Nucleic Acid Sequence Design As A Graph Colouring Problem

DISSERTATION
zur Erlangung des akademischen Grades
Doktor rerum naturalium

Vorgelegt der
Fakultät für Chemie
der Universität Wien

von
Mag. Ingrid G. Abfalter

am Institut für Theoretische Chemie und Molekulare Strukturbiologie

im November 2005

Dank an alle,

die zum Gelingen der vorliegenden Arbeit beigetragen haben:

Peter Stadler, Christoph Flamm, Ivo Hofacker, Peter Schuster.
Stephan Bernhart, Lukas Endler, Claudia Fried, Kurt Grünberger, Jörg Hackermüller, Ulli Langhammer, Rainer Machne, Ulli Mückstein, Stefan Müller, Sonja Prohaska, Bärbel Stadler, Camille Stephan-Otto Attolini, Roman Stocsits, Andreas Svrcek-Seiler, Andrea Tanzer, Caroline Thurner, Stefan Washietl, Stefanie Widder, Christina Witwer, Michael Wolfinger, Judith Ivansits, Judith Jakubetz.

Meiner Familie sowohl für die finanzielle als auch seelische Unterstützung.

Meinen Freunden für die Aufmunterung und Ablenkung. Matthew Darlison für den kreativen Austausch und Lydia Brühwasser für Graphikunterstützung.

Meine Arbeit wurde unterstützt durch ein DOC-Stipendium der Österreichischen Akademie der Wissenschaften.

Abstract

The aim of the work described in this thesis was the creation of a software tool to support the rational design of RNA molecules capable of forming two or more alternative metastable structures. This required the creation of a logical information model, thus isolating relevant aspects of the biological problem as posed, and incorporating these into a graph-based mathematical model. The algorithm we developed based on this model reduces the problem to vertex coloring the union of all prescribed outerplanar secondary structure graphs, called *dependency-graph*. Starting from a decomposition of this dependency graph, colorings are then produced by a dynamic programming procedure. In the final step sequences can then be optimized for particular properties by means of standard optimization heuristics. The connection between sequence design and vertex-colorings has hitherto not been described in literature.

Contents

1 Introduction **4**
- 1.1 Functions of single-stranded Nucleic Acid Molecules 4
- 1.2 Molecular Switches 5
  - 1.2.1 Examples of Riboswitches 6
- 1.3 Graph Theory in the Context of Biopolymers 7
  - 1.3.1 The Significance of Graph Theory 7
  - 1.3.2 Graphs of Biopolymers 8
  - 1.3.3 The Structure of Single Stranded Nucleic Acid Molecules . 9
- 1.4 Sequence Design and its Applications 12
- 1.5 Organisation of the Thesis 13

2 Basic Definitions **16**
- 2.1 Secondary Structures of Nucleic Acid Molecules 16
- 2.2 Representations of Secondary Structure 17
- 2.3 Graphs 19
  - 2.3.1 Components of Graphs 19
  - 2.3.2 Graph Colouring 20
- 2.4 Basic Algorithms 20
  - 2.4.1 Testing for Biconnectivity 20
  - 2.4.2 Breadth First Search 21
  - 2.4.3 Ear Decomposition 22
  - 2.4.4 Dynamic Programming 24

3 Sequence Design **26**
- 3.1 The Dependency Graph 26
- 3.2 RNA Structures and Compatible Sequences 26
  - 3.2.1 The Size of the Intersection 27
  - 3.2.2 Finding Random Sequences in $\mathbf{C}[\Theta_1] \bigcap \mathbf{C}[\Theta_2]$: 28
- 3.3 The Generalized Intersection Theorem 28
  - 3.3.1 The Dependency Graph of Multi-Stable Switches 30

4 The Design Algorithm **33**
- 4.1 Testing for Bipartiteness 33
- 4.2 Decomposing the Graph 33
  - 4.2.1 The Splitting Operation 34

4.2.2 Sequence Design as Graph Coloring 34
4.3 Ear Decompositon and Associated Graphs 35
4.4 Colourings 36
4.4.1 Implementation- Forward Recursion 39
4.4.2 Stochastic Backtracking 41
4.5 Applicability of the Design Algorithm 42

5 Optimization 44
5.1 Energy Landscape of RNA Molecules 46

6 Computational Results and Discussion 49
6.1 Feasibility of Mulit-stable Switches 49
6.2 Characterization of RNA Dependency Graphs 51
6.2.1 The Width of the Dependency Graphs 51
6.2.2 Tree-likeness of Dependency Graphs 51
6.3 Example applications of the design algorithm 53
6.3.1 Example 1 53
6.3.2 Example 2 54
6.3.3 Example 3 57
6.4 Dependency Graphs of Switches with Stack-size constraint 57
6.4.1 Characterization 57
6.4.2 Width of Dependency Graphs with stack constraints . . . 60

7 Summary and Discussion 61

8 Summary and Outlook 64
8.1 The Woffle Theory 64
8.2 Small networks 67

List of Figures 70

List of Algorithms 70

References 71

1 Introduction

1.1 Functions of single-stranded Nucleic Acid Molecules

DNA and RNA were long considered to have little cellular importance beyond their essential role in carrying and transmitting genetic information. However the discovery of special self-splicing RNAs led to increasing recognition that particularly RNAs play roles in many more and far more diverse cellular processes than had previously been considered.

The 1989 nobel prize for chemistry was awarded to Tom Cech and Sid Altman "for their discovery of catalytic properties of RNA", work which had taken the concrete form of characterizing the Tetrahymena Group I Introns and researching the role of RNA in RNAseP thus laying the foundations for the concept of catalytically active RNA [42, 28]. Although reactions catalyzed by RNA in vivo are often aided by protein interaction, in-vitro studies have shown that even without the help of proteins, RNAs can play a wide range of catalytic roles while being sequence specific and working much faster than expected. Research over several decades has shown that these specific RNAs, called ribozymes and initially known for phosphorylation reactions inside the cell, can do a lot more: In-vitro selection demonstrated that RNAs can catalyze acyl-transfer, Diels-Alder reactions, function as kinases etc. [89]. Interestingly, the activity of these ribozymes can also be controlled allosterically through binding of another RNA-strand, a property reminiscent of proteozymes. The abundance of short nucleotides as cofactors of many naturally-occuring enzymes has been interpreted as indicating that these cofactors date from an era when nucleotides were the main agents of cellular catalysis - the so-called "RNA-world". As more and more ribozyme functions are discovered, the hypothesis of a pre-biotic world in which RNAs performed the tasks of today's enzymes is gaining ground. Accordingly, the catalytic activity of RNAs has been assigned great evolutionary importance.

Very recent publications (reviewed in [34]) concerning non-coding RNAs (ncRNAS) suggest that the role of ribozymes in the regulation of vital cellular functions may even be as important as that of proteins. Two large classes of ncRNAs have been identified: (i) small nucleolar RNAs (snoRNAs [41]); and (ii) microRNAs (miRNAs) and small interfering RNAs (siRNAs). The assignment of the myriad of as-yet-unclassified candidate ncRNAs to categories constitutes another avenue of research. The cellular functions of these ncRNAs are diverse and important. While snoRNAs have been shown to be involved in rRNA modification and were proposed as RNA chaperones, miRNAs and siRNAs have been demonstrated to inhibit the translation of target mRNAs. Other functions including gene regulation at the level of imprinting, transcriptional interference and methylation modification have been postulated for other, as yet unclassified, ncRNAs. Moreover ligand-induced ncRNAs have been found that switch conformation and ultimately result in change of downstream regulated genes, and further research is ongoing to determine how common this phenomenon is. The near future should reveal how big the overall contribution of ncRNAs really is to the genetic activity of living organisms.

Against this background, research on ribozymes is no longer of purely mechanistic and evolutionary interest. Besides being necessary for understanding cellular activity this new knowledge even indicates possible therapeutic applications. siRNAs and miRNAs can both be considered as molecular switches capable of regulating gene expression. siRNAs can be easily administered to cells by lipotransfection[17], thus allowing the expression of certain targeted disease related genes to be suppressed. Clinical trials using such ribozymes in this way are already being conducted with promising results [69, 47].

1.2 Molecular Switches

The function of a nucleic acid molecule is largely determined by its three-dimensional structure, which in turn is influenced by the folding path of the molecule. Recent years have seen an accumulation of evidence indicating that the folding process of RNA is hierarchical in nature. At first, stable secondary structural elements are quickly formed which go on to determine the assembly of tertiary contacts [12, 75]. The fraction of Free Energy of the complete folding process involved in the formation of secondary structures is relatively large compared to that of the tertiary assembly [22]. Thus the secondary structure can serve as a model of the actual RNA structure that, although simplified, is both practical and representative and which has the added bonus of being computationally cheap.

Several efficient algorithms have been designed for the prediction of RNA secondary structures (see for example [66]). which allow for the fast computation of thermodynamic properties as well as the detailed exploration of conformational landscapes.

Thorough analysis of conformational energy landscapes of RNA molecules shows that non-native conformations with energies comparable to the ground state can exist, but may be separated from the native state by high energy barriers. The occurrence of such (meta-) stable conformations with considerable life spans has also been experimentally verified [18, 25, 31]. Furthermore it has been demonstrated that stable alternative conformations of the same RNA molecule can also fulfill completely different functions [6, 59]. This feature of RNA is used in nature to regulate and control a variety of biological processes. RNA molecules with two ore more (meta-) stable structures can thus be understood as molecular switches. The time-dependent behaviour of such a switch is mainly determined by the height of the energy barrier that has to be overcome by the molecule to change from one conformation into to the other. The barrier height can be modulated by external signals such as for example temperature changes or binding of other molecules (proteins, antisense-RNA etc.).

1.2.1 Examples of Riboswitches

A good example of a simple and nautrally-occuring temperature sensitive switch is SV11, a relatively small RNA molecule that is replicated by $Q\beta$ replicase [9, 8]. It has two dominating conformations- one is metastable and functions as the replicase's template and the other is the ground state that is not recognized by $Q\beta$. Melting and rapid quenching reconverts this molecule from the inactive and stable ground state to its meta-stable active state [88]. Several other thermosensitive switches have been suggested (e.g. [37, 53]. For example some genes that encode small heat shock proteins were found to contain a conserved sequence in their 5' untranslated regions referred to as the ROSE (repression of heat shock gene expression) element [56]. The SD box and AUG start codon are normally hidden in a stem structure, melting at heat-shock-inducing temperature makes these sequences accessible and thus allows for the translation of the associated mRNAs [58].

In 1997 J. Tang and R. Breaker demonstrated for the first time that engineered

RNAs can function as allosteric molecular switches and respond to small metabolites [73]. Since then, several natural metabolite-dependent riboswitches have been identified that control the expression of adjoining genes without the involvement of a protein factor. The list of metabolites known to modulate such switches includes $coenzyme - B_{12}$ [55], Thiamine pyrophosphate (TPP), flavin monocucleotide (FMN) [52], guanine [49].

Animal and plant miRNAs and siRNAs both influence gene expression by making use of an antisense mechanism (see [5] for an overview). A large group of miRNAs in Drosophila, for example, have been shown to form RNA duplexes with certain classes of sequence motifs. These miRNAs exhibit perfect complementarity to 3'UTR sequence motifs which in turn have been proven to mediate negative post-transcriptional regulation [44].

1.3 Graph Theory in the Context of Biopolymers

1.3.1 The Significance of Graph Theory

Graph Theory is a branch of mathematics that investigates the properties of graphs and their relations. At first glance it may seem to be a rather abstract and theoretical mathematical field that came into existence when Leonhard Euler solved a much celebrated problem known as the *Königsberg Bridges Problem* in 1736 [19]. Since then it has been exploited for the solution of numerous problems and to this day practical aspects of graph theory have found a wide range of applications.

Structures that can be presented as graphs are ubiquitous and many problems of practical interest can thus be modelled as graph theoretical problems. In the early days, graph theory was used for example by Kirchhoff to study electrical circuits [40] and Cayley to enumerate chemical isomers [14]. Nowadays it is an important tool in computer sciences as many algorithmic problems can be understood as graph problems and conversely many solutions to graph theoretical problems are based on algorithms. Network analysis and computational complexity theory are two more examples where graph theory plays a central role. In physics, graph theory is used to simulate complicated three dimensional atomic structures [24]. In chemistry we find two main types of correspondency between graphs and chemical categories: (i) The structural or constitutional graph that corresponds to a molecule or a group of molecules, where the nodes symbolize the atoms and edges symbolize covalent bonds (see Fig.1). It was this type of graph that inspired Cayley to develop a procedure for counting the constitutional isomers of alkanes [14], and (ii) the reaction graph that corresponds to a reaction mixture, where points symbolize chemical species and lines represent conversions between them (see Fig. 2). Vol'kenshtein and Gold'shtein were the first to use reaction graphs for kinetic studies (see [80, 81, 82, 83]). One of the pioneers in the field of reaction graphs is A.T. Balaban who reviewed the topic in [4]. This kind of graph plays an ever increasing role in explaining and rationalizing chemical rearrangements.

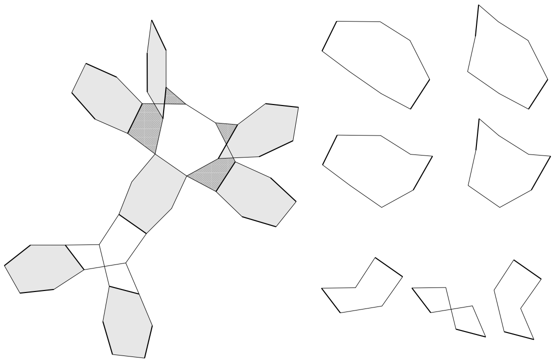


Figure 1: Organic carbon compounds may exhibit elaborate polycyclic structures. The example shown here is `Compound 8` from [39] (aromatic "double bonds" are indicated by thick lines).

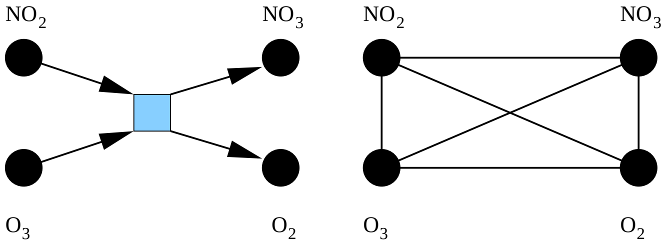


Figure 2: Representations of the reaction $NO_2 + O_3 \rightarrow NO_3 + O_2$ in hypergraph form: (left) the equivalent directed bipartite graph (right) as part of a substrate graph.

1.3.2 Graphs of Biopolymers

Biopolymers such as RNA, DNA or proteins fold into well-defined three dimensional structures that are of the utmost importance for their biological functions. The most fundamental features of the 3D shape of these molecules are captured

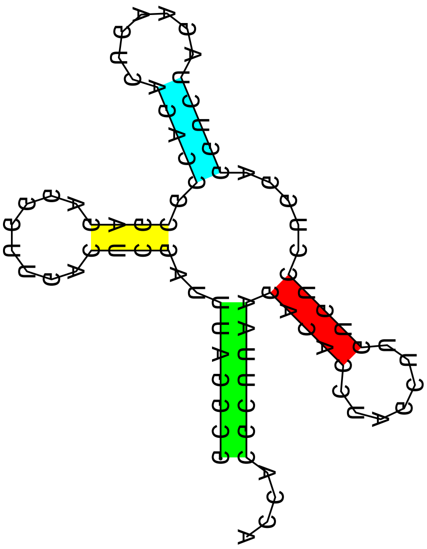


Figure 3: RNA secondary structure representation

in so-called *connection graphs* which have the atoms of small molecules or the monomers of a biopolymer as their vertices and the connections between spatially adjacent objects as their edges. Obviously, this simplification discards many structural details, yet it retains and exposes a wealth of structural information that can be gained from a variety of experimental and computational methods.

Biopolymers share a number of common features that distinguish them from other classes of molecular contact graphs. They have a spanning path that corresponds to the covalent backbone and the remaining non-covalent bonds determine the fold of the 3D structure of the molecule. Nucleic acids in particular form a special type of contact structure called *secondary structure* (see Fig. 1.3.2 for the conventional representation).

1.3.3 The Structure of Single Stranded Nucleic Acid Molecules

The monomers of nucleic acids are called nucleotides and consist of three components: A phosphate group, a pentose sugar and a nitrogenous heterocyclic base (either a purine or a pyrimidine). The periodic chain of phosphates and pentose sugars forms the backbone of the nucleic acid molecule and a base is bound to the χ position of each of the sugars (see Fig. 1.3.3).

References in the following text to "RNA" refer to nucleic acids in general. Ac-

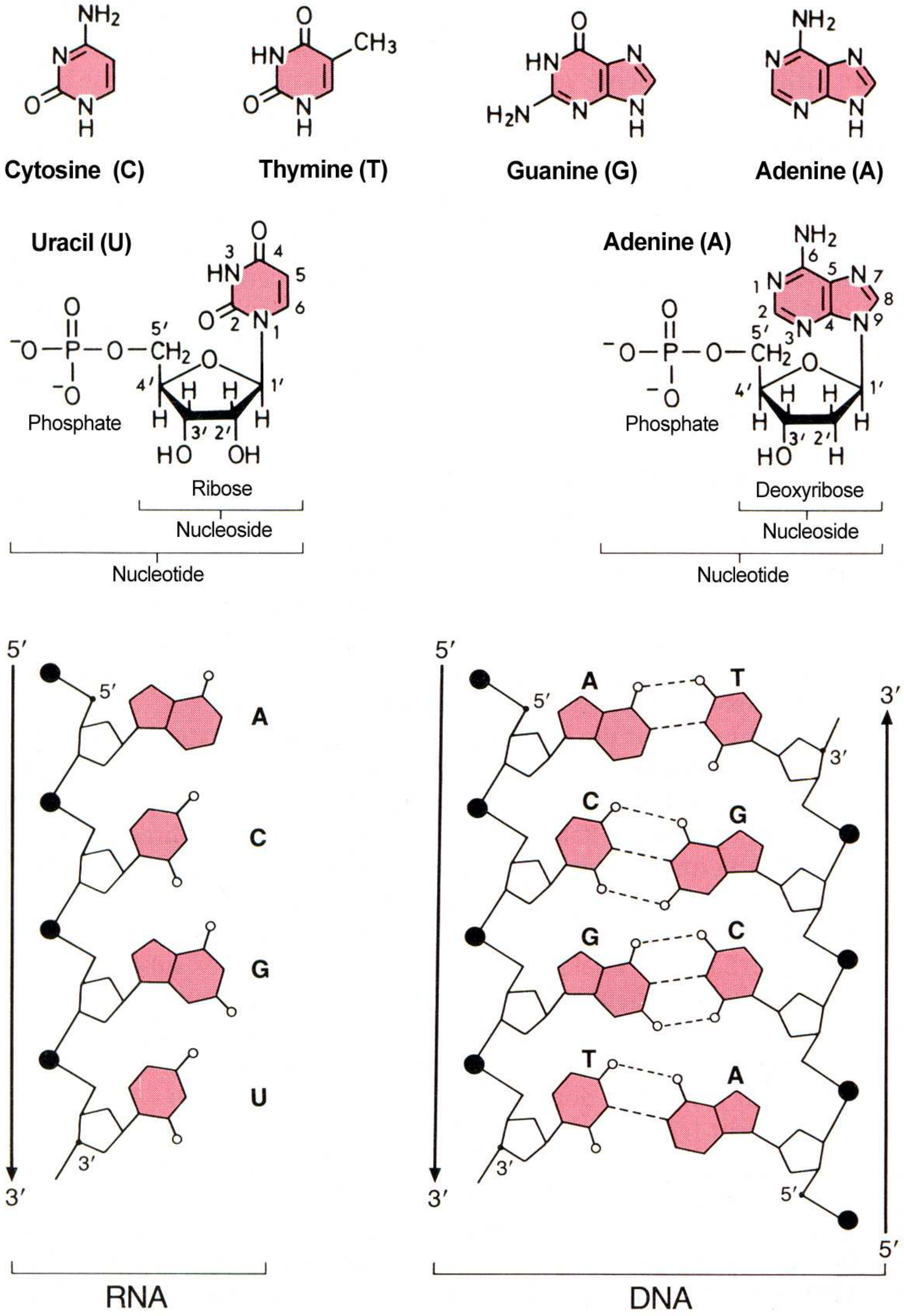


Figure 4: DNA and RNA structure and differences (from [86] and translated)

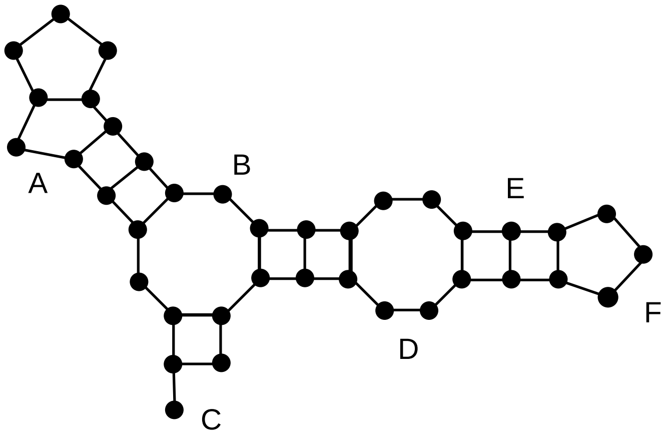


Figure 5: Secondary structure of an RNA molecule. As the molecule folds back onto itself, several secondary structure elements are formed: A) bulge, B) multi-loop, C) external vertex D) internal loop E) stacked basepairs F) hairpin loop.

cordingly, all definitions given for single stranded RNA apply equally to *single stranded* DNA with the exception that DNA consists not of ribose sugars, but of 2'deoxyribose sugars and that the alphabet of bases is guanine(G), adenine(A), thymine(T) and cytosine(C) for DNA and guanine, adenine, cytosine and uracil(U) for RNA. DNA molecules can form the Watson-Crick base pairs GC, CG, TA and AT, since RNA uses uracil instead of thymine it can also form the "wobble"-base pair GU, UG [16, 43]. Usually, DNA forms a double helical structure with a complementary strand [85] within the living cell, whereas RNA mostly occurs in the single stranded form and folds back onto itself. The stacking energy of the base pairs is the major driving force of the RNA structure formation.

The *primary structure* of RNA is simply the sequence of nucleotides, the *secondary structure* is represented by a graph where the vertices are the bases and the edges depict the contacts (hydrogen bonds) between these bases and the backbone along the sequence (see Fig. 1.3.3). Thus the secondary structure is actually a *topology* that indicates which sequence positions are adjacent, but says nothing about the spatial distances between the positions. Therefore, it is not a real two-dimensional representation of the structure. The so-called *tertiary structure* depicts the spatial arrangement of the secondary structure elements, i.e. the real three dimensional fold of the molecule (see Fig. 1.3.3).

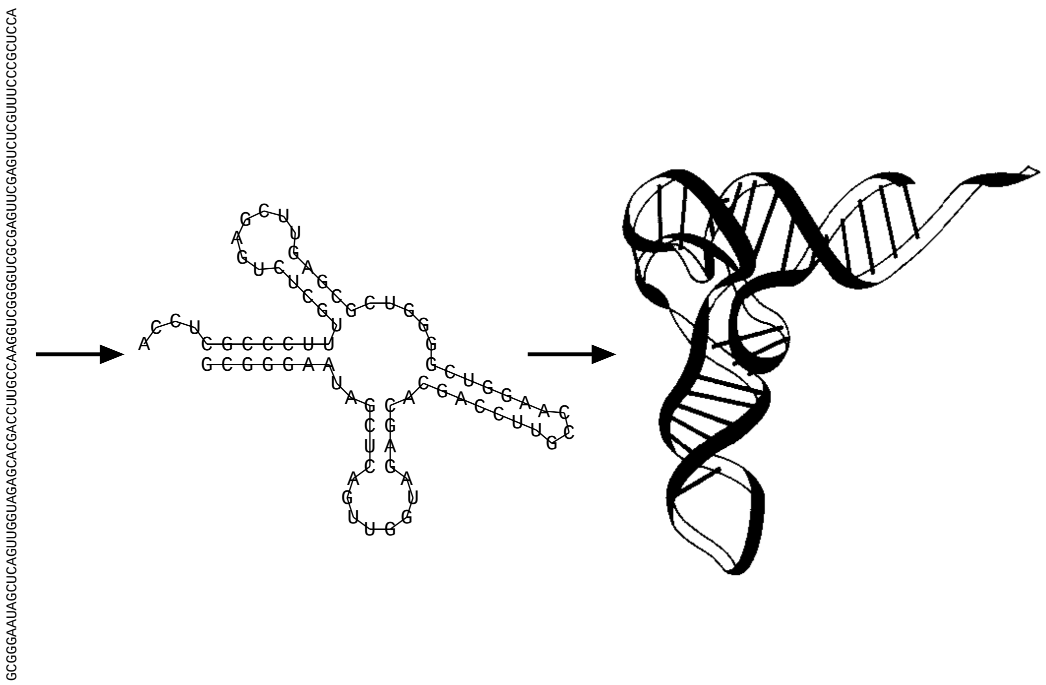

GCGGGAAUAGCUCAGUUGGUAGAGCACGACCUUGCCAAGGUCGGGGUCGCGAGUUCGAGUCUCGUUUCCCGCUCCA

Figure 6: Primary, secondary and tertiary structure of a tRNA molecule. The secondary structure shows the typical clover-leaf structure of tRNAs and the tertiary structure depicts the L-shape three-dimensional fold characteristic of tRNAs.

1.4 Sequence Design and its Applications

Nucleic acid design presents a key-step in many biotechnological applications. DNA microarrays for example consist of target-specific oligonucleotides that are immobilized on a surface and are used for experiments including genotyping and polymorphism analysis [21]. They require diligent design of probes, since they have to show maximum specific hybridisation with complementary targets in a complex solution while minimizing unspecific cross-hybridisation. Furthermore, DNA can be used as a template for organic synthesis, processes which rely on the rational design of DNA strands that direct the production of a library of small polymers [26, 30]. Another interesting field of design applications are DNA computing experiments where calculations are carried out at the molecular level [1, 46, 10]. Further motivation for nucleic acid design results from the fact that ribozymes provide new paths to drug design [77, 64] and can be used for engineering ligand-specific biosensors on microarrays based on the work of Breaker et al. [68].

There are numerous biocomputational approaches to address the requirements of this wide range of design tasks. Several tools have been implemented that use thermodynamic and combinatorial models to design both DNA probes for microarrays [7, 76, 38] and oligonucleotides than can be utilized as information storage media in biomolecular computations [72]. Software packages and tools have been created that deal with RNA secondary structure prediction [32, 91] and new variants of algorithms constantly emerge to keep step with the development of new tasks [3, 2].

That biocomputational methods can facilitate experimental research is demonstrated by the many instances in which nucleic acid design has been successfully applied. An excellent design example of a riboswitch that was also verified experimentally is described in the work of Bartel et al. [65]. Two ribozyme folds were selected that share no evolutionary history and not even a basepair in the secondary structure: the synthetic class III self-ligating ribozyme and the hepatitis delta virus (HDV) self-cleaving ribozyme. A sequence was then designed that can fold into both predescribed strutures and fulfill both respective enzymatic reactions (see Fig. 7). These results provide new insight into the possible mechanisms of ribozyme evolution, since it suggests that RNAs with novel structures and activities could arise from previously existing catalytic RNAs without an intermediate step of a non-functional sequence. Nucleic acid design also has practical applications in demonstrating the functionality of certain RNA structures as described for instance in [45]. By means of comparative sequence analysis and free energy calculations Lanz et al. studied and determined structured regions in SRA (steroid receptor RNA activator) and then went on to prove their functional importance by site-directed mutagenesis. Since all mutations that disrupted the secondary structure of the proposed SRA motifs were ”silent” mutations, i.e. did not alter the coding sequence, SRA-mediated coactivation was shown to occur on the RNA level through distinct RNA motifs and not to be executed by encoded proteins.

1.5 Organisation of the Thesis

Graphs of secondary structure form the basis and starting point of the design program presented in this work. In chapter 2 we describe the secondary structure of nucleic acids in more detail and provide the most important concepts of

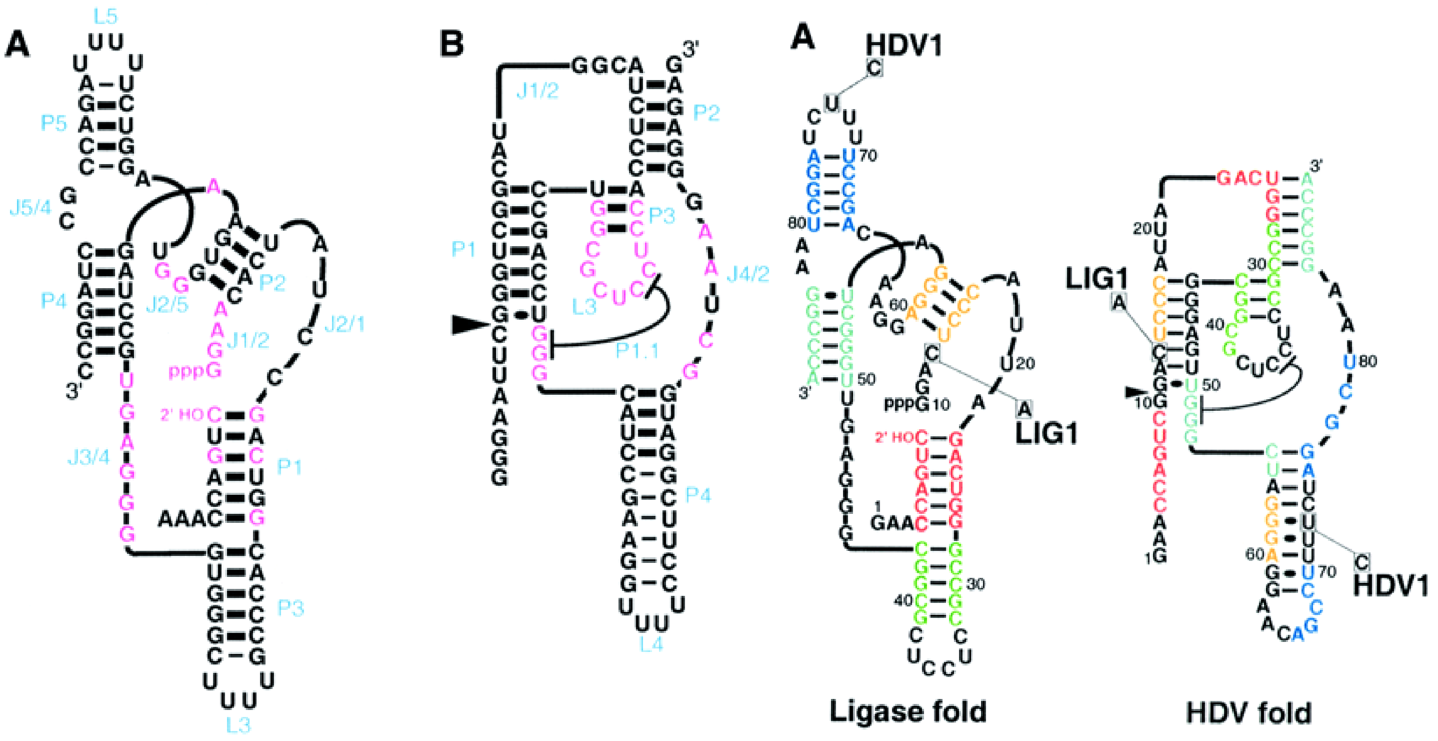


Figure 7: Design of a riboswitch with two functions. (left) Secondary structures of the prototype ribozymes: the synthetic class III self-ligating ribozyme and the hepatitis delta virus (HDV) self-cleaving ribozyme. (right) The designed sequence that can fold into both secondary structures (taken from [65])

graph theory that will be used throughout this thesis. Furthermore, we introduce some basic algorithms that were essential for this work.

In chapter 3 we proceed with the introduction of the theorems which lay the foundations for our approach. We briefly describe how the design task for bistable switches has previously been solved and then present the complications that arise for multi-stable switches.

The following chapter describes in detail all the steps of the design program for multi-stable switches and introduces the mathematic background for a dynamic programming algorithm that can be used to count sequences compatible with a block given an ear-decomposition.

WIRD NOCH VERVOLLSTAENDIGT

Chapter 5 Optimization

Chapter 6 Computations and Discussion

Chapter 7 Summary and Outlook

2 Basic Definitions

2.1 Secondary Structures of Nucleic Acid Molecules

The primary structure of a single stranded nucleic acid molecule is by definition the linear sequence of nucleotides $x = x_1, x_2, ..x_n$ of the length n where $a_i \in A, U, C, G$. As previously mentioned, the secondary structure is a graph with the bases as vertices and the contacts between the bases as edges. There are two kinds of contacts: (1) the covalent ribophosphate bridges between consecutive bases which form the backbone and (2) the set of base pairs formed by hydrogen bonds. The backbone is of little interest since it is defined by the sequence, it's the set of base pairs that characterizes a secondary structure. A sequence can form various secondary structures which differ according to their energies.

Definition 1. Secondary Structure
A nucleic acid secondary structure can be understood as the set Θ of base pairs. The graph of the secondary structure consists of a set of vertices (bases) $V = 1, 2, ..., i, ..., n$ and a set E of edges $i \cdot j, 1 \leq i \leq j \leq n$ with the following constraints:
(i) (backbone) $\forall i < n$: $i \cdot (i+1) \in E$ and
(ii) (binary pairing) for each i there is at most one $k \neq i-1, i+1$ where $i \cdot k \in E$, and
(iii) (pseudoknots are not allowed) if $i \cdot j \in E$ and $k \cdot l \in E$ and $i < k < j$ then $i < l < j$.

According to this definition the set of vertices just contains the enumerated set of nucleotides of the sequence of the length n. The set of edges contains the backbone of the chain and the base pairs where base pairing between consecutive positions i and $i+1$ is ruled out. In practice, this does not represent a limitation, since steric reasons make it almost impossible for the bases i and $i+1$ to form a basepair. To be able to compute secondary structures we now have to define a set $\mathcal{B} \subset \mathcal{A} \times \mathcal{A}$ of allowed basepairs for the alphabet $\mathcal{A}$ of riobonucleotides. These are the Watson-Crick pairs AU, UA, GC, CG and the *wobble pair* GU, UG.

Criterion (iii) of our definition rules out the occurrence of so-called *pseudoknots* (see for example [70]), i.e. crossing basepairs, the reason being that most efficient RNA-folding algorithms can't deal with them.

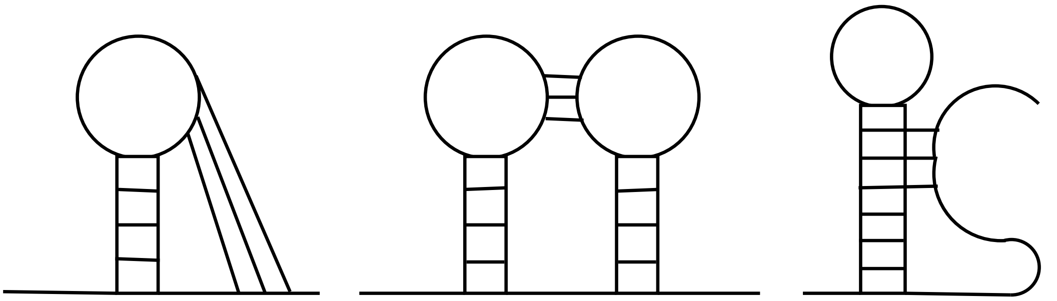


Figure 8: RNA pseudoknots and base triples are excluded by our secondary structure definition. This figure shows an H-type pseudoknot (left), kissing loops (middle) and a base triplet (right).

2.2 Representations of Secondary Structure

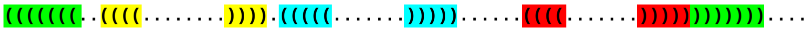
There are several kinds of secondary structure representations, Fig. 9 presents an overview of the graphs.

The *dot-bracket* or *string*-representation depicts unpairing positions as dots '.' and opening and closing positions of pase pairs as parentheses '(' and ')'.

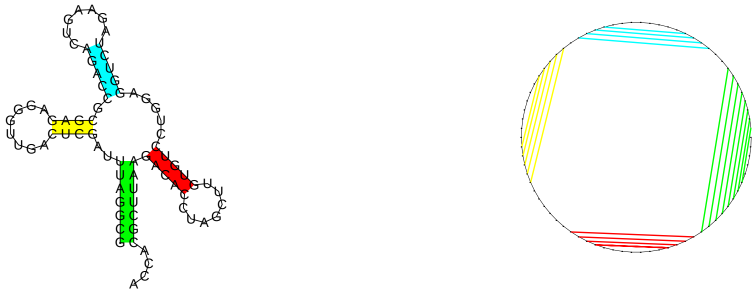
The *dot plot* presents base pairs (i, j) as predicted by the McCaskill algorithm [51] in the form of squares in row i and column j in the upper right side of the plot. The size of the square is proportional to the predicted base-pairing probability. A square in row j and column i on the lower left side of the dot plot indicates a base pair (i, j) which is part of the minimum-free-energy structure of the sequence.

The *mountain representation* is especially useful for comparing large structures [33]. The directions "horizontal", "up" and "down" are used analogously to the symbols '.', '(' and ')' in the string representation. It follows that slopes depict the opening and closing basepairs of stems, plateaus show the unpaired bases in loops and valleys indicate unpaired regions between branches or external bases if the height is zero.

The representation of particular importance to this thesis is the *cycle representation* in which the 5' - 3' nucleotide sequence is depicted as a circle and the basepairs appear as chords between pairing positions. The pseudoknot constraint means that no crossing chords may occur in the graph.

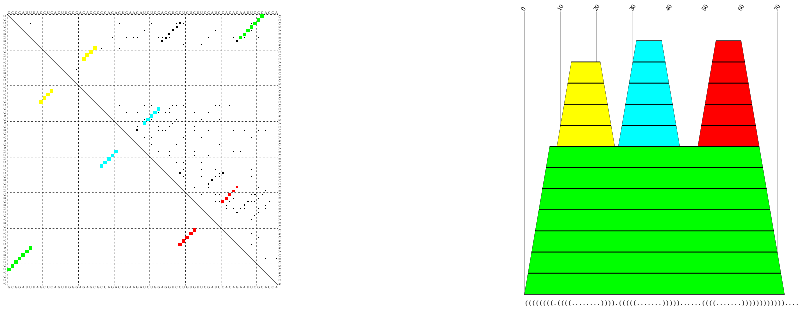


String



Conventional

Circle



Dot plot

Mountain plot

Figure 9: Different representations of RNA secondary structure, the individual secondary structure elements are depicted in the same colours in all representations.

2.3 Graphs

Informally, a graph consists of points or dots and lines or arrows that connect the points. The shape or length of lines and the angle between lines are usually not of any importance, so a graph can rather be considered as a topological than a geometric object.

Scientifically speaking, a graph $G = (V, E)$ consists of two sets: a finite set of vertices V (the points) and a finite set of edges E (lines, arrows) that link together the vertices. We distinguish between *undirected* graphs, where edges are non-ordered pairs of vertices and *directed* graphs, where edges are considered as ordered pairs.

The degree $d(v)$ of a vertex v is the number of edges with which it is incident. Two vertices are *adjacent* if they are incident to a common edge. The set of *neighbours*, $N(v)$, of a vertex v is the set of vertices which are adjacent to v.

A *walk* in a graph is a finite alternating sequence of vertices $v_0, v_1, v_2, ..., v_k$ such that $(v_{i-1}, v_i), 1 \leq i \leq k$, is an edge in the Graph G. A walk is *open* if its end vertices are distinct, otherwise it is *closed.* A walk is a *trail* if its edges are distinct. An open trail is a *path* if all its vertices are distinct. A closed trail is a *circuit* or *cycle* if all its vertices except the end vertices are distinct.

A graph $G' = (V', E')$ is a *subgraph* of a graph $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. The subgraph of G *induced* by V' is the grpah $H = (V', F)$ where $F = (u, v) \in E \mid u, v \in V'$.

A graph is called *acyclic* if it has no circuits. A *tree* of a graph $G$is a connected acyclic subgraph of G. A *spanning tree* of a graph G is a tree of G having all the vertices of G.

Furthermore, an undirected graph $G = (V, E)$ is *bipartite*, if its vertex set V can be partitioned into two subsets V_1 and V_2 such that each edge E has one end vertex in V_1 and one in V_2.

2.3.1 Components of Graphs

An undirected graph G is *connected*, if there exists a path between every pair of vertices in G. A *connected component* of a graph G is a maximal induced subgraph of G which is connected. A vertex v of a connected graph G is a *cut-vertex* if and only if there exist two vertices u and w distinct from v such that v is on every $u - w$ path. A connected graph is *biconnected* if it contains at least three vertices and has no cut-vertices. A *biconnected component* or *block* of G is a maximal induced subgraph of G which is biconnected.

2.3.2 Graph Colouring

A graph colouring is a consecutive assignment of "colours" to certain objects in a graph: these objects can be vertices, edges or faces or a mixture of all of them. Most important for our purposes is the vertex colouring, usually assuming that no two adjacent vertices are allowed to be assigned the same colour. A colouring of the graph G with k colors is called a $k - colouring$ and is a function $f : V(G) - > 1, 2, ..., k$, such that no edge $e = (u, v)$ has $f(u) = f(v)$.

2.4 Basic Algorithms

2.4.1 Testing for Biconnectivity

The procedure for determining biconnected components is based on a depth first search that incorporates a criterion to identify cut-vertices. A *depth first search* (DFS) is an algorithm for traversing a graph in such a manner that starting from the root vertex one explores as far as possible along each branch before backtracking. The DFS progresses by expanding the first child of a search tree that appears and thus goes deeper and deeper, until a goal state is found or it hits a node that has no children. Then the search backtracks and starts off on the next node with children.

Algorithm 1 is a recursive procedure starting from a root vertex r, all vertices are white (unvisited) initially, grey if they have been visited and black if they have been fully explored. The $dfs - counter$ variable labels each vertex with a depth first search number ($DFN(u)$) in the order it was found. A vertex $v \neq r$ is a cut-vertex of G if and only if v has a son s sucht that low$(s) \geq v$. The function low(u) thus introduces the criterion for identifying cut-vertices in our algorithm and is defined as follows

Algorithm 1 Algorithm for finding cut-vertices: CUT(u)

```
Require: root $u \in V$
dfs-counter++
$color[u] \leftarrow$ GRAY
$low[u] \leftarrow$ dfs-counter
$DFN[u] \leftarrow$ dfs-counter
for all vertices $v \in Adj[u]$ do
    if $color[v] =$ WHITE then
        $\pi[v] \leftarrow u$
        call CUT($v$)
    else if $v \neq \pi[u]$ then
        if $v \neq \pi[u]$ then
            $low[u] \leftarrow min\{low[u], DFN[v]\}$
        end if
    end if
end for
$color[u] \leftarrow$ BLACK
if $low[u] \geq DFN(\pi[u])$ then
    add $\pi[u]$ to the list of cut-vertices
end if
$low(\pi[u]) \leftarrow min\{low(\pi[u]), low[u]\}$
```

$\text{low}(u) = \min(\{u\} \cup \{\text{low}(s)\text{—}s \text{ is a son of } v\} \cup \{w|(v, w) \text{ is a back edge}\})$

2.4.2 Breadth First Search

A *breadth first search* (BFS) is another way of searching a graph starting from a root node, but in contrast to traversing the graph “vertically” as in the DFS algorithm, it is explored “horizontally”, visiting all the neighbouring vertices. Then one continues by searching for all the unexplored neighbour nodes to each of those nearest nodes and so on, until the goal is met.

Algorithm 2 uses two colour-attributes for each vertex u: (i) BFS-color(u), which tells if a vertex has not been visited (white), has been visited, but not all its adjacent vertices Adj(u) (gray), or if it has been fully explored (black); and (ii) color(u) for 2-colouring the graph. The vertices are all white at the beginning

Algorithm 2 Breadth First Search BFS(u)

```
Require: root $u \in V$
enqueue start-vertex $u$
while queue not empty do
    fetch $Adj[u]$
    for all vertices $v \in Adj[u]$ do
        if $BFS - color[v] = \text{WHITE}$ then
            $BFS - color[v] \leftarrow \text{GRAY}$
            if $color[u] = \text{RED}$ then
                $color[v] \leftarrow \text{BLACK}$
            else
                $color[v] \leftarrow \text{RED}$
                enqueue vertex $v$
            end if
        else if $color[u] = color[v]$ then
            exit {graph is not bipartite}
        end if
    end for
    $BFS - color[u] \leftarrow \text{BLACK}$
    dequeue vertex $u$
end while
```

and then coloured red and black alternatingly. The program exits if two adjacent vertices have the same colour red/black.

2.4.3 Ear Decomposition

An *ear decomposition* is a graph searching method for biconnected components that decomposes a block into a collection of labelled paths. An undirected graph $G = (V, E)$ has an ear decomposition if and only if it is biconnected [87].

An ear decomposition $D = [P_0, P_1, ..., P_{r-1}]$ of an undirected graph $G = (V, E)$ is a partition of E into and ordered collection of edge-disjoint simple paths $P_0, ..., P_{r-1}$ such that P_0 is an edge, $P_0 \cup P_1$ is a simple cycle. Each endpoint of P_i, for $i > 1$, is contained in some $P_j, j < i$ and none of the internal vertices of P_i are contained in any $P_j, j < i$. We call the paths in D ears.

Algorithm 3 Ear Decomposition EAR(G, u)

```
Require: connected graph $G = (V, E)$, root $u \in V$, $|V| = n$
$color[u] \leftarrow$ GRAY
$DFN[u] \leftarrow$ counter
counter ++
for each vertex $v \in Adj[u]$ do
    if $color[v] =$ WHITE then
        $\pi[v] \leftarrow u$
        call EAR$(G, v)$
        if $low[v] \geq DFN[v]$ then
            $ear(\pi[v], v) \leftarrow (\infty, \infty)$
        else if $low[v] \leq DFN[v]$ then
            $ear(\pi[v], v) \leftarrow ear[v]$
            $low[u] \leftarrow min\{low[u], low[v]\}$
            $ear[u] \leftarrow lexmin\{ear[u], ear[v]\}$
        end if
    else if $color[v] =$ GRAY then
        if $v \neq \pi[u]$ then
            $low[u] \leftarrow min\{low[u], DFN[v]$
            $ear(v, u) \leftarrow (DFN[v], DFN[u])$
            $ear[u] \leftarrow lexmin\{ear[u], ear(v, u)\}$
        end if
    end if
end for
$color[u] \leftarrow$ BLACK
{sort the earl labels of the edges in non-decreasing order and relabel distinct
    labels in order as 1,2,..;}
```

The algorithm (see Alg.3) used in this thesis was taken from a chapter of the book *Synthesis of Parallel Algorithms* written by V. Ramachandran [61] and based on [74]. This recursive procedure computes an ear decomposition (i.e. ear labeling of the edges) while constructing a depth-first search tree of the graph G. Initially all vertices are unvisited (white). As in the DFS-procedure, gray indicates if a vertex has been visited and black if it's been fully explored. $\pi(u)$ denotes the parent of the vertex u in the DFS-order, the *counter* variable labels each vertex in the DFS-order it was found. The functions $low(u)$ and $ear(u)$ are defined as follows, assuming that the vertices are numbered in the DFS order and that the graph has n nodes:
$low(u)$ =min($\{v|v$ lies on the fundamental cycle of a nontree edge incident on a descendant of $u\} \cup \{n\}$)
$ear(u)$ =lexmin($\{(v,x)|(v,x)$ is a nontree edge with x a descendant of $u\} \cup \{n,n\}$)

2.4.4 Dynamic Programming

Dynamic Programming is an extremely important and useful technique for solving optimization problems. It can be applied for problems that exhibit the properties of *optimal substructure*, which means that the optimal solution of subproblems can be used to find the optimal solution of the overall problem. In general, we can solve a problem with optimal substructure using a three-step process:

(i) Break the problem into smaller subproblems.

(ii) Solve these problems optimally using this three-step process recursively.

(iii) Use these optimal solutions to construct an optimal solution for the original problem.

The subproblems are, themselves, solved by dividing them into sub-subproblems, and so on, until we reach some simple case that is easy to solve. The procedure thus makes use of a *bottom-up technique*, in which the smallest sub-instances are explicitly solved first and the results of these used to construct solutions to progressively larger sub-instances. The subproblems need not be independent from each other, it is only important that optimality at any given point of time t does not destroy the optimality of previously found solutions.

The advantage of dynamic programming over other prodecures, like for example the divide-and-conquer method, lies in reducing the runtime of algorithms, since it avoids the repeated computation of identical sub-instaces that arise when the subproblems are not independent. The method accomplishes this by maintaining a table of sub-instance results.

Well-known examples for dynamic programming algorithms in biocomputing are sequence alignment algorithms [57, 67] and the maximum matching algorithms for finding RNA secondary structures [92].

3 Sequence Design

3.1 The Dependency Graph

Each of the structures that are predefined in our design task can be illustrated by a cycle representation as described in chapter 2. We now superimpose all these graphs of secondary structure in a composite graph- the *dependency graph* (Fig. 10), which forms the starting point for the development of our algorithm. The dependency graph Ψ of a collection of secondary structures $\{\Theta_i\}$ with n nucleotides consists of n vertices and edges connecting k with l if and only if (k, l) is a basepair in at least one of the secondary structures Θ_i.

Our approach is based on the following theorem (described in the sections 3.2 and 3.3) that incorporates the concept of the depency graph and characterizes the realizability of a collection of M distinct secondary structures by a single RNA sequence. We start off by describing the mathematical background of the solution for designing bistable switches as presented in [23] and then generalize the problem for switches with more than two predefined structures.

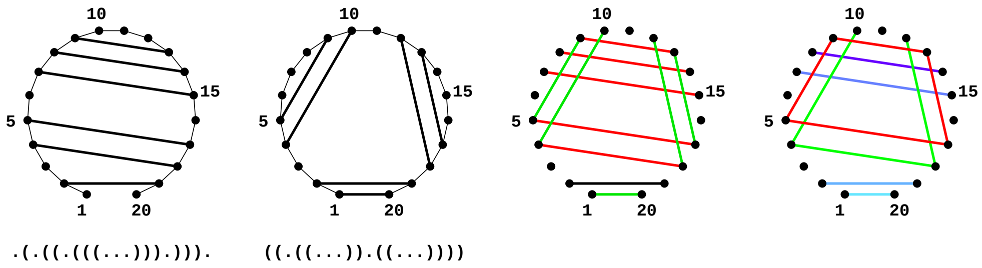


Figure 10: The dependency graph Ψ. (left) Circle representation of secondary structures 1 and 2. (middle) The dependency graph is constructed by super-imposing the circle representations of the two structures. Edges that can only be found in structure 1 are green, those only in structure 2 red, edges contained in both structures are black. (right) Paths are coloured blue and green, cycles red [23].

3.2 RNA Structures and Compatible Sequences

As previously defined, we denote Θ as the set of base paired sequence positions and Υ as the set of unpaired positions. The base pairing rules for RNA allow only

6 types of base pairs out of the 16 possible combinations, thus given a secondary structure Θ our choice of sequences compatible with Θ is restricted, since for each pair $i, j \in \Theta$ and each sequence x, the positions x_i and x_j must be able to form one of the permitted base pairs. We write $\mathbf{C}[\Theta]$ for the set of all sequences that are compatible with Θ in the sense that every basepair $(i, j) \in \Theta$ is realized by a pair $(x_i, x_j) \in \mathcal{B}$ of pairing nucleotides. It follows that the number of sequences compatible with a given structure $\mid \mathbf{C}[\Theta] \mid$ is

$$\mid \mathbf{C}\,[\Theta] \mid = \mid \mathcal{A} \mid^{|\Upsilon|} \mid \mathcal{B} \mid^{|\Theta|} \tag{1}$$

since for each $i \in \Upsilon$ we can choose any random letter of the nucleic acid alphabet and for each pair we may choose one of the possible base pairs.

The first obvious question that arises when trying to design multi-stable sequences is: Can sequences be found that fold into two (and consequently more) predefined structures and if so, how many, i.e. what is the size of the cardinality of the intersection of two given secondary structures? This is answered by the *Intersection Theorem.*

Theorem 2. *Intersection Theorem: If the nucleic acid alphabet admits at least one type of complementary base pairs, then, for any two secondary structures* Θ_1 *and* Θ_2 *there exists at least one sequence that is compatible with both structures, in symbols:*

$$\mathbf{C}[\Theta_1] \cap \mathbf{C}[\Theta_2] \neq \emptyset \tag{2}$$

This means whenever we have symmetric base pairs, i.e. $XY \in \mathcal{B}$ implies $YX \in \mathcal{B}$, we can always find sequences that can fold into both Θ_1 and Θ_2. An abstract group-theoretical proof can be found in [63], for a purely combinatorial proof see [23].

3.2.1 The Size of the Intersection

In order to determine the cardinality of the intersection of Θ_1 and Θ_2 we display the graph Ψ as the conjunct union of its connected components: $\Psi = \dot{\bigcup}\psi$. Depending on the predefined structures there can be found three kinds of components:

(i) Positions that are unpaired in both structures correspond to isolated vertices in the graph.

(ii) Positions that are paired in both secondary structures correspond to paths of the length one.

(iii) Base-pairing positions that occur only in one of the two different structures are part of paths or cycles.

Using this definition we can write:

$$|\mathbf{C}[\Theta_1]\bigcap\mathbf{C}[\Theta_2]| = \prod_{\text{components } \psi \text{ of } \Psi} F(\psi) \tag{3}$$

where $F(\psi)$ describes the number of sequences compatible with a connected component ψ of Ψ. For (i) an isolated vertex K_1 we get $F(\{i\}) = F(K_1) = |\mathcal{A}|$, i.e. the number of nucleotides in the alphabet, for (ii) paths of the length 1 $F(K_2) = |\mathcal{B}|$, the number of allowed base pairs, for (iii) the compatible sequences of paths and cycles can be counted by using recursive formulae called *Fibonacci-numbers* (paths) and *Lucas-numbers* (circles):

$$F(P_n) = 2(Fib(n) + Fib(n+1)) = 2Fib(n+2) \tag{4}$$

$$F(C_n) = 2(Fib(n-1) + Fib(n+1)) = 2Lucas(n) \tag{5}$$

3.2.2 Finding Random Sequences in $\mathbf{C}[\Theta_1]\bigcap\mathbf{C}[\Theta_2]$:

The recursive counting formulae shown in the previous section can be used inversely for creating a generator of uniformly distributed objects- in this case for sequences of the intersection $\mathbf{C}[\Theta_1]\bigcap\mathbf{C}[\Theta_2]$: Each connected component ψ of Ψ is independent from all others, since sequence constraints are only defined by the edges in Ψ. Hence, the task of generating a sequence in the intersection is reduced to assigning sequences to each separate connected component. (For an implementation see [23].)

3.3 The Generalized Intersection Theorem

The intersection theorem as postulated in the previous section does not directly generalize to more than two sequences. However, in [23] it was shown that based on the conecpt of colouring the dependency graph we obtain the following:

Theorem 3. *Generalized Intersection Theorem: Suppose $\mathcal{B} \subseteq \mathcal{A} \times \mathcal{A}$ contains at least one symmetric base-pair then:*

1. $\boldsymbol{C}[\Theta_1] \cap \boldsymbol{C}[\Theta_2] \cap \cdots \cap \boldsymbol{C}[\Theta_k] \neq \emptyset$ *if the dependency graph* $\Psi = \Theta_1 \cup \Theta_2 ... \cup \Theta_k$ *is bipartite.*

2. *There are*

$$\prod_{\text{components } \psi \text{ of } \Psi} F(\psi) \text{ sequences in } \bigcap_j \boldsymbol{C}[\Theta_j]$$

   where $F(\psi)$ is the number of sequences that are compatible with a connected component ψ of Ψ.

3. *For the biophysical alphabet holds:* $\bigcap_j \boldsymbol{C}[\Theta_j] \neq \emptyset$ *if and only if Ψ is a bipartite graph.*

It follows that designing switches with more than two metastable structures is much more complicated than the bistable case: Firstly, according to the Generalized Intersection Theorem the dependency graph Ψ has to be bipartite. It was proven in [23] that the dependency graph of bistable switches is always bipartite, if we predefine more than two structures then this is not the case anymore. Loosely speaking, a graph is bipartite if we can divide the set of vertices into two subsets, so that all edges have one vertex in one subset and the other vertex in the other. This means that no edges may occur within one subset. The constraint of bipartiteness arises simply from the base pairing rules. The simplest case of a non-bipartite graph is a triangle, let us call into mind that the basepairing rules demand that each purine base can only pair with a pyrimidine and vice versa. If we try to assign a sequence to a triangle (choose one color for pyrimidines, another one for purines), we immediately see that this task cannot be solved, since there will always be either two purines or two pyrimidines connected by an edge (see Fig. 11).

Secondly, the dependency graph of more than two predefined structures is more complicated and thus the design task not merely a trivial extension of the bistable case. The main complication arises from the fact that we additionally find blocks

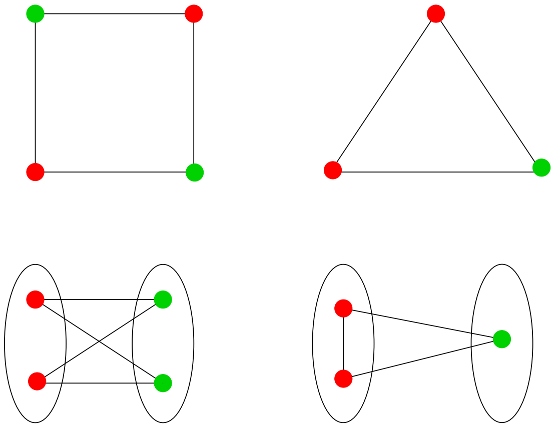


Figure 11: The bipertite property. (left) The square is a simple form of a bipartite graph, its vertices can be assigned to two subsets in such a manner that no edge occurs within one subset(below). (right) The triangle is the simplest form of a non-bipartite graph. Its edges cannot be assigned to two subsets without an edge occuring within one subset (below).

in these dependency graphs that are connected by cut-vertices and thus the biconnected components are not independent anymore (see Fig.12).

3.3.1 The Dependency Graph of Multi-Stable Switches

As with the bistable case, we are again interested in finding recursions that can be used for determining the size of the intersection. For this purpose we first identify connected components and then have to decompose or *split* these into their biconnected components. If we explore the graph along the cut-vertices we obtain a tree-like structure (see Fig. 12). The number of sequences compatible with the graph G_i given the base X at the cut-vertex x is thus the result of

$$\sum_{X \in \mathcal{A}} \prod_{\psi \subset \Psi} F(G_i | x = X) \tag{6}$$

For the case pictured in Fig. 12 we can write the following:

$$\sum_{X \in \mathcal{A}} F(G_1 | x = X) F(G_3 | x = X) \sum_{Y \in \mathcal{A}} F(G_2' | x = X \wedge y = Y) F(G_2'' | y = Y) \tag{7}$$

G_1 and G_2' are paths of the length 1, G_3 is a cycle of the size 4. For these components we have already found recursions in the previous section. Graph

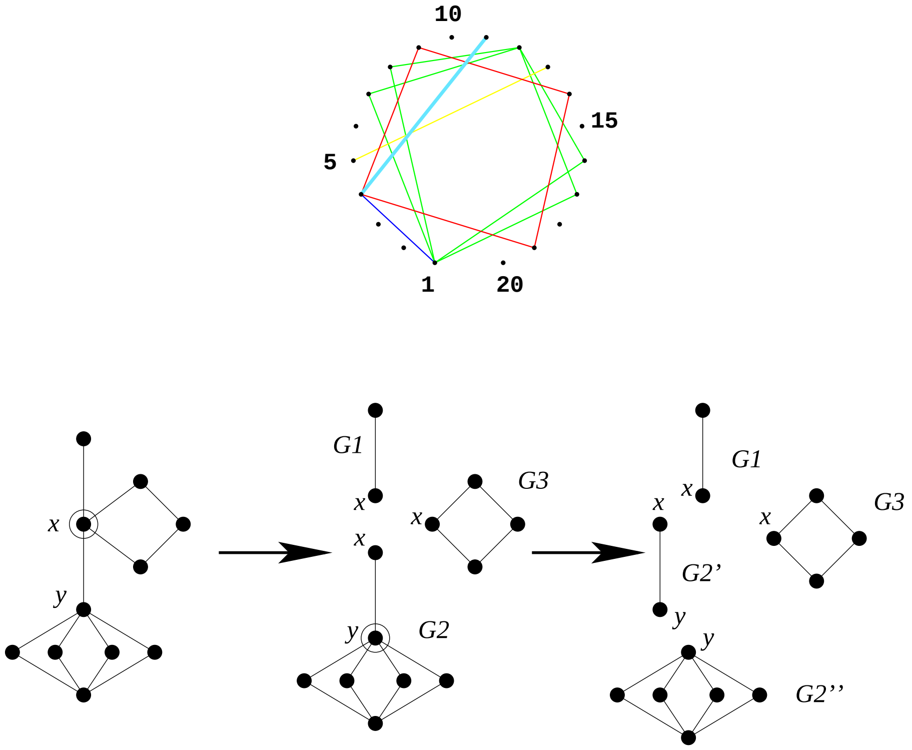


Figure 12: The dependency graph Ψ of more than two structures may consist of several and more complicated connected components that need further decomposition. (top) This shows a dependency graph of four superimposed secondary structures with two connected components: a path (yellow) and a complex component (blue, green, cyan). (below) The second connected component of the dependency graph above is represented here. It is decomposed at the cut-vertices x, y into two paths of the length 1, a circle of the length 4 and a block.

G_2'' is a complex block and therefore has to be dealt with in a new and different way: Structural characterizations of graphs are often based on so-called *ear decompositions* as described by Whitney [87]. In essence, this procedure assembles a given graph by starting off with a simple sub-graph (e.g. a circle) and proceeds by repeated attaching of simple elements, the "ears" (e.g. paths), until the graph is complete [87, 48]. Inversely, we can decompose a graph by successive removal of the "ears" until a simple object is left, see Fig. 13. In the next chapter we describe in detail how the splitting operation of the graph works and how compatible sequences can be found using the ear decomposition.

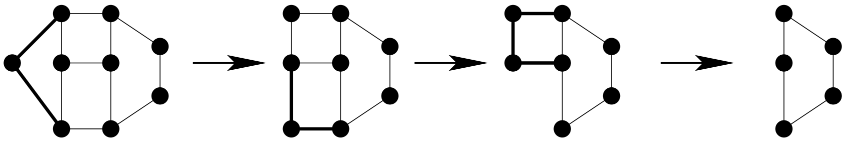


Figure 13: The ear decompositon: In each step a path is removed until a simple object (circle) is left.

4 The Design Algorithm

In detail, the algorithm consists of the following steps:

1. INPUT: Predefine a set of secondary structures.
2. Draw the dependency graph Ψ.
3. Test for the bipartite property of the graph Ψ using a simple breadth-first search.
4. Decompose the graph into its connected components, then further into the biconnected components and finally decompose also the blocks by Whitney's Ear Decomposition.
5. Count the number of compatible sequences.
6. Generate sequences with uniform distribution on the set of compatible sequences.
7. Optimize the sequence according to an energy criterion.
8. OUTPUT: Optimized nucleic acid sequence compatible with all predefined structures.

4.1 Testing for Bipartiteness

According to criteron (i) of the Intersection Theorem we can only find sequences compatible with all predescribed structures if the resulting dependency graph is bipartite. Consequently, the first task in our program is to test for bipartiteness which we achieve by two-colouring the graph using a simple breadth-first search (see section 2.4.2).

4.2 Decomposing the Graph

In order to be able to design sequences without *a priori* bias in the sequence composition, we need to count and sample uniformly from the sequences compatible with a set of structures. According to assertion 2 of the Generalized Intersection Theorem each connected component can be treated independently and all their combinations are used to compute the cardinality of the intersection. Thus the next task of our algorithm is to identify connected components and find recursions to count compatible sequences. For this purpose we use the biconnectivity algorithm for finding *cut-vertices* (see section 2.4.1) and then decompose or *split* the connected components further into their biconnected components. At this point we want to call to attention that by definition paths of length n are decomposed into n subpaths of length 1, since each subpath presents a biconnected component. It is a lot more efficient to avoid this fragmentation, as paths of arbitrary length can be coloured in a more effective manner by our recursive algorithms explained in sections 4.4.1 and 4.4.2. Therfore, we only split our graph at cut-vertices $c(k)$ if $deg(c) \geq 3$, which means we screen for cut-vertices at bifurcation points and ignore those that are contained in paths.

4.2.1 The Splitting Operation

For two graphs $G_1(V_1, E_1)$ and $G_1(V_2, E_2)$ we define the the union $(G_1 \cup G_2)(V_1 \cup V_2, E_1 \cup E_2)$. Intersection, difference, etc. are defined analogously. Furthermore, G_1 is a subgraph of G_2, in symbols $G_1 \subseteq G_2$, iff $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$. We sometimes write $V(H)$ and $E(H)$ or V_H and E_H for the vertex and edge set of a graph H.

Let $G(V, E)$ be a connected graph and let $A \subseteq V$. The *splitting* of G at A is the graph $G^{\rangle G \langle}$ obtained from G by (i) replacing each vertex in A by $\deg_G(x)$ copies, one attached to each edge incident with x, and (ii) identifying again those copies of x that are contained in the same connected component after step (i). See Fig. 14 for an illustration of the splitting operation. If follows that either $G^{\rangle G \langle}$ is not connected any more or $G^{\rangle G \langle} = G$.

The classes of biconnected components we thus get by decomposing the graph at its cut-vertices are paths, cycles and complex blocks. For the case of paths and cycles, the counting and sampling problem is solved in [23]. The blocks are now decomposed by means of ear decomposition.

4.2.2 Sequence Design as Graph Coloring

The important observation is that a sequence that is compatible with all secondary structures can be viewed as a colouring of the vertices of the dependency graph Ψ such that adjacent vertices have colours $(a, b) \in \mathcal{B}$, the alphabet of

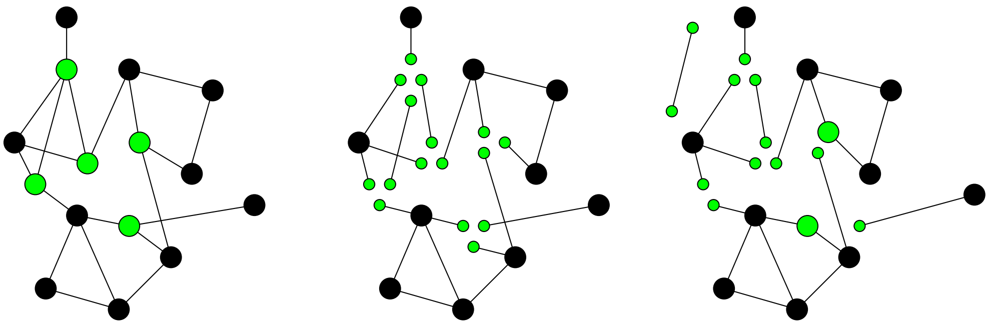


Figure 14: Splitting of G proceeds in two steps. (i) replacing each split vertex $x \in A$ by end points of its incident edges, and (ii) identifying copies of x that reside in the same connected component of the intermediate graph. The resulting graph is $G^{\rangle G \langle}$.

base-pairs. In the following sections we describe how we developed a dynamic programming algorithm that counts the compatible sequences of a block and produces colourings during the backtracking procedure. For this purpose we approach the task from a more abstract graph theoretical perspective.

4.3 Ear Decompositon and Associated Graphs

Let $G(V, E)$ be a finite 2-connected graph with vertex set V and edge set E. Let $\mathfrak{E} = (P_0, P_1, \ldots, P_\mu)$ be an ear decomposition of G, i.e., a sequence of paths P_i such that

(E0) P_0 is a single vertex.

(E1) The endpoints of P_i are contained in the union of the paths P_j, $j < i$

(E2) The graph G is the union of all paths in $\mathfrak{E}$.

It is well known that a graph has an ear decomposition if and only if it is biconnected [87]. Furthermore $\mu = |E| - |V| + 1$, the cyclomatic number of G [20].

We define two series of partial graphs associated with the ear decomposition $\mathfrak{E}$:

$$G_k = \bigcup_{i=0}^{k} P_i \qquad \overline{G}_k = \bigcup_{i=k+1}^{\mu} P_i \qquad A_k = G_k \cap \overline{G}_k \tag{8}$$

We observe that G_k is biconnected for all $k > 0$. Be definition $G_0 = P_0$, $G_1 = P_1$, $G_\mu = G$, $\overline{G}_0 = G$ and $\overline{G}_\mu = \varnothing$, the empty graph. We therefore have

$$\overline{G}_k = P_{k+1} \cup \overline{G}_{k+1} \tag{9}$$

The graphs $\overline{G}_k$ are the *complementary* graphs of each decomposition step and are not necessarily connected. The graphs A_k are disconnected and consist of the *attachment points* of $\overline{G}_k$ on G_k.

Furthermore we define the *width* $\alpha(\mathfrak{E})$ of an ear decomposition $\mathfrak{E}$ of G as

$$\alpha(\mathfrak{E}) = \max_{0 \le k \le \mu} |A_k| \tag{10}$$

i.e., the maximal number of attachment points along the decomposition sequence.

It seems natural to define the *ear width* of a graph as

$$\alpha(G) = \min_{\mathfrak{E}} \alpha(\mathfrak{E}) \tag{11}$$

and to ask whether this parameter can be determined (efficiently).

4.4 Colourings

A vertex *colouring* of G is a map $V \to \mathfrak{A} : c \mapsto c_x$ where $\mathfrak{A}$ is an "alphabet" of colours and c_x denotes the colour assigned to the vertex x. We will write this map as $\mathbf{c}$. Let $H \subseteq G$. A partial colouring $\mathbf{c}_W$ of H is a map $W \to \mathfrak{A}$ for some subset $W \subseteq V(H)$. Our interpretation of a partial colouring is that all colour assignments are considered on the vertices $V(H) \setminus W$. If $U' \cap U'' = \emptyset$ then $\mathbf{c}_{U' \cup U''} = \mathbf{c}_{U'} \circ \mathbf{c}_{U''}$, the concatenation of the colouring map on U' and U'', i.e., the map consisting of colouring both U' and U''. Two colourings on U' and U'', respectively, are consistent if their restrictions to $U' \cap U''$ are identical.

Let $\mathbb{X}$ be a set endowed with two operations $\wedge$ and $\vee$ that are associative and commutative. We consider an abstract evaluation Ω of partial colourings on subgraphs of G with values in $\mathbb{X}$ that satisfy the following conditions:

$$\begin{aligned} &\Omega(H, \mathbf{c}_U) = \bigvee_{\mathbf{c}_{W \setminus U}} \Omega(H, \mathbf{c}_U \circ \mathbf{c}_{W \setminus U}) \qquad \text{for all } U \subseteq W \\ &\Omega(H, \mathbf{c}_U) = \Omega(H_1, \mathbf{c}_{U \cap V(H_1)}) \wedge \Omega(H_2, \mathbf{c}_{U \cap V(H_2)}) \\ &\qquad\qquad \text{for all } H_1, H_2 \subseteq H \text{ and all } U \text{ such that } V(H_1 \cap H_2) \subseteq U \end{aligned} \tag{12}$$

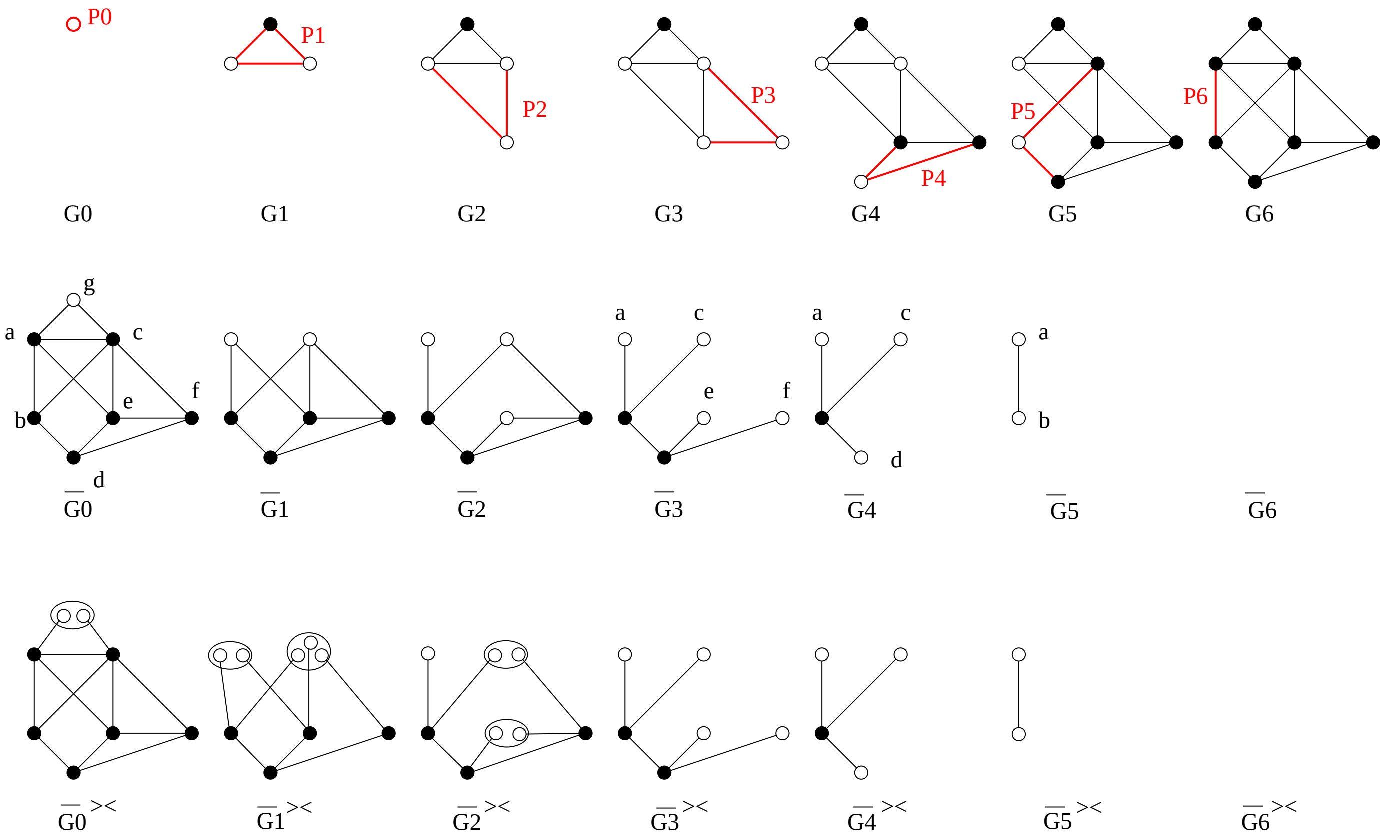


Figure 15: Graphs associated with an ear-decomposition. (top) The graphs G_k that form the union of the Ears P_k in each step are shown here. If we follow the graphs from the right side G_6 to G_0 on the left side, then we observe an ear decomposition: consecutively a path (red) is removed until a central cycle is left. The attachment vertices A_k are depicted as unfilled circles. (middle) For each decomposition step there is a complementary graph $\overline{G}_k$. (bottom) The graphs $\overline{G}_k^{\rangle\langle}$ result from splitting at the attachment vertices as illustrated.

Table 1: A few colouring problems.

Question	$\Omega(e; c', c'')$	$\bigwedge$	$\bigvee$
minimal number of colour conflicts	$\delta(c', c'')$	$+$	min
number of conflict free colourings	$1 - \delta(c', c'')$	$\times$	$+$
partition function of colourings	$\exp(-\delta(c', c''))$	$\times$	$+$
list all conflict free colourings[†]	$[c', c'' \mid c' \neq c'']$	$\circ$	$\cup$
count compatible RNA sequences	$1 \iff \{c', c''\} \in \mathcal{B}$	$\times$	$+$

[†] The concatenation $\circ$ of two lists of colourings is defined as the concatenation of all pairs of colourings.

The second equation allows in particular to write $\Omega(H, \mathbf{c}_{V(H)})$ in terms of the edges of H:

$$\Omega(H, \mathbf{c}_{V(H)}) = \bigwedge_{e \in E(H)} \Omega(e, \mathbf{c}_e) \tag{13}$$

where $\Omega(e, \mathbf{c}_e) = \Omega(e, c_x, c_y)$, is the evaluation of a particular colouring of the two end points x, y, of the edge $e = \{x, y\}$.

It follows that we can evaluate the set of all colouring on a graph formally as

$$\Omega(G, \varnothing) = \bigvee_{\mathbf{c}_V} \Omega(G, \mathbf{c}) = \bigvee_{\mathbf{c}_V} \left(\bigwedge_{e \in E} \Omega(e, \mathbf{c}_e) \right) \tag{14}$$

Note that this is by no means surprising since $\bigwedge_{e \in E}$ tells us to consider all edges of G and $\bigvee_{\mathbf{c}_V}$ means to consider all $|\mathfrak{A}|^{|V|}$ colouring maps. The question hence becomes whether we can use equ.(12) to compute $\Omega(G, \varnothing)$ more efficiently than the brute-force approach of equ.(14).

Graph colouring is a well known NP-complete problem ([36]). Of course our approach cannot overcome this in general. We can, however, search for a decomposition of G that allows us to apply equ.(12) with acceptable resource requirements.

A few colouring problems, like for example the number of conflict free colourings given a graph and a set of colours, are summarized in Table 1.

Erlaeuterung der colouring-probleme wird eingefuegt

Let us now return to the ear decomposition of G. We are interested in evaluating a colouring scheme for G. We do this iteratively for the subgraphs $\overline{G}_k$, starting with $\overline{G}_{\mu-1}$ and proceeding towards $\overline{G}_0 = G$.

We consider the step from $\overline{G}_{k+1}$ to $\overline{G}_k$. The attachment vertices of $\overline{G}_k$ are $A_k = G_k \cap \overline{G}_k$. Recall that $\overline{G}_k = P_{k=1} \cup \overline{G}_{k+1}$. We have to distinguish two classes of attachment of $\overline{G}_{k+1}$ points:
(1) The set $A_{k+1} \setminus A_k$ contains those attachment points of $\overline{G}_{k+1}$ that are buried in the interior of P_k and hence will not play a role in further steps.
(2) The set $A_{k+1} \cap A_k$ contains those attachment points of $\overline{G}_{k+1}$ that are also attachment points of $\overline{G}_k$. These points either are the end-points of the path P_k or they are not contained in P_k at all.
The points of P_k are divided into the interior points $A_{k+1} \setminus A_k$ and the end-points A'_k of P_k.

We can now directly apply the recursion (12) to obtain the general recursion for passing from $\overline{G}_{k+1}$ to $\overline{G}_k$:

$$\Omega(\overline{G}_k; \mathbf{c}_{A_k}) = \bigvee_{\mathbf{c}_{A_{k+1}\setminus A_k}} \left[\Omega(\overline{G}_{k+1}; \mathbf{c}_{A_{k+1}\setminus A_k} \circ \mathbf{c}_{A_{k+1}\cap A_k}) \wedge \Omega(P_k; \mathbf{c}_{A_{k+1}\setminus A_k} \circ \mathbf{c}_{A'_k}) \right] \quad (15)$$

The path P_k is subdivided by the points in $A_{k+1} \setminus A_k$ into $|A_{k+1} \setminus A_k| + 1$ subpaths for which the weights can be computed recursively. Suppose the vertices $x \in (A_{k+1} \setminus A_k) \cup A'_k$ are ordered linearly along the path so that x_0 is the inital vertex and x_L is the terminal one. Thus

$$\Omega\big((x_0, ..., x_L); \mathbf{c}_{\{x_i|0\leq i\leq L\}}\big) = \bigwedge_{k=1}^{L} \Omega(e_k; c_{k-1}, c_k) \quad (16)$$

It follows that effort for this step is $\alpha^{|A_{k+1}\cup A_k|}$ since we need to evaluate all possible colouring of the interior attachement vertices ($x \in A_{k+1} \setminus A_k$) for all possible colourings of the exterior attachement vertices ($x \in A_k$). The evaluation of $\Omega(\overline{G}_{k+1}; \dots)$ is a single table lookup, while the evaluation of $\Omega(P_k; \dots)$ requires $L \leq |A_{k+1}| + 2$ table lookups and $\wedge$ operators. The performance of this way of colouring the graph is therefore determined by the maximal number of attachment vertices in a decomposition step for $0 \leq k \leq \mu$, which is represented by the width $\alpha(\mathfrak{E})$ of the ear decomposition of the graph.

4.4.1 Implementation- Forward Recursion

Now that we have the fully decomposited graph and have obtained the ear decompositions of the blocks we initiate the counting and colouring routine of each connected component ψ of the dependency graph Ψ. We start by colouring blocks and then proceed outwards towards the paths that are connected via cut-vertices until each connected component is completely coloured. The cut-vertices are treated just like attachment points, since the mechanism of counting, colouring and concatenation is exactly the same.

The compatible sequences of blocks are counted recursively starting from the utmost ear proceeding inwards to the central circle. In each step of the counting procedure while concatenating the ears, we have to compute and memorize the number of compatible colourings of paths given each consistent assignment of the attachment points A_k.

Algorithm 4.4.1 summarises in pseudo-code the recursive dynamic programming procedure for determining the number of sequences that start with a certain base i and end in a certain base k for all path lengths $l \leq n$ (n is the maximum path length in our graph). We generate the matrix entries in the following manner: The number of paths with length l that start with base i and end in base k is the sum over all path matrix entries $\mathrm{PM}(i, j)$ of the path length $l - 1$ times 1, if (j, k) is an allowed base pair, and times zero if (j, k) cannot form a basepair. The solution to the smallest subproblem of this dynamic programming algorithm- finding assignments for paths of the length one- is clearly the base-pairing matrix. Proceeding bottom-up from this sub-instance we can generate the path matrices for paths of arbitrary length.

The implementation of this counting procedure proved to be very complex although it is based on a dynamic programming algorithm. The generation of the path colouring tables themselves is rather straight forward as the algorithmth 4.4.1 implies. It is the upkeep and update of the colouring matrices that store consistent assignments for attachment points and their multiplicity that result from each concatenation step which require a sophisticated table management. The reason is that as we concatenate the ears of the graph the table size does not always grow steadily but also decreases again when colourings become impossible due to constraints of attachment vertices in later steps.

When counting on the blocks is finished, the counting procedure continues in an analogous manner on all remaining biconnected components of each connected component, with the exception that path concatenation now considers fixed assignments of cut vertices rather than attachment points. The result is a table of

Algorithm 4 Counting Algorithm - constructing path matrices

```
Require: list of base-pairs $\mathcal{B}$, alphabet of bases $\mathcal{A}$, maximal path length $n$
while $length \leq n$ do
    for $k \leftarrow$ [A,U,G,C] do
        for $i \leftarrow$ [A,U,G,C] do
            for $j \leftarrow$ [A,U,G,C] do
                $entry = \sum PM(i,j)^{length-1} * canpair(j,k)$
            end for
            push entry into new row
        end for
        add row to new matrix
    end for
    push new matrix into list of path matrices
end while
```

consistent assignments for all attachment points and cutvertices and the number of compatible sequences, given each assignment, which is then used to sample colourings with uniform distribution by means of stochastic backtracking.

4.4.2 Stochastic Backtracking

In the backtracking procedure we start off with choosing an assignment for the first set of attachment vertices $a_1, a_2, ..., a_n$. The probability of choosing a certain assignment for these vertices is the number of of colourings that are possible given the chosen colours at the attachment points divided by the number of all possible assignments.

$$Z_{a_1..a_n} = \frac{|[a_1..a_n]|}{\sum_{a1..a_n} |[a_1..a_n]|} \tag{17}$$

Our task then is to generate internal colors for each path with given begin and end colors (at the attachmain points). The probability of choosing colour u at position k given that the colour z was chosen for position $k+1$ equals the number of paths that start in x and end in u at step k divided by the number of paths that start in x and end in z at step $k+1$ times 1 if u, z is an allowed base pair and times 0 if not (see Fig. 16).

$$Z_k = \frac{|[x,u;k]|}{|[x,z;k+1]|} \times \begin{cases} 1 & : \quad if(u,z) \in \mathcal{B} \\ 0 & : \quad otherwise \end{cases} \tag{18}$$

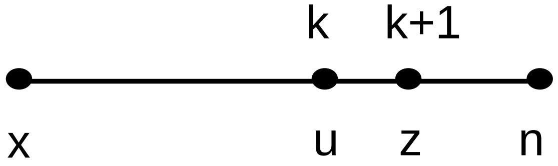


Figure 16: Stochastic Backtracking

At each step we generate a random number ξ that has a value between 0 and 1. We add up the probabilities Z_k until the sum of Z_ks is greater or equal to ξ.

$$\sum_{k=1}^{n} Z_k \geq \xi \tag{19}$$

We draw the colour of the respective Z_k we reached when the ξ constraint was fulfilled. This procedure is comparable to turning a roulette wheel where each colour occupies a segment of the wheel where the size corresponds to the number of colourings that are consistent with this particular color at step k. It follows that the color with the largest segment is also the most likely one to be chosen.

The backtracking continues in this way until the whole graph is coloured and the obtained sample sequence can be fed into the optimization algorithm.

4.5 Applicability of the Design Algorithm

Our algorithm can easily be adapted for different colouring tasks, as outlined in table 1, by simply defining another set of pairing rules. We can also provide fast solutions for colouring problems that are not in a biological context. The important constraint though is that the underlying graph displays a small value for the graph width $\alpha(G)$. Alternatively to accepting secondary structures and thus constructing graphs, our design tool also processes files in the GML format- the standard file format for many graph software packages. The width of the input graph can be calculated by means of the design program and thus determined whether the graph can be coloured efficiently using our algorithm.

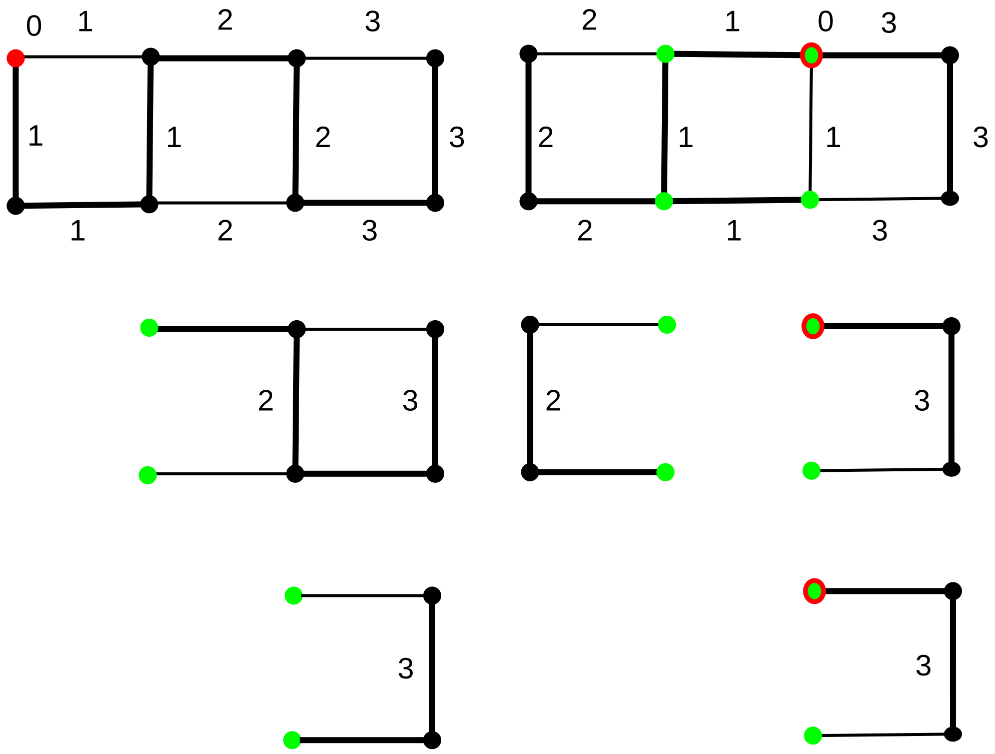


Figure 17: The width of an ear decomposition $\alpha(\mathfrak{E})$, i.e. the maximal number of attachment points in a decomposition step, depends on the selected spanning tree. Bold lines illustrate the tree edges in the graph, the root vertex of the ear decomposition is coloured red and the respective attachment points green. The numbers show the labels of each ear-edge. If one of the outer squares is the central circle of the decomposition, $\alpha(\mathfrak{E}) = 2$. If the central square also reprensents the central circle of the decomposition, then $\alpha(\mathfrak{E}) = 4$.

The width of an ear decomposition $\alpha(\mathfrak{E})$ is directly associated with the spanning tree that generates the ear-labeling. Fig. 17 illustrates how different spanning trees imply different ear decompositions and thus a different number of attachment points in each decomposition step. Therefore, a tool that finds spanning trees that minimize $\alpha(\mathfrak{E})$ would help cutting down on computation time and make our algorithm even more efficient. In chapter 7 we describe possible approaches for solving this task.

5 Optimization

Uniform samples of colouring (i.e., sequences that are compatible with all prescribed secondary structures) can be obtained by means of a standard backtracking procedure when the numbers of colourings with given colors on the Attachment vertices A_k and cut-vertices are tabulated for each k. These sequences must then be used to initialize optimization heuristics.

The RNA folding problem has been solved by dynamic programming procedures [92, 90, 51] that make use of a so-called *nearest neighbour model* for which most energy parameters were carefully measured [35, 84, 50]. The `Vienna RNA-Package` for example evaluates a folding function Φ to compute the secondary structure $\Phi(x)$ of a given sequence x [32]. It is straight forward to invert this combinatoric problem and look for a sequence x given a certain secondary structure Θ. Thus we write for the *inverse folding problem*:

$$\text{Find } x \in \mathbf{C}[\Theta] \text{ such that } \Xi(x) = D(\Theta, \Phi(x)) \rightarrow min. \tag{20}$$

which means that the sequence x can only fold into the structure Θ, i.e. $\Phi(x) = \Theta$, if the structural distance $D(\Theta(x), \Phi(x))$ equals zero (implemented in the `RNAinverse` program of the excellent `Vienna RNA-Package`). For most design purposes is it sufficient to just minimize the distance.

The combinatorial optimization problem can be easily solved by means of *adaptive walks* that search the set of compatible sequences for those that optimize a cost-function Ξ. The algorithm starts with a random colouring sample and then mutates single nucleotides at unpaired sequence positions Υ or exchanges base pairs in Θ. A mutant is accepted, if the value for the cost function $\Xi(x)$ decreases. For example we can modify equation 20 in such a manner, that the ground state is more stable than any structural alternative. Be $E(x, \Theta)$ the energy of the structure Θ of sequence x and $G(x)$ the Free Energy of sequence x that can be computed using the McCaskill-algorithm [51], then we can find suitable sequences by minimizing the following equation:

$$\Xi(x) = E(x, \Theta) - G(x) = -RT \ln p \tag{21}$$

where p denotes the probability of the structure Θ in the Boltzmann-ensemble of the sequence x.

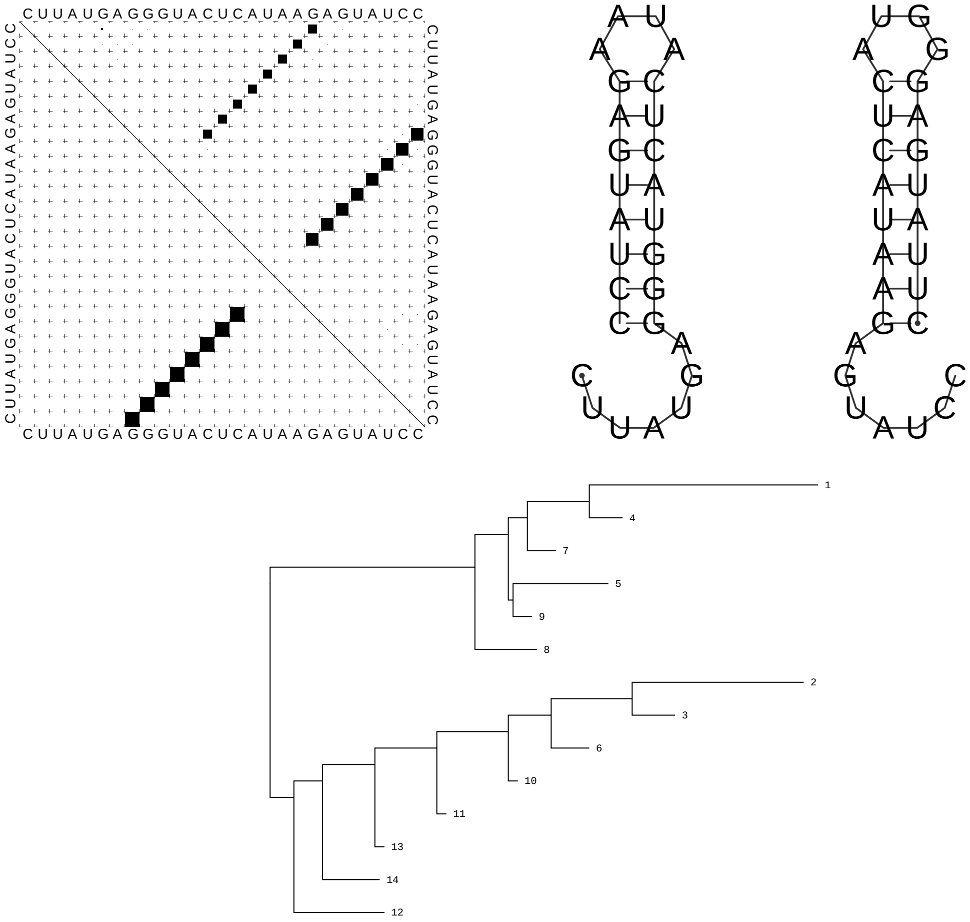


Figure 18: Example of a bistable switch molecule. (top from left to right) Dotplot: both structures have approximately the same statistical weight within the thermodynamic equilibrium; MFE structure; Metastable structure; note that the two structures have no base pair in common; (below)Tree of local minima: Shown are the two (meta-)stable conformations, that are seperated by an energy barrier of ~ 11.2kcal/mol.

The function Ξ encapsulates desired properties of the molecule, so we can modify equation 20 in a similar manner to construct cost functions Ξ for more complicated design problems such as (nearly) equal energies for all prescribed secondary structures and constraints on the energy barriers between these metastable states. Ξ is only definded for sequences that fold into all predescribed structures, thus the optimization should be restricted to the interesection $\bigcap_j \mathbf{C}[\Theta_j]$ for j given structures. An example of such a designed RNA switch is shown in Fig. 18.

Having optimized our output sequences we want to determine if our design goal was also met with regard to thermodynamic properties. Thus, we need a means of evaluating our results. In the next section we describe how the energies of RNA secondary structures are determined and how they are visualized by energy landscapes and barrier trees.

5.1 Energy Landscape of RNA Molecules

RNA secondary structures can be decomposed uniquely into a set of loops of different types: stacked base pairs, bulges, interior loops and multi-branched loops 1.3.3. The standard energy model **??** describes the energy of an RNA secondary structure as a sum of sequence-dependent contributions for each loop. Dynamic programming algorithms are known to exactly and efficiently compute the minimum free energy structure **??**, the base pairing probability matrix **??**, the density of states **??**, certain sub-optimal structures **??** or all structures with an energy below a threshold value **??**. A suite of these algorithms is implemented in the

`Vienna RNA package`

, which was used to evaluate the success of our design algorithm **??**.

Within the framework of the folding landscape we can meaningfully speak of local minima or metastable states, their basins of attraction, and the saddle points separating them. Formally, a structure $x \in X$ is a local minimum of E if $E(x) \leq E(y)$ for all its neighbors, $(x, y) \in \mathfrak{M}$. A gradient walk is defined as follows: starting from $x \in X$ we move to its neighbor y with minimal energy if $E(y) < E(x)$. If the minimum energy neighbor y of x is not uniquely defined we use a deterministic rule to break the tie, for instance, by choosing the structure that comes lexicographically first. The step from x to $y = \gamma(x)$ is repeated until we reach a local minimum where the walk terminates, $\gamma(x) = x$. The local minima are therefore the attractors of the map $\gamma : X \to X$ and each $x \in X$ is mapped to a unique local minimum $z = \gamma^{\infty}(x) = \gamma^{t}(x)$ by a finite number t of applications of γ. The basin of attraction of a local minimum z, $\mathcal{B}(z)$, consists of all structures that are mapped to it by the gradient walk, i.e. $\mathcal{B}(z) = \{x \in X | \gamma^{\infty}(x) = z\}$. Below we will need the (trivial) fact that these "gradient basins" of the local minima form a partition of X.

Let us now turn to the transitions between local minima. The energy of the lowest saddle point separating two local minima x and y is

$$E[x,y] = \min_{\mathbf{p}\in\mathbb{P}_{xy}} \max_{z\in\mathbf{p}} E(z) \tag{22}$$

where $\mathbb{P}_{xy}$ is the set of all paths $\mathbf{p}$ connecting x and y by a series of subsequent moves. The saddle-point energy $E[\,.\,,\,.\,]$ is an ultra-metric distance measure on the set of local minima, see e.g. [62].

In the simplest case the energy function is non-degenerate, i.e., $f(x) = f(y)$ implies $x = y$. Then there is a unique saddle point $s = s(x,y)$ connecting x and y characterized by $E(s) = E[x,y]$. This definition of a saddle point is more restrictive than in differential geometry where saddles are not required to separate local optima. [78]. For each saddle point s there exists a unique collection of configurations $\mathcal{V}(s)$ that can be reached from s by a path along which the energy never exceeds $E(s)$. In other words, the configurations in $\mathcal{V}(s)$ are mutually connected by paths that never go higher than $E(s)$. This property warrants to call $\mathcal{V}(s)$ the *valley below the saddle* s. Furthermore, suppose that $E(s) < E(s')$. Then there are two possibilities: if $s \in \mathcal{V}(s')$ then $\mathcal{V}(s) \subseteq \mathcal{V}(s')$, i.e., the valley of s is a "sub-valley" of $\mathcal{V}(s')$, or $s \notin \mathcal{V}(s')$ in which case $\mathcal{V}(s) \cap \mathcal{V}(s') = \emptyset$, i.e., the valleys are disjoint. This property arranges the local minima and the saddle points in a unique hierarchical structure which is conveniently represented as a tree, termed *barrier tree* (see Fig. **??**). Since saddle points separate local optima, each valley $\mathcal{V}(s)$ contains (in the non-degenerate case at least two) local minima $z_1, \dots, z_k$. Conversely, $\mathcal{V}(s) \subseteq \bigcup_k \mathcal{B}(z_k)$, i.e., the valley of s is contained in the union of the basins of attraction of the metastable states "below" s. The metastable states therefore form the tips (or leafs) of a tree. We calculate this barrier tree by a software package called

`barriers`

. This program constructs the barrier tree directly from an energy sorted list of all configurations **??**. Starting with the lowest energy configurations,

`{barriers}`

explicitly builds the valleys $\mathcal{V}(s)$ and subtrees by checking for each configurarion whether it (a) is a local minimum, (b) uniquely belongs to the basin of a local minimum that was encountered earlier in the list or (c) merges two or more basins (which equals a saddle point). The

`{Vienna RNA package}`

also forms the basis for the computations of the barrier algorithm.

6 Computational Results and Discussion

In the previous chapters the theoretical background as well as the underlying models and algorithms were introduced. We will now give some examples of our calculations on this basis.

6.1 Feasibility of Mulit-stable Switches

In order to get at least a rough idea on how easy or hard it is to design multi-stable switches we consider the probability that the intersection $\bigcap_i \mathbf{C}[\Theta_i]$ is non-empty for random samples of n secondary structures. Otherwise, corresponding switches obviously do not exist. As proven in [23] the dependency graphs of bistable switches are always bipartite, consequently, compatible sequences can always be found. The ease of designing bistable switches indicates that RNA switches are equally easily accessible in evolution which is also confirmed by experimental data, see e.g. [9, 8, 18, 60, 15, 13] for a wide variey of experimentally known bistable self-induced RNA switches. Self-induced RNA switches thus seem to be a wide-spread and elegant way to regulate biological activity that has to be limited to a certain time window.

The situation is different with $s \geq 3$ structures since it is simple to construct triples of structures with conflicting base-pairs that lead to a triangle in Ψ (and thus a non-bipartite graph). In order to estimate the probability of a non-empty intersection we sample random structures (generated by means of stochastic backtracking as described in [71]) and check whether their intersection graph is bipartite. For $s \geq 3$ we find an exponential decrease with sequence length, see Fig. 19. The exponent is very small for $s = 3$, however, so that tri-stable switches do not seem to be evolutionary inaccessible.

So far no naturally occuring *self-induced* s-stable RNA switches with $s \geq 3$ that are completely self-induced have been described in literature. One switch system that comes close to a self-induced tri-stable RNA switch is the *Hok/Sok* system of plasmid R1 in E.coli [29, 27], which regulates gene expression via an intricate cascade of secondary structural rearrangements. The Hok/Sok system mediates plasmid maintenance by expressing the Hok toxin which kills plasmid-free segregates. The plasmid encodes for a highly stable mRNA, which is translated

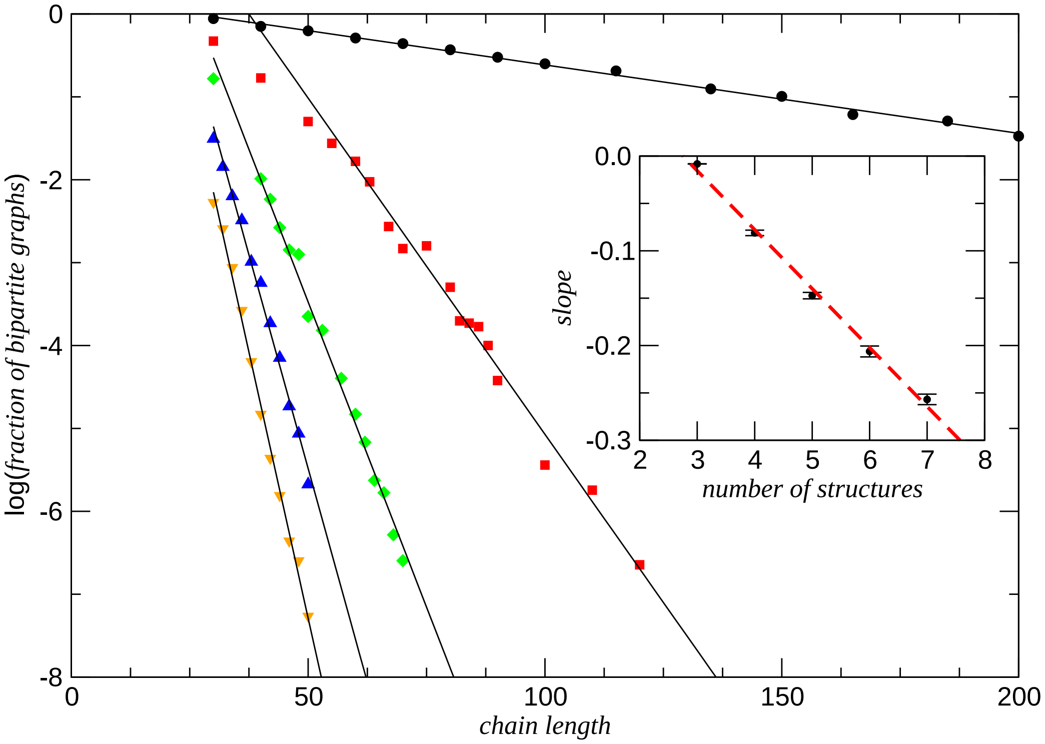


Figure 19: Statistics of the fraction of bipartite graphs versus the sequence length. Shown here are the results for switches with different numbers s of pre-defined structures. ● $s = 3$, ■ $s = 4$, ◆ $s = 5$, ▲ $s = 6$ and ▼ $s = 7$.

to the Hok toxin if the mRNA is in its activated conformation, and a labile anti-sense RNA (Sok) which acts as an antidote by binding to the activated *hok* mRNA, leading to rapid degradation of the resulting duplex. When completely transcribed, the *hok* mRNA switches from the unfavourable inactive structure it adopted during transcription to a more stable structure without the help of any effector molecule and thus forms a pool of stable inactive mRNAs. The next switching step is induced in trans by the RNA degradation machinery which, in time, truncates the 3'-end and causes the *hok* mRNA to refold into yet another structure that is finally translationally active. Then both locations, the Hok gene and the Sok binding site, become accessible. If the plasmid was lost, the pool of antidote Sok is quickly deptleted, since it is conseiderably less stable than the *hok* mRNA. Hence, Hok protein is produced and the killing of the cell induced. For a recent review on self-induced RNA switches see [54, 11].

6.2 Characterization of RNA Dependency Graphs

Since it is necessary to decompose the dependency graphs of nucleic acid switches in the course of the design algorithm, we were particularly interested in the features of these graphs. For this purpose we, again, generated random RNA-structures as described in the previous section and computed several statistics.

6.2.1 The Width of the Dependency Graphs

The most important feature of the graphs in connection with our algorithm is the width $\alpha(G)$, as it determines the computational "costs". We conducted a statistic analysis on tri-stable switches for sequence lengths reaching from 30 to 140 nucleotides. The result we obtained showed that the average ear-width $\alpha(\mathfrak{E})$ was almost exactly 2.0 independent from the sequence length. It follows that the graph width is small enough for our design algorithm to work efficiently.

6.2.2 Tree-likeness of Dependency Graphs

Fig. 20 depicts the number of tree-edges, i.e. edges that are not part of a block component, which increases almost precisely in a linear manner in dependence of the sequence length. Switches of three and four predefined structures display almost the same slope, whereas the increase of tree-edges in switches with five predefined structures is less drastic. This can be explained by the fact that switches with $s = 5$ define much more complex dependency graphs and thus the ratio of block edges rises. The statistics for $s = 5$ switches stop at a sequence length of $n = 70$ nucleotides, since the probability of defining a bipartite dependency graph dramatically decreases at a bigger sequence length as shown in the statistics of Fig. 19. Therefore, it cannot be considered reasonable to attempt designing switches beyond this number of nucleotides.

In Fig. 21 we proceeded by exploring the average number of ear decompositions of the dependency graphs with s predefined structures, which rises linearly with the sequence length. Interestingly, we observe one ear decomposition per dependency graph on average starting from switches of four predefined structures and a length of at least 60 nucleotides.

The effective number of ear edges, i.e. edges that are part of a block, is depicted

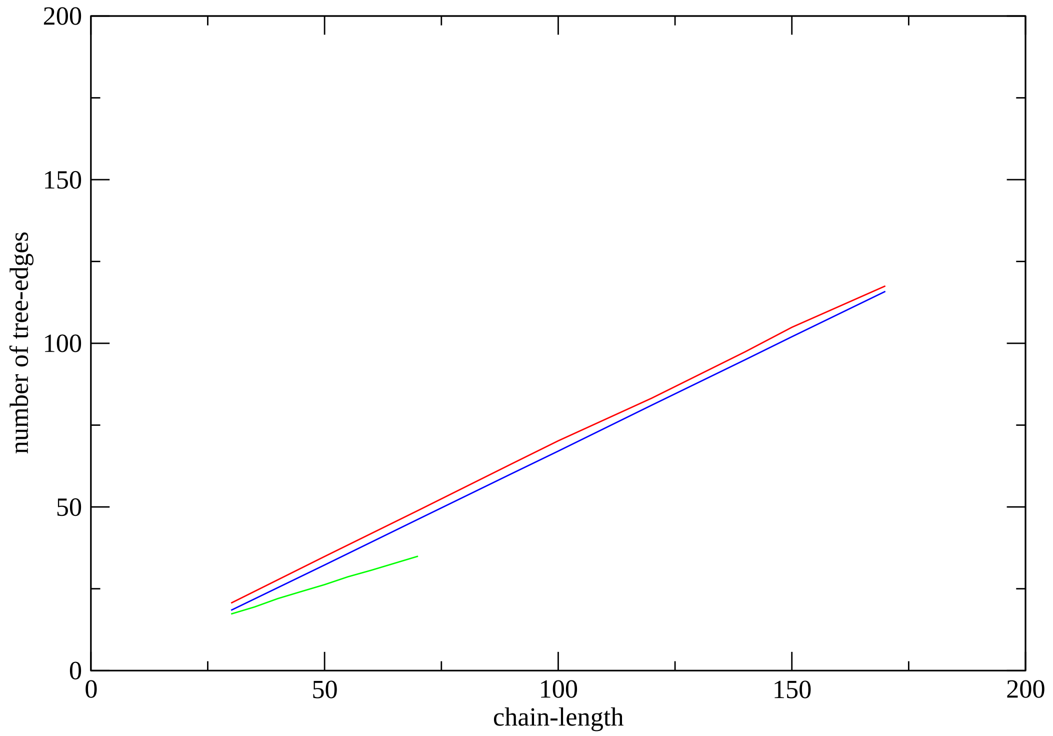


Figure 20: Statistics of the number of tree-edges in dependency graphs of switch structures versus sequence length. Shown are the results for switches with different numbers s of predefined structures: $\circ$ $s = 3$, $\triangle$ $s = 4$, $\square$ $s = 5$.

in Fig. 6.2.2. Remarkably, the number of ear-edges of switches with four and five structures is identical. This means that, according to Fig. 20 switches of four predefined structures have about the same number of tree-edges on average as switches of three structures, at the same time they display the same number of ear-edges as switches of five structures, although more ear-decompositions occur with the latter.

If we compare the number of tree-edges and ear-edges we can clearly conclude that the dependency graphs of multi-stable nucleic acid switches are primarily of a *tree-like structure*.

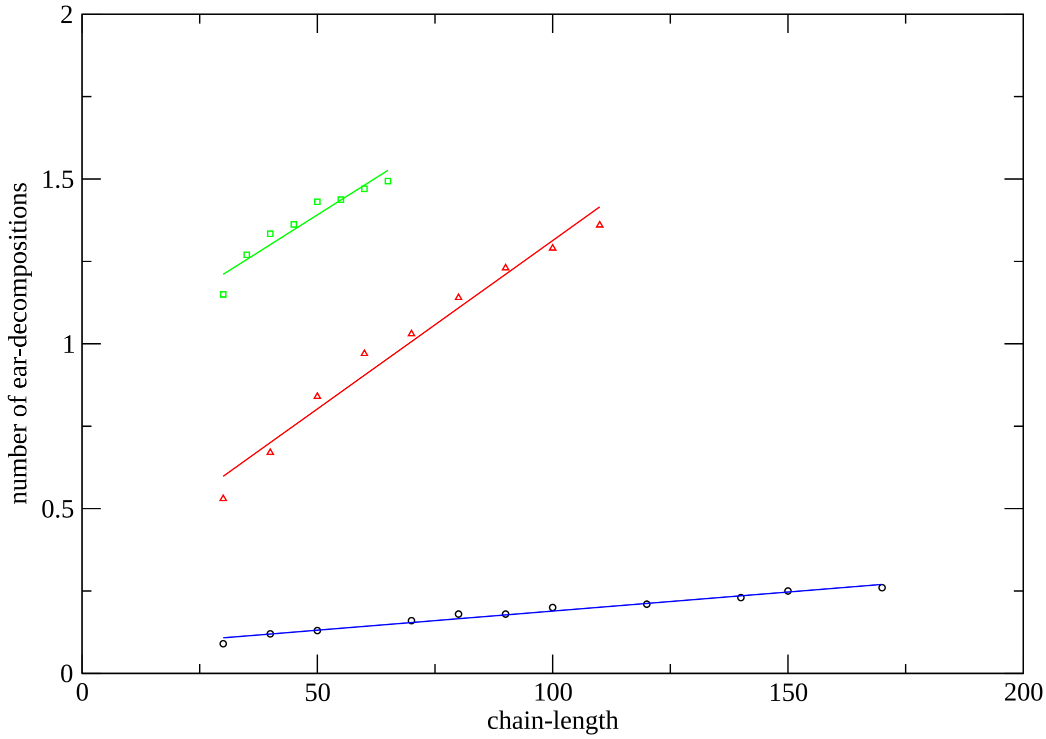


Figure 21: Statistics of the number of ear-edges in dependency graphs of switch structures versus sequence length. Shown are the results for switches with different numbers s of predefined structures: ○ $s = 3$, △ $s = 4$, □ $s = 5$.

6.3 Example applications of the design algorithm

6.3.1 Example 1

In order to find a set of three samples for the switch design we generated random RNA structures of the length $n = 40$ and obtained the following structures (see Fig. 23). The respective dependency consists of seven connected components and 21 isolated vertices in this graph. (see Fig. 23). The complex connected component is now decomposed into a block and two paths and then all the components are fed into the colouring procedure.

For the connected component containing the block there are 1072 different colouring possibilities, i.e. 1072 sequences that fulfill the structure constraints. Whereas these sequences can all theoretically fold into the predescribed structures, the optimization steps shows that the secondary structures that generate the block are not very stable, since the resulting energies are rather unfavourable. Thus, a

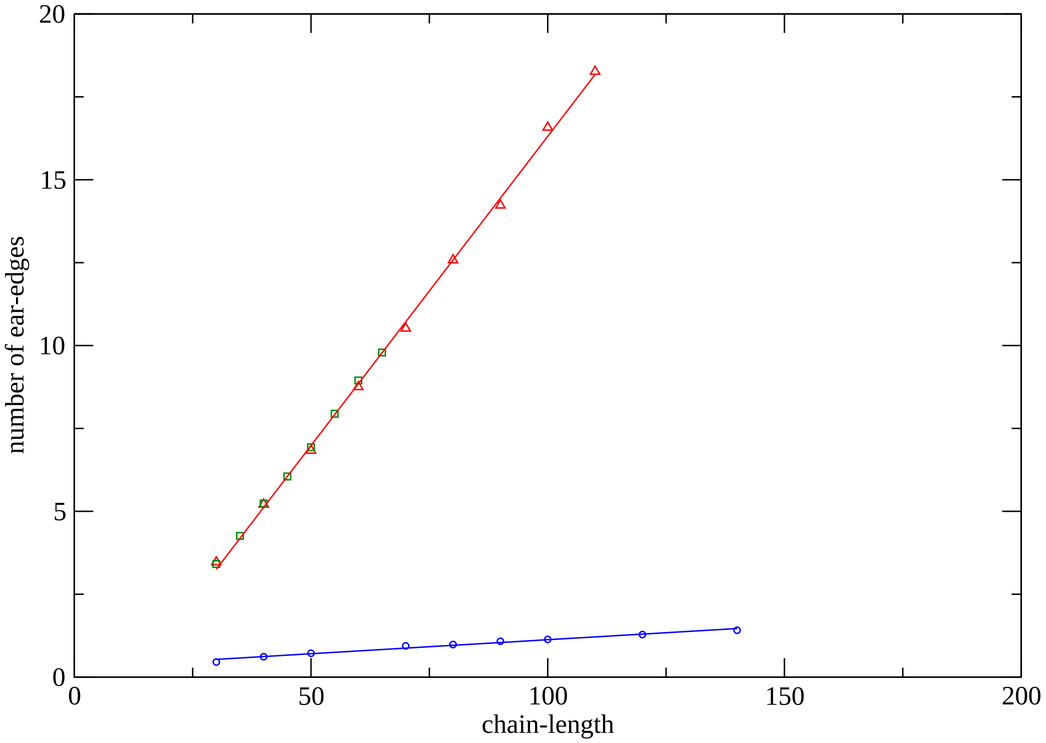


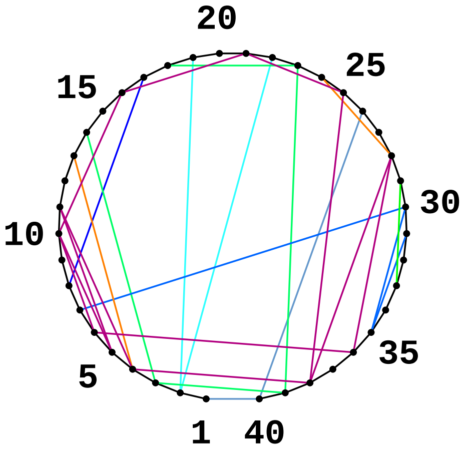
Figure 22: Statistics of the number of ear-decompositions in dependency graphs of switch structures versus sequence length. Shown are the results for switches with different numbers s of predefined structures: $\circ$ $s = 3$, *triangle* $s = 4$, $\square$ $s = 5$.

suitable *optimized* sequence can not be found. This can be explained by the fact that the randomly generated structures contain many isolated basepairs, as we did not put any constraints on the stack-size.

6.3.2 Example 2

Biologically relevant RNA structures usually exhibit a minimum stack-size of at least three or four basepairs. We therefore generated a new random sample set of structures that incorporates this constraint. Fig. 24 shows the selected structures, the resulting dependency graph and the connected components that are fed into the colouring algorithm.

Expectedly, this time the optimization run was more successful and we could observe the cost function scores drop during the adaptive walk. Yet, further examination proved that none of the predescribed structures were minimum free energy (mfe) structures for the optimized output sequences. We conclude that



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

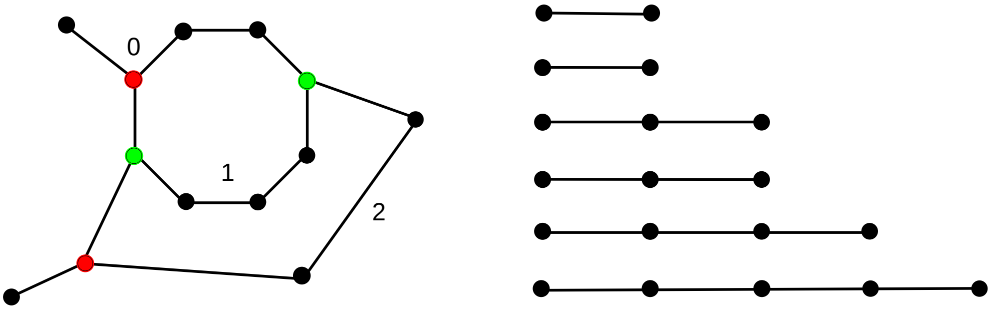
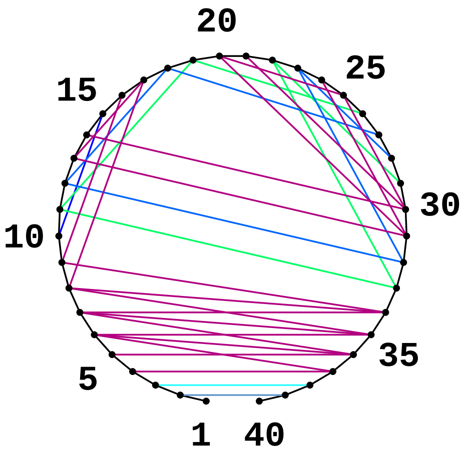


Figure 23: Sample set of structures for the design algorithm in bracket-dot notation and the resulting dependency graph. (top) The graph consists of six connected components that are paths of length 1-4 (blue, green) and a complex connected component containing a block (red, purple). (bottom) The complex component consists of two paths of the length 1 connected to a block via cut-vertices (red). The numbers 0,1,2 label the root and the two ears of the ear decomposition, the corresponding attachment vertices are coloured in green.



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

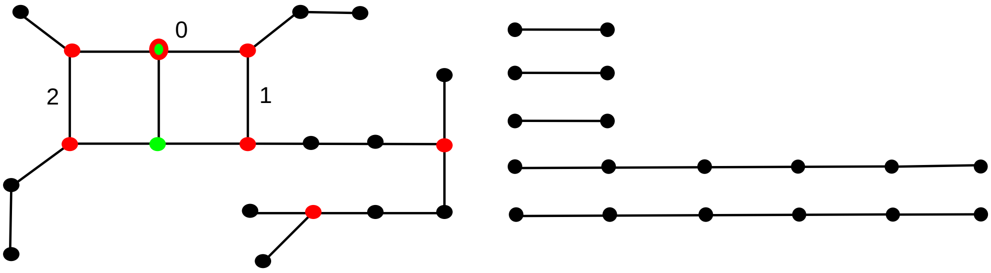


Figure 24: Second sample set of structures for the design algorithm in bracket-dot notation and the resulting dependency graph. (top) The graph consists of five connected components that are paths of length 1 and 5 (blue, green) and a complex connected component containing a block (purple). (bottom) The complex component is made up of three paths and a tree-like component connected to a block. The numbers 0,1,2 label the root and the two ears of the ear decomposition, the corresponding attachment vertices are coloured in green. Cut-vertices are depicted in red.

in this case an adaptive walk did not present the ideal method of optimization, since our walk was trapped in a local minimum.

6.3.3 Example 3

As a third sample set we chose the structures shown in Fig.25. We examined the optimized output sequences with the Barriers software **??** and identified the candidate CCGCACAGCGGGCAGUGCCC as an ideal switch-sequence for the predescribed structures. Fig. 26 depicts the output of the Barrier program. The dotplot shows three (sub)optimal structures on the upper right side and the mfe on the lower left. These structures are consistent with the three predefined structures of the design task. The barrier tree in Fig. 26 confirms the results of the dotplot: There are three minima (numbered 1-3) that correspond to the pre-described structures, one of them being the mfe structure ((((...)))).........

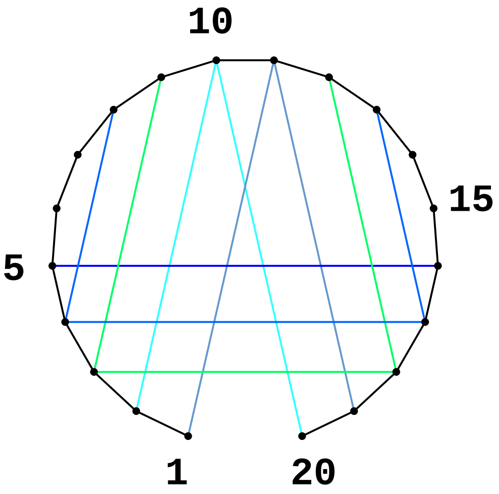
6.4 Dependency Graphs of Switches with Stack-size constraint

6.4.1 Characterization

Since we introduced a new constraint on our sample set, a minimum stack-size for stems in the secondary structure, we were interested in how this affects the features of our dependency graphs. In particular we wanted to know whether the resulting dependency graphs display more complicated connected components. Therefore, we performed a statistical analysis of the number of ear decomposition versus the sequence length for dependency graphs of tri-stable switches with varying stack-size constraints.

Fig. 27 shows that the number of ear decompositions does in fact rise with the assigned stack-size. Yet, the absolute number of ear decompositions is still so small that the increase does not change anything about the principally tree-like nature of our dependency graphs.

Srictly speaking, cycles are also biconnected components, and are thus included in our previous statistics concerning ear decompositions. Yet, cycles do not have to be labeled in an ear decomposition in order to generate colourings and it follows



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

Figure 25: Second sample set of structures for the design algorithm

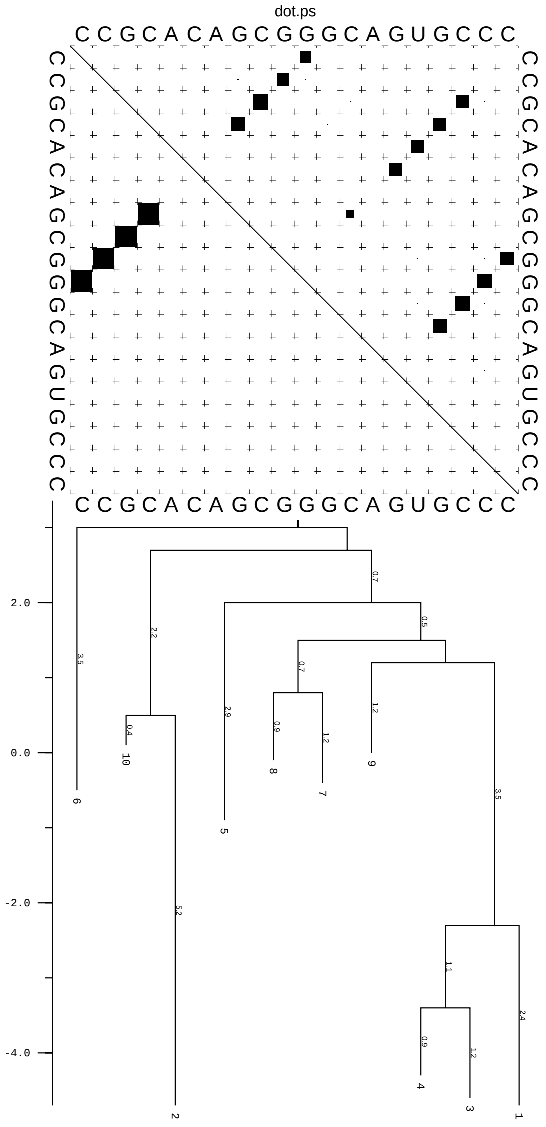


Figure 26: Dotplot and Barrier-Tree after Optimization

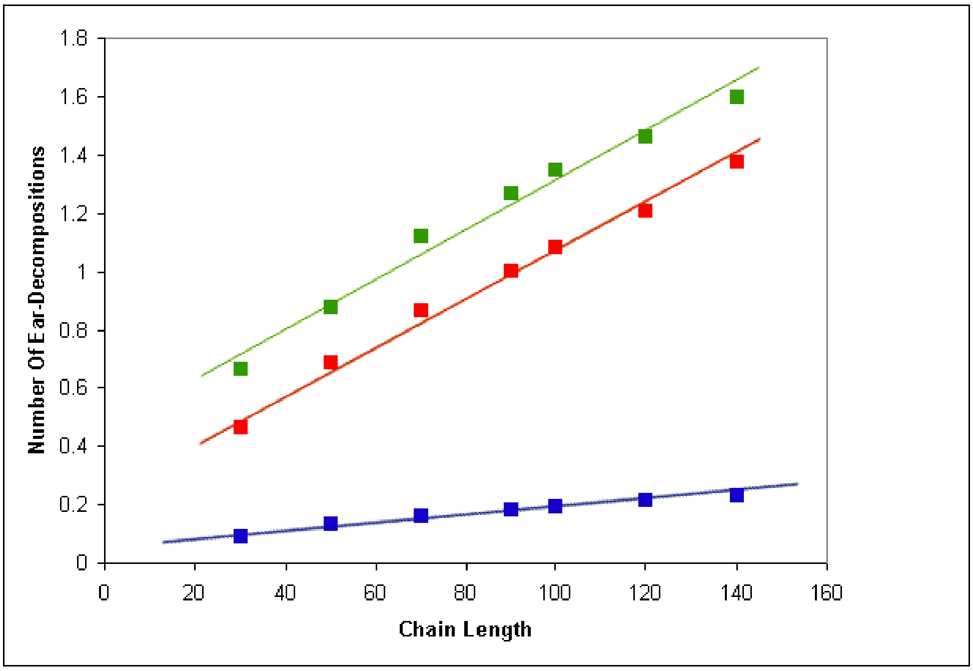


Figure 27: Comparison of the number of ear decompositions of dependency graphs with varying stack-size constraint. This calculation was conducted on tristable switches. Depicted are the fractions of ear-decomposition versus the sequence length for dependency graph with no stack-size constraint, with a minimum stack-size of three and with a minimum stack-size of four.

that the computational effort to find compatible sequences is far smaller than with complex components. To get an idea how frequent complex blocks are in our dependency graphs we issued another statistical analysis that counts ear decompositions only on complex blocks and not on cycles. As Fig. 28 illustrates, ear decompositions are a rather rare event, though they occur much more frequently in switches with stack-size constraints than in those without. The number of complex components in tristable switches with no stack-size constraint is in fact that small (between 1.5 and 4.6 per 1000 switches), that we included the results for the total number of ear decompositions instead.

6.4.2 Width of Dependency Graphs with stack constraints

As the previous section shows the number of ear-decomposition increases with a predescribed stack-size, we thus proceeded with investigating the impact on the width of the graphs $\alpha(G)$. A statistical analysis of dependency graphs of tristable switches with stack-size three and four, respectively, obtained the following

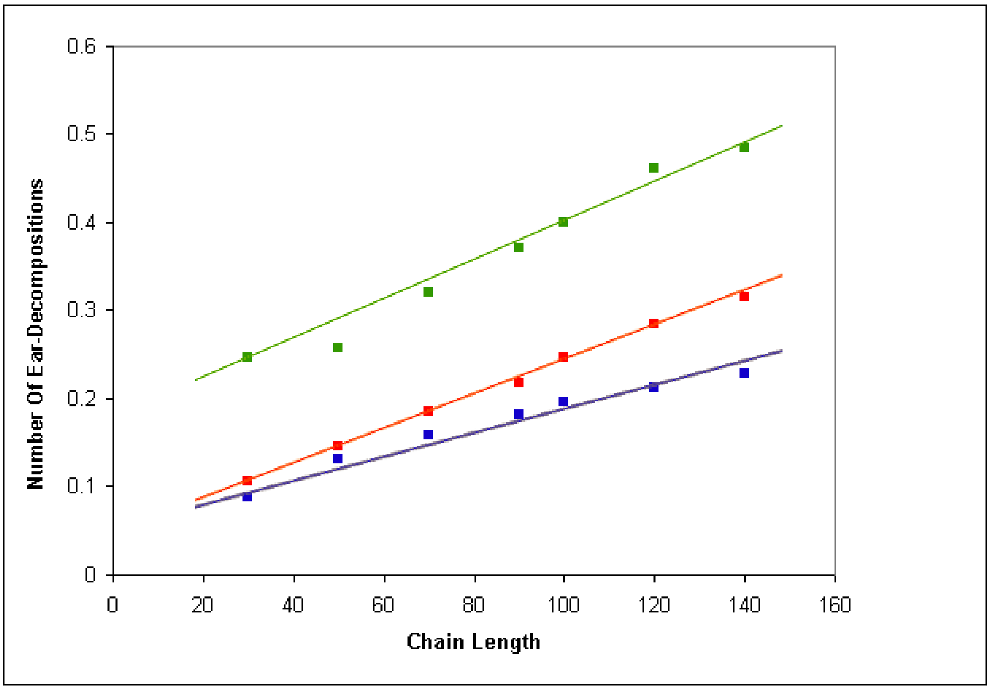


Figure 28: Comparison of the number of ear decompositions(not including cycles) of dependency graphs. This calculation was conducted on tristable switches with varying stack-size constraint. Depicted are the fractions of ear-decomposition versus the sequence length for dependency graphs with no stack-size constraint, with a minimum stack-size of three and with a minimum stack-size of four. The number of complex components in tristable switches with no stack-size constraint is that small, that we included the total number of ear decompositions (including cycles) in this figure instead.

results: On average, the maximal number of attachment points in a decomposition step is between 2.5 and 3 for switches of the length $n = 30$ to $n = 140$ and a stack-size constraint of three. The ear-width of switches with a stack-size constraint of four lies between 2.9 and 3.8. This analysis implicates that the width of these graphs is still sufficiently small for our algorithm to work efficiently.

7 Summary and Discussion

Sequence design represents an integral part of research on nucleic acids and is equally important for industrial applications. The development of new theoretical approaches and hence the implementation of new program packages that assist with the rational design of nucleic acids is therefore of fundamental ***interest***. ...

With the discovery of an abundance of riboswitches in recent years the ***interest*** in catalytically active RNA has vastly increased. Riboswitches perform vital tasks within the cell, they are of evolutionary as well as of functional interest and even lay the way to the development of a new class of drugs that are extremely target-specific. Against this background...

The aim of the work described in this thesis was the creation of a software tool to support the rational design of RNA molecules capable of forming two or more alternative metastable structures. This required the creation of a logical information model, thus isolating relevant aspects of the biological problem as posed, and incorporating these into a graph-based mathematical model. The algorithm we developed based on this model reduces the problem to vertex coloring the union of all prescribed outerplanar secondary structure graphs, called *dependency-graph*, which represents a completely new approach to nucleic acid design. Starting from a decomposition of this dependency graph, colorings are then produced by a dynamic programming procedure. In the final step sequences can then be optimized for particular properties by means of standard optimization heuristics. The connection between sequence design and vertex-colorings has hitherto not been described in literature.

We have not only solved the design problem in theory, but also provide a software package that is at free disposal for the scientific community and presents an excellent tool for many fields of research on nucleic acids. In addition to enabling the design of biologically relevant riboswitches our algorithm also has applications in the graph-theoretical field. We can efficiently calculate colourings, given that the underlying graph of the colouring task displays a small width.

Our design program always delivers solutions to the combinatoric problem of finding sequences that fulfill the basepairing constraints of the predescribed structures. As example 2 illustrates, we were not always able to screen for sequences in the optimization procedure that also showed sufficient thermodynamic stability when folding into the switch structures. Further progress can be made by introducing different optimization techniques, to overcome this problem. An adaptive walk is a relatively cheap method in terms of computation time, but it has the inherent problem that an optimization run is easily trapped in a local minimum and thus the basin of the mfe structure, the global minimum, can not be found. Several approaches are within reach to address this problem. Simulated annealing, for example, is a technique that enables us to overcome energy barriers between basins of minimima in the energy landscape. Since our design algorithm is independent from the optimization method used, our program can be easily adapted for the optimization technique of choice.

In the next chapter we provide the theoretical basis on how our design algorithm can be further improved by defining a special kind of ear decomposition, the woffle decomposition. Since the approach we describe there demands a lot of additional graph-theoretical background work the exhaustive investigation of this task exceeded the scope of this thesis, but presents a very promising starting point for future development. Furthermore we outline how our software can be adjusted for the design of whole RNA switching networks.

8 Summary and Outlook

8.1 The Woffle Theory

In the course of researching the graph theoretical background of our design problem, we found yet another approach to speed up the colouring procedure. For this purpose we have to identify a special kind of ear decomposition, the *woffle decomposition* described in the following section.

There is a natural decomposition of the graph $\overline{G}_k$ by splitting it at its set A_k of attachment points, see Fig. 29. We call $\overline{G}_k$ a *woffle* if it is connected and all connected components of $\overline{G}_k^{\rangle \overline{G}_k \langle}$ are the *subwoffles* of $\mathfrak{E}$ at step k. We write $A(\Psi)$ for the attachment points of the woffle Ψ. (See the appendix for the origin of the name woffle.)

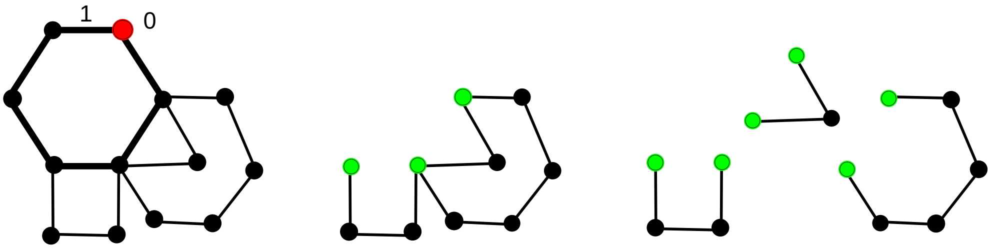


Figure 29: Woffles of an ear decomposition. The graph G has an ear decomposition in which P_0 is the vertex marked in red and P_1 is the cycle outlined in bold. The graph $\overline{G}_1$ consists of one connected component- a woffle- which is split into three subwoffles at the attachments points shown in green.

The path P_k is contained in exactly one of the woffles at step k. If follows that each woffle at step $k+1$ is contained in one of the woffles at set k. More precisely, if Ψ is a woffle at step $k+1$ but not a woffle at step k then there is a (unique) woffle Θ at step k that contains both Ψ and P_k. In this case we call Ψ an *immediate sub-woffle* of Θ and write $\Psi \sqsubset \Theta$.

Consequently, if Ψ and Θ are two woffles, then either $\Psi \cap \Theta = \varnothing$, $\Psi \subseteq \Theta$, or $\Theta \subseteq \Psi$. Thus the woffles of $\mathfrak{E}$ are partially ordered w.r.t. set inclusion. The woffle graph of $\mathfrak{E}$ is the directed graph corresponding to the Hasse diagram of this partial order, see Fig. 30. There is a directed edge from Θ to Ψ iff $\Theta \sqsubset \Psi$.

If the graphs $\overline{G}_k$ are all connected and do not contain a separating set of attach-

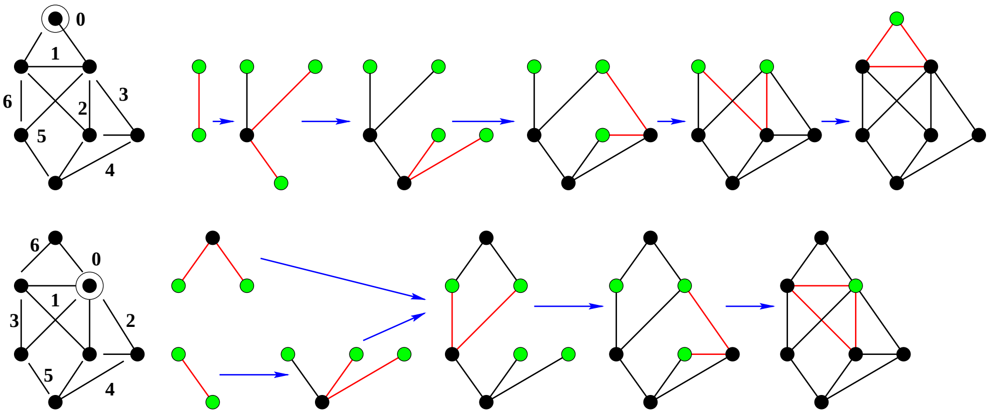


Figure 30: A graph G, two different ear decompositions of G and their woffle sequences. The attachement vertices of the woffles are marked in green. Note that the upper decomposition is a woffle decomposition since each graph $\overline{G}_k$ is connected and thus a woffle.

ment vertices then each $\overline{G}_k$ is a woffle. Let us call an ear decomposition a *woffle decomposition* of G if each $\overline{G}_k$ is a woffle.

Question 1. *Does every biconnected graph have a woffle-decomposition. If not, which graphs do have a woffle decomposition?*

Analogously to section 4.3 we define the woffle-width $\beta(\mathfrak{E})$ of an ear decomposition $\mathfrak{E}$ of G as

$$\beta(\mathfrak{E}) = \max_{\text{woffle } \Psi} |A(\Psi)| \tag{23}$$

and the *woffle width* of a graph as

$$\beta(G) = \min_{\mathfrak{E}} \beta(\mathfrak{E}) \tag{24}$$

Question 2. *How are the ear width and the woffle width of a graph G related to other graph parameters?*

The effort for tabulation can be significantly reduced by decomposing $\overline{G}_{k+1}$ into its woffles since

$$\Omega(\overline{G}_{k+1}; \mathbf{c}_{A_{k+1}\setminus A_k} \circ \mathbf{c}_{A_{k+1}\cap A_k}) = \bigwedge_{\text{woffles } \Psi \text{ of } \overline{G}_{k+1}} \Omega(\Psi; \mathbf{c}_{A(\Psi)\setminus A_k} \circ \mathbf{c}_{A(\Psi)\cap A_k}) \tag{25}$$

As the woffles form a tree w.r.t. to inclusion we can write down equ.(15) separately for each woffle Θ of $\overline{G}_k$. Let

$$A^* = \bigcup_{\Psi \sqsubset \Theta} A(\Psi) \tag{26}$$

be the attachment vertices of all immediate sub-woffles Ψ of Θ; let P_Θ the path of the ear decomposition such that Θ is the union of P and the woffles Ψ satisfying $\Psi \sqsubset \Theta$; and let A' be the endpoints of P_Θ. With this notation equ.(15) becomes

$$\Omega(\Theta; \mathbf{c}_{A(\Theta)}) = \bigvee_{\mathbf{c}_{A^*\setminus A(\Theta)}} \left[\Omega\big(P_\Theta; \mathbf{c}_{A^*\setminus A(\Theta)} \circ \mathbf{c}_{A'}\big) \wedge \bigwedge_{\Psi \sqsubset \Theta} \Omega\big(\Psi; \mathbf{c}_{A(\Psi)\setminus A(\Theta)} \circ \mathbf{c}_{A(\Psi)\cap A(\Theta)}\big) \right] \tag{27}$$

Equ.(27) admits a rather straightforward interpretation: the terms in the square bracket are the decomposition of the woffle Θ into the path P_Θ and the immediate subwoffles of Θ into which Θ decomposed when the path P_Θ is removed and the remainder is split at the attachment vertices $A(\Theta)$. The $\bigvee$-operation then iterates over all colourings of the attachment vertices of the sub-woffles that are not attachment vertices of Θ itself. We obtain an Ω-table indexed by the colourings of the attachement vertices of the woffle Θ. We may regard equ.(27) therefore as the recursion of a dynamic programming algorithm on the woffles of G.

In order to estimate the performance of such an algorithm we argue as follows: Let us recall that $\alpha(\mathfrak{E})$ is the width and $\beta(\mathfrak{E})$ is the woffle width of an ear decomposition. The table $\Omega(\Psi; \mathbf{c}_{A(\Psi)})$ describes all colorings of the attachment vertices of the woffle Ψ. The largest of these tables obviously contains $|\mathfrak{A}|^{\beta(\mathfrak{E})}$ entries. There are not more than μ woffles hence, the evaluation of $\Omega(\overline{G}_{k+1}, \dots)$ requires not more than μ table lookups and $\wedge$ operators. Clearly $|A_k| \leq 2\mu$, thus one recursion step can be performed in at most $\mathcal{O}\left(\mu|\mathfrak{A}|^{2\alpha(\mathfrak{E})}\right)$. The iteration terminates after at most μ steps with the entire graph G. Therefore, we obtain the following upper bounds on CPU and memory consumption

$$\text{CPU} \leq \mathcal{O}\left(\mu^2|\mathfrak{A}|^{2\alpha(\mathfrak{E})}\right) \qquad \text{MEM} \leq \mathcal{O}\left(\mu|\mathfrak{A}|^{\beta(\mathfrak{E})}\right) \tag{28}$$

where we assume that the table entries $\Omega \in \mathbb{X}$ are “numbers” i.e., require $\mathcal{O}(1)$ memory and access time.

It follows that we can evaluate colorings efficiently on graphs with bounded ear-width $\alpha(G) < \alpha_0$.

In order to further investigate the questions raised in this section a tool is currently implemented that generates representative sample spanning trees of biconnected graphs and their respective ear decompositions. As discussed above, the effectiveness of the design algorithm depends on the width $\alpha(\mathfrak{E})$ of the ear decomposition (see Fig. 17), thus we are interested in finding spanning trees which define ear decompositions that minimize the variable $\alpha(\mathfrak{E})$. We anticipate that a probability-driven approach of this kind will help us identify a pattern for determining the ideal spanning tree that minimizes $\alpha(\mathfrak{E})$. Furthermore, we hope that it will lead to new approaches on how to find woffle decompositions in biconnected components.

The term *woffle* has been taken from a short satire on mathematical papers from the web site `http://www.kfunigraz.ac.at/imawww/pages/humor/`.

8.2 Small networks

Our design program can easily be adapted for cofolding tasks, which enables especially the research of genetic switching networks on the basis of alternative folding of nucleic acids. We can achieve this by incorporating the program

`{RNAcofold}`

of the

`{Vienna RNA package}`

into our design algorithm. Thus, we gain an *excellent tool* for a design kit of switch topologies and the theoretical exploration of networks as well as for the development of important models for experimental work.

Fig. 31 shows a simple example of a *self-regulating network with negative feedback control* (similar to the p53-network, see for example [79]). Molecule A has a conformation in which the promoter region (filled red triangle) can be read and thus transcription takes place. In this course the molecule A' is synthesized. A' is able to cofold with another RNA-molecule B, forming the complex A':B. B alone is in a conformation with an inactive promoter (unfilled green triangle), building a complex with A' makes the promoter region accessible though (filled green triangle) and transcription of B' starts. B' in turn can cofold with A which makes the promotor region of A inaccessible and results in the inactivation of transcription of A.
If A is heavily transcribed, we get lots of prodcut A', which then forms a complex with B and enables the synthesis of B'. B' forms a complex with A and so

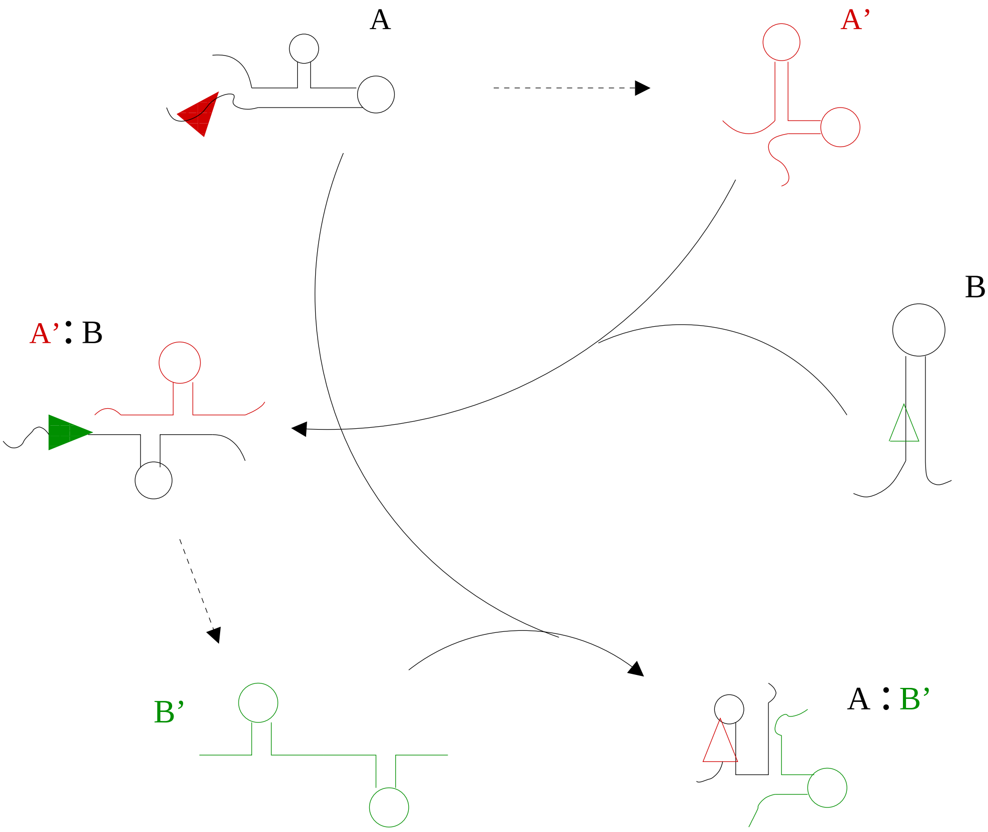


Figure 31: Example of the simplest case of an RNA switch network with negative feedback-loop. The filled triangles depict molecules in the active transcriptional mode, unfilled triangles represent the inactive mode. Dashed arrows indicate transcription.

further transcription of A is downregulated. Since it’s necessary to transcribe A to produce sufficient B’, we will not have enough B’ left to inactivate A at some point of time and therefore the transcription of A increases again.

List of Figures

1 Organic carbon compounds 8

2 Graph representation of a chemical reaction 8

3 RNA secondary structure representation 9

4 DNA and RNA structure and differences. 10

5 Secondary structure of an RNA molecule. 11

6 Primary, secondary and tertiary structure of a tRNA molecule. . . 12

7 Design of a riboswitch with two functions. 14

8 RNA pseudoknots and base triples 17

9 Different representations of RNA secondary structure 18

10 The dependency graph Ψ. 26

11 The bipertite property. 30

12 The dependency graph Ψ of more than two structures may consist of several and more complicated connected components that need further decomposition. 31

13 The ear decompositon. 32

14 Splitting of G proceeds in two steps. 35

15 Graphs associated with an ear-decomposition. 37

16 Stochastic Backtracking 42

17 The width of an ear decomposition $\alpha(\mathfrak{E})$ depends on the selected spanning tree. 43

18 Example of a bistable switch molecule. 45

19 Statistics of the fraction of bipartite graphs versus the sequence length. 50

20 Statistics of the number of tree-edges in dependency graphs of switch structures versus sequence length. 52

21 Statistics of the number of ear-edges in dependency graphs of switch structures versus sequence length. 53
22 Statistics of the number of ear-decompositions in dependency graphs of switch structures versus sequence length. 54
23 Sample set of structures for the design algorithm 55
24 Second sample set of structures for the design algorithm 56
25 Second sample set of structures for the design algorithm 58
26 Dotplot and Barrier-Tree after Optimization 59
27 Comparison of the number of ear decompositions of dependency graphs with varying stack-size constraint. 60
28 Comparison of the number of ear decompositions (not including cycles) of dependency graphs with varying stack-size constraint. . 61
29 Woffles of an ear decomposition. 64
30 A graph G, two different ear decompositions of G and their woffle sequences. 65
31 Example of the simplest case of an RNA switch network with negative feedback-loop. 68

List of Algorithms

1 Algorithm for finding cut-vertices: CUT(u) 21
2 Breadth First Search BFS(u) 22
3 Ear Decomposition EAR(G, u) 23
4 Counting Algorithm - constructing path matrices 41

References

[1] LM Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266:1021–1024, 1994.

[2] M Andronescu, R Aguirre-Hernandez, A Condon, and HH Hoos. RNAsoft: A suite of RNA secondary structure prediction and design software tools. *Nucleic Acids Res*, 31:3416–3422, 2003.

[3] M Andronescu, AP Fejes, F Hutter, HH Hoos, and A Condon. A new algorithm for RNA secondary structure design. *J. Mol. Biol.*, 336:607–624, 2004.

[4] AT Balaban, D (eds Bonchev, and Mekenyan O). *Graph Theoretical Approaches to Chemical Reactivity*. Kluwer Academic Publishers, Dordrecht, Netherlands, 1992.

[5] D. P. Bartel. MicroRNAs: genomics, biogenesis, mechanism, and function. *Cell*, 116:281–297, 2004.

[6] T. Baumstark, A. R. Schroder, and D. Riesner. Viroid processing: Switch from cleavage to ligation is driven by a change from a tetraloop to a loop E conformation. *EMBO J.*, 16:599–610, 1997.

[7] A Ben-Dor, R Karp, B Schwikowski, and Yakhini Z. Universal dna tag systems: a combinatorial design scheme. *J. Comput. Biol.*, 7:503–519, 2000.

[8] CK Biebricher, S Diekmann, and R Luce. In vitro recombination and terminal elongation of RNA by Qβ replicase. *EMBO J.*, 11(13):51129–5135, 1992.

[9] CK Biebricher and R Luce. Structural analysis of self-replicating RNA synthesized by Qβ replicase. *J. Mol. Biol.*, 154:629–648, 1982.

[10] RS Braich, N Chelyapov, C Johnson, PW Rothemund, and L Adleman. Solution of a 20-variable 3-SAT problem on a DNA computer. *Science*, 296:499–502, 2002.

[11] RR Breaker. Engineered allosteric ribozymes as biosensor components. *Curr. Opin. Biotechnol.*, 13:31–39, 2002.

[12] P. Brion and E. Westhof. Hierarchy and dynamics of RNA folding. *Annu. Rev. Biophys. Biomol. Struct.*, 26:113–137, 1997.

[13] Y Cao and SA Woodson. Refolding of rRNA exons enhances dissociation of the tetrahymena intron. *RNA*, 6(9):1248–1256, 2000.

[14] A Cayley. On the analytic forms called trees, with applications to the theory of chemical combinatons. *Rept. Brit. Assoc. Adv. Sci.*, 45:257–305, 1875.

[15] DM Chadalavada, SM Knudsen, S Nakano, and PC Bevilacqua. A role for upstream RNA structure in facilitating the catalytic fold of the genomic hepatitis delta virus ribozyme. *J. Mol. Biol.*, 301(2):349–367, 2000.

[16] FHC Crick. Codonanticodon pairing: The wobble hypothesis. *J. Mol. Biol.*, 19:548–555, 1966.

[17] SM Elbashir, J Harborth, W Lendeckel, A Yalcin, K Weber, and T Tuschl. Duplexes of 21-nucleotide RNAs mediate RNA interference in cultured mammalian cells. *Nature*, 411:494–498, 2001.

[18] VL Emerick and SA Woodson. Self-splicing of the tetrahymena pre-rRNA is decreased by misfolding during transcription. *Biochemistry*, 32(50:14062–14067, 1993.

[19] L. Euler. SOLUTIO PROBLEMATIS AD GEOMETRIAM SITUS PERTINENTIS. *Comm. Acad. Sci. Imp. Petropol.*, 8:128–140, 1736. (Latin).

[20] L. Euler. ELEMENTA DOCTRINÆ SOLIDORUM. *Novi Comm. Acad. Sci. Imp. Petropol.*, 4:109–140, 1752. (Latin).

[21] JB Fan, X Chen, MK Halushka, A Berno, X Huang, T Ryder, RJ Lipshutz, DJ Lockhart, and A Chakravarti. Parallel genotyping of human snps using generic high-density oligonucleotide tag arrays. *Genome Res.*, 10:853–860, 2000.

[22] Christoph Flamm, Walter Fontana, Ivo Hofacker, and Peter Schuster. RNA folding kinetics at elementary step resolution. *RNA*, 6:325–338, 2000.

[23] Christoph Flamm, Ivo L. Hofacker, Sebastian Maurer-Stroh, Peter F. Stadler, and Martin Zehl. Design of multi-stable RNA molecules. *RNA*, 7:254–265, 2001.

[24] DS Franzblau. Computation of ring statistics for network models of solids. *Physical Review B (Condensed Matter)*, 44:4925–4930, 1991.

[25] J. R. Fresco, A. Adains, R. Ascione, D. Henley, and T. Lindahl. Tertiary structure in transfer ribonucleic acids. *Cold Spring Habor Symp. Quant. Biol.*, 31:527–539, 1966.

[26] ZJ Gartner, BN Tse, R Grubina, JB Doyon, TM Snyder, and DR Liu. DNA-templated organic synthesis and selection of a library of macrocycles. *Science*, 305:1601–1605, 2004.

[27] K Gerdes, AP Gultyaev, T Franch, K Pedersen, and ND Mikkelsen. Antisense RNA-regulated programmed cell death. *Annu. Rev. Genet.*, 31:1–31, 1997.

[28] C. Guerrier-Takada, K. Gardiner, T. Marsh, N. Pace, and S. Altman. The RNA moiety of ribonuclease p is the catalytic subunit of the enzyme. *Cell*, 35:849–857, 1983.

[29] AP Gultyaev, T Franch, and K Gerdes. Programmed cell death by hok/sok of plasmid R1: coupled nucleotide covariations reveal a phylogenetically conserved folding pathway in the hok family of mRNAs. *J. Mol. Biol.*, 273(1):26–37, 1997.

[30] DR Halpin and PB Harbury. Dna display i. sequence-encoded routing of dna populations. *PLoS Biol.*, 2:173, 2004.

[31] E. R. Hawkins, S. H. Chang, and W. L. Mattice. Kinetics of the renaturation of yeast tRNA_3^{Leu}. *Biopolymers*, 16:1557–1566, 1977.

[32] Ivo L Hofacker, Walter Fontana, Peter F Stadler, L Sebastian Bonhoeffer, Manfred Tacker, and Peter Schuster. Fast folding and comparison of RNA secondary structures. *Monatsh. Chem.*, 125:167–188, 1994.

[33] Paulien Hogeweg and Ben Hesper. Energy directed folding of RNA sequences. *Nucleic Acids Research*, 12(1):67–74, 1984.

[34] A. Hüttenhofer, P. Schattner, and N. Polacek. Non-coding RNAs hope or hype. *Trends in Genetics*, 21:289–297, 2005.

[35] JA Jaeger, DH Turner, and M Zuker. Improved predictions of secondary structures for RNA. *Proc. Natl. Acad. Sci. USA*, 86:7706–7710, 1989.

[36] Tommy R. Jensen and Bjarne Toft. *Graph Coloring Problems.* John Wiley & Sons, New York, 1994.

[37] J Johansson, P Mandin, A Renzoni, C Chiaruttini, M Springer, and P Cossart. An RNA thermosensor controls expression of virulence genes in listeria monocytogenes. *Cell*, 110:551–561, 2002.

[38] AB Kahng, II Mandoiu, PA Pevzner, S Reda, and AZ Zelikovsky. Scalable heuristics for design of dna probe arrays. *J. Comput. Biol.*, 11:429–447, 2004.

[39] S Kammermeier, H Neumann, F Hampel, and R Herges. Diels-alder reactions of tetrahydrodianthracene with electron-rich dienes. *Liebigs Ann.*, pages 1795–1800, 1996.

[40] G Kirchhoff. über die auflösung der gleichungen, auf welche man bei der untersuchung der linearen verteilung galvanischer ströme geführt wird. *Poggendorf Ann. Phys. Chem.*, 72:497–508, 1847.

[41] Tamas Kiss. An abundant group of noncoding rnas with diverse cellular functions. *Cell*, 109:145–148, 2002.

[42] K. Kruger, P. J. Grabowski, A. J. Zaug, J. Sands, D. E. Gottschling, and T. R. Cech. Self-splicing RNA; autoexcision and autocyclization of the ribisomal RNA intervening sequence of terahymena. *Cell*, 31:147–157, 1982.

[43] JE Ladner, A Jack, JD Robertus, RS Brown, D Rhodes, BFC Clark, and A Klug. Structure of yeast phenylalanine transfer RNA at 2.5 åresolution. *Proc. Natl Acad. Sci. USA*, 72:4414–4418, 1975.

[44] Eric C. Lai. Micro RNAs are complementary to 3' utr sequence motifs that mediate negative post-transcriptional regulation. *Nature Genetics*, 30:363–364, 2002.

[45] RB Lanz, B Razani, AD Goldberg, and BW O'Malley. Distinct RNA motifs are important for coactivation of steroid hormone receptors by steroid receptor RNA activator (SRA). *Proc. Natl. Acad. Sci. USA*, 99:16081–16086, 2002.

[46] Q Liu, L Wang, AG Frutos, AE Condon, RM Corn, and LM Smith. DNA computing on surfaces. *Nature*, 403:175–179, 2000.

[47] Christina Lorenza, Philipp Hadwigera, Matthias Johna, Hans-Peter Vornlochera, and Carlo Unverzagt. Steroid and lipid conjugates of siRNAs to enhance cellular uptake and gene silencing in liver cells. *Bioorg. Med. Chem. Lett.*, 14:4975–4977, 2004.

[48] L. Lovász and M. D. Plummer. *Matching theory*, volume 29 of *Annals of Discrete Mathematics*. North-Holland, 1986.

[49] M Mandal, B Boese, JE Barrick, Winkler WC, and Breaker RR. Riboswitches control fundamental biochemical pathways in bacillus subtilis and other bacteria. *Cell*, 113:577–586, 2003.

[50] D Mathews, J Sabina, M Zucker, and H Turner. Expanded sequence dependence of thermodynamic parameters provides robust prediction of RNA secondary structure. *J. Mol. Biol.*, 288:911–940, 1999.

[51] JS McCaskill. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, 29:1105–1119, 1990.

[52] AS Mironov, I Gusarov, R Rafikov, LE Lopez, K Shatalin, RA Kreneva, DA Perumov, and Nudler E. Sensing small molecules by nascent RNA: a mechanism to control transcription in bacteria. *Cell*, 111:747–756, 2002.

[53] MT Morita, Y Tanaka, TS Kodama, Y Kyogoku, H Yanagi, and T Yura. Translational induction of heat shock transcription factor sigma32: evidence for a built-in RNA thermosensor. *Genes Dev.*, 13:655–665, 1999.

[54] JH Nagel and CW Pleij. Self-induced structural switches in RNA. *Biochimie*, 84:913–923, 2002.

[55] A Nahvi, JE Barrick, and RR Breaker. Coenzyme b12 riboswitches are widespread genetic control elements in prokaryotes. *Nucleic Acid Res.*, 32:143–150, 2004.

[56] F Narberhaus, R Kaser, A Nocker, and H Hennecke. A novel dna element that controls bacterial heat shock gene expression. *Mol. Microbiol.*, 28:315–323, 1998.

[57] SB Needleman and CD Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J.Mol.Biol.*, 48:443–453, 1970.

[58] A Nocker, T Hausherr, S Balsiger, NP Krstulovic, H Hennecke, and F Narberhaus. mRNA-based thermosensor controls expression of rhizobial heat shock genes. *Nucleic Acids Res.*, 29:4800–4807, 2001.

[59] A. T. Perrotta and M. D. Been. A toggle duplex in hepatitis delta virus self-cleaving RNA that stabilizes an inactive and a salt-dependent pro-active ribozyme conformation. *J. Mol. Biol.*, 279:361–373, 1998.

[60] RA Poo, NV Tsareva, IV Boni, and J van Duin. RNA folding kinetics regulates translation of phage MS2 maturation gene. *Proc. Natl. Acad. Sci.*, USA 94(19):10110–10115, 1997.

[61] Vijaya Ramachandran. *Parallel Open Ear Decomposition with Applictions to Graph Biconnectivity and Triconnectivity.* Chapter in Synthesis of Parallel Algorithms by Morgan Kaufmann, 1992.

[62] R. Rammal, G. Toulouse, and M. A. Virasoro. Ultrametricity for physicists. *Rev. Mod. Phys.*, 58:765–788, 1986.

[63] C. M. Reidys, P. F. Stadler, and P. Schuster. Neural networks of RNA secondary structures. *Bull. Math. Biol.*, 59:339–397, 1997.

[64] JJ Rossi. The application of ribozymes to hiv infection. *Curr. Opin. Mol. Ther.*, 3:316–322, 1999.

[65] A Schultes and DP Bartel. One sequence, two ribozymes: Implications for the emergence of new ribozyme folds. *Science*, 289:448–452, 2000.

[66] Peter Schuster, Peter F. Stadler, and Alexander Renner. RNA structures and folding: From conventional to new issues in structure predictions. *Curr. Opinions Structural Biol.*, 7:229–235, 1997.

[67] TF Smith and MS Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197, 1981.

[68] GA Soukup, EC DeRose, M Koizumi, and RR Breaker. Generating new ligand-binding rnas by affinity maturation and disintegration of allosteric ribozymes. *RNA*, 4:524–536, 2001.

[69] J Soutschek, A Akinc, B Bramlage, K Charisse, M Constien, Rand Donoghue, S Elbashir, A Gelck, P Hadwiger, J Harborth, M John, V Kesavan, G Lavine, RK Pandey, T Racie, KS Rajeev, I Röhl, I Toudjarska, G Wang, S Wuschko, D Bumcrot, V Kotellansky, S Limmer, M Manoharan, and HP Vornlocher. Therapeutic silencing of an endogenous gene by systemic administration of modified siRNAs. *Nature*, 432:173–178, 2004.

[70] DW Staple and SE Butcher. Pseudoknots: RNA structures with diverse functions. *PLoS Biol*, 3:213, 2005.

[71] Manfred Tacker, Peter F. Stadler, Erich G. Bornberg-Bauer, Ivo L. Hofacker, and Peter Schuster. Algorithm independent properties of RNA structure prediction. *Eur. Biophy. J.*, 25:115–130, 1996.

[72] F Tanaka, A Kameda, M Yamamoto, and A Ohuchi. Design of nucleic acid sequences for dna computing based on a thermodynamic approach. *Nucleic Acids Res.*, 33:903–911, 2005.

[73] Jin Tang and Ronald R. Breaker. Rational design of allosteric ribozymes. *Chem. Biol.*, 4:453–459, 1997.

[74] RE Tarjan. Depth first search and linear graph algorithms. *SIAM J. Computing*, 1:146–160, 1972.

[75] Ignatio Tinoco Jr and C. Bustamante. How RNA folds. *J. Mol. Biol.*, 293:271–281, 1999.

[76] D Tulpan, M Andronescu, SB Chang, MR Shortreed, A Condon, HH Hoos, and LM Smith. Thermodynamically based dna strand design. *Nucleic Acids Res.*, 33:4951–4964, 2005.

[77] N Usman and J McSwiggen. Catalytic RNA (ribozymes) as drugs. *Annu. Rep. Med. Chem.*, 30:285–294, 1995.

[78] A. M. Vertechi and M. A. Virasoro. Energy barriers in SK spin glass models. *J. Phys. France*, 50:2325–2332, 1989.

[79] Levine AJ Vogelstein B, Lane D. Surfing the p53 network. *Nature*, 408:307–310, 2000.

[80] MV Vol'kenshtein. Physics of enzymes. *Sciences, Moscow*, 1965.

[81] MV Vol'kenshtein and BN Gol'dshtein. *Dokl. Akad. Nauk SSSR*, 170:963, 1965.

[82] MV Vol'kenshtein and BN Gol'dshtein. Models of allosteric enzymes and their analysis using the graph theory method. *Biokhimiia*, 4:679–686, 1966. (Russian).

[83] MV Vol'kenshtein and BN Gol'dshtein. A new approach to the problems of stationary kinetics of enzyme reactions. *Biokhimiia*, 3:541–547, 1966. (Russian).

[84] AE Walter, DH Turner, J Kim, MHw Lyttle, P Müller, DH Mathews, and M Zuker. Co-axial stacking of helixes enhances binding of oligoribonucleotides and improves predictions of RNA folding. *Proc. Natl. Acad. Sci. USA*, 91:9218–9222, 1994.

[85] JD WATSON and FHC CRICK. Molecular structure of nucleic acids. *Nature*, 171:737–738, 1953.

[86] Rüdiger Wehner and Walter Gehring. *Zoologie.* Georg Thieme Verlag, 1995.

[87] H. Whitney. Non-separable and planar graphs. *Trans. Am. Math. Soc.*, 34:339–362, 1932.

[88] H. Zamora, R. Luce, and C. K. Biebricher. Design of artificial short-chained RNA species that are replicated by Qβ replicase. *Biochemistry*, 34:1261–1266, 1995.

[89] B. Zhang and T. R. Cech. Peptidyl-transferase ribozymes: trans reactions, structural characterisation and ribosomal RNA -like structures. *Chem. Biol.*, 5:539–553, 1998.

[90] M. Zuker. On finding all suboptimal foldings of an rna molecule. *Science*, 244:48–52, 1989.

[91] M Zuker, DH Mathews, DH Turner, J (eds Barciszewski, and BFS) Clark. *Algorithms and thermodynamics for RNA secondary structure prediction: a practical guide in* RNA Biochemistry and Biotechnology. Kluwer Academic Publishers, 1999.

[92] M. Zuker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acid Res.*, 9:133–148, 1981.

Curriculum vitae

Mag. Ingrid G. Abfalter

Persönliche Daten

Geburtsdatum: 2. August 1976

Geburtsort: St. Pölten, Österreich

Staatsbürgerschaft: Österreich

Schulbildung

1982 – 1986	Volksschule Institut der Englischen Fräulein in St. Pölten
1986 – 1994	Bundesgymnasium und Bundesrealgymnasium St. Pölten, humanistischer Zweig
1994	Matura mit Auszeichnung bestanden

Studium

1994 – 2000	Studium der Chemie, Studienzweig Biochemie bzw. ab 1999 Studium der Chemie, Schwerpunkt Biochemie an der Universität Wien
1999 – 2000	Diplomarbeit am Institut für Medizinische Biochemie, Vienna Biocenter, Universität Wien, Titel: *Vergleich der proteolytischen Spaltung der beiden In-Vivo-Substrate der* $FMDV - Lb^{pro}$
2000	Diplomprüfung mit Auszeichnung bestanden
2001 – 2004	*DOC Stipendium für hochqualifizierte Doktoranden* der Österreichischen Akademie der Wissenschaften
2001 – 2005	Doktorarbeit am Institut für Theoretische Chemie und Molekulare Strukturbiologie, Universität Wien, Titel: *Sequence Design As A Graph Colouring Problem*