i,j-1}$, so we do not need to consider those if we initialize properly:

$$p_{i,j} = p_{i,j-1} \tag{5}$$

Additionally, there may be rainbows connecting to the last (jth) position, and those are not yet included in our $p_{i,j}$, so we must still add those. It should be remarked that in our notation, the partition sum from some position $i+k$ to the final position $(i+j-1)$ is denoted as $p_{i+k,j-k}$. Thus, to find the partition sum $p_{i,j}$, we can write:

```
$p_{i,j} = p_{i,j-1}$
for $k = 0$ to $j - j_0 - 2$
  if $s_{i+k} = C(s_{i+j-1})$
    $p_{i,j} = p_{i,j} + p_{i,k} \cdot p_{i+k+1,j-k-2} \cdot e^{-\beta \Delta G_{s_{i+k}}}$
```

Here, $C(s_i)$ is a function which takes the complement, such that $C(s_i)$ is the nucleotide complementary to s_i. The exponent in the last line is the energy contribution due to the rainbow that connects nucleotides $i + k$ and $i + j - 1$, which also depends on the $\Delta G_{s_{i+k}}$ parameter for the hybridization free energy of a nucleotide of the type that is found at position $i + k$.

Note that for low k, we may encounter terms like $p_{i+j,-1}$ if j_0 is very small.[24] Furthermore, when we encounter a partial partition sum that is too small to sustain any rainbows (that is, shorter than length 2), it should be taken as having a value of 1: the only thermodynamic state possible for such a strand is the one without any connections, whose contribution to the partition sum is $e^0 = 1$.

All in all, this is a most elementary order-n^3 algorithm. From this point on, we shall be trying to extend and improve this algorithm, adding more and more flexibility and detail step by step in order to create a model that is able to predict RNA/DNA hybridization efficiently.

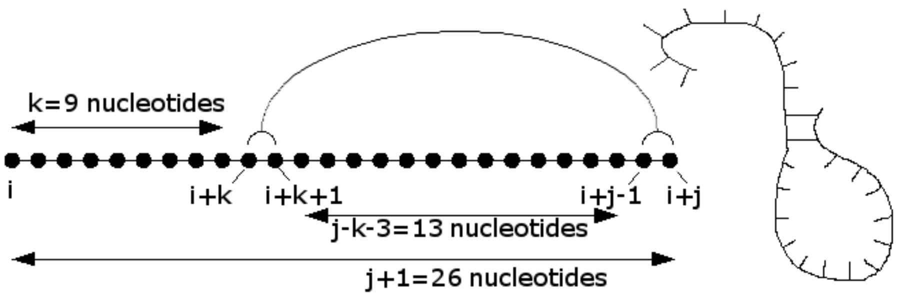

Figure 9: An example of the nearest neighbor model. Two neighboring nucleotides somewhere on the strand, are connected to the rightmost pair of nucleotides.

3.2.2 Minimal algorithm for computing the many-state partition sum

Now that we have this result, we should try to extend it towards an algorithm that does not work with a rainbow between just one nucleotide and another, but rather a rainbow that describes a connection between a pair of nucleotides somewhere along the strand, and another pair somewhere else. This would make the algorithm one that is compatible with nearest-neighbor model assumptions and restrictions. The most straightforward generalization, which is still incorrect, yields:

[24]For the case $j_0 = 0$ (which physically is not very relevant), we see that for the largest k in a loop (namely $k = j - j_0 - 1$, $p_{i+k+1,j-k-2}$ reads as $p_{i+j,-1}$. If it is desired that this is possible, this case must be taken special care of.

```
function $p_{i,j}$ Version 1 *(Missing some configurations)*
  $p_{i,j} = p_{i,j-1}$
  for $k = 0$ to $j - 4 - j_0$
    if $s_{i+k} = C(s_{i+j-1})$ and $s_{i+k+1} = C(s_{i+j-2})$
      $p_{i,j} = p_{i,j} + p_{i,k} \cdot p_{i+k+2,j-k-4} \cdot e^{-\beta \Delta G_{s_{i+k}, s_{i+j-2}}}$
  return
```

We have now incorporated the two-base length of a connection, and improved ΔG in a particular way to represent a base pair doublet. Unfortunately, this algorithm is *incorrect*, for the following reason. The problem lies with the *inner partial partition sum*, the partial partition sum that resides inside the rainbow we consider.[25] This partition sum should contain, among others, configurations that include a rainbow connecting the $i+k+1, i+k+2$ bases to the $i+j-3, i+j-2$ bases. This is the largest possible rainbow in the partition sum $p_{i,j}$ that does not use the first nucleotide of the substrand (the nucleotide at position $i+k$) or the last nucleotide of the substrand (at position $i+j-1$). Configurations of this type obviously do not contribute to the partial partition sum $p_{i+k+2,j-k-4}$ since they concern nucleotides outside the scope of that partition sum. Unfortunately, this means that a certain type of configurations is excluded from our calculations:

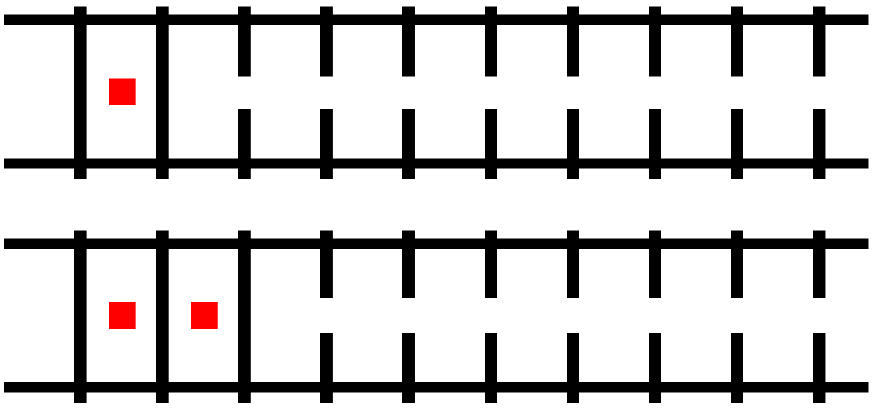

Figure 10: An illustration of missing configurations in Version 1 of function $p_{i,j}$. The long vertical lines indicate base pairs, and the red squares represent base pair doublets. The upper configuration is included in our calculations, as it should. However, none of the four used nucleotides can be reused in our recursive function, and the lower configuration is discarded even though it is a valid configuration for the nearest-neighbor model.

Configurations missing in the inner partial partition sum
Suppose there is a configuration which has a triplet of nucleotides at $i, i+1, i+2$ and another triplet of nucleotides $i+j-3, i+j-2, i+j-1$ for some $j > 5$ such that the triplets are complementary. This situation corresponds to the lower half of Figure 10. Consider the state s which has no base pairs, except for the three base pairs formed by

[25] In the algorithm we just described, the *inner partition sum* is given by $p_{i+k+2,j-k-4}$.

hybridizing the two triplets. Then, although by the nearest-neighbor model this state should be included in the partition sum, this state fails to be included in the partition sum that would be calculated by using Version 1 to calculate $p_{i,j}$, and this method can not be used to obtain results for $p_{i,j}$.

This is a major problem when considering complementary DNA and RNA strands. We just cannot hope to calculate correctly any state that has more than 2 consecutive base pairs, as the nearest-neighbor model is not compatible with this algorithm.

The first attempt to fix this, would be to include also the contribution from the rainbow between $i+k, i+k+1$ and $i+j-2, i+j-1$. However, if we allow the *entire* partial partition sum $p_{i+k+1,j-k-2}$ to be included, we are using nucleotides twice, and that is also not something we want.

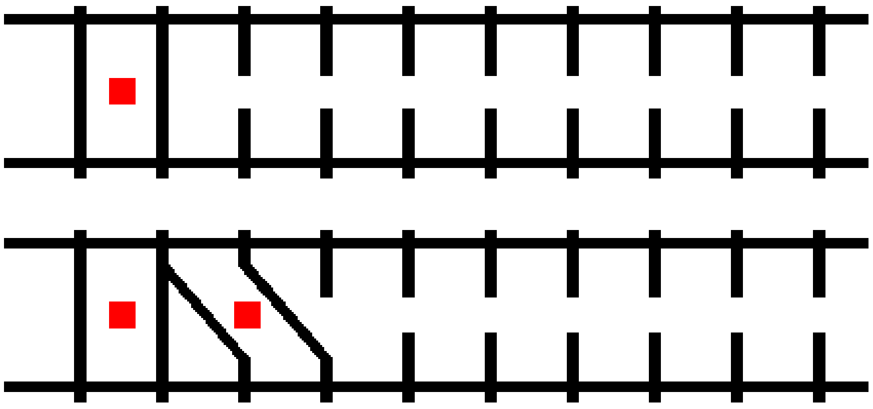

Figure 11: An example of an illegal configuration that arises when we try to allow configurations such as the lower configuration in Figure 10. When we allow the rightmost pair of nucleotides of a base pair doublet to take part in another base pair doublet, we risk using a nucleotide in two different base pairs, which is forbidden.

Illegal configurations in the inner partial partition sum
We might try to repair our algorithm by using $p_{i+k+1,j-k-2}$ instead of $p_{i+k+2,j-k-4}$ as the partial partition sum corresponding to the configurations that may form inside the rainbow. However, consider a rainbow contained in $p_{i+k+1,j-k-2}$ that connects to the first nucleotide of the partial strand, but not to the last. Then the first nucleotide is involved in two distinct base pairs, a configuration that is not allowed by the nearest-neighbor model. See Figure 11 for an example of such an illegal configuration.

We have seen two incorrect approaches, one which incorrectly discards configurations that are perfectly allowed by the nearest-neighbor model, and one which creates illegal configurations. We see that while we have to allow nucleotides to take part in two distinct base pair doublets, there have to be some restrictions to make sure that the nucleotide has only one partner with which it forms a base pair. This means that only a certain subset of the configurations in $p_{i+1,j-2}$ should be used. It thus turns out that we cannot compute $p_{i,j}$ using only order-n operations on smaller partial partition sums, because the inner partial partition sum cannot be written as a simple combination of partial partition sums.

> ***We can not fix the inner partial partition sum***
> There is no way to compute a partial partition sum solely from other, smaller, partial partition sums. This follows because we cannot exclude the innermost base pair of a rainbow (see how Version 1 failed), but we cannot include it either, as our attempt to do so, has failed as well. It is thus not clear how to incorporate the innermost base pair into the inner partial partition sum.

Fortunately, not everything is lost, and we can solve the problem as follows. We have to keep track of where the rainbows are, and distinguish between partition sums which have a rainbow between their extremes (endpoints), and partition sums that do not. We can then use those two partition sums at different intervals, and combine them. The partial partition sums with a *start-end bow* (a rainbow that connects the first nucleotides, i and $i+1$, to the last nucleotides, $i+j-2$ and $i+j-1$) will be called $\hat{p}_{i,j}$, and from now on $p_{i,j}$ will refer to a partial partition sum of all configurations on the substrand $(i, i+j-1)$, with the limitation that start-end bows are not allowed. The partial partition sum of all configurations possible on the substrand is thus given by $p_{i,j} + \hat{p}_{i,j}$. The altered version of the algorithm, for the new $p_{i,j}$, is then given by

```
**function** $p_{i,j}$ Version 2 (Minimal extended NN model)
  $p_{i,j} = p_{i,j-1}$
  **for** $k = 0$ **to** $j - j_0 - 4$
    **if** $s_{i+k} = C(s_{i+j-1})$ **and** $s_{i+k+1} = C(s_{i+j-2})$
      **if** $k \neq 0$
        $p_{i,j} = p_{i,j} + (p_{i,k} + \hat{p}_{i,k}) \cdot (p_{i+k+2,j-k-4} + \hat{p}_{i+k+1,j-k-2}) \cdot e^{-\beta \Delta G_{s_{i+k}, s_{i+j-2}}}$
      **else**
        $\hat{p}_{i,j} = p_{i,j} + p_{i,k} \cdot (p_{i+k+2,j-k-4} + \hat{p}_{i+k+1,j-k-2}) \cdot e^{-\beta \Delta G_{s_{i+k}, s_{i+j-2}}}$
```

As before, if the substrand length j is too small (smaller than the length $j_0 + 4$ of the smallest nontrivial partial partition sum possible), the program must treat it as a special case and enter 1 instead of the array value, which may not be initialized for too low values. In bad cases, the second argument might even be negative, leading to segmentation faults, so some care is required when this algorithm is used in a computer program. Additionally, the reason for choosing for $\Delta G_{s_{i+k+1}, s_{i+j-2}}$ is as follows. The first index corresponds to the outer base pair, and the second index corresponds to the inner base pair. However, we also must distinguish between DNA/DNA, RNA/DNA and RNA/RNA hybridization. Thus, the first index corresponds to the strand type (DNA or RNA) of the left end of the rainbow, and the second index corresponds to the strand type at the right end of the rainbow. Thus, there are 64 parameters associated to $\Delta G_{s_{i+k+1},s_{i+j-2}}$, since $s_i \in dA, dC, dG, dT, rA, rC, rG, rU$. For example, $\Delta G_{dC,rU}$ would correspond to DNA CA forming a base pair doublet with RNA UG. In the manner specified, $\Delta G_{s_{i+k+1},s_{i+j-2}}$ contains all the information necessary to determine the parameter corresponding to the nearest-neighbor doublet formed.

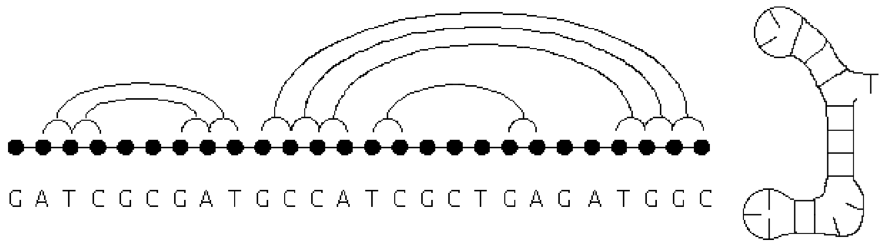

Figure 12: One of many configurations that contribute to the partition sum. Note how the introduction of $\hat{p}$ is required in order to allow all configurations from the nearest neighbor model to appear in the partition sum.

The parameters for bulges, hairpins, terminal A·T and *internal* terminal A·T effects[26] are not covered in this algorithm. However, these parameters may be incorporated into the algorithm in a straightforward manner and adding them would degrade the comprehensibility of the algorithms because of all the special cases required, so this will only be discussed at a later time, and not be included into the pseudocode for the algorithms.

The above algorithm confirms that there is indeed an order-n^3 algorithm to calculate the partition sum of all available partially-hybridized states, even though the actual number of different configurations is order $n!$. We have translated our observation, on how partition sums may be split up, into an algorithm that may be implemented in a computer program. In the following, we will delve into possible improvements and extensions of this algorithm, and discuss the possible consequences of the choices we make.

3.3 The Recursive Rainbow Algorithm

The above algorithm is, indeed, an order n^3 algorithm which does exactly what we need. However, it would be interesting to see if there is any room for improvement. For instance, our algorithm must perform the same check (whether or not a rainbow exists between $i + k, i + k + 1$ and $i + j - 2, i + j - 1$) very often, while in most cases such a rainbow will not exist. Statistically speaking, for a random strand the probability of two randomly chosen base pairs to be complementary is fairly small. Perhaps we can find a way to save ourselves this effort.

Furthermore, many of the partition sums calculated, are not necessary to find the result for the complete partition sum, so perhaps we can avoid treating them at all. These considerations suggest that we should develop a more sophisticated

[26]Recall that we have introduced internal A·T pairs in Section 2.3.2 as the bulge or hairpin equivalent of a terminal A·T pair.

algorithm, which does not work bottom-up (filling a table of $n \times n$ values of (i, j) entirely) but instead performs the computation recursively, such that it only calculates something when the result is required to find the final answer.

3.3.1 Listing all possible bows

First of all, we must find an alternative to checking which rainbows are possible. If we perform order-n^3 calculations, in the preceding algorithm, we also have to check n^3 times whether a rainbow is allowed.

A rainbow may be formed only when two nucleotides are complementary to two others. But this is a rare event! Generally speaking, assuming that the DNA sequence is random, the odds for a valid connection are only $\frac{1}{16}$. It thus seems beneficial to start off by creating a list of possible bows. This list can then be used to determine possible bows quickly. This replaces the expensive procedure of finding bows inside the main loop of the algorithm by straightforwardly carrying out checks.

One method is as follows. For each i, let us make a list of all k's such that the nucleotides s_i and s_k are complementary, as well as the nucleotides s_{i+1} and s_{k-1}. The most direct way for this would be an order-n^2 algorithm listing those k's. The resulting list would be quite large in system memory, an undesirable result.

Fortunately, another method is possible, which requires less memory. This involves a list of nucleotides that is sorted by its base pair type. We will skip further specifications of the n^2-type, as we shall not be using this method.

Avoidance of n^2-type lists is preferable especially for the longer sequences. We have to make only two passes to create two lists. Thus, we could remember a position in a list of candidates positions, and we would only have to save each position only once (rather than once for every position corresponding to a complementary doublet). Each base pair doublet type can have its own list (note that when considering complements, RNA and DNA do not need to be treated individually) such that we may create 16 lists (of total length n, as each position occurs only once and only in one of the 16 lists). Subsequently we can make another pass and make a list of starting points — positions of the first matching (complementary) doublet to the left. The starting points will help us find the first complementary pair, and from that point onwards we only need to iterate through the list (backwards).[27] The idea behind this setup will become clear in the recursive algorithm that will be given in the following section. Creating these two lists is not a very complicated matter, so the pseudocode is omitted here.

Finally, it should be remarked that as this part of the algorithm is not of order n^3, it is probably worth the effort of creating these two lists. We will call the type-sorted list K, because it is an ordered array of k candidates. The list of

[27]We iterate backwards through the list of k candidates because we consider rainbows between $i+k, i+k+1$ and $i+j-2, i+j-1$. It is then easiest to start with rainbows whose left doublet is close to $i+j$, and then decrement the position. Note that our list of partner pairs is necessarily a global list, as it is to be used for many different choices of i and j. It is thus impossible to iterate through the list of k candidates in the forward direction, because there would be no way to tell where to start the iterations — if we were to keep a list of starting points for forward iteration, we must note that the starting position would depend on both i and j and any such n^2-behavior is undesired. Summarizing, the only efficient way of iterating is backwards, and thus that is how we shall do it.

starting points will simply be referred to as *left*, because the positions referred to, correspond to the nearest complementary doublet *to the left* of the position. Finally, because we have to know where the 16 sub-lists begin and end, we keep a list of length 16 with the positions of the first doublet of each doublet type, and we call this list *first*.

While the conventions may seem confusing at first sight, the application and use of these structures in the algorithm in the next section should make more clear how they are useful.

3.3.2 Functions for the recursive algorithm

Now that we have a list of k's, we are ready to start working on our desired recursive algorithm. Our eventual goal is to develop a function that takes a DNA or RNA strand as its argument, and that returns the value of the (self-hybridization) partition sum. Our next step is to build a function which calculates the partition sum, calling itself in a recursive manner. Because we need both $p_{i,j}$ and $\hat{p}_{i,j}$, we will need two recursive functions.

In the calculations, we will also use the parameters for the nearest-neighbor model. They reflect the strength of the bond between the DNA and RNA, and thus are the elements used to construct the partition sum. Owing to the algorithm we use, we have to use the exponent $e^{-\beta \Delta G_{s_{i+k}, s_{i+j-2}}}$ for the energy contribution of a rainbow. We do not wish to calculate this many times, so we will carry along the exponentiated values, denoted by $g_{s_{i+k}, s_{i+j-2}} := e^{-\beta \Delta G_{s_{i+k}, s_{i+j-2}}}$, having calculated them only once. We only have to do multiplications then, and we do not have to perform many expensive exponentiations inside the algorithm.

Furthermore, we intend to recycle previously derived results. So, we will have to store them somewhere. For this purpose, we build a matrix, $M_{i,j}$, of partition sums from starting position i, and of length j. $M_{i,j}$ will be initialized at 0. Then, if we need the partition sum $p_{i,j}$, we will look at $M_{i,j}$. If $M_{i,j} = 0$, we have not yet calculated the partition sum. If $M_{i,j} > 0$, we have already got the partition sum, and there is no need to calculate it again. Note also that the partition sum always evaluates to a value of 1 or more, such that no ambiguities exist.

Additionally, as noted in the preceding (more intuitive) algorithm in section 3.2.2, we need both a matrix for partition sums with a bow connecting the extremal positions, and a matrix for partition sums that may have any bow except for the one between the first and last position. The matrix for the bows will be denoted $\hat{M}_{i,j}$. This matrix will be given a 0 for combinations of i and j that do not correspond to a valid bow, and 1 otherwise.[28]

It is actually fairly easy to fill this matrix with zeroes first, and then do an order-n^2 run for all n starting points through the K array (of length n) to determine all possible rainbows, setting some of the entries to 1. If we would not do this, we would have to determine at a later stadium whether a start-end bow was possible, and this checking is something we are trying to avoid. Thus

[28]Note that entries in $\hat{M}_{i,j}$ correspond to partition sums with forced start-end bows, and for most combinations of i and j no start-end bow is possible, and thus the entry in this matrix should be zero. For the few cases where a start-end bow will exist, we will put a 1 such that we see immediately that we need to compute something when we come across it.

it is important that we mark the entries in some way, such that we do not have to do any checking anymore while running the recursive algorithm.

In the following, *type*(r) refers to the type of the doublet at positions $r, r+1$, while *cotype*(r) refers to the complementary type. Note how we have adopted the convention to denote a doublet by the position of the left nucleotide.

Using the variables and arrays introduced above, we are now able to give the following pseudocode for the function $p_{i,j}$:

```
function $p_{i,j}$ Version 3 (Minimal extended NN model in recursive form)
  if $j \leq 3$ return 1
  if $M_{i,j} \neq 0$
    return $M_{i,j}$
  $sum = p_{i,j-1} + \hat{p}_{i,j-1}$
  $t = cotype(i + j - 2)$
  $index = left(i + j - 2)$
  while $index \geq first(t)$ and $K(index) > i$
    $k = K(index) - i$
    $sum = sum + (p_{i,k} + \hat{p}_{i,k}) * \hat{p}_{i+k,j-k}$
    $index = index - 1$
  $M_{i,j} = sum$        /* store sum, since this is the
  return $sum$              first time it was computed */
```

A lot happened here. We first had to initialize. We chose to initialize from the left end, $p_{i,j} = p_{i,j-1}$, such that we would be looking for rainbows ending at the nucleotide doublet $i+j-2, i+j-1$. Then we would iterate through the list of k candidates until either we found a candidate to the left of the leftmost nucleotide of our substrand (i) or until we exhausted the list of candidates, such that no more complementary doublets existed to the left of $i+j$. For every such rainbow, we found the contribution to the partition sum to be

$$(p_{i,k} + \hat{p}_{i,k}) * \hat{p}_{i+k,j-k}. \tag{6}$$

This is a simplification of the formula in Version 2, where we have used that

$$g_{i+k,i+j-2} \cdot (p_{i+k+2,j-k-4} + \hat{p}_{i+k+1,j-k-2}) = \hat{p}_{i+k,j-k}. \tag{7}$$

In the algorithm, we use a lot of function calls, and we start the function by checking if we have already computed a value. In programming practice, it is often preferable to check whether a call is required, and only make the call if that is indeed necessary.[29]

After the initialization procedure, all relevant rainbows were added to the sum. As we have prepared a list of such bows, we only have to walk through the list until we either receive a position beyond i (such that it is not part of the substrand), or exhaust the part of the list which relates to the base pair doublet type currently under consideration. The value discovered for $p_{i,j}$ is then stored in $M_{i,j}$ for future use.

After all these considerations, we still have to describe the function for $\hat{p}_{i,j}$ that sofar has been used in $p_{i,j}$ but remains undefined at this time. The function for $\hat{p}_{i,j}$ turns out to be a lot less complicated than the function for $p_{i,j}$, and the

[29]Function calls supposedly slow down computations, and should be avoided if possible.

computational effort is not of order n^3. In fact, we can find the required value fairly straightforwardly, though it depends on both p and $\hat{p}$.

```
function $\hat{p}_{i,j}$ (used for $p_{i,j}$ Version 3)
    if $j \leq 3$ return 0
    if $\hat{M}_{i,j} = 0$ return 0
    if $\hat{M}_{i,j} > 1$ return $\hat{M}_{i,j}$
    $\hat{M}_{i,j} = g_{i,i+j-2} * (p_{i+2,j-4} + \hat{p}_{i+1,j-2})$
    return $\hat{M}_{i,j}$
```

Note how, in the above function, we exploit the fact that we have initialized the $\hat{M}$ matrix in a particular way, such that 0 entries are found when no bow exists, and 1 entries are found when a bow exists but the corresponding partition sum $\hat{p}_{i,j}$ has not been calculated yet. When such a partial partition sum is calculated, it is stored in $\hat{M}$, and the next time we run $\hat{p}_{i,j}$ for this choice of i and j we can use the value we found the last time, a value which must be strictly greater than 1.[30]

These algorithms make up a quite efficient method to determine the partition sum of this system, and thus are helpful in finding the effective self-hybridization energy. The extension to DNA-RNA hybridization is straightforward from here: it is sufficient to tape the DNA and RNA strands together (though, due to directionality, some care is required) and consider the partition sum for the whole, combined strand.

3.3.3 Inclusion of a Reduction Matrix for more efficiency

One more remark should be made on the current structure of our algorithm. Many partition sums (in particular, partition sums for shorter strands) are identical to the partition sum for a smaller strand contained in itself. That is to say, if a strand's rightmost doublet cannot form a rainbow to any doublet on the substrand, it might as well be omitted from the calculations, and it need not be calculated at all. Such a strand is said to be *reducible*, and we may create a reduction matrix R which redirects calculations for reducible strands to the related irreducible ones. This does not really alter the algorithm, but instead reduces the number of times a function is called. For example, a substrand $ATATATAT$ would have the same partition sum as the substrand $ATATATATC$ and the latter substrand would be *reducible.* A possible setup for the reduction matrix would be as follows. We use the notation $(i, i+j-1)$ to refer to the substrand composed of the nucleotides i through $i+j-1$. The notation thus denotes the first and last nucleotide of a substrand, and may be thought of as some kind of interval notation.

[30]It need not be greater than 1, in fact, when there are $\Delta G_{s_{i+k},s_{i+j-2}}$ parameters that are nonpositive (which would cause the exponent to decrease below 1). If this is the case, another convention would have to be adopted, but the current choice is good enough for illustrating the technique used.

Entry at $R_{i,j}$	Meaning
-3	A rainbow (a base pair doublet) may form between positions $(i, i+1)$ and $i+j-2, i+j-1$ (such that all four together form a base pair doublet).
-2	No rainbow may form between the nucleotide doublets $(i, i+1)$ and $(i+j-2, i+j-1)$, but both the first and the last nucleotide may take part in some rainbow contained in the substrand $(i, i+j-1)$, so the substrand is irreducible.
-1	No rainbows are possible on the substrand $(i, i+j-1)$.
$\{i', i'+j'-1\}$	The substrand is reducible and its partition sum is equal to the partition sum of the irreducible substrand $(i', i'+j'-1)$, which is contained in $(i, i+j-1)$.

Under these conventions, when we need a certain $p_{i,j}$ we may look in the reduction matrix what we should do. If $R_{i,j}$ is smaller than -1 we will have to call $p_{i,j}$. However, if $R_{i,j} = -1$ we know that $p_{i,j} = 1$, and if $R_{i,j}$ points to some $\{i', i'+j'-1\}$ we can call $p_{i',j'}$ instead.

To give an example, the reduction matrix for the symmetric strand

$$ATGGCGATCXXXXGATCGCCAT$$

(the X's are used in gluing the strands together) will be given below, using a slightly different coding system to give a visual idea of the structure of the reduction matrix. The (anti)symmetry of the strand will also give rise to some symmetry in the reduction matrix.

In the following, $B = -3$ (start-end rainbows), x $= -2$ (irreducible), . $= -1$ (trivial), and $<$ corresponds to a reducible substrand. On the horizontal axis we have put the substrand length j, while the vertical axis corresponds to the leftmost nucleotide of the substrand, i. Both i and j start at zero in the lower left corner. The upper triangle corresponds to invalid substrands since $i + j$ exceeds the strand length of $n = 22$ for these values of i and j.

3.4 The Hybrid Algorithm

The techniques developed above for the recursive algorithm, can also be adjusted for use in iterative algorithms. This leads to a hybrid kind of algorithm, which works iteratively but uses the tables and lists from the recursive algorithm for more efficient computations. Thus, the computations are performed in an iterative way, starting out at the small j and then calculating larger j. However, our reasoning is more like that seen in the recursive algorithm, as we hope to be able to tell in advance what computations are truly necessary.

In this section, we will give an example of a *hybrid algorithm*, an algorithm that makes use of the best of two worlds. The hybrid algorithm performs exactly the same calculations as the recursive algorithm, but the order in which the calculations are performed is iterative, which motivates the use of the word *hybrid algorithm.* Eventually, in the next chapter 4, we shall compare various algorithms to determine their performance under different circumstances.

```
    ——————————————————————————————————————————————
    . ————————————————————————————————————————————
    . . ——————————————————————————————————————————
    . . . ————————————————————————————————————————
    . . . . ——————————————————————————————————————
    . . . . . ————————————————————————————————————
    . . . . . . ——————————————————————————————————
    . . . . . . . ————————————————————————————————
    . . . . . . . . ——————————————————————————————
    . . . . . . . . B ————————————————————————————
    . . . . B < < < < x ——————————————————————————
    . . . . . < < < < < x ————————————————————————
 ↑  . . . . . . < < < < < < ——————————————————————
 i  . . . . . . . < < < < < < ————————————————————
    . . . . . . . . < < < < < < ——————————————————
    . . . . . . . . . < < < < < < ————————————————
    . . . . . . . . B < x < < < < x ——————————————
    . . . . . . . . . < B x < < < < B ————————————
    . . . . B < < < < < x x B < < < < x ——————————
    . . . . . < < < < < < < < < B < < < x ————————
    . . . . . . < < < < < < < < < < B < < x ——————
    . . . . . . . < < < < < < < < < < < B < x ————
    . . . . . . . . < < < < < < < < < < < < B x ——
    . . . . . . . . B x < < < < < x B x x x x x B
   0

                       → j
```

Table 1: An example of a Reduction Matrix. i, j pairs corresponding to a rainbow $i, i+1 \rightarrow i+j-2, i+j-1$ are denoted by a B. Pairs corresponding to irreducible partial partition sums are marked with an x, while reducible partial partition sums are marked with a $<$. Finally, partial partition sums that are trivially 1 are denoted by a period. Note that the upper triangle corresponds to partial partition sums $p_{i,j}$ for which $i+j-1$, the end of the substrand, is outside of the strand, such that these combinations of i, j are not used.

3.4.1 Combining the recursive and iterative algorithms

The techniques outlined in the discussion of the recursive algorithm, may also be applied to a non-recursive algorithm which is more along the lines of the basic algorithm from the end of Section 3.2.2. The list of k's may also be incorporated into the iterative algorithm, like the reduction matrix. In fact, the only difference with respect to the basic algorithm is that more thought is given to the question what calculations are really necessary, and consequently what calculations may be omitted. The implementation is thus straightforward: the loop over all k is replaced by a loop over k's in the list of k's. Furthermore, not all partition sums are calculated, because the reduction matrix contains information on what partition sums are required for the final result.

The pseudocode for the hybrid algorithm is given below. The code covers all relevant combinations of i and j, while the loop over j should be the outermost loop. In the following, we have renamed the $M_{i,j}$ and $\hat{M}_{i,j}$ arrays by $p_{i,j}$ and $\hat{p}_{i,j}$, respectively. $p_{i,j}$ should be initialized at 1 for all $j < j_0 + 4$. $\hat{p}_{i,j}$ should be initialized at 0 everywhere. The total partition sum for the strand is given by $p_{0,n} + \hat{p}_{0,n}$.

```
**Hybrid Algorithm** (Minimal extended NN model in hybrid form)
  for $j = j_0 + 4$ to $n - 1$               /* strand length is $n$,
    for $i = 0$ to $n - j - 1$                positions are $0, \ldots, n-1$ */
      if $R_{i,j} = -1$
        $p_{i,j} = 1$
      else if $R_{i,j} = \{i', j'\}$
        $p_{i,j} = p_{i',j'}$
      else
        $sum = p_{i,j-1} + \hat{p}_{i,j-1}$
        $t = cotype(i + j - 2)$
        $index = left(i + j - 2)$
        while $index \geq first(t)$ and $K(index) > i$
          $k = K(index) - i$
          $sum = sum + (p_{i,k} + \hat{p}_{i,k}) * \hat{p}_{i+k,j-k}$
          $index = index - 1$
        $p_{i,j} = sum$
      if $R_{i,j} = -3$              /* Calculate $\hat{p}_{i,j}$ if necessary */
        $\hat{p}_{i,j} = g_{i,i+j-2} * (p_{i+2,j-4} + \hat{p}_{i+1,j-2})$
```

The reason that this algorithm is considered interesting, is that it is recursive without doing any function calls whatsoever. Or, alternatively, it is iterative, but in a smart way such that no checks need to be performed and no unnecessary computations are carried out.

4 Algorithms compared

Having introduced and described various possible algorithms, we would like to examine implementations for each algorithm and find out how they perform, compared to each other, and see how the algorithms differ. Additionally, we would like to see what approach performs best: to write a recursive algorithm, or to try to mix recursive elements into the basic algorithm to arrive at a more advanced hybrid algorithm. Or, perhaps, to keep things simple and straightforward and skip the complications.

4.1 Algorithm running times

4.1.1 Implementations and setup

Because the creation of a reduction matrix costs quite a bit of time, and also because the use of the reduction matrix is more difficult, the recursive and hybrid algorithms were tested both with and without making use of the reduction matrix. To this end, the algorithms have been compared by a series of test runs for a multitude of DNA strands.[31] Five different algorithms were considered:

1. The basic (iterative) algorithm (sec. 3.2)
2. The recursive algorithm, using only the list of k's (sec. 3.3)
3. The hybrid algorithm, using the list of k's (sec. 3.4)
4. The recursive algorithm, (2) while also using a reduction matrix
5. The hybrid algorithm, (3) while also using a reduction matrix

Furthermore, as some algorithms require other preliminary calculations than others, a number of preprocessing steps was examined[32]:

6.	Creating the list of k's	Used by: 2,3,4,5
7.	Creating a list of starting points in the list of k's	Used by: 2,3,4,5
8.	Initializing matrices to zero	Used by: 2,3,4
9.	Creating the reduction matrix	Used by: 4,5

Random DNA was created using a Mersenne Twister-based random number generator [19]. The length varied from 4 nucleotides for the shortest strand, to a little less than 800 for the longest. These limits were chosen because no rainbows can form in strands consisting of less than 4 nucleotides, and because model limitations cause a practical limit of about a hundred nucleotides, beyond which effects like kissing hairpins and pseudoknots become important, and our model falls short of physical reality. Nonetheless, we also investigated higher lengths as this might prove interesting to people who want to build algorithms based on the algorithms described here.

All results were obtained using an AMD Opteron 848 CPU running at 2.2 GHz, and using 4 GB of RAM.

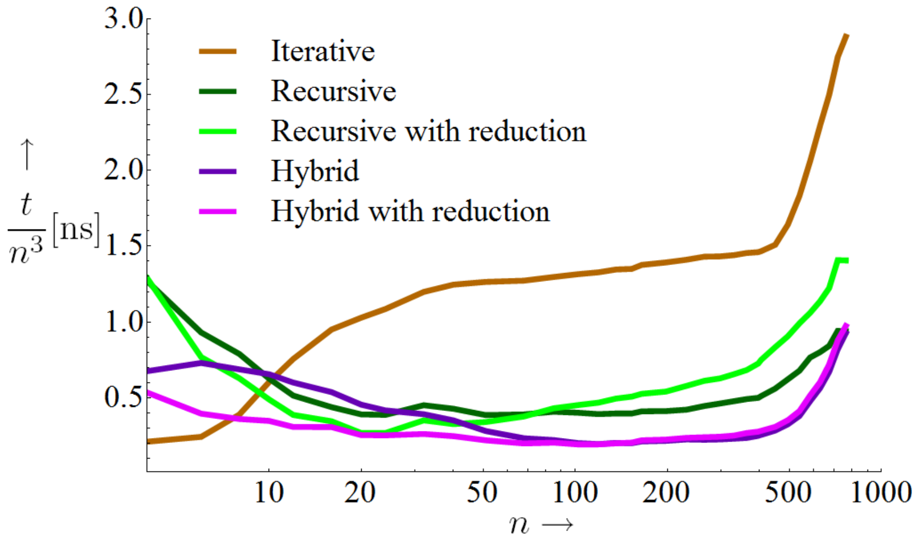

Figure 13: Comparison between the iterative, recursive and hybrid algorithms for randomly generated DNA strands. For each length n, 32 strands were generated and tested. The average time t required for the computations (in ns) was divided by n^3, where n is the strand length.

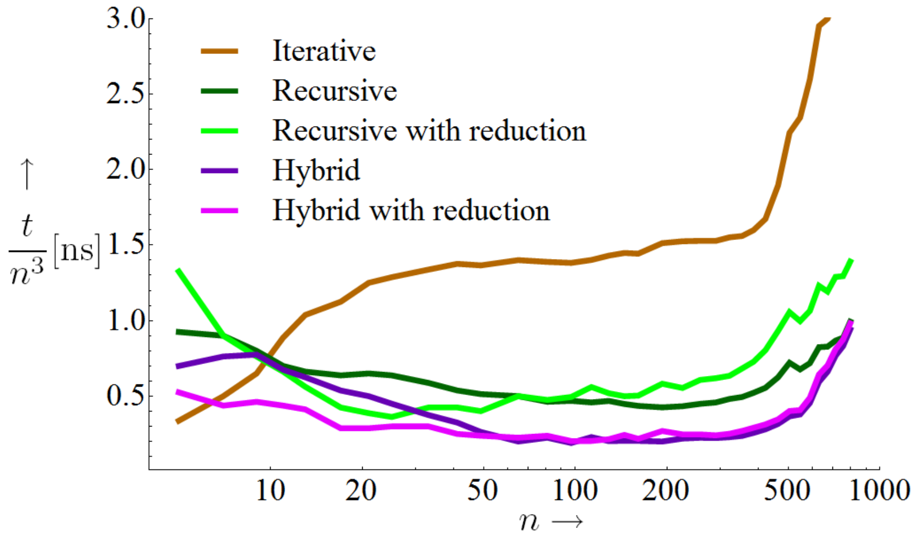

Figure 14: Comparison between the iterative, recursive and hybrid algorithms for 'symmetric' DNA strands. Motivated by the fact that microarrays require hybridization of two complementary strands of DNA, we generated random strands, then glued these together with their own complement, as would be done for microarray calculations. Once again, the image was scaled by dividing by n^3.

4.1.2 Timings of the algorithms

Besides testing the algorithm on randomly generated DNA strands, we also investigated the performance of this algorithm when used to perform calculations on symmetric DNA. That is to say, for example in microarrays, we are primarily interested in the algorithm's performance when applied to two complementary strands. As an antisymmetric strand is not quite a *random* strand, it was not at all obvious whether the different algorithms would respond differently. Such a difference, however, is not found, as can be seen from Figures 13 and 14, and there appears to be little difference between the two types of strands as far as our algorithms are concerned.

There is, however, a distinctive difference between the types of algorithms. First, Figures 13 and 14 show that the improved iterative algorithms achieve the best performance. For the smaller strands, the use of a reduction matrix appears to cut down computation time substantially. However, for the longest strands there is no gain, and perhaps even a slight decrease in performance, when the reduction matrix is used in the calculations.

4.1.3 Preprocessing

While the improved algorithms do indeed speed up calculations, a few extra steps have become necessary, too. Because those are not intimately related to the algorithms, they have been treated separately. Furthermore, if the algorithm is run multiple times, but with different energy parameters (which will turn out to be useful in due time), these steps do not need to be repeated, and thus they are not really a part of the algorithm, but rather a part of its implementation. For many applications, however, they need to be carried out for every strand used as input by the algorithm, however, and it is important to know their contribution to the computation time, relative to that of the algorithms themselves.

The creation of the list of partner doublets and the list of starting points is linearly proportional to the strand length n, as the lists are both of length n. This is shown in Figure 15. The matrix initialization and reduction matrix, as shown in Figure 16 are both of size n^2 and their creation times are thus expected to be proportional to n^2.

A few jumps become obvious. Apparently, at certain points, caching effects cause sudden increases in computation time. Other than that, it is confirmed that, in line with our expectations,[33] the matrix initialization and reduction matrix creation are of order n^2, and creating the list of k's and the list of starting points are of order n.

Figure 17 shows that the preprocessing steps are only important for very short lengths, $n < 10$, and marginally important for $10 < n < 30$, depending on the chosen algorithm. Beyond this length, the effort required to complete all

[31]For the algorithm comparison, it doesn't matter if RNA nucleotides are omitted, we are primarily interested in the efficiency, which is not expected to depend on the strand type.

[32]The hybrid algorithm without reduction matrix (3) must initialize $\hat{M}$, but when the reduction matrix is present (5), we can also check whether $R_{i,j} = -3$ and we do not need to initialize anything.

[33]The matrices contain n^2 entries, and the lists contain n entries, so if filling them is anything near efficient, we would only expect a constant parameter to show up next to the n^2 or n, respectively.

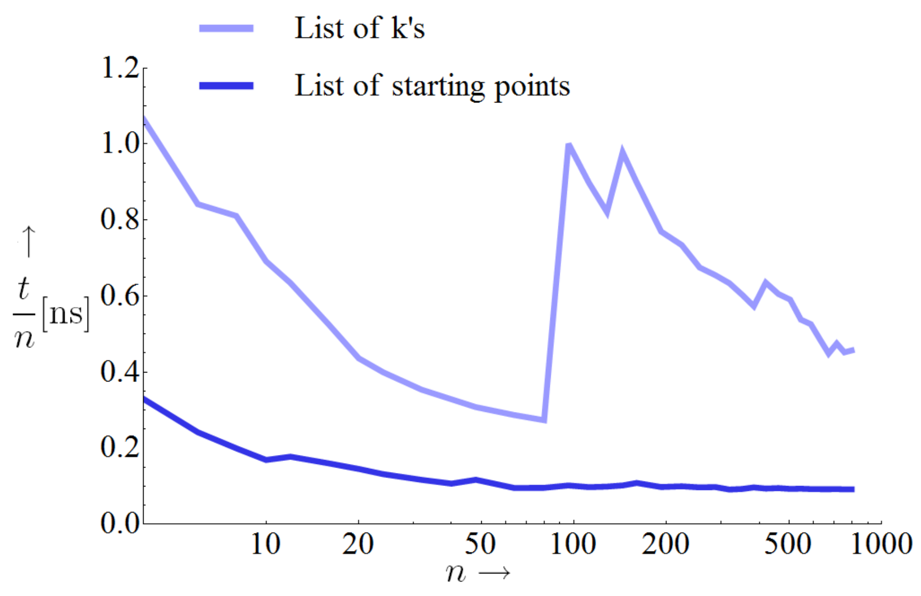

Figure 15: Creation time required for the list of k's. The linear dependence on n is made clear by dividing by n. These arrays are required for all algorithms except the iterative algorithm.

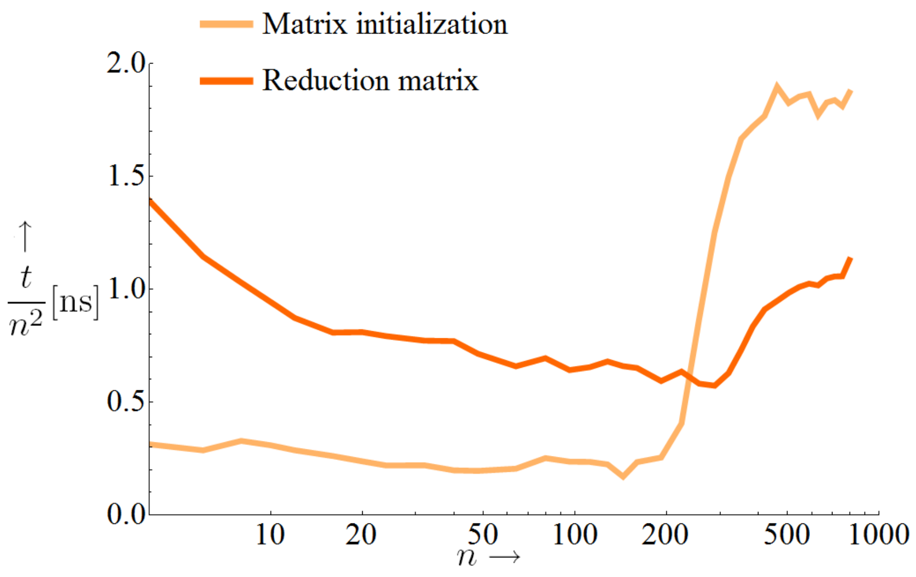

Figure 16: Time required to initialize the arrays for result storage, and the time it took to create a reduction matrix. Note that these preprocessing steps are not required for all algorithms. The results were divided by n^2 to emphasize deviations from perfect n^2-proportionality.

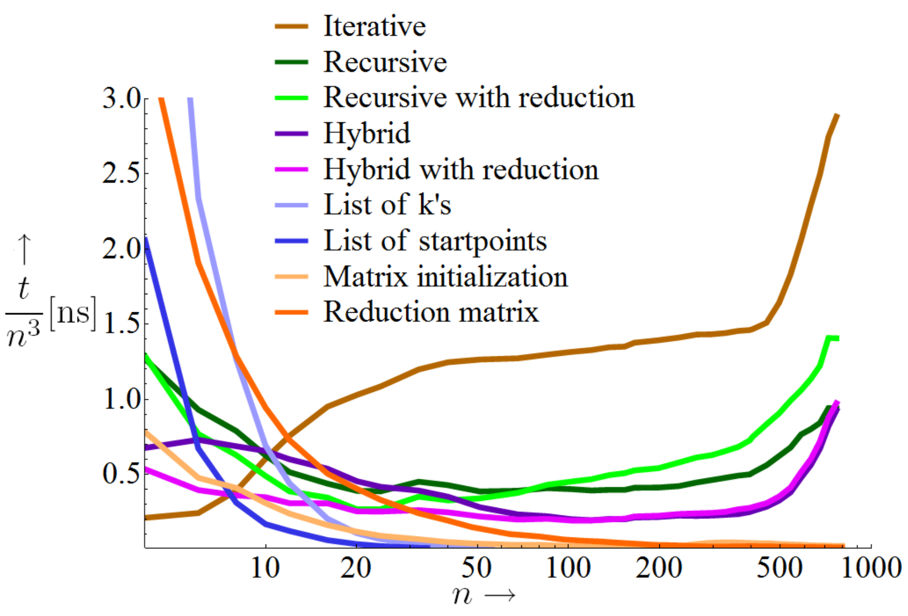

Figure 17: In this figure, the preprocessing steps have been drawn as thin lines through the algorithm performance graph, Figure 13. All graphs have been divided by n^3.

preprocessing steps quickly becomes less than 10 % of the total time, and the effect is negligible. That is to say, in comparing algorithms, we can conveniently ignore the preprocessing costs for $n > 30$. For calculations on microarrays of length 25, which involve calculating partition sums of length 50 for RNA/DNA hybrids, the iterative algorithm with reduction matrix is found to be the most efficient one.

4.2 Algorithm analysis

A different way to assess algorithms, is to count some core elements of the algorithms. For example, one might count the number of times that two floats are multiplied, or the number of function calls, or the number of different arcs on a strand. In this section, we seek to give a clear analysis of the difference between the algorithms, in terms of such counts.

4.2.1 The number of flops

A good way to estimate the computational load of a program, is to compute the number of *flops*, floating-point operations. As there are different possible definitions of the word *flop*, we shall define it in this context as *a multiplication between two numbers in double precision*. In fact, such multiplications do not show up often, they are only used when products of partial partition sums are computed, and when any $\hat{p}$ is calculated. The results are given in Figure 18.

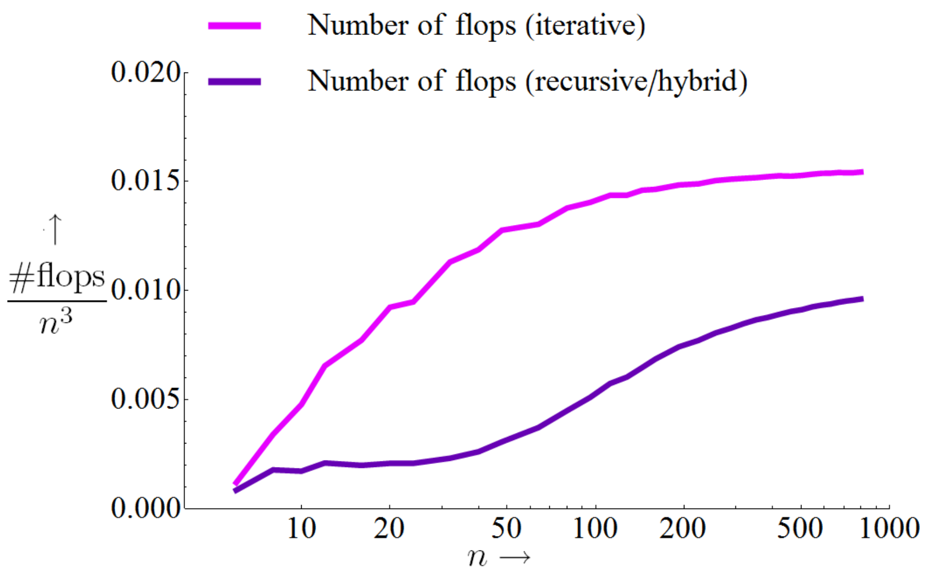

Figure 18: The number of flops for random strands, both for the iterative and the recursive/hybrid algorithm, divided by n^3. The number of flops is the same with and without reduction matrix, because in both cases the number of products taken between partition sums is minimal. Moreover, there is no difference between flop counts for the recursive and the hybrid algorithm, because they perform the same multiplications, only in a different order.

4.2.2 The number of irreducible partition sums

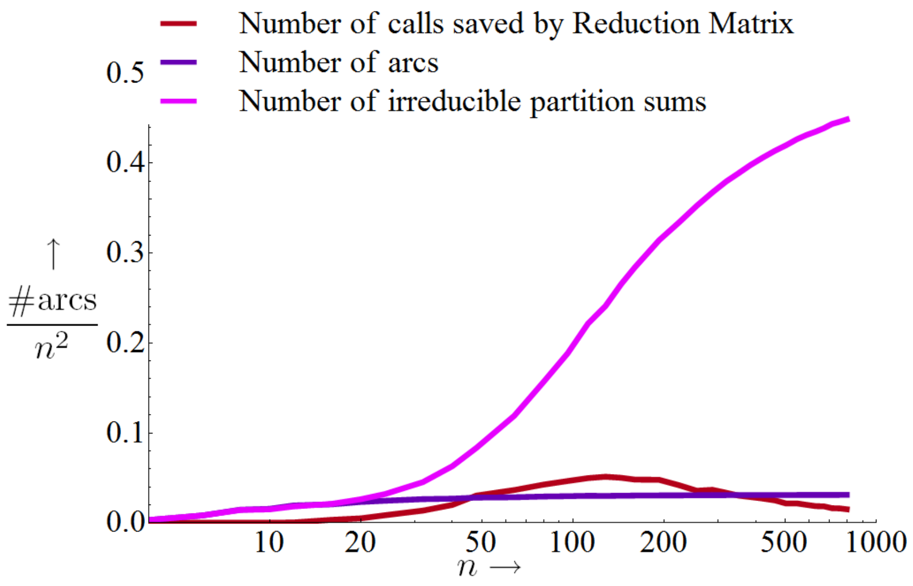

Figure 19: The number of arcs, and the number of irreducible partition sums (also including the number of arcs). The results were divided by n^2 to illustrate how the counts converge towards large-n limit values. Furthermore, for comparison purposes the effect of the reduction matrix is also included, to show which portion of the irreducible partition sums may profit from the use of the reduction matrix.

Figure 19 displays a few of the most interesting statistics. The number of rainbows is seen to be proportional to $\frac{n^2}{32}$. This may be well understood: in a strand of length n, there are about $\frac{n^2}{2}$ possible rainbows, but because of the nature of the rainbows, the probability for any combination of two pairs of nucleotides to be complementary is only $\left(\frac{1}{4}\right)^2$. The number of irreducible partition sums is also of order n^2. Naturally, there are at most about $\frac{n^2}{2}$ different partition sums, and for long strands, few are reducible. However, likely because of our reduction matrix, the curve approaches $\frac{n^2}{2}$ only for large n. At $n = 50$, only a fraction 1/5 of the maximum number needs to be computed.

Also included is the work done by the reduction matrix. It follows that at certain lengths, up to 20 % of the function calls may be saved by the reduction matrix. Although this may not seem a lot when compared to the difference between the recursive algorithm and the iterative algorithm, a 20 % better performance is still noteworthy, and indeed a significant improvement is seen for short strands in Figure 13. Furthermore, the reduction matrix is known to improve the algorithm in more ways than reducing partition sums for the hybrid algorithm, where it is used to determine whether a rainbow exists. This is indeed seen in Figure 13, too, as the inclusion of a reduction matrix improves the performance for the hybrid algorithm up to about $n = 100$.

4.2.3 The number of array lookups

Another parameter of interest is the number of array lookups and writes, presented in Figure 20. The intensity of retrieving data from memory or cache, and storing results, is also an indication of algorithm performance. We see that the iterative algorithm does not need as many lookups for short strands (while the other algorithms use k lists, the reduction matrix, et cetera, which requires many lookups), but that the number of lookups is high at long lengths. This is likely caused due to excessive checking of possible bows, which we have repaired in the other algorithms by introducing a list of possible partner pairs. The prevention of all these checks is a likely cause for the significantly better performance of the recursive and hybrid algorithms with respect to the iterative algorithm.

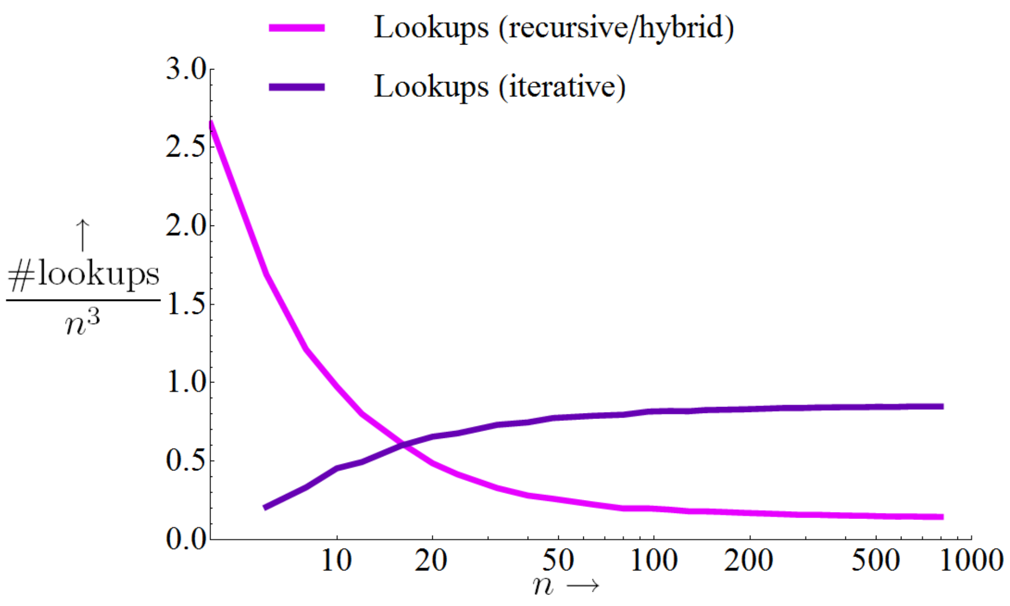

Figure 20: The number of flops for random strands, both for the iterative and the recursive/hybrid algorithm. The number of flops is the same with and without reduction matrix, because in both cases the number of products taken between partition sums is minimal. Moreover, there is no difference between flop counts for the recursive and the hybrid algorithm, because they perform the same multiplications, only in a different order. We thus see that for small n, the iterative algorithm needs the fewest lookups, while the recursive and hybrid algorithms are more efficient for larger n. Fewer lookups in general means easier caching, and the reduced number of lookups might be responsible for the improved performance of the recursive and hybrid algorithms with respect to the iterative algorithm at large n.

5 DNA and RNA Thermodynamic parameters

In our calculations on microarrays, the parameters for the binding energy between nearest-neighbor pairs of DNA and RNA are crucial. These parameters may seem well-known, but in fact uncertainties are large. In addition, the parameters were derived using the two-state approximation for the partition sum, and we would like to derive a new set of parameters using the many-state model that we have developed. In deriving this set, we will put our freshly made algorithms to the test.

5.1 Existing Experimental Results

In the 1990s, a number of groups have performed experiments to determine the parameters involved in DNA and RNA thermodynamics [21, 24, 27]. These groups all used the procedure known as *UV melting* to determine the values of $\Delta G(i,j)$.

'UV melting' works as follows. One considers a single strand, and its complement. First, one produces a solution of these DNA and/or RNA strands. This solution consists of a range of chemicals, most notably salt. It is known that such a solution becomes less transparent to UV radiation when the fraction of duplexes (hybridized strands) decreases. This is known to occur when the temperature increases. Since the rest of the solution does not change its translucency as temperature changes, this offers an opportunity to establish a relation between the temperature, and the fraction of strands that is part of a duplex.

Aside from the temperature dependence, the fraction of hybridized strands is also dependent on the salt concentration, and on the strand concentration. At constant salt concentrations, repeated experiments at different concentrations will thus yield different melting curves. From these curves, the point where exactly half of all strands are part of a duplex can be discerned. The corresponding temperature is known as the *melting temperature*. A plot of these melting temperatures versus the logarithm of the concentration may then be used to determine values for ΔS and ΔH, the entropy and enthalpy, respectively. Note that these represent values that correspond to a certain strand of known configuration.

From the values of ΔS and ΔH, we derive the value for ΔG by the thermodynamic relation

$$\Delta G = \Delta H - T\Delta S. \tag{8}$$

In this manner, one can determine ΔG parameters for any strand at any temperature. As was discussed before, when these parameters are assumed known, for any duplex, the value of ΔG may be predicted. Thus, once a significant amount (typically about a hundred) of strands has been examined, a set of parameters can be determined.[34] The most commonly used method is to perform a least-squares fit: try different sets of parameters, and find which choice of parameters yields the lowest sum of square deviations.

[34]Obviously, the number of data points (i.e. the number of strands) should be large compared to the number of parameters.

5.2 Thermodynamic Parameters for the Extended Model

The current parameters used in the nearest-neighbor model have been derived from experimental results for ΔG parameters at 37 °C. However, these parameters were derived using the two-state model. Thus, it would be particularly interesting to rederive the parameters for our model, as we are using a many-state model that likely needs different parameters. Thus, in the following we will refit the parameters for the nearest-neighbor model, based on the experimental data used for melting experiments of small hybrids in solution. This is the same experimental data that has been used to determine the values currently known in the literature [1, 24, 27]. However, we have now used the previously described algorithm to compute new values for ΔG, such that we generate a set of parameters that is well suited to our model.

This refitting is a first application of our improved model. The resulting refitted values for the parameters in the nearest-neighbor model are presented in Tables 2 and 3, together with the total squared deviation between experiment and the prediction obtained with the nearest-neighbor model. Since our model avoids a poorly justified assumption in the existing two-state nearest-neighbor model, a better agreement with experimental values is anticipated.

Because now hairpins are allowed as well, the parameter set for DNA/RNA hybridization also depends on the parameters for RNA/RNA and DNA/DNA hybridization. Thus, the parameters were fitted simultaneously, minimizing the total square deviation over all available experimental results combined. Hairpin loops are assumed to contain at least four consecutive nucleotides that are not part of a base pair. A fixed parameter was used, such that for any hairpin, regardless of its configuration or size, the same penalty is added.

5.2.1 Experimental data

To redetermine the parameters, experimental data for ΔG were required. The experimental results used by Allawi [1], Xia [27] and Sugimoto [24] for DNA/DNA, RNA/RNA and RNA/DNA respectively, were used in the analysis. Note in particular that these values are from experiments, so they do not contain the assumptions from the INN-HB model with its two-state approximation for the partition sum. The experimental results are thus model-independent and may be used to derive a new parameter set.

Because of the small length of the strands, self-hybridization is likely a minor contribution and hairpin formation is minimal. However, it cannot be completely neglected, especially for the self-complementary DNA/DNA and RNA/RNA strands. Thus, using the algorithm described in the previous chapter, a fit was made for the stacking free energy (thermodynamic) (ΔG) parameters, also called in our extended model, for RNA/RNA, DNA/DNA, and DNA/RNA. Fits were made for all three because RNA/RNA and RNA/DNA parameters are required in order to perform calculations with self-hybridization.

5.2.2 Results using the Rainbow model

A few additions to the set of parameters are required, because some new situations arise in our Rainbow model. Because of partial unzipping, O-type loops may form. After a few test runs, it was decided not to include a free energy penalty to a partial unzipping, because the parameter was small and unreliable, taking values both below and above zero in different setups. However, an internal AU parameter (similar to the terminal AU parameter introduced earlier, but now corresponding to AU pairs at the location of unzipping) was used, and turned out to be a significant improvement to the model. Furthermore, hairpins were only allowed whenever there were at least 4 nucleotides contained in the loop. A penalty was assigned to any hairpin spanning four or more nucleotides, for both DNA-DNA and RNA/RNA loops.

The addition of these parameters strains the limited pool of experimental data used to derive the original parameter sets. The addition of the terminal AU/AT and internal AU/AT to the RNA/DNA set, for example, increases the number of parameters to 19, while only 64 data points are available. However, while more verification using larger data sets is desirable, adding these two extra parameters cuts the total square error down from 6.3 to 5.3. This was considered sufficiently significant for inclusion in the model.

For the fit of RNA/RNA parameters, 90 sequences were available. Furthermore, there were 108 DNA/DNA sequences, and 64 RNA/DNA sequences. Because the RNA/DNA sequences may also form hairpins, which in turn need the DNA/DNA and RNA/RNA parameters, it was chosen to fit all parameters simultaneously.

parameters	[27]	many-state	parameters	[1]	many-state
AA/UU	-0.93	-0.86	AA/TT	-1.00	-0.88
AU/UA	-1.10	-1.04	AT/TA	-0.88	-0.64
UA/AU	-1.33	-1.27	TA/AT	-0.58	-0.62
CU/GA	-2.08	-2.04	CT/GA	-1.28	-1.14
CA/GU	-2.11	-2.05	CA/GT	-1.45	-1.58
GU/CA	-2.24	-2.25	GT/CA	-1.44	-1.35
GA/CU	-2.35	-2.39	GA/CT	-1.30	-1.42
CG/GC	-2.36	-2.28	CG/GC	-2.17	-2.16
GG/CC	-3.26	-3.23	GG/CC	-1.84	-1.81
GC/CG	-3.42	-3.45	GC/CG	-2.24	-2.26
ΔG_{init}	4.09	4.16	ΔG_{init}	1.82	2.27
termAU	0.45	0.45	termAT	0.05	0.09
intAU^a	-	0.78	intAT^a	-	0.68
hairpin	-	4.54	hairpin	-	9.80
Error	9.40	8.32	Error	14.86	17.00

Table 2: The nearest-neighbor model thermodynamic parameters for $\Delta G(a, b)$ for RNA/RNA (left) and DNA/DNA (right), for the two-state models from the literature, and for our many-state model. ^{a} To aid partial hybridization, a parameter was introduced for terminal A·T and A·U pairs, similar to the usual terminal A·T penalty, but now for partial unzipping inside the hybrid. See Section 2.3.2.

doublet type	[24]	[10]	this work
AA/TT	-1.00	-0.95	-0.78
AC/TG	-2.10	-1.76	-0.91
AG/TC	-1.80	-1.54	-0.59
AU/TA	-0.90	-0.71	-0.51
CA/GT	-0.90	-1.25	-2.12
CC/GG	-2.10	-1.92	-1.88
CG/GC	-1.70	-1.78	-1.77
CU/GA	-0.90	-1.07	-1.73
GA/CT	-1.30	-1.82	-2.76
GC/CG	-2.70	-2.62	-2.66
GG/CC	-2.90	-2.64	-2.63
GU/CA	-1.10	-1.36	-2.27
UA/AT	-0.60	-0.75	-0.86
UC/AG	-1.50	-1.28	-0.39
UG/AC	-1.60	-1.37	-0.58
UU/AA	-0.20	-0.34	+0.09
ΔG_{init}	3.10	2.90	3.25
termAU	-	-	-0.79
intAU	-	-	0.98
Error	15.26	7.12	5.26

Table 3: Thermodynamic parameters $\Delta G_{m,m+1}$ in kcal/mol for DNA/RNA. The parameters of [10] are based on the experimental data in [24], using a different fitting procedure.

5.3 Discussion of the new parameter set

5.3.1 RNA/DNA parameters

It is surprising to see that the fit for RNA/DNA hybridization actually converged to a parameter set where terminal AU pairs are given a negative free energy contribution - this would mean that A · U (or T · A) pairs are actually stronger than C · G pairs. As C · G pairs have three hydrogen bonds while A · U pairs have only two, this is unexpected, and there is no obvious reason for this. It is quite possible that this is not a physical effect, but instead an artifact of fitting many parameters on relatively few data points.

The parameters for pairs which have A's and T's have decreased significantly. As a larger termAU parameter and smaller parameters for pairs containing A, T or U nucleotides tend to compensate each other, large deviations in parameters are possible with only small changes in the predicted hybridization free energy for a given pair of DNA and RNA strands. Consequently, the parameters must be regarded with caution. On the other hand, this effect is encountered in every fit, and this also means that literature values may be far from the true values.

Our model and parameters predict hybridization free energies which are closer to experimental values, and the total square error decreased by 66 % (with respect to [24]) and 26 % (with respect to [10]), respectively. Thus, our many-state model, making use of this new set of parameters, is expected to be able to predict hybridization probabilities much more accurately.

5.3.2 RNA/RNA parameters

The changes in the parameters for RNA/RNA are not very significant. The total square error decreased by 11 %, but it is hard to tell how much of this effect is due to the additional two parameters. If the hairpin parameter would be very small, no hairpin formation would be possible. However, at its current value, some of the hairpins still contribute significantly to a strand's self-hybridization partition sum, and it seems that hairpin formation is possible even at these lengths.

5.3.3 DNA/DNA parameters

The many-state model failed to improve results for DNA/DNA parameters. The most likely cause is the small size of the strands, which reduces the effect of partial hybridization and self-hybridization. It is expected that for longer strands, the many-state model will still yield a better prediction. This will require new experiments, however, as to our knowledge no extensive data sets on DNA/DNA hybridization are available. It should be noted that the average deviation between prediction and experiment is rather large, in other words the total square error is 17.00 for 108 sequences used to fit 14 parameters. Fitting 17 parameters on 64 sequences, the total square error for RNA/DNA was only 5.26. Thus, it appears that for reasons unknown, DNA/DNA hybridization free energies are relatively hard to predict.

5.4 Inclusion of more parameters

5.4.1 Motivation

In the above fit, there were no special contributions for bulges and dangling ends, to keep the number of parameters low. This choice was made because adding bulge and dangling end parameters hardly affected the total square error. However, inclusion of these extra parameters in the fit turns out to cause a significant shift in some parameters, even if it does not significantly improve the quality of the fit. Thus, including the extra parameters and doing another fit is a good way to determine which parameters are reliable, and which parameters are unreliable. After all, adding a parameter will never make the fit worse, and in general will improve the fit.

Because of this, the fitting was done again with the most complete parameter set possible - including dangling ends, bulges, hairpins, internal and terminal A·T pairs, et cetera. It is expected that this procedure gives a good indication of parameter stability.

Finally, another important reason for performing this fit is that we will also use bulge and dangling end parameters when performing calculations on microarrays. In order to be able to fully compare these parameters to the ones used for microarray analysis, it is interesting to do a complete fit.

5.4.2 Comparison between the two parameter sets

The minimal fit, as described in Section 5.2.2, deviates significantly from literature values, while still being characterized by a lower square error. It is tempting to conclude that the literature values are therefore unreliable, but such a conclusion is certainly not justified by the results of this section, as we shall see that the two-state model is not defeated yet.

The results for the extended fit, which has had parameters for bulges and dangling ends added with respect to the minimal fit, are shown in Tables 4 and 5.

In Table 4, we see a striking similarity between the parameters of Gray [10] and the parameters from the extended fit. However, the parameters from the minimal fit are quite different from either of these parameter sets. This is, of course, not a coincidence: the bulge parameters have diverged to very negative values, and bulge effects are almost forbidden. However, without bulge effects, there is no partial unzipping, and as the hairpins also suffer a large penalty, the remaining many-state model is little different from the two-state model. It is rather surprising that, while it seems we have hardly changed anything, we still have managed to cut down the total square error from 7.12 to 5.79, an improvement of 19 %. Apparently, the fact that there still is the option for partial unzipping at the ends of a hybrid (because dangling ends are still allowed, as are terminal A·T pairs) remains an improvement with respect to the two-state model.

The move towards two-state model properties is also seen clearly in Table 5. First of all, it must be noted that the total square error for DNA/DNA hybridization, which was notable for being larger for the minimal fit, is now almost equal to the total square error found by Allawi *et al* [1], being only 2.6 % larger. A remarkable difference is the difference in the parameters for initiation and terminal A·T pairs.

doublet type	minimal	extended	[10]
AA/TT	-0.78	-0.73	-0.95
AC/TG	-0.91	-1.74	-1.76
AG/TC	-0.59	-1.44	-1.54
AU/TA	-0.51	-0.50	-0.71
CA/GT	-2.12	-1.30	-1.25
CC/GG	-1.88	-1.88	-1.92
CG/GC	-1.77	-1.79	-1.78
CU/GA	-1.73	-0.80	-1.07
GA/CT	-2.76	-1.89	-1.82
GC/CG	-2.66	-2.74	-2.62
GG/CC	-2.63	-2.64	-2.64
GU/CA	-2.27	-1.33	-1.36
UA/AT	-0.86	-1.03	-0.75
UC/AG	-0.39	-1.43	-1.28
UG/AC	-0.58	-1.52	-1.37
UU/AA	+0.09	+0.06	-0.34
ΔG_{init}	3.25	3.13	2.90
termAU	-0.79[a]	-0.03	-
intAU	0.98	-0.83[a]	-
bulge	-	7.63	-
dangling end	-	2.57	-
Error	5.26	5.79	7.12

Table 4: Comparison of RNA/DNA thermodynamic parameters for the minimal fit, as described in Section 5.2.2, and the extended fit, which also has parameters for bulges and dangling ends. For comparison, the results by Gray [10] have also been included. It is remarkable how the extended model resembles the results by Gray much more closely than the minimal model does. [a]The minus sign is remarkable, as this suggests that this type of A·U pairs is more stable than C·G pairs.

parameters	minimal	extended	parameters	minimal	extended	[27]
AA/UU	-0.86	-0.87	AA/TT	-0.88	-0.98	-1.00
AU/UA	-1.04	-1.03	AT/TA	-0.64	-0.91	-0.88
UA/AU	-1.27	-1.24	TA/AT	-0.62	-0.52	-0.58
CU/GA	-2.04	-2.04	CT/GA	-1.14	-1.23	-1.28
CA/GU	-2.05	-2.02	CA/GT	-1.58	-1.46	-1.45
GU/CA	-2.25	-2.22	GT/CA	-1.35	-1.48	-1.44
GA/CU	-2.39	-2.36	GA/CT	-1.42	-1.37	-1.30
CG/GC	-2.28	-2.26	CG/GC	-2.16	-2.15	-2.17
GG/CC	-3.23	-3.21	GG/CC	-1.81	-1.83	-1.84
GC/CG	-3.45	-3.42	GC/CG	-2.26	-2.29	-2.24
ΔG_{init}	4.16	4.03	ΔG_{init}	2.27	2.41	1.82
termAU	0.45	0.65	termAT	0.09	0.37	0.05
intAU	0.78	0.44	intAT	0.68	1.23	-
hairpin	4.54	9.89	hairpin	9.80	13.57	-
bulge	-	7.70	bulge	-	7.82	-
dangling	-	1.91	dangling	-	0.77	-
Error	8.32	7.95	Error	17.00	15.25	14.86

Table 5: Comparison of RNA/RNA and DNA/DNA thermodynamic parameters for the minimal fit and the extended fit. The three sets for RNA/RNA are rather similar, so the results by [27] were not presented here again. For DNA/DNA, the parameter set from the minimal fit is again somewhat different from the literature set and the set derived from the extended fit, so it is included for comparison. The only additions were bulges and dangling ends.

The shift in RNA/DNA parameters is also remarkable because it shows that the literature square error Q can be improved both with and without allowance of bulge formation. The total square error for RNA/DNA has increased somewhat in the extended model. This might seem odd - a degraded performance following the addition of new parameters - but this is likely a result of hairpins no longer being allowed. Moreover, the total square error for RNA/DNA was so low already, that the difference between 5.79 and 5.26 is smaller than some of the individual square errors.

On the other hand, we have improved the total square error for RNA/RNA hybridization some more, such that the improvement for the total square error here becomes 15 %.

The most notable effects of extending the fit are the disappearance of configurations with internal unzipping and hairpins. Apparently, by insisting that bulges should exist with no energy penalty (as was assumed in the minimal fit), RNA hairpins were made possible, at an energy penalty of 4.54. Nonetheless, the hairpins are gone in the extended fit, as the energy penalty of about -10 kcal/mol is so high that no configuration with a hairpin can contribute significantly to the partition sum. The same is true for the bulges - even though they have not diverged to $-\infty$, they play no meaningful role at energy penalties of around 8 kcal/mol. Because there are no bulges, the internal A·T parameters have also become obsolete, and although they are presented in the tables for completeness, they cannot be expected to have any physical meaning.

These considerations do however increase our confidence that our model should be an improvement of the two-state model. The similarity between our results and those known for the two-state model, are an indication that the extra parameters are meaningful, as they do decrease the square error, but they do not shift the parameters in the original model much. Even though we have not yet managed to establish the importance of self-hybridization or internal unzipping, the current improvements seem to imply that including more configurations in the partition sum is worth some effort.

5.4.3 Values for ΔH and ΔS

In order to provide a complete overview of thermodynamic parameters, the extended fit was repeated a number of times to also generate the corresponding results for ΔH and ΔS. More precisely, a range of temperatures was covered in the fit, where for each temperature, the ΔG value for each parameter was fitted. The ΔS and ΔH values can be derived from a linear fit to those values, as ΔS is the slope and ΔH the cutoff value at zero temperature of the fitted line.

A fit has been made for the temperatures $T = 27, 29, 31, \ldots, 47\,^{\circ}\mathrm{C}$. This leads to 11 complete sets of parameters, such that each parameter has a certain value of ΔG at each temperature. This allows us to also determine ΔS and ΔH for each parameter, by performing a linear fit through these eleven data points.

The values for the bulges have become extremely small, and their corresponding values for ΔH and ΔS are of no significance. The values of ΔH and ΔS for the internal A·T and the hairpins, are somewhat unreliable, but unlike the bulges they do seem to have a serious effect on the fit. Finally, the effects of dangling ends appeared to be all but discarded by the fit, as the corresponding parameters were rather small. However, the parameters for DNA/DNA and RNA/DNA both seem to have at least some effect, since DNA/DNA has only a dangling end penalty of 0.77 at 37 °C, and RNA/DNA has a dangling end penalty of 2.57 but an internal A·T bonus of -0.83, such that a penalty of only 1.74 is taken into account for a configuration which has a single unzipping. Whether this is a physically relevant point remains questionable, it is still well possible that there are simply too many parameters on too few data points, such that some parameters do not correspond to physical effects.

The only bulge formation in the current data set, is by internal unzipping of a perfect hybrid (a hybrid formed by hybridization of two mutually complementary strands), but these effects are likely also contained in the nearest-neighbor pair parameters. To estimate bulges, and the internal A·T parameters that also depend on bulges, it would be better to have hybridization of pieces of DNA and/or RNA that are not perfectly complementary. However, this type of hybridization is scarcely present on short strands such as the ones we currently use in our fitting, and it is thus not very surprising that the parameters fluctuate wildly. It is likely that the actual improvement of the fit is only due to the fact that the extra parameters allow the exclusion of certain states, and is not because the parameters are physically relevant.

doublet type	ΔH [27] (kcal/mol)	ΔH (this work) (kcal/mol)	ΔS [27] (cal mol^{-1} K^{-1})	ΔS (this work) (cal mol^{-1} K^{-1})
AA/UU	-6.82	-7.30	-19.0	-20.7
AU/UA	-9.38	-9.57	-26.7	-27.6
UA/AU	-7.69	-7.47	-20.5	-20.1
CU/GA	-10.48	-9.87	-27.1	-25.3
CA/GU	-10.44	-9.79	-26.9	-25.1
GU/CA	-11.40	-11.62	-29.5	-30.3
GA/CU	-12.44	-12.51	-32.5	-32.7
CG/GC	-10.64	-9.54	-26.7	-23.5
GG/CC	-13.39	-12.84	-32.7	-31.0
GC/CG	-14.88	-15.66	-36.9	-39.5
ΔG_{init}	3.61	-1.56	-1.5	-18.0
termAU	3.72	0.00	10.5	-2.1
intAU	-	-8.78	-	-29.8
hairpin[a]	-	-11.88	-	-67.1
dangling end	-	-0.78	-	-9.0

Table 6: Parameters for ΔH and ΔS for RNA/RNA. [a] The ΔG parameter for hairpins was very nonlinear as a function of temperature, varying from -7.1 at 27 °C, to -10.5 at 39 °C, to -8.5 at 47 °C. While other parameters often are not exactly linear, deviations this strong are not seen in other parameters. The nonlinearity means, however, that ΔH and ΔS variables make little sense.

doublet type	ΔH [1] (kcal/mol)	ΔH (this work) (kcal/mol)	ΔS [1] (cal mol^{-1} K^{-1})	ΔS (this work) (cal mol^{-1} K^{-1})
AA/TT	-7.9	-7.50	-22.2	-21.0
AT/TA	-7.2	-7.49	-20.4	-21.3
TA/AT	-7.2	-5.56	-21.3	-16.2
CT/GA	-7.8	-8.12	-21.0	-22.2
CA/GT	-8.5	-8.54	-22.7	-22.8
GT/CA	-8.4	-9.55	-22.4	-26.0
GA/CT	-8.2	-8.17	-22.2	-21.9
CG/GC	-10.6	-10.48	-27.2	-26.9
GG/CC	-8.0	-8.89	-19.9	-22.8
GC/CG	-9.8	-10.23	-24.4	-25.6
ΔG_{init}	0.2	-3.28	-5.6	-18.4
termAT	4.8	0.87	13.8	1.6
intAT	-	-12.39	-	-44.0
hairpin^a	-	-73.85	-	-281.8

Table 7: Parameters for ΔH and ΔS for DNA/DNA. ^{a} The hairpin parameters were strongly nonlinear, see also the remark at Table 6.

doublet type	ΔH [10] (kcal/mol)	ΔH (this work) (kcal/mol)	ΔS [10] (cal mol^{-1} K^{-1})	ΔS (this work) (cal mol^{-1} K^{-1})
AA/TT	-6.62	-5.80	-18.3	-16.4
AC/TG	-7.82	-7.66	-19.5	-19.1
AG/TC	-7.27	-8.03	-18.5	-21.2
AU/TA	-7.12	-6.95	-20.7	-20.8
CA/GT	-7.46	-7.77	-20.0	-20.9
CC/GG	-6.06	-6.68	-13.4	-15.5
CG/GC	-12.37	-12.34	-34.1	-34.1
CU/GA	-3.13	-2.82	-6.7	-6.5
GA/CT	-9.24	-8.15	-23.9	-20.2
GC/CG	-10.29	-10.25	-24.7	-24.2
GG/CC	-10.74	-11.63	-26.1	-29.0
GU/CA	-8.81	-7.56	-24.0	-20.1
UA/AT	-9.50	-10.47	-28.2	-30.5
UC/AG	-8.09	-8.90	-21.9	-24.1
UG/AC	-7.59	-8.20	-20.1	-21.5
UU/AA	-4.86	-5.10	-14.6	-16.7
ΔG_{init}	1.9	-6.87	-3.9	-32.2
termAU	-	0.00	-	0.09
intAU	-	-19.75	-	-61.1

Table 8: Parameters for ΔH and ΔS for RNA/DNA.

6 The many-state model and microarrays

In the previous chapters of this thesis, we have introduced the many-state model for partition sums in DNA and RNA hybridization. We have subsequently created algorithms for this model, tested these algorithms, and used them to create a new parameter set.

These results now allow us to return to the microarrays that we have introduced in the first chapter. Recall that we introduced the many-state partition sum because it appeared to be a much better approximation of the actual physics of the system. It thus seems a logical conclusion to this research to end with a chapter on the analysis of microarrays using the tools that have been developed.

6.1 Testing the many-state model on microarrays

6.1.1 Comparison of the two-state and many-state models

In the following, we want to compare the two-state model and the many-state model, to determine which model is better suited for analysis of microarrays. To allow for a good comparison, each model should have an algorithm associated with it, that can be used to estimate the hybridization energies ΔG.

For the two-state model, $\Delta G_{\text{free}} = 1$ for a single, unhybridized strand, while ΔG_{hybr}, the hybridization free energy for two fully hybridized strands, is equal to a sum over the nearest-neighbor model parameters corresponding to the base pair doublets found on the strand, as seen in Section 2.2. For the many-state model, the values of ΔG are computed using one of the algorithms from Section 3.

Both models lead eventually to a certain estimate of ΔG_{eff},[35] the effective stacking free energy for a hybrid in solution. The concentration c_s of the RNA corresponding to probe s can be estimated from the measured intensity I and ΔG_{eff}, from

$$c_s = \frac{C}{\alpha Z_s \left(\frac{A}{I_{\text{m}}} - 1\right)}, \tag{9}$$

where I_{m} is the measured intensity for the probe, C is a proportionality constant, A is a normalization factor for the intensity,[36] and Z_s is the partition sum given by

$$Z_s = 1 + e^{\Delta G_{\text{eff}}}. \tag{10}$$

Here, α is a fitting parameter to compensate for target-target hybridization,

$$\alpha = \frac{1}{1 + c_0 e^{\beta' \Delta G_{\text{hybr}}}}, \tag{11}$$

where we use the assumption that the likeliness of target-target hybridization is dependent on the sequence. A large target-probe hybridization free energy means that it is likely that the target is also highly likely to bind to other targets,

[35]For the two state model, $\Delta G_{\text{eff}} = \Delta G_{\text{hybr}}$. However, when we take into account the self-hybridization for the many-state model, it is reasonable to use $\Delta G_{\text{eff}} = \frac{\Delta G_{\text{hybr}}}{\Delta G_{\text{free,DNA}} \Delta G_{\text{free,RNA}}}$, such that we find that the self-hybridization free energies serve as a sequence-dependent normalization.

[36]Because we are usually far away from saturation, the -1 can be ignored, and A may be absorbed into C.

which might decrease the available concentration of RNA. The definition of α contains two parameters, c_0 and β', which should be fitted. See also Ref. [11] for a more detailed explanation of α. Note, however, that their α also may account for hairpin formation, which we have already covered, such that it is likely that our α will have a smaller effect on the accuracy of our predictions. However, it was found that calculations without any use of α were significantly less accurate than calculations using α.

When we compare the fitted concentration to the actual experimental concentration, the difference serves as an estimate of the quality of the model used to estimate ΔG_{eff}.

6.1.2 Thermodynamic parameters for microarrays

The first thing to be done, is to determine what parameters to use for the many-state model. We have developed a set of parameters for RNA/DNA hybridization in Section 5, but these were for 37 °C, while the microarray experiments we want to consider, are performed at 45 °C. We have also determined the ΔH and ΔS values, so we might use the thermodynamic relation from eq. (8) to determine a value ΔG at 45 °C. Alternatively, we can fit the experiments from Section 5 at 45 °C directly, by calculating the experimental ΔG values for 45 °C for each sequence. Both methods lead to a possible parameter set.

Another possibility is to fit a new parameter set using the experimental data available for microarrays. There are various advantages to this approach. First of all, it is quite possible that there are differences between the experimental setups, that are unknown to us. The experiments used for Section 5 were done under well-known conditions, using only one or two DNA sequences. The microarray experiments, on the other hand, have a completely different setup. The RNA hybridizes to DNA that is mounted on a chip, and the RNA and DNA on microarrays is much longer than those in section 5. These and other considerations motivate us to perform a new fit on microarray experiments.

Before we go into further detail regarding to the fitting, let us introduce the set of experimental data that we shall be using in our calculations.

6.2 The Latin Square data set

6.2.1 Introduction to the Latin Square experiments

To encourage research being conducted on microarrays, microarray producer Affymetrix, based in Santa Clara, California, United States, released experimental data from a so-called *latin square* series of experiments. These are *spike-in* experiments, in which known concentrations of certain strands of RNA are added to a natural biological background.

In the case at hand, fourteen distinct experiments were carried out, and each experiment was repeated three times. There were 42 different *probe sets* (a probe set represents an RNA strand of several hundred nucleotides long, as is common for protein-coding RNAs), and each probe set has had about a dozen intervals of 25 positions selected, whose complements appear on the microarray. Consequently, every one of the 42 probe sets is represented at about 11 positions on the microarray. This means that all 11 *probes* (DNA strands of length 25 on the microarray chip) are expected to measure the same concentration in the ideal case. Of course, uncertainties and errors of various kinds will blur the data, such that a range of different concentration estimates will be found. One of the challenges in microarray analysis, is to select which concentration estimates should be included.

In a given experiment, a certain probe set will be added to the experiment in either one of 14 possible concentrations. The concentrations are in the range $\{0, 0.125, 0.25, 0.5, 1, \ldots, 128, 256, 512\}$ picomolar, and the 42 probe sets are distributed evenly over these concentrations, so that there are three different probe sets at a given concentration. The concentrations are rotated cyclically, such that every probe set is found at every possible concentration in a single set of three experiments.

The data obtained after performing an experiment, is in the form of intensities. This means that a laser is used to detect RNA at the DNA probes. Thus, we shall use our many-state model to estimate the relationship between measured intensity and the concentration.

6.2.2 Choosing the right probes to use for analysis

One of the least intuitive problems is to select the probes that are thought to represent believable values. We would like to derive an independent parameter set for microarray analysis, as temperatures and circumstances are radically different from the experiments seen in chapter 5. However, the fit was found to be complicated, and it took some effort to get a realistic parameter set. Only by carefully discarding certain probes, a physically believable set of parameters was obtained.

There are various ways to decide whether the result found at a probe is likely to give a good estimate of the concentration. We shall give a short discussion of every method used to discard probes.

Mismatch intensities. In each experiment, we find both the intensity of the DNA probe, and that of its mismatch twin, which has an altered central (13th) nucleotide. As the partition sum for the hybridization of mismatches and RNA targets is significantly lower than the partition sum for perfect matches and RNA targets, a measurement is considered to be unreliable if the mismatch intensity is close to or exceeding the perfect match intensity. In our fit, we have decided to discard probes for which the mismatch intensity was in excess of 70 % of the perfect match intensity.

Low concentration. As there is plenty of data available, it seems an unnecessary complication to try and predict very low concentrations. Thus, only the top 4 concentrations (64, 128, 256 and 512 picomolar) were used in the fitting procedure, as they are most significant with respect to the background intensities, and thus most likely to be accurate data points.

Median-based sifting. We expect to have 11 probes left for each probe set, although a very high mismatch intensity might have sized down our probe set somewhat. They are expected to all yield the same concentration estimate. Thus, the decision was made to discard the highest two and the lowest two values, such that only the intermediate estimates remained. This means that extreme outliers are not included in the fit, as our fitting method is a least-squares fit, and a small number of outliers would dominate the total square error, making a reliable fit impossible.

Weighed errors. If a probe set, for some reason, has lost several probes because of high mismatch intensities, it is probable that something is just plain wrong with the fit. Thus, the square error was weighed by the size of the list (usually this is just 7) of probes whose perfect match intensity was significantly higher than their corresponding mismatch intensity.

At this point, in most cases, 7 probes are still left for each probe set, which means that a single experiment still contains $7 \cdot 3 \cdot 4 = 84$ data points. Fitting on 14 experiments thus yields 1176 data points, which is more than enough for us. There are at most 50 parameters or so in our fit, and we actually have about four times more data than we had in the parameter set we derived earlier in this thesis.

6.3 Fitting the many-state model for microarrays

When we have done this pre-selection of probes, we are ready to estimate concentrations for the probes that still remain. The fit is similar to the fit described in chapter 5. Again, we develop a measure Q which represents the total deviation. In this case, this is done by taking the square of the difference between the logarithms of the estimated concentration and spike in concentration:

$$Q = \sum m_s (\log \hat{c}_s - \log c_s)^2, \tag{12}$$

where we have defined the concentration $\hat{c}_s$ as the spike-in concentration of RNA targets complementary to probe s. We have also introduced m_s as the number of probes available for the probe set to which probe s belongs. This weight factor will only affect Q when a certain probe set is missing members. If only the probe sets with a high spike-in concentration are considered (see the preceding Section 6.2.2), m_s is rarely lower than its maximum of 7, and the inclusion of m_s barely affects the fitting procedure.

6.3.1 Parameters to be fitted

First let us decide which parameters shall be fitted. As it is expected that using extrapolated values from the parameters of Section 5 is insufficient, we shall try to fit every parameter encountered in our model. However, because there still are only a few hundred data points available, we shall only fit a few general bulge, hairpin and dangling end parameters. This means that we are introducing a significant error in our model. However, it is expected that the resulting model will still outperform the two-state model. The parameters included in either model are presented in Table 9.

6.3.2 Testing for overfitting of the data set

One of the dangers we have to anticipate, is overfitting. Overfitting is the fitting of too many variables on too few data points, such that some of our parameters are not physically relevant, but serve only to bring down the error. This is the case when, for example, we introduce a parameter which is only used in a single probe. The parameter would bring down the total square error by decreasing

parameter type	many-state	two-state
Nearest-neighbor pairs	36	16
Initiation	0 (3)	0 (1)
Initiation with dangling ends	3	0
Hairpins	2	0
Bulge formation	3	0
Terminal A·T pairs	3	0
Internal A·T terminal pairs	3	0
β (Inverse temperature)	1	1
C (Proportionality constant)	1	1
β' (Fitting parameter of α)	1	1
c_0 (Fitting parameter of α)	1	1
Total	54	20

Table 9: An overview of the parameters to be fitted on the Latin Square data set, for both the many-state model and the two-state model. The number of initiation parameters is decreased because of redundancy: the hairpins of DNA and RNA may absorb the DNA/DNA and RNA/RNA initiation parameters, and the only resulting initiation parameter for RNA/DNA may be absorbed into the proportionality constant C. This absorption is indicated by the number in brackets following the number of initiation parameters. On the other hand, there are three distinct initiation penalties for dangling ends.

	Two-state model	Many-state model
Q after fitting on 1-4	148.4	127.16
Q for Expt. 8-11	230.8	228.0
Q for Expt. 1-14	702.3	658.6

Table 10: Estimating the degree of overfitting when only four experiments are used in the fitting. The values of the total square error, Q, are given after a least-squares fit on the experiments 1-4. The value of Q for experiments 8-11 was calculated without any fitting, using only the parameters calculated in the fit on the experiments 1-4.

the error corresponding to the probe. However, the parameter does not help in predicting any other concentration.

This shows that we will have to check whether our parameters are still helpful in predicting concentrations, when they are used to predict concentrations of probes other than the probes that were used in the fitting procedure.

To estimate the relative importance of overfitting, the parameter set for each model was fit only on the experiments 1,2,3 and 4. Then, the parameter set was used directly to estimate the concentrations for the experiments 8,9,10 and 11, which do not have any common data points with the experiments 1,2,3 and 4 used in the fit.

The results are shown in Table 10. We immediately see that we are indeed overfitting. While the many-state model appears to give a much better fit, the difference is all but gone when we apply the model on a different set of probes. Although this does not necessarily mean that there is no physical relevance in

	Two-state model	Two-state refit	Many-state model	Many-state refit
Q after fitting on 1-4	148.4	167.7	127.1	151.6
Q for Expt. 8-11	230.8	217.7	228.0	197.9
Q for Expt. 5-7 and 12-14	323.1	311.6	303.5	278.9
Q for Expt. 1-14	702.3	697.0	658.6	628.4

Table 11: Comparison between the performance of the original fit and the refit. Again, the values of the total square error, Q, are given after a least-squares fit on the experiments 1-4. The value of Q for experiments 8-11 was calculated without any fitting, using only the parameters calculated in the fit on the experiments 1-4. The values in the refit were calculated by using only a very small number of iterations in the fitting procedure, cutting off the fit even while the fit has not yet reached the optimal point. While the fits on the experiments 1-4 are of a lower quality, the verification experiments 8-12 are predicted better by the parameters from the interrupted refit.

the parameters we have found, it certainly provides a clear warning that the parameters are not reliable.

During fitting, it became clear that the fit converged rapidly up a certain point, after which it continued to converge at a very low pace. It was conjectured that the point where rapid convergence ended, $Q \approx 150$ for the many-state model, was the point where the parameters were fitted after noise effects and no longer corresponded to physical effects. The choice was made to cease the fit fairly early, and see whether this reduced the severity of overfitting.

The results presented in Table 11 show that indeed the overfitting effects are much less pronounced when the fitting was stopped at an early time. While the two-state model shows a moderate improvement, it is primarily the many-state model that benefits from this early stopping. At this time, we also see confirmed that the many-state model is an improvement when compared to the two-state model, as the many-state model results on the verification experiments 8-11 outperform the two-state model by 19.8, or about 9.1 %. Because we have not used any of these data in our fit, the overfitting effect caused by the extra parameters cannot account for this decrease in Q. The fact that there are about a hundred data points in each set ($7 \cdot 4 \cdot 4 = 112$) also appears to indicate that the deviation is larger than might have been expected from random variations. More so, for the refit, it should be noted that in the fit on experiments 1-4 (9.6 %), the verification on experiments 8-12 (9.1 %), and in the fit on the remaining six experiments (10.5 %), the performance of the many-state model is repeatedly better than the two-state model performance by about 10 %. This also suggests that the many-state model is indeed an improvement with respect to the two-state model.

6.3.3 Parameter sets for calculations on microarrays

The refits from the preceding section, which yielded the best fit to the verification data set, lead to the following parameter sets for microarray analysis. Because of the nature of the problem, values for ΔH and ΔS cannot be derived, and only values for ΔG are given.

Shared parameters			*Many-state parameters*			
RNA/DNA parameter	many-state	two-state	RNA/RNA parameter	many-state	DNA/DNA parameter	many-state
AA/TT	-0.21	-0.83	AA/UU	-1.21	AA/TT	-1.10
AC/TG	-3.44	-3.15	AU/UA	-0.43	AT/TA	+0.32
AG/TC	-1.24	-1.40	UA/AU	-0.23	TA/AT	-0.17
AU/TA	-1.77	-2.22	CU/GA	+0.96	CT/GA	+0.25
CA/GT	-1.19	-1.09	CA/GU	-2.23	CA/GT	-1.62
CC/GG	-0.05	-0.19	GU/CA	-0.39	GT/CA	-0.52
CG/GC	-1.34	-1.97	GA/CU	-1.14	GA/CT	-0.54
CU/GA	-1.63	-2.83	CG/GC	-0.05	CG/GC	-0.79
GA/CT	-1.19	-1.32	GG/CC	-2.14	GG/CC	-0.64
GC/CG	-3.26	-2.31	GC/CG	-4.61	GC/CG	-0.97
GG/CC	-3.10	-3.16				
GU/CA	-4.34	-4.82				
UA/AT	+1.15	+1.45				
UC/AG	+1.28	+1.68				
UG/AC	+1.08	+1.65				
UU/AA	+0.67	+0.10				

Table 12: The parameters found after the fitting of both the two-state and the many-state nearest-neighbor model on the data of the Latin Square microarray experiments. The RNA/DNA parameters, which are shared by both sets, are somewhat mutually similar. None of the parameters is very much like the parameters found in Section 5, so extrapolated values from those parameter sets were not included. The fitting parameters C, β, c_o and β' were not included in the table.

Effect	RNA/DNA	RNA/RNA	DNA/DNA
hairpin	-	3.04	4.52
bulge[a]	7.56	5.88	5.66
dangling end	-0.33	-0.05	0.57
terminal A·T	-1.00	0.93	0.84
internal A·T	-0.74	1.00	1.10

Table 13: The remaining parameters from the many-state model, for bulge, hairpin, A·U and dangling end effects. [a]Unfortunately, the bulge parameters depend quite a bit on their initial values. This is an unfortunate result of our solution for the overfitting, and the bulge values may be up to 2 kcal/mol off their currently listed values.

In these results, Tables 12 and 13, several unexpected observations show up. For example, RNA/DNA parameters for terminal and internal A·T pairs are negative, which means that A·T base pairs appear to be more stable connections than C·G base pairs. This is contrary to known results from the literature [27], and in general there is little doubt that C·G base pairs are more stable than A·T base pairs. It remains to be determined whether this is solely a coincidence, or a trace of some physical effect we have not yet understood.

The hairpin parameters are significantly present. Whereas the parameters were almost meaningless in the parameter fits in Section 5, their current values are roughly according to expectations, as they are indeed in the range of hairpin parameters known from the literature [3]. Because of the lengths of $n = 25$ of strands in the current data set, hairpins indeed do contribute, primarily via self-hybridization and the corresponding decrease in availability in solution.[37]

Unfortunately, the bulge parameters remain large. When the bulge parameters were set at 25 each, the total square error instantly increased by about 15 %, which indicates that the bulge parameters are not completely irrelevant. However, it seems odd that the bulge parameters should be higher than initiation parameters. After all, the initiation corresponds to two strands finding each other in spite of entropy. Suppose however that these two strands were already hybridized to each other at another point, somewhere else on the strand. It would seem that the two strands then would be more likely to hybridize also at another point. The fact that this does not happen, is perhaps something worthy of further research. One possible explanation is that it is very hard for a double helix not to form base pairs between a pair of nucleotides when base pairs are present on either side, but at this point we have no results at our disposal to back up this conjecture.

The dangling ends, meanwhile, seem not to have a strong preference to be either above or below zero. Values from experiments for dangling ends [6] show a wide spread in parameters, so we cannot assign much meaning to these single parameters, which are tremendous simplifications with respect to the many parameters in Ref. [6].

Finally, considering the parameters for the base pair doublets, it is quickly seen that the parameters are not at all near the extrapolated values at 37 °C. The two-state model even has a parameter, GU/CA, at -4.82 kcal/mol. The many-state model's strongest parameter is RNA/RNA GC/CG at -4.61 kcal/mol. These are a lot larger than the parameters found in Section 5, and in fact they are so large that it is hard to believe they can be real. In addition, lacking material for comparison, it is difficult to say much about the parameter sets. Obviously, because of our early termination of the fitting procedure, different initialization of parameters will lead to different sets of parameters. In fact, we initialized the variables at the values that can be found from the ΔH and ΔS values from section 5. Nothing of these values remains, however, as the values are completely different in Tables 12 and 13.

One last finding worth some attention is the similarity between the two-state and many-state parameter sets for the RNA/DNA base pair doublet parameters. Even though they are not quite the same, it cannot be denied that there is some similarity between both sets, indicating that the many-state model and the two-

[37]That is to say, it is known that in microarray experiments, much of the RNA is hybridized in solution and not available for hybridization on probes.

state model remain related, and that the parameters are still a representation of physical properties of DNA and RNA hybridization on microarrays.

7 Conclusion and summary

In this thesis, we have extended the nearest-neighbor model for DNA and RNA hybridization. The common two-state assumption is not justified for the lengths encountered in microarray analysis. Thus, we have created a many-state model, which takes into account many currently known configurations not yet covered by the two-state model. The configurations we have not taken into account, such as those with kissing hairpins, are expected not to have a significant impact on the partition sum, as those effects are only expected to become important at lengths much larger than the strands (of lengths ranging from 20 to at most 60) that are known from microarrays. However, throughout the course of this thesis, we have repeatedly seen confirmation that the many-state model gives better results than the two-state model.

To perform computations using this model in an efficient way, we have first developed an algorithm to carry out the necessary computations. This algorithm, which we called the *Rainbow algorithm*, was developed as an order-n^3 algorithm, using the assumption that the partition sum can be split into two independent parts. We then discussed several improvements that could be made to make the algorithm more efficient, and that would help skip redundant checks and computations.

There turned out to be several candidate algorithms. Each of the candidates was implemented and tested. Results indicated that simple iterative algorithms are most efficient for short lengths. However, at these lengths, there is virtually no hairpin or bulge formation, and the effect of partial unzipping is only minor. For the lengths at which hairpins and bulges are known to form, it was found that a hybrid algorithm, using both iterative and recursive strategies, performs best. The differences between the algorithms were found to be quite significant, with a speedup factor of 7 for the hybrid algorithm with respect to the iterative algorithm at intermediate lengths, $n \approx 50$.

With an efficient algorithm at our disposal, the decision was made to first test the new model on existing literature values. For this purpose, the model was implemented in a fitting program that computed a new set of parameters for the nearest-neighbor model. Because implementation of bulges and dangling ends was somewhat complicated, these parameters were skipped in the early fitting. However, to be fully compatible with the parameter set to be used for microarrays, the bulge and dangling end parameters were eventually included. The significance of the removal of bulges was somewhat surprising, and caused a shift in the other parameters. This shift might have been thought of as an overfitting effect, if it were not for the striking similarity between the new set and the two-state model sets from the literature. However, the two sets still represent two different approaches that both seem quite able to predict DNA and RNA hybridization, but that have parameters that lie far apart, especially where RNA/DNA parameters are concerned. However, both parameter sets also resulted in a total square error that was much lower than previously known. It is hardly possible to say anything about the parameters, except that the parameters for RNA/DNA hybridization remain uncertain. Nonetheless, the improvement does seem to suggest that the many-state model is more accurate in predicting DNA and RNA hybridization than the two-state model.

Finally, the algorithm was used in a new fit program to predict concentrations for microarrays. Without the availability of a parameter set, we have fitted some of the parameters. The initial guess, being only an extrapolation to 45 °C of parameters known for 37 °C, turned out to perform poorly. Thus, it was decided to fit all parameters once more. As there were another four parameters related to concentration estimation, this brought the total number of parameters to 54 for the many-state model. Fits were done using these parameters, but the results were suspicious, so tests were done to check for overfitting. For comparison, the two-state model was also implemented and fitted, using only 20 parameters. Overfitting was indeed found, in serious amounts for the two-state model, but the amount of overfitting in the many-state model fit was so severe that it was deemed necessary to acquire a set of parameters using a different method of fitting. It turned out that cutting the fit short was sufficient to remove much of the overfitting effects. While it is inevitable that some overfitting effects remain, the results were sufficient to conclude that the many-state model indeed improves upon the two-state model when used in microarray analysis.

7.1 Outlook for future research

The results of this thesis confirm that the many-state model improves calculations on DNA and RNA hybridization. This raises a number of new questions and problems. First of all, it would be desirable to do fits on larger data sets to gather more decisive evidence on the parameters of the model. Furthermore, there are many different parameters for bulges and hairpins, and when more data is available, this also opens up possibilities for inclusion of all of these parameters, rather than a single parameter for 'all bulges in DNA/DNA hybridization', for instance. After all, bulges take many different forms, and the nucleotide content of the bulges can be expected to play a major role in the tendency of a bulge to exist. With or without these additional parameters, it is expected that the many-state model we have developed will help future analysis of microarrays.

The model is applicable to many other problems in DNA and RNA hybridization, too. A lot of research worldwide deals with computations on hybridization, such as RNA folding, and it is very well possible that the many-state model will improve results in many different situations.

Acknowledgements

First and most of all, I would like to thank my supervisors, Gerard Barkema of the Institute for Theoretical Physics and Rob Bisseling of the Mathematics Institute for their support and guidance, and for many useful discussions and valuable advice. I consider myself lucky to have been writing my thesis under their supervision.

Additionally, I want to thank Enrico Carlon and his group from the K.U. Leuven. Also I want to thank Geert Mulders, another student of the University of Utrecht who also wrote his thesis on microarray analysis, and who has helped me in getting started.

Furthermore, I would like to thank my friend and co-student Bas Fagginger Auer for a lot of feedback on my programming and computational problems, and also for giving me the code of his Mersenne Twister-based random number generator. Additionally, I do not think I could have worked so hard on this thesis if I would not have seen him working even harder on his thesis.

References

[1] H. T. Allawi and J. SantaLucia, Jr, *Thermodynamics and NMR of internal G.T mismatches in DNA*, Biochemistry, 36 (1997), pp. 10581 – 10594.

[2] M. Andronescu, A. Condon, H. H. Hoos, D. H. Mathews, and K. P. Murphy, *Efficient parameter estimation for RNA secondary structure prediction*, Bioinformatics, 23 (2007), pp. i19 – i28.

[3] V. P. Antao and I. Tinoco, *Thermodynamic parameters for loop formation in RNA and DNA hairpin tetraloops*, Nucleic Acids Research, 20 (1992), pp. 819 – 824.

[4] T. W. Barnes and D. H. Turner, *C5-(1-propynyl)-2'-deoxy-pyrimidines enhance mismatch penalties of DNA:RNA duplex formation*, Biochemistry, 40 (2001), pp. 12738 – 12745.

[5] S. H. Bernhart, I. L. Hofacker, and P. F. Stadler, *Local RNA base pairing probabilities in large sequences*, Bioinformatics, 22 (2006), pp. 614 – 615.

[6] S. Bommarito, N. Peyret, and J. SantaLucia, Jr, *Thermodynamic parameters for DNA sequences with dangling ends*, Nucleic Acids Research, 28 (2000), pp. 1929 – 1934.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press and McGraw-Hill, 2nd ed., 2001.

[8] J. M. Diamond, D. H. Turner, and D. H. Mathews, *Thermodynamics of three-way multibranch loops in RNA*, Biochemistry, 40 (2001), pp. 6971 – 6981.

[9] C. B. Do, D. A. Woods, and S. Batzoglou, *CONTRAfold: RNA secondary structure prediction without physics-based models*, Bioinformatics, 22 (2006), pp. e90 – e98.

[10] D. M. Gray, *Derivation of nearest-neighbor properties from data on nucleic acid oligomers. II. Thermodynamic parameters of DNA · RNA hybrids and DNA duplexes*, Biopolymers, 42 (1997), pp. 795–810.

[11] T. Heim, J. K. Wolterink, E. Carlon, and G. T. Barkema, *Effective affinities in microarray data*, Journal of Physics: Condensed Matter, 18 (2006), pp. S525–S536.

[12] G. Held, G. Grinstein, and Y. Tu, *Modeling of DNA microarray data by using physical properties of hybridization*, Proc. Natl. Acad. Sci. USA, 100 (2003), pp. 7575 – 7580.

[13] I. Hofacker, W. Fontana, P. Stadler, L. Bonhoeffer, M. Tacker, and P. Schuster, *Fast folding and comparison of RNA secondary structures*, Monatshefte fur Chemie, 125 (1994), pp. 167 – 188.

[14] D. J. Fish et al., *DNA multiplex hybridization on microarrays and thermodynamic stability in solution: a direct comparison*, Nucleic Acids Research, 35 (2007), pp. 7197 – 7208.

[15] A. Jacobson, L. Good, J. Simonetti, and M. Zuker, *Some simple computational methods to improve the foldings of large RNAs*, Nucleic Acids Research, 12 (1984), pp. 45 – 52.

[16] J. A. Jaeger, D. H. Turner, and M. Zuker, *Improved predictions of secondary structures for RNA*, Proc. Natl. Acad. Sci. USA, 86 (1989), pp. 7706–7710.

[17] J. S. Kim, J.-W. Lee, Y.-K. Nog, J.-Y. Park, D.-Y. Lee, K.-A. Yang, Y. G. Chai, J. C. Kim, and B.-T. Zhang, *An evolutionary Monte Carlo algorithm for predicting DNA hybridization*, Biosystems, 91 (2008), pp. 69 – 75.

[18] B. E. Lang and F. P. Schwarz, *Thermodynamic dependence of DNA/DNA and DNA/RNA hybridization reactions on temperature and ionic strength*, Biophysical Chemistry, 131 (2007), pp. 96 – 104.

[19] M. Matsumoto and T. Nishimura, *Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator*, ACM Transactions on Modeling and Computer Simulation, 8 (1998), pp. 3–30.

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

[21] J. SantaLucia, Jr, H. T. Allawi, and P. A. Seneviratne, *Improved nearest-neighbor parameters for predicting DNA duplex stability*, Biochemistry, 35 (1996), pp. 3555 – 3562.

[22] S. J. Schroeder and D. H. Turner, *Thermodynamic stabilities of internal loops with GU closing pairs in RNA*, Biochemistry, 40 (2001), pp. 11509 – 11517.

[23] S. Smit, K. Rother, J. Heringa, and R. Knight, *From knotted to nested RNA structures: a variety of computational methods for pseudoknot removal*, RNA, 14 (2008), pp. 410 – 416.

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

[25] N. Sugimoto, S. Nakano, M. Yoneyama, and K. Honda, *Improved thermodynamic parameters and helix initiation factor to predict stability of DNA duplexes*, Nucleic Acids Research, 24 (1996), pp. 4501 – 4505.

[26] F. Tanaka, A. Kameda, M. Yamamoto, and A. Ohuchi, *Thermodynamic parameters based on a nearest-neighbor model for DNA sequences with a single-bulge loop*, Biochemistry, 43 (2004), pp. 7143 – 7150.

[27] T. Xia, J. SantaLucia, Jr, M. E. Burkard, R. Kierzek, S. J. Schroeder, X. Jiao, C. Cox, and D. H. Turner, *Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson-Crick base pairs*, Biochemistry, 37 (1998), pp. 14719 – 14735.

[28] X. Ying, H. Luo, J. Luo, and W. Li, *RDfolder: a web server for prediction of RNA secondary structure*, Nucleic Acids Research, 32 (2004), pp. W150 – W153.

[29] M. Zuker, *On finding all suboptimal foldings of an RNA molecule*, Science, 244 (1989), pp. 48 – 52.