Computational Design of Hammerhead Ribozymes for Logic Computing and Disease Treatment

Nicolas Kamel

A Thesis
in
The Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy (Electrical and Computer Engineering) at
Concordia University
Montréal, Québec, Canada

March 2024

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Nicolas Kamel**

Entitled: **Computational Design of Hammerhead Ribozymes for Logic Computing and Disease Treatment**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Otmane Ait Mohamed

_____ External Examiner
Dr. Malcolm Iain Heywood

_____ Arm's Length Examiner
Dr. Maureen McKeague

_____ Examiner
Dr. Habib Benali

_____ Examiner
Dr. Steve Shih

_____ Thesis Supervisor
Dr. Nawwaf Kharma

_____ Co-supervisor
Dr. Joey Paquet

Approved by _____
Dr. Jun Cai, Graduate Program Director

April 9, 2024 _____
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Computational Design of Hammerhead Ribozymes for Logic Computing and Disease Treatment

Nicolas Kamel, Ph.D.
Concordia University, 2024

This thesis explores the intersection of two techniques: evolutionary algorithms (EAs) and allosteric ribozymes (ARs). EAs are population-based search heuristics inspired by natural evolution and ARs are catalytic non-coding RNA (ncRNA) whose activity can be modulated via shape changes induced by external molecules. We present two EAs that design biological devices based on ARs. The first, TruthSeqEr, designs ribogates, logic gates that take short RNA strands as inputs and produce a short RNA output strand as output. Compared to existing approaches, TruthSeqEr is easy-to-use and produces ribogates that are more versatile and that have a greater computational capacity. *In silico* results show that TruthSeqEr successfully designs ribogates implementing all instances from representative sets of 1, 2, and 3-input functions, including linearly inseparable functions. Motivated by a desire to understand these complex ribogates at an intuitive level, we developed an abstract model that represents each ribogate as a small graph. Analysis of these graphs showed that ribogates can be classified into families of varying complexity based on shared structural motifs and that ribogates act as a more general version of an artificial neuron. Our second EA, TriCleaver, designs selective ribozymes (sRzs) that cleave the pathogenic mutant mRNA transcript associated with a trinucleotide repeat expansion disorder (TRED) while leaving the functional wild-type (WT) transcript intact. TREDs are debilitating genetic disorders that result in a severe reduction in quality of life, and in many cases, death. *In silico* results reveal that compared to existing approaches, TriCleaver is more general, being able to design sRzs that target TREDs in which the mutant is much longer than the WT, as well as TREDs in which the mutant and WT are close in length. In addition, *in vitro* results in mammalian cells showed that two sRzs designed by TriCleaver selectively silenced the mutant transcript associated with a TRED called OPMD. Altogether, these results highlight the therapeutic promise of combining EAs and ARs.

Acknowledgments

To begin, I thank Dr. Nawwaf Kharma, my supervisor, for providing me with the opportunity to perform research at the intersection of two cutting-edge fields, for encouraging me to focus on the long-term picture, and to stay the course whenever unforeseen challenges arose. I would also like to thank Dr. Joey Paquet, my co-supervisor, for helping me become a better researcher by emphasizing the importance of rigor and precision. I extend my sincere thanks to Dr. Jonathan Perreault for patiently helping me close gaps in my knowledge of molecular biology. I express my gratitude to Dr. Aida Abu-Baker and Ms. Pegah Hadavi for their tireless efforts validating designs generated by my algorithms. Finally, I thank my parents for their unconditional love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Protein

All living organisms are made of cells, and each cell is a sophisticated biochemical factory. The "machines" of this factory are macromolecules called *proteins*. Proteins perform a remarkable set of functions, which include, but are by no means limited to: converting energy from one form to another, generating linear and translation motion, acting as structural supports, relaying biological signals, and accelerating chemical ractions [4]. Proteins are comprised of units called *amino acids* linked together by peptide bonds. There are 20 standard amino acids, each with a unique set of chemical properties. By chaining together different sequences of amino acids, proteins with different structures and functionalities are obtained [41].

1.2 DNA

The instructions to synthesize a cell's proteins are encoded in another type of macromolecule called *deoxyribonucleic acid* (*DNA*). DNA is a polymer composed of monomers called *nucleotides*. Each nucleotide is composed of three functional groups: a *phosphate*, a sugar called *deoxyribose*, and a *nucleobase*, often simply referred to as a *base*. Deoxyribose contains 5 carbon atoms, labeled 1' through 5'. Carbons 1' to 4' are arranged in a ring, along with an oxygen atom (O). In addition, the 1' carbon is attached to the nucleobase, the 2' carbon to a hydrogen atom (H), the 3' atom to a hydroxyl group (OH), the 4' carbon to the 5' carbon, and the 5' carbon to the phosphate group. There are four different DNA bases: *adenine*, *cytosine*, *guanine*, and *thymine*, abbreviated as A, C, G, T, respectively. DNA strands are formed by a *polymerization* reaction, in which the 3' carbon of an *upstream* nucleotide becomes linked to the 5' carbon of a *downstream* nucleotide via ia *phosphodiester* bond. The most upstream nucleotide contains a free phosphate attached to its 5' carbon, this is denoted as the *5' end* of the strand, while the most downstream nucleotide contains a free hydroxyl attached to its 3' carbon, this is denoted as the *3' end* of the strand. In general, the phosphodiester bonds and sugar groups are uniform along the DNA strand; what distinguishes one DNA strand from another is the *sequence* of bases

encountered when traversing the strand from the 5' end to the 3' end. One can therefore view DNA as a sequence of bases attached to a *sugar-phosphate backbone*. This sequence is often represented as a string of characters from the set {A,C,G,T } [4]. Not all regions of DNA code for proteins, but in those that do (called coding genes), a well-established *genetic code* provides a mapping from bases to amino acids [4]. In other words, a gene's base sequence determines the amino acid sequence of the protein it codes for, and ultimately that protein's structure and function. Certain pairs of bases can bind to each other via hydrogen bonds; they are said to form a *base-pair*. There are two types of standard base-pairs: A-T and C-G. In cells, DNA is not single stranded, but instead consists of two *complementary* strands, in which each base of one strand is base-paired with a base from the other strand. This extensive base-pairing results in the folding of the two strands into a three dimensional structure called the double helix [4].

1.3 RNA

So far, we have been introduced to two biomolecules: proteins which perform various cellular tasks, and DNA which acts as a store of information. There is, however, a third molecule missing from this picture: *ribonucleic acid* (*RNA*). RNA is generated from DNA by a process called *transcription* [4], and it shares characteristics with both DNA and protein. Its chemical structure is similar to that of DNA: it consists of four types of nucleobases attached to a sugar-phosphate backbone. Three of these bases (A, C, and G) are shared with DNA, and the fourth, *uracil* (U) is similar to DNA's thymine. Like DNA bases, RNA bases are able to form base-pairs. The standard base-pairs are A-U and C-G, although the non-standard *wobble* base-pair G-U also plays an important role [54]. Like DNA, RNA can store information: such RNA is called *messenger RNA* (*mRNA*), and it is used as a template to synthesize protein by a process called *translation* [4].

Despite the similarities between DNA and RNA, there are important differences [25] that enable *non-coding RNA* (*ncRNA*) to perform roles other than information storage. Unlike DNA which is usually *double-stranded*, RNA is usually *single-stranded*. Whereas DNA bases are (usually) constrained to form *inter*-molecular base-pairs with their complementary strand, RNA bases can form *intra*-molecular base-pairs with other bases in the same strand. These intra-molecular base-pairs cause the backbone to bend and the RNA strand to fold into a *secondary* structure consisting of regions base-paired RNA called *helices* (also known as *stems*) connected by regions of unpaired RNA called *loops* [99]. Once the secondary structure has formed, different types of molecular interactions between the loops and helices cause it to adopt a more elaborate *tertiary* structure [25]. RNA strands with different sequences will form different sets of base-pairs and therefore adopt different secondary and tertiary structures. In other words, RNA's sequence determines it structure [54]. The structures RNA adopts serve various roles such as sequestering important regions of RNA [9], serving as recognition motifs for *RNA-binding proteins* (*RBPs*) [38], and tightly binding to small molecules [45]. Another consequence of RNA being single-stranded is that it frees it up to bind to many possible RNA strands, and not just a single complementary strand as for DNA. Furthermore, RNA bases can base-pair with

DNA bases, enabling the formation of RNA-DNA duplexes [95]. Via complementary base-pairing, an RNA strand can recognize and bind to specific regions on DNA or other cellular RNAs. Many ncRNAs are associated with proteins, and they *guide* these proteins to complementary RNA or DNA targets [65, 94]. In addition to being more structurally versatile than DNA, RNA is also more chemically reactive than DNA. This is due to RNA's sugar group being a *ribose* instead of a deoxyribose, meaning that it has a hydroxyl group on its 2' carbon instead of a hydrogen atom [25].

1.4 Ribozymes

RNA's increased chemical reactivity and structural versatility allow it to function as an *enzyme* and accelerate naturally occurring chemical reactions by orders of magnitude [25]. RNA enzymes are called *ribozymes*, and as will become soon apparent, form the cornerstone of this thesis. Since RNA is capable of both storing information and performing chemical catalysis, it is hypothesized that RNA evolved before DNA and protein, and was able to catalyze its own replication [4]. This is called the RNA world hypothesis. While ribozymes were once believed to be common, today ribozymes are much rarer than protein enzymes [10]. Indeed, there are only three known chemical reactions catalyzed by natural ribozymes in modern cells: *peptidyl transferase*, *hydrolysis*, and *transesterification* [59]. Peptidyl transferase is catalyzed by a large protein/RNA complex called the ribosome (introduced in Section 2.3.3) which joins two amino acids via a peptide bond during protein translation. Hydrolysis is catalyzed by an enzyme called RNase P which splits a specific ncRNA into two strands by *cleaving* its backbone. Transesterification is catalyzed by two types of ribozymes. The first is a large protein/RNA complex called the spliceosome (introduced in Section 2.3.2) which removes RNA segments from percursor mRNA. The second is a class of small ribozymes called *nucleolytic* ribozymes which cleave their own backbone. In this thesis, we will make extensive use of a type of nucleolytic ribozyme called the *hammerhead* ribozyme. We will therefore discuss in more detail the mechanism by which nucleolytic ribozymes catalyze their own clevage.

As previously explained, RNA is more chemically reactive than DNA due to its 2' hydroxyl (OH) group. This is because the oxygen atom of the hydroxyl group is more electronegative than the hydrogen, resulting in a partial negative charge on the oxygen. This negative charge makes the hydroxyl a *nucleophile* that is attracted to the positively charged nucleus of certain atoms such as the phosphorus in the phosphodiester bond linking two nucleotides. Consider a nucleotide M connected to an immediate downstream nucleotide N via a phosphodiester bond. The 2' hydroxyl of M can chemically attack the phosphorus atom in the phosphodiester bond, resulting in a bond forming between the phosphorus and the oxygen of the hydroxyl, and a bond breaking between the phosphorus and the 5' oxygen of N [25]. In other words, the RNA backbone will be *cleaved* between the nucleotide M and N, resulting in two separate strands. Under normal physiological conditions, this reaction is slow, but nucleolytic ribozymes significantly accelerate it [25]. The ribozyme's structure and sequence both play key roles in catalyzing the cleavage reaction. The structure can help correctly position the atoms involved in the reaction, while certain specific nucleobases

participate in the reaction by acting as *general bases* or *general acids* which accept or donate protons (H+), respectively [25].

Since an enzyme's function is determined by its structure, it is clear that changes in its structure can result in changes to its function. Indeed, an important class of enzymes are *allosteric* enzymes that change shape upon the binding of an *effector* molecule [4]. This allows cellular pathways to tightly modulate an enzyme's function based on the concentration of its effector molecule [4]. In nature, virtually all allosteric enzymes are proteins; indeed, currently there is only one known example of an *allosteric ribozyme* (*AR*) [78]. However, several synthetic ARs have been designed [84, 11, 55]. In one especially relevant case [81], one of the hairpins of a hammerhead ribozyme was replaced with an *oligonucleotide binding site* (*OBS*) complementary to a DNA or RNA effector strand. In absence of the effector, the OBS interfered with proper ribozyme folding, preventing it from adopting the structure required for it to catalyze the cleavage of its backbone. However, when the effector strand was added, the OBS base-paired with the effector instead of the ribozyme, allowing the ribozyme to correct fold and cleave itself. One exciting application of ARs is using them to construct biological *logic circuits* [81, 80] that can potentially operate in cells of living organisms. These circuits could sense various biochemical signals and conditionally trigger a response based the values of these signals [18]. For example, a logic circuit could determine whether small RNA strands were present in a cell in a combination indicative of cancer, and if so, trigger an immune response attacking that cell [76]. The focus of this thesis is the design of new, more sophisticated devices based on ARs (specifically allosteric hammerhead ribozymes). Figure 1 recaps several features of RNA that we have discussed so far.

1.5 RNA structure prediction

We have seen that an RNA strand's sequence determines the structure it folds into, and that its structure plays a key role in its cellular function. Therefore, being able to predict the structure of an RNA strand from its sequence greatly facilitates the study of natural RNA and the design of synthetic RNA. Unfortunately, simulating the three-dimensional conformation of a molecule such as RNA, at the molecular level and in near real-time, is completely unfeasible [41]. Therefore, *folding algorithms* such as ViennaRNA [61] make several simplifications to make RNA structure prediction computationally tractable. First, they usually neglect tertiary structure and only predict secondary structure. For RNA, this is a reasonable assumption because tertiary interactions only make a minor contribution to the structure's stability [99]. Second, instead of integrating a set of physical equations to predict how an RNA structure changes over time, algorithms like ViennaRNA compute its associated *partition function* [39]. We will briefly explain the meaning of the partition function. The probability of a strand sampling a given secondary structure in equilibrium is governed by the Boltzmann equation [54]. Each secondary structure has an associated *free energy*; the lower the free energy, the more stable the structure and the more likely it is to be observed in equilibrium. In order to evaluate the probability of observing a given secondary structure, the denominator of the Boltzmann distribution must be computed: this

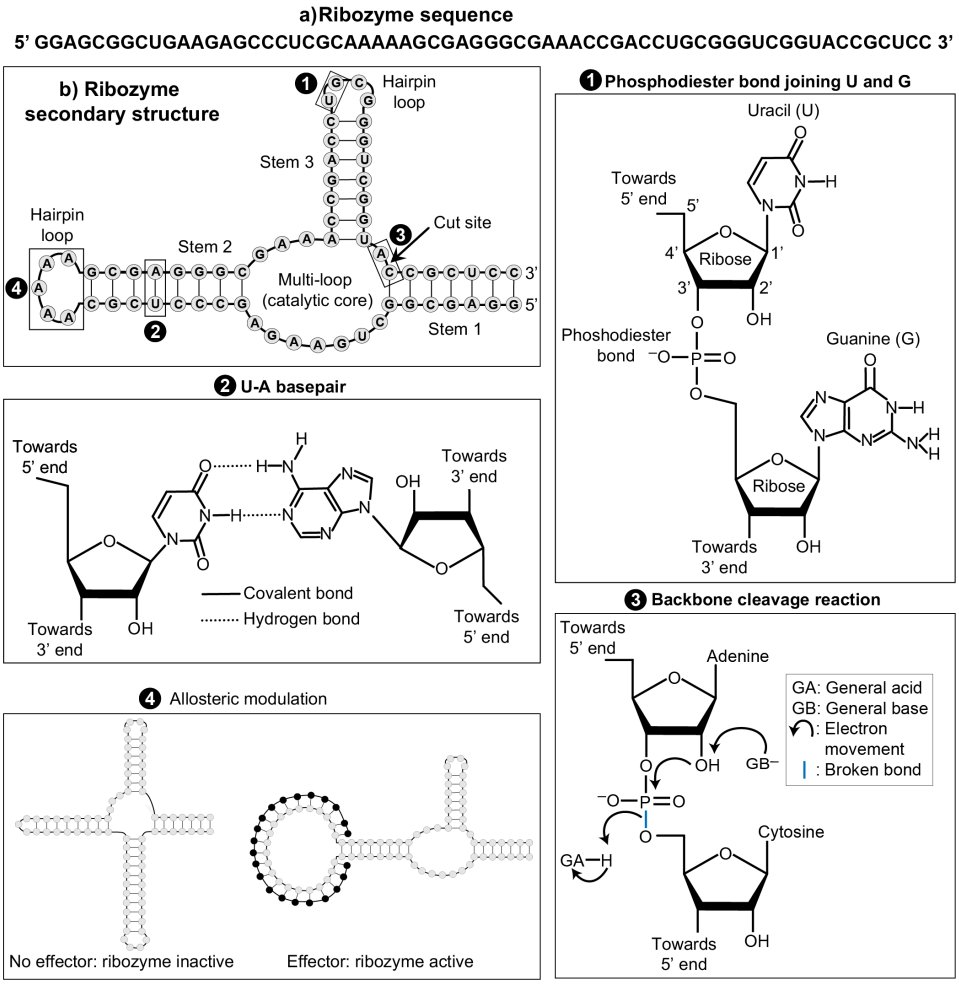


Figure 1: RNA summary. a) An RNA strand's structure is determined by its sequence of nucleotides from the set {A,G,C,U}. The start (upstream end) and end (downstream end) of the strand are denoted by the symbols 5' and 3, respectively. b) RNA folds into a secondary structure consisting of base-paired regions called stems linked together by unpaired regions called loops. This example contains three stems and three loops (two hairpins and a multi-loop). b1) Two consecutive nucleotides, uracil and guanine, attached to the sugar-phosphate backbone by covalent bonds. The sugar molecule (ribose), contains 5 carbons, labeled 1' through 5'. RNA is distinguished from DNA by the OH group on its 2' carbon. b2) Base-pair between uracil and adenine. The two bases are linked with hydrogen bonds, which are weaker than covalent bonds, meaning that base-pairs form and break much more frequently than the backbone. b3) The RNA strand depicted is actually a ribozyme that catalyzes its own self-cleavage, causing the backbone to break at the location indicated by the straight arrow (the cut site). Cleavage is caused by 2' OH initiating a nucleophilic attack on the phosphorus. Nucleolytic ribozymes accelerate this reaction through various means such as providing nucleobases that act as general acids and general bases. b4) An allosteric ribozyme is obtained by replacing one of the hairpin loops with a site complementary to an effector RNA strand (black). When the effector is absent, the ribozyme folds into an inactive state; when the effector strand is present, the ribozyme folds into an active state.

is the partition function [54]. From the partition function, ViennRNA can generate several outputs [39], two of which are relevant for this work: the *base-pairing probability matrix* (*BPPM*) and the *minimum free energy* (*MFE*) structure. The BPPM encodes the probability of any two bases being paired and the MFE structure is the most probable structure that the strand adopts.

1.6 Evolutionary algorithms

Equipped with folding software, we can now develop computational methods to automatically design RNA devices that perform various tasks. In this thesis, we will craft customized *evolutionary algorithms* (*EAs*) that design new types of allosteric-ribozyme based devices. An *evolutionary algorithm* (*EA*) generates a population of candidate solutions to a given problem by performing repeated cycles of *selection* and *reproduction with variation*. The performance of an individual on the target task is quantified by its *fitness*; fitter individuals are more likely to produce offspring and/or survive to the next generation. Offpsring are not carbon copies of their parents, but are varied to a certain extent by operations such as *mutation* and *crossover*. The initial population of candidate solutions is randomly generated and will generally have low fitness. If the elements of the EA (i.e. representation, fitness evaluation, selection, etc.) are appropriately crafted and a sufficient amount of compute is available, the EA will eventually *converge* to a set of satisfactory solutions to the specified problem.

EAs have several advantages over other search heuristics: they can easily be parallelized over multi cores, they do not require the problem domain to differentiable, they can overcome local optima, they find multiple solutions instead of a single one, and they can easily handle multiple objectives. An exciting class of EAs are *quality diversity* (*QD*) approaches. These EAs focus not just on maximizing the performance of the population on the provided task, but also reward individuals that exhibit novel behaviors [58]. In other words, they can evolve solutions that all solve the given problem, but by exploiting different strategies. We believe that this is relevant to the design of biological devices because it helps mitigate the lack of a complete computational model of the cell. Much about the cellular environment remains a mystery, and even if it wasn't, simulating the cell in its entirety would be computationally intractable. As we saw, major simplifications must be made to predict the structure of just a single RNA strand. Therefore, even the best search heuristic is ultimately limited by the model it uses to assess the performance of its candidate solutions. Biological interventions that are predicted to work *in silico* often fail *in vitro* and *in vivo* for unforeseen reasons. If a computational method designs biological devices that all use the same mechanism, and if for some reason that mechanism fails in reality, all the designs would fail. We believe that a quality diversity approach can help mitigate this uncertainty by developing RNA devices that use many different mechanisms to achieve the same goal.

1.7 Contributions

We have now laid the groundwork to summarize our contributions. In this thesis, we present two evolutionary algorithms, *TruthSeqEr* and *TriCleaver*, both of which follow a quality diversity approach, and both of which design RNA devices based on allosteric ribozymes. TruthSeqEr designs *ribogates*, logic gates that take small RNA strands as input and that produce a small RNA strand as output via ribozyme self-cleavage. TruthSeqEr accepts as input the truth table of a 1, 2, or 3-input Boolean function and produces a diverse population of ribogates. Our *in silico* results show that ribogates can implement sophisticated 3-input linearly inseparable functions. In order to study *how* such ribogates functioned, we developed an abstract model called *additive segment competition* (*ACS*), in which each ribogate is represented by a *mechanism graph*. By comparing and contrasting different mechanism graphs, we were able to group ribogates into different families and uncover design principles shared within families. Through mechanism graph analysis we also showed that a ribogate acts as a more general version of an artificial neuron, further highlighting its computational power. TriCleaver designs *selective ribozymes* (*sRzs*), ribozymes that silence a target pathogenic *mutant* mRNA transcript, while sparing a similar, but shorter, healthy *wild-type* transcript. TriCleaver accepts as inputs a wild-type and a mutant mRNA transcript associated with a type of genetic disorder called a *trinucleotide repeat disorder* (*TRED*) and produces a diverse population of sRzs. Our *in silico* results show that TriCleaver can successfully design sRzs targeting two different TREDs, OPMD and Huntington's disease (HD). Further *in vitro* results show that two of these sRzs are functional against OPMD in mammalian cells.

The rest of this thesis is structured as follows. In Chapter 2, we explore RNA in more depth, examining its synthesis, processing, function, and degradation. In Chapter 3, we introduce TruthSeqEr, overviewing related work, explaining its computational methodology, and presenting and analyzing ribogates generated by it. In Chapter 4, we introduce our graph-based ribogate model and use it to analyze ribogates evolved by TruthSeqEr in the previous chapter. In Chapter 5, we introduce TriCleaver, overviewing related work, explaining its computational methodology, and presenting and analyzing sRzs generated by it. Finally, in Chapter 6, we present our conclusions and outline future work.

Chapter 2

Background

2.1 Life of an RNA

In this section, we provide a high level view of the life of an RNA molecule, from its synthesis to its destruction.

2.1.1 Synthesis

RNA is produced from DNA by a protein enzyme called *RNA polymerase* via a process called *transcription*. Transcription consists of three steps: *initiation*, *elongation*, and *termination*. During initiation, RNA polymerase is recruited to a *promoter*, a region of double stranded DNA located near the transcription start site (TSS), where transcription begins. During elongation, the double stranded DNA helix is progressively unwound as RNA polymerase moves along it. One of the strands serves as a *template* strand, and the other as a *coding* strand. One by one, nucleotides that are complementary to the template strand are added to the growing RNA strand. The resulting RNA strand has the same base sequence as the coding strand, but with uracils in place of thymines. Transcription terminates when RNA polymerase encounters a termination signal, which causes it to release the RNA strand and dislodge from the DNA [4]. Initiation, elongation, and termination are all sophisticated processes in their own right, each compromising several sub-steps and involving many additional protein factors [25]. Furthermore, the details of these processes vary depending on which class of RNA polymerase is doing the transcribing. There are (at least) three types of eukaryotic RNA polymerases: Pol I, Pol II, and Pol III. Each type is responsible for transcribing certain types of RNA, and each does so in different ways [25].

2.1.2 Processing and localization

Depending on its ultimate function in the cell, RNA can be processed into a more mature form, either co-transcriptionally (while transcription is occurring) or post-transcriptionally (after transcription has finished). This processing can consist of removing certain regions of RNA, adding special sequences, or even chemically modifying specific nucleotides [4].

In addition, RNAs often assemble with other proteins into *ribonucleoprotein* (*RNP*) complexes [25].

Eukaryotes contain several *membrane-bound organelles* that act as specialized compartments. Transcription occurs inside an organelle called the *nucleus* [4]. The nucleus also contains several compartments that do not contain a membrane. These *membraneless* compartments behave in a liquidlike manner due to multiple weak interactions between their constituent proteins and RNAs. They accelerate specific cellular processes by bringing the factors involved in such processes in close proximity with each other [4].

Depending on the type of RNA, it may stay in the nucleus after transcription or may it be exported out of the nucleus and into the cytoplasm [85]. In some cases, the RNA is exported out of the nucleus where it associates into a ribonucleoprotein complex and is then reimported back into the nucleus [65].

2.1.3 Function

Once an RNA strand has been processed into its mature form and has been transported to an appropriate location, it can perform a variety of functions. An overview of some of these functions and the types of RNA that perform them are shown below [25]:

1. Messenger RNA (mRNA). Contains the instructions necessary to synthesize a given protein.

2. Primer RNA. Required for DNA polymerase to begin replicating segments of DNA.

3. Small nuclear RNA (snRNA). Forms part of the spliceosome RNP that removes segments called introns from pre-mature messenger RNA.

4. Ribosomal RNA. Forms part of the ribosome RNP that synthesizes proteins from messenger RNA.

5. Small nucleolar RNA (snoRNA). Helps process pre-ribosomal RNA into mature ribosomal RNA.

6. Small Cajal body associated RNA. Helps process small nuclear RNA into a more mature form.

7. MicroRNA (miRNA). Guides protein complexes to target mRNA strands to silence them.

8. Long non-coding RNA (lncRNA). Serves various roles such as acting as an miRNA sponge and recruiting protein factors to activate or silence transcription.

9. tRNA. Serves as a molecular adapter between RNA and amino acids.

Many of these RNAs will be explained in more detail in Section 2.3.

2.1.4 Destruction

Eventually all RNA molecules are degraded by cellular enzymes called *nucleases*. There are three classes of nucleases [98]:

1. *5'-3' exonucleases* which degrade RNA starting from the 5' end.
2. *3'-5' exonucleases* which degrade RNA starting from the 3' end.
3. *Endonucleases* which cleave RNA at an internal location.

RNA strands with unprotected 5' and/or 3' ends are vulnerable to exonuclease attack and are quickly degraded. The ends of RNAs with extended half-lives typically contain special features such as:

1. Complex secondary and tertiary structures. For example, two lncRNAs, MALAT1 and NEAT1, have a tertiary *triplex* structure on their 3' end which prevents exonuclease loading [85].
2. Chemical modifications. For example, mRNA (see Section 2.3.1) possess a chemical structure on its 5' end that prevents 5' to 3' exonucleases from binding to it [22]. In addition, synthetic RNA strands used in therapeutics are often heavily modified to extend their half-life and increase their potency [93].
3. RNA-binding proteins. For example, microRNA (see Section 2.3.5) is stable when bound to the Argonaute protein, but when released it is quickly degraded [94].
4. Poly(A) tails. The 3' end of mRNA contains a *tail* comprised of multiple adenine (A) bases (an average of 200 in mammals). The tail serves as a binding site for multiple poly(A) binding proteins (PABPs) which protect the mRNA from exonucleases [79].
5. Being non-existent. Some RNAs are *circularized*, meaning that their 5' and 3' ends are covalently linked together [13].

The stability of a specific RNA molecule can be tightly regulated through the actions of proteins that make it more vulnerable to exonuclease attack. Such proteins can remove the 5' cap [22], shorten the poly(A) tail [79] and unwind complex secondary structures [47]. In addition, certain RNAs contain sequence motifs that result exonucleases being recruited to that RNA. Some of these motifs are intrinsic to the RNA itself while others are added by enzymes as a degradation signal. *AU-rich elements*, segments of adenines (A) and uracils (U) present in certain mRNAs are examples of the former [47], whereas tails of uracils added to RNAs with short poly(A) tails are examples of the latter [60, 27]. RNA stability is regulated for two main reasons: to control gene expression (i.e. longer lived mRNA transcripts can serve as templates for more proteins) and degrade faulty transcripts [25].

Endonucleases cleave RNA internally, with different types of endonucleases often targeting RNA with specific structures and/or sequences. For example, RNase H degrades the RNA strand that is part of a DNA-RNA duplex [88], Drosha cleaves primary microRNA

(pri-miRNA) with certain conserved nucleotides and structural features [94], and Ago2 cleaves a target mRNA with a region fully complementary to a small interfering RNA (siRNA) [93]. Endonuclease cleavage results in two strands, one with an exposed 3' end and one with an exposed 5' end. These exposed ends are vulnerable to exonuclease attack and are quickly degraded [4].

2.2 Transcriptional regulation

Complex organisms such as mammals are comprised of many different cell types. Although all cell types are genetically identical, they can differ widely in terms of morphology and function [6]. This is possible because two cells via the same genes can produce different levels of proteins and ncRNA by regulating *gene expression* [6]. The central dogma of molecular biology states that DNA is transcribed into messenger RNA which is translated into protein. Much of the cell's machinery is dedicated to regulating this process through four main approaches [4]. First, the act of transcription can be regulated. Second, mRNA can be modified after it is transcribed but before it is translated. Third, the act of translation can be regulated. Fourth, proteins can be modified after translation. These are referred to as *transcriptional*, *post-transcriptional*, *translational*, and *post-translational* regulation, respectively. In this section, we concentrate on transcriptional regulation.

2.2.1 The preinitiation complex and chromatin

Eukaryotic RNA polymerase cannot initiate transcription on its own; instead, a *preinitiation complex* (*PIC*) must form at the promoter. This complex consists of RNA polymerase and set of proteins called *general transcription factors* (*GTFs*) [34]. The preinitiation complex forms sequentially, with one GTF recognizing a specific motif on the promoter [25]. This GTF in turns recruits other GTFs which serve roles such as correctly positioning RNA polymerase, unwinding the DNA double helix, and chemically modifying RNA polymerase so that it assumes an active form [64].

However, by default, it is difficult for the PIC to assemble at the promoter. The reason for this is that eukaryotic DNA is packed with proteins into a much more condensed form called *chromatin*, which makes the promoter less accessible to the PIC proteins [34]. The main units of chromatin are *nucleosomes*, which consist of short stretches of DNA wrapped around a core composed of eight proteins called *histones*. These nucleosomes are chained together by short stretches of unpacked linker DNA. Multiple nucleosomes can stack on top of each other, further compacting the DNA structure. The tightness of the chromatin packing can be regulated by various means, and the tighter the packing, the more difficulty proteins will have binding to DNA [6].

2.2.2 Transcription factors

Given the inaccessibility of chromatin, robust transcription usually requires the intervention of additional proteins called *activators*. Activators bind to regulatory elements on the

genome called *enhancers*. Activators are a type of *transcription factor* (*TF*), but unlike the general transcription factors involved in the direct recruitment of RNA polymerase, activators are more specific, with a given activator only being recognized by certain enhancers. This specificity allows for tailored control of transcription. Another type of TF is the *repressor* which inhibits transcription by a binding to regulatory element called a *silencer* [50]. The physical proximity between the enhancer/silencer and the promoter is of paramount importance. Chromatin is arranged in loops, and regions within these loops are more likely to interact with each other than with regions in other loops [50]. In some cases, these loops can be modified [31], bringing some enhancers closer in contact to the promoters they modulate while moving others further away.

TFs exert their effect on transcription by recruiting other proteins called *cofactors*, with three categories of cofactors having be proposed [50]. The first category consists of *nucleosome remodelers*, protein complexes that alter the structure of chromatin [6]. They can make chromatin more accessible by moving or ejecting nucleosomes, or make it less accessible by assembling more chromatin or removing activating *histone markings* (discussed shortly) [16]. The second category consists of *histone modifiers*. The histone proteins that make up the core of the nucleosome contain long tails that are readily accessible by other proteins. Histone modifiers alter these tails by adding or removing various types of chemical markings. Through means that are not completely understood, certain markings activate transcription while others repress it [6]. For example, some loosen chromatin through electrostatic interactions [6], others stabilize units of the pre-initiation complex [57], and many are recognized by nucleosome remodelers, facilitating their recruitment [16]. The third category includes a single cofactor called *Mediator*. Mediator is a large protein complex that can enhance transcription by various means [5], three of which we list here. First, it is involved in DNA loop formation, which as previously discussed, brings enhancers closer to their associated promoters. Second, it can simultaneously bind to multiple GTFs, thereby stabilizing the PIC. Third, it participates in the chemical modification of RNA polymerase required for its activation.

We have seen that by binding to enhancers, TFs help the PIC overcome nucleosome barriers and enable RNA polymerase to initiate transcription. However, this leads to another question: how are the TFs themselves able to bind to nucleosome dense regions in enhancers? One mechanism is *cooperative binding*, in which the binding of one TF makes it easier for subsequent TFs to bind and displace nucleosomes. Another explanation is that some TFs are especially equipped to bind to nucleosome dense regions. Such TFs are called *pioneering factors* [4].

2.3 Types of RNA

We have seen at a high level how RNA is transcribed, modified, and degraded. In this section, we shall examine the various roles of RNA in more depth. Unless otherwise noted, it is assumed that these RNAs are operating in eukaryotic cells. The goal this chapter is not to be comprehensive, but rather to provide a glimpse into RNA's versatility.

2.3.1 Messenger RNA (mRNA)

mRNA stores the information required to synthesize proteins. mRNA is not directly transcribed; instead, it is generated via the co-transcriptional processing of *precursor* mRNA (*pre-mRNA*) [4]. pre-mRNA is transcribed from Pol II, and during transcription, three types of processing events occur [4]. First, a chemical structure called the *5' cap* is added to the 5' end of the growing pre-mRNA strand. Second, segments called *introns* are spliced out of the pre-mRNA, one after another, by a ribonucleoprotein complex called the *spliceosome*. Third, a sequence on the pre-mRNA called the polyadenylation signal is recognized by a protein complex which cleaves the strand at a nearby downstream site. After cleavage, a *poly(A) tail* consisting of a large number of adenine (A) nucleotides is added to the 3' end of the strand. The resulting mRNA strand consists of five components [91]: a 5' cap, a 5' *untranslated region* (*UTR*), an *open reading frame* (*ORF*), a 3' UTR, and a poly(A) tail. The 5' cap and 3' tail protect the mRNA from exonuclease degradation and also serve many other roles [22, 79]. The ORF is *translated* into a protein, and as their names suggest, the 3' and 5' UTRs are not [4]. Certain proteins bind to the mature mRNA's 5' cap and facilitate its export from the nucleus into the cytoplasm [22]. In the cytoplasm, translation is performed by the *ribosome*, a complex comprised of both RNA and protein. For every three consecutive nucleotides in the ORF, called *codons*, the ribosome adds one amino acid to the growing protein [4]. The are 64 possible codons: 61 of them code for amino acids while the other 3 are *stop* codons that act as signals for translation termination [4]. The mapping from codons to amino acids is called the *genetic code* [4]. Since there are more (non stop) codons (61) than amino acids (20), it means that the genetic code is *degenerate* [4]. In other words, multiple codons can specify the same amino acid.

2.3.2 Spliceosomal RNA

pre-mRNA consists of an alternating series *exons* and *introns*. Exons are segments that remain in the mature mRNA while introns are segments that are removed by *splicing*. Splicing is performed by the *spliceosome*, a sophisticated ribonucleoprotein complex. It consists of five *small nuclear RNAs* (*snRNA*) (U1, U2, U4, U5, and U6) and a large number of proteins [25].

U1, U2, U4, and U5 are transcribed by Pol II but are processed differently than mRNA. Their 5' end is capped, allowing them to be exported to the cytoplasm, but their 3' end is not polyadelynated. Instead, their 3' end forms a stem-loop structure and contains binding sites for multiple *Sm* proteins. Once in the cytoplasm, they assemble into RNP complexes with these Sm proteins and undergo more processing. Their 5' cap is then modified, allowing them to be *reimported* back into the nucleus, specifically to regions known as Cajal bodies. Once in the Cajal bodies, they are further modified, this time by another class of ncRNA called *small Cajal body RNAs* (*scaRNA*) [65]. Although U6 is similar in structure to the other snRNAs (it contains a 3' stem loop and protein binding sites), its biogenesis is markedly different. It is transcribed by Pol III instead of Pol II, and it assembles with a different class of protein. Crucially, it has a non-standard 5' cap which prevents its export from the nucleus [65].

Although different pre-mRNAs generally have different sequences, they share certain *conserved* sequences at specific locations that are recognized by the splicesome, allowing it to distinguish introns from exons [25]. Consider two exons, denoted as the 5' and 3' exons, respectively, separated by an intron. The spliceosome recognizes three sites [25]:

- The 5' splice site, which marks the boundary between the 5' exon and the intron.
- The 3' splice site, which marks the boundary between the intron and the 3' exon.
- The branchpoint, located inside the intron.

Splicing proceeds via two consecutive transesterification reactions, similar to the one described in Chapter 1 [25]. The specific mechanism of splicing is beyond the scope of this thesis, but the end result is that the intron is released in the form of a loop called a *lariat*, and the 3' end of the 5' exon is joined to the 5' end of the 3' exon [25].

Usually, the removed introns serve no further purpose and are quickly degraded, but in some cases they are functional and avoid degradation. For example, some microRNAs (miRNA) and most small nucleolar RNAs (snoRNA) [25] are derived from introns. Since identification of the introns and exons depends the spliceosome recognizing splice sites, splicing can be regulated by masking splice sites [88] or sequestering splicing factors [102]. In some cases, such *alternative splicing* is desirable, since it allows a single gene to code for multiple proteins; such proteins are called *isoforms* [4]. Changing the splicing pattern can also generate new types of RNAs such as circular RNAs [13]. On the other hand, incorrect splicing is associated with various diseases such as cancer [32]. Although the splicesome consists of both RNA and protein, it appears that the actual splicing reactions are catalyzed in large part, if not entirely, by RNA and not protein [105]. This makes the spliceosome a ribozyme.

2.3.3 Ribosomal RNA (rRNA)

The ribosome is an RNP that synthesizes protein from mRNA. It is comprised of two subunits: the small subunit and the large subunit, both of which consist of proteins associated *ribosomal RNA* (*rRNA*). rRNA is transcribed and processed in a membraneless compartment of the nucleus called the *nucleoulus*. The small subunit contains one type of rRNA, called 18S, while the large subunit contains three types: 5.8S, 27S, and 5S. 5S is transcribed from Pol III while 18S, 5.8S, and 28S are generated from a precursor strand called 47S. The 47S rRNA is transcribed by Pol I and its 5' end is not capped, nor is its 3' end polyadenylated [4]. 47S rRNA undergoes several rounds of cleavage performed by all three types of nucleases: 5'-3' exonucleases, 3-5' exonucleases, and endonucleases [37]. Additional factors are required for the correct cleavage of 47S, including a type of a ncRNA called *small nucleolar RNA* (*snoRNA*) [37]. snoRNA is generated from introns removed from certain genes during splicing [25]. Several ncRNAs (such as U3, U14, and U17) are believed to function as *chaperones*, ensuring that the rRNA folds correctly and is cleaved at the appropriate locations [107]. In addition to being cleaved into smaller strands, rRNA undergoes another type of processing: chemical modifications [65]. These modifications

are performed by proteins that are guided to specific locations on the rRNA by snoRNAs [4]. Not only do ncRNAs have a key role in the processing of ribosomal RNA, the ribosome is itself a ribozyme, with the rRNA in the large subunit catalyzing the reaction that joins two amino acids together [24].

2.3.4 Primer RNA

When a cell divides, its DNA is replicated by an enzyme called DNA polymerase. However, unlike RNA polymerase which can synthesize a new RNA strand from scratch, DNA polymerase can only append nucleotides to a base-paired 3' end of an existing nucleic acid strand. Therefore, a special type of RNA polymerase synthesizes short RNA strands called *primers* [4]. During replication, the DNA double helix is opened, allowing the single stranded DNA to form RNA-DNA duplexes with the primers. DNA polymerase uses these primers to start synthesizing DNA fragments. The RNA is then removed, replaced with DNA, and the fragments are ligated together. The primer RNA is removed by an enzyme called Rnase H which specifically cleaves the RNA strand of DNA-RNA duplexes [106]. This cleavage activity can be leveraged by a form a gene therapy involving *antisense oligonucleotides* (*ASOs*). ASOs are short DNA strands that are antisense (i.e. reverse complementary) to a target mRNA molecule. Since they are reverse complementary, the ASO and target mRNA form a strong DNA-RNA duplex that is recognized by Rnase H, which cleaves the mRNA strand, resulting in gene silencing [88].

2.3.5 MicroRNA (miRNA)

MicroRNAs (*miRNAs*) are small ncRNAs involved in post-transcriptional gene silencing [4]. There are many ways in which miRNA is generated, but the following pathway is by far the most common [94]. First, *primary miRNA* (*pri-miRNA*) is transcribed by Pol II. pri-miRNA has a secondary structure consisting of a terminal loop, an upper stem, and a lower stem that is recognized by a protein complex called *Microprocessor*. Microprocessor cleaves the pri-miRNA at a specific location, resulting in a new strand called *pre-miRNA*. pre-miRNA has a secondary structure consisting of a terminal loop, a stem, and 3' overhang that is recognized by a protein called Dicer. pre-mRNAs are exported to the cytoplasm, and are cleaved by Dicer at a specific location, resulting in an miRNA duplex. One strand, the *guide*, is loaded onto a protein called Argonaute, while the other, the *passenger* is discarded. Other proteins such as GW182 then interact with Argonaute to form an *RNA-induced silencing complex* (*RISC*). Although rarer, miRNA can also be generated from different precursors, such as introns of spliced pre-mRNA [100]. The guide strand leads RISC to a target mRNA by binding to a complementary region in the mRNA strand [100]. The outcome of this interaction depends on the specific type of Argonaute protein and the extent of the base-pairing between the miRNA and its target mRNA. If the miRNA binds completely to the target region, and if it is assembled with a specific Argonaute protein called Ago2, Ago2 will cleave the target mRNA [25]. However, if the base-pairing is limited to a *seed* region, or if one of the other argonaute proteins is involved, silencing

will occur via another pathway [25]. For instance, proteins may be recruited that remove the 5' cap and 3' poly(A) tail, making the mRNA strand vulnerable to degradation by exonucleases [43]. There is also a different pathway which does not directly degrade the mRNA, but instead prevents translation initation [43]. Interestingly, if both ends of the miRNA strand are base-paired with the mRNA, but the middle is *not*, then the miRNA will be degraded instead of the target mRNA [94].

2.3.6 Small interfering RNA (siRNA)

Small interfering RNAs (*siRNAs*) are small ncRNAs involved in post-transcriptional gene silencing, in a manner similar to miRNAs [4]. A key difference between miRNA and siRNA lies in the way in which they are produced. As we just saw, miRNAs are endogenously produced by the cell. In contrast, siRNAs are derived from long regions of double-stranded RNA typically associated with viral genomes [4]. The vast majority of viruses produce double-stranded RNA during replication, and in plants and invertebrates, this double-stranded RNA is detected by Dicer and cleaved into siRNAs [19]. These are loaded into catalytically active argonaute proteins which are guided to other viral RNAs via base-pairing, resulting in viral supression [19]. In mammalian differentiated cells, Dicer is unable to cleave this type of double-stranded RNA (although it can cleave pre-miRNA as previously discussed) [83]. Instead, viral defense is performed by the immune system. However, in stem cells, an isoform of Dicer (generated via alternative splicing), is able to cleave double-stranded viral RNA [83]. Finally, miRNA and siRNA can be synthetically generated, allowing researchers to silence target genes for experimental and therapeutic purposes. Different types of miRNA and siRNA precursors such as short hairpin RNA, Dicer substrate siRNA, and single-stranded siRNA can be introduced into the cell and processed into mature miRNA or siRNA via existing pathways [93].

2.3.7 Long non-coding RNA (lncRNA)

Long non-coding RNAs (*lncRNAs*) are RNAs greater than 200 nucleotides in length that do not code protein [85]. Many, but not all, lncRNAs are synthesized in a manner similar to mRNA: they are transcribed by Pol II, and undergo 5' capping and poly(A) tail addition [85]. lncRNAs often act in *cis*, by remaining near their transcription site in the nucleus, but they can also act in *trans* and move to distal genes in the nucleus or be exported to the cytoplasm [31]. lncRNAs are versatile molecules that perform a variety of a functions through unique mechanisms[108]. Enumerating all reported roles of lncRNAs is beyond the scope of this thesis; instead we will focus on a small subset of these.

Cis-acting lncRNAs play a key role in transcriptional regulation [31]. Some contain binding sites for protein complexes such as nucleosome remodelers [111] or histone modifiers [20] and they guide these complexes to a nearby promoter. This results in the epigenetic state of the promoter being altered, resulting in the activation or repression of transcription. Other lncRNAs regulate the three-dimensional organization of the gene. For example, one lncRNA interacts with Mediator which results in increased DNA looping and

transcription of nearby genes [56].

Some lncRNAs are *circular*. These circular RNAs (*circRNA*) are generated by *back-splicing* which can occur if exons are spliced out of order [13]. As previously mentioned, circular RNAs do not have any exposed 5' or 3' ends, and are thus very stable. Some circRNAs act as miRNA *sponges*. The binding of miRNAs to circRNAs spares their target mRNAs from RISC silencing, thus upregulating their expression [13].

One lncRNA, NEAT1, acts a protein scaffold and is essential for the formation of membraneless compartments called paraspeckles [29]. Paraspeckles can regulate gene expressing by sequestering RNAs containing a certain type of hairpin structure [29]. They can also co-localize pri-miRNAs with Microprocessor (see Section 2.3.5), thereby accelerating the processing of pri-miRNA into pre-miRNA by Microprocessor [42].

Finally, in one reported case, a circRNA interacts with a target mRNA via hairpin secondary structure motifs [112]. This interaction stabilizes the mRNA by disrupting the binding of a protein that induces the decay of mRNAs containing AU-rich elements [112].

2.3.8 Nucleolytic ribozymes

In this chapter, we have seen two ribozymes: the spliceosome and the ribosome, both of which are large molecular machines consisting of multiple ncRNAs and proteins. In Chapter 1 we were also introduced to the nucleolytic ribozymes, a class of small ribozymes that cleave their own backbone. To date, nine nucleolytic ribozymes have been discovered: pistol, hatchet, twister, twister-sister, hairpin, VS, HDV, *glmS*, and hammerhead [70]. In this work, we design two types of devices (ribogates and selective ribozymes) that depend on a nucleolytic ribozyme switching ON or OFF under different conditions. Specifically, we utilize a variant of the hammerhead ribozyme called the *minimal hammerhead ribozyme* (*mhRz*). The structure and sequence required for the mhRz to cleave itself are known [82] and are shown in Figure 2 a). An advantage of working with the mhRz is that it has no long range tertiary interactions; this means that knowledge of its secondary structure is sufficient to predict its activity. The mhRz is active if four secondary *structural motifs* are present: three *stems* and one 3-part *loop* called the *core* [82]. A stem consists of a series of *contiguous* base-pairs. It can be viewed as two paired nucleotide *segments*. Note that for a stem to form, the strand must fold back on itself in such a way that the first nucleotide of the first segment is paired with the last nucleotide of the second segment. In other words, the two segments must be reverse complementary. An n-part loop consists of n unpaired nucleotide segments linked together by n base-pairs. A 1-part loop called a *hairpin* is located at the end of stems 2 and 3 of the mhRz. The 3-part core links the three stems together. While the stems can generally assume any pair of complementary nucleotides (indicated by Ns in the figure), much of the core is constrained to have a specific sequence. The mhRz shown in Figure 2 a) is *cis*-acting: when it is active, it cleaves *itself* into two uneven segments at the location indicated by the arrow. The smaller segment will eventually dissociate from the larger one, allowing it to be used as an output signal.

By splitting the *cis*-acting ribozyme into two strands, one obtains a *trans*-acting ribozyme, shown in Figure 2 b). The *trans*-acting ribozyme contains stem 2 in its entirety,

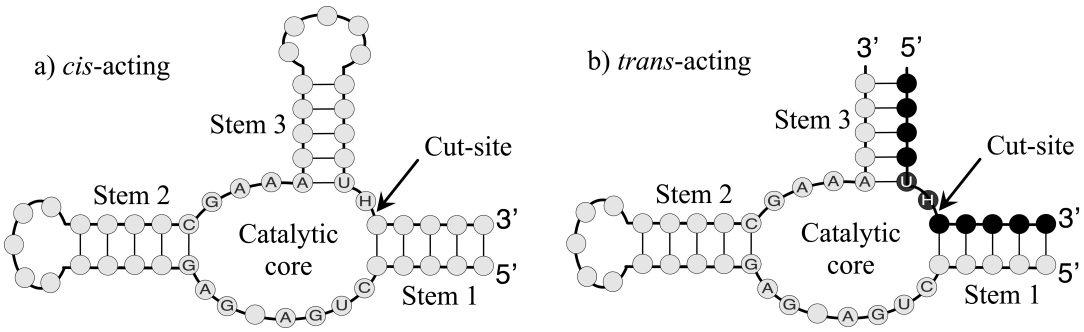


Figure 2: Active structure of a minimal hammerhead ribozyme (mhRz). a) A *cis*-acting mhRz consists of a single strand. Its active structure consists of three stems attached to a catalytic core. When active, the ribozyme will cleave itself into two at the position indicated by the arrow. b) A *trans*-acting mhRz consists of two strands: the ribozyme strand and the substrate strand (black). The active structure of the *trans*-acting mhRz also consists of three stems and a loop, but these motifs are split between the two strands. When the ribozyme strand binds to the substrate strand correctly, it will cleave the substrate at the location indicated by the arrow. In both the *cis* and *trans*-acting cases, nucleotides at certain position in the core must have specific values, indicated by letters. An H indicates any possible nucleotides except for a G. A lack of a letter indicates that the nucleotide is unconstrained.

but only half of stem 1, half of stem 3, and part of the core. The halves of stem 1 and stem 3 are referred to as *arms*. When the *trans*-acting ribozyme's arms bind to a second strand called the *substrate*, stem 1, stem 3, and the core are completed, and the substrate is cleaved.

Chapter 3

TruthSeqEr

3.1 Introduction

In the introductory chapter, we argued for the potential of smart biological devices to help with the diagnosis and treatment of diseases. We saw that these devices consist of three types of components: sensors, a logic circuit, and an actuator. There are many ways to build biological logic circuits, each with their own sets of advantages and disadvantages. They can operate at the transcriptional [75], post-transcriptional [33], translational [49] levels, or post-translational [15]. They can utilize biological components that already exist in nature [75] or design them *de novo* [33, 15, 90]. The logic circuit can consist of a single *co-localized* device [80, 33], or a network of multiple simpler components [75, 30]. The circuits also differ in the types of environments in which they were validated. Some are more theoretical and have only been validated *in silico*, while others have been validated in test tubes [81], bacteria [75], yeast [14], or even mammalian animal models [76]. Many of these biological logic circuits are designed in part or in entirely using computational tools [81, 90, 75].

In this chapter, we focus on logic circuits designed using allosteric ribozymes (ARs) (see Section 1.4). Using ARs for circuit implementation offers several theoretical advantages over alternative approaches. First, they are composed entirely of RNA, which makes predicting their structure easier compared to devices that based on proteins [99]. Second, they operate at the post-transcriptional level, meaning they don't require translation to occur, thereby accelerating logic computation and reducing energy consumption [33]. Third, they are intrinsically catalytically active, and therefore don't need to divert endogenous proteins from their natural pathways [7].

Penchovsky & Breaker [81], designed logic gates by replacing one of the hairpins of a hammerhead ribozyme with an *extension region*. Input signals were represented by short DNA or RNA strands called *oligonucleotides*. A 1 was represented by adding an input strand to a test tube and a 0 was represented by leaving it absent. The input strands had a strong affinity for segments of the extension region called *oligonucleotide binding sites* (*OBSs*). Different combinations of input strands binding to the extension region caused it to adopt different structures. The extension region had the *potential* to bind to the ribozyme,

thus disrupting its structure, rendering it inactive, and preventing the output signal from being cut and released. Whether it *actually* did so was determined by the sequences of the logic gate and input strands, as well as the specific combination of input strands placed in the test tube. Penchovsky & Breaker used *random search* to find sequences of ribozyme-based logic gates that implemented YES, NOT, AND, and OR functions. These gates were then validated *in vitro*. Many random *candidate* sequences were generated, subjected to a multi-stage evaluation, and rejected if they failed any stage. For each input state (combination of input strands), the BPPM of the gate was predicted using folding software. The BPPMs were then assessed to see whether the ribozyme was active for target ON states (i.e. states where the target function outputs a 1) and inactive for target OFF states. Surviving candidates were then subjected to additional criteria such as ensuring an optimal free energy gap between the ON and OFF states.

While this method was a significant contribution to the field of biological logic circuits, we highlight some important limitations. The algorithm is powered by random search, and as we will show in Section 3.4, random search scales extremely poorly, and the solutions it does find have little diversity. In addition, the designs were not created from scratch: they were obtained by modifying a relatively small portion of an existing well-characterized ribozyme. It is also unclear how applicable some of the evaluation criteria and their parameters are to more general sets of gates or to different conditions, such as cellular or *in vivo* environments. Finally, the method was specific to four particular 1 and 2-input gates, and one could not specify another target Boolean function by simply providing its formula (or equivalent truth table).

Some of these limitations were addressed in subsequent in work. In [80], 2-input AND and OR gates designed using the method in [81] were *manually* modified to produce experimentally validated 3-input AND and multiplexer gates. While this showcased the computational power of a single ribozyme-based logic gate, we must point out that creating these gates required specialized molecular biology knowledge. Ramlan & Zauner [86], used a more sophisticated *inverse folding* algorithm instead of random search. This method was shown to generate all possible 2-input logic gates (*in silico*; no *in vitro* validation was performed). However, they did not attempt to generate 3-input gates. Furthermore, the inverse folding algorithm required the user to specify *partial target structures* for each state of the device. We stress that this requires serious domain knowledge on the part of the user, and the number of partial structures that need to be provided increases exponentially with the number of inputs to the device. These partial structures also limit the diversity of the generated logic gates. Therefore, despite the potential of ribozyme-based logic gates, there is at present no published method that accepts a user-specified 1-3 input Boolean function and generates a set of diverse ribozyme-based logic gates.

To help address these limitations, we introduce a multi-objective evolutionary algorithm called *TruthSeqEr* that designs ribozyme-based logic gates that we refer to as *ribogates*. TruthSeqEr advances the state of the art in three ways. First, it is easy to use and requires no biological domain knowledge: the user simply specifies a target Boolean function and TruthSeqEr produces a population of ribogates implementing that function. Second, it is able to design 3-input ribogates that implement sophisticated linearly inseparable functions.

Third, it produces populations of ribogates that are structurally diverse. In the next section, we present in detail our ribogate model, and in Section 3.3, we provide an algorithmic description of TruthSeqEr.

3.2 Ribogate model

Our ribogate template is very similar to the allosteric ribozyme-based logic gate introduced in [81]. It consists of an extension region attached to a minimal hammerhead ribozyme. The extension region contains binding sites complementary to specific short RNA inputs. Inputs binding to the ribogate will cause it to change conformation and hence logical state. An input with a logical value of 1 is represented by the corresponding RNA input strand being present in the biological environment, whereas a 0 is represented by that RNA strand being absent. An output value of 1 is represented by the ribozyme folding into its active conformation, cleaving itself into two uneven strands, and releasing the shorter of the strands into the environment. Unlike previous work, we allow for up to 3 inputs, and consequently, the extension region may contain up to three binding sites. In addition, we add a new type of segment to the extension region called a *negator*. The purpose of the negator is to disrupt the ribozyme when all binding sites are occupied. The negator is only present for functions that output a 0 when all inputs are present (e.g. NOT, NAND). In Figure 3, we show the ribogate model for a 2-input XNOR gate. Note that the secondary structures presented in this thesis were visualized in entirety or in part using the FORNA software package [46].

3.3 Computational methodology

TruthSeqEr designs a *population* of ribogates over several *generations*. The first generation proceeds as follows. First, the representation of candidate ribogates is *configured* based on the target Boolean function. A population of random *individuals* is *initialized* and these individuals are each assigned a *fitness* measuring their performance on a set of objectives. Certain individuals are then selected as *parents*, who produce *offspring* through *mutation*. These offspring also have their fitness evaluated and a set of *survivors* is selected from the combined set of parents and offspring. This concludes the first generation. Subsequent generations cycle through parent selection, reproduction, fitness evaluation, and survivor selection. The survivors at the end of one generation become the population at the beginning of the next generation. This cycle *terminates* after a fixed number of generations. The final population may then undergo an optional post-processing step called *reliability assessment* which eliminates designs that are deemed biologically unreliable. Finally, the results are reported to the user. An overview of the TruthSeqEr algorithm is shown in Figure 4.

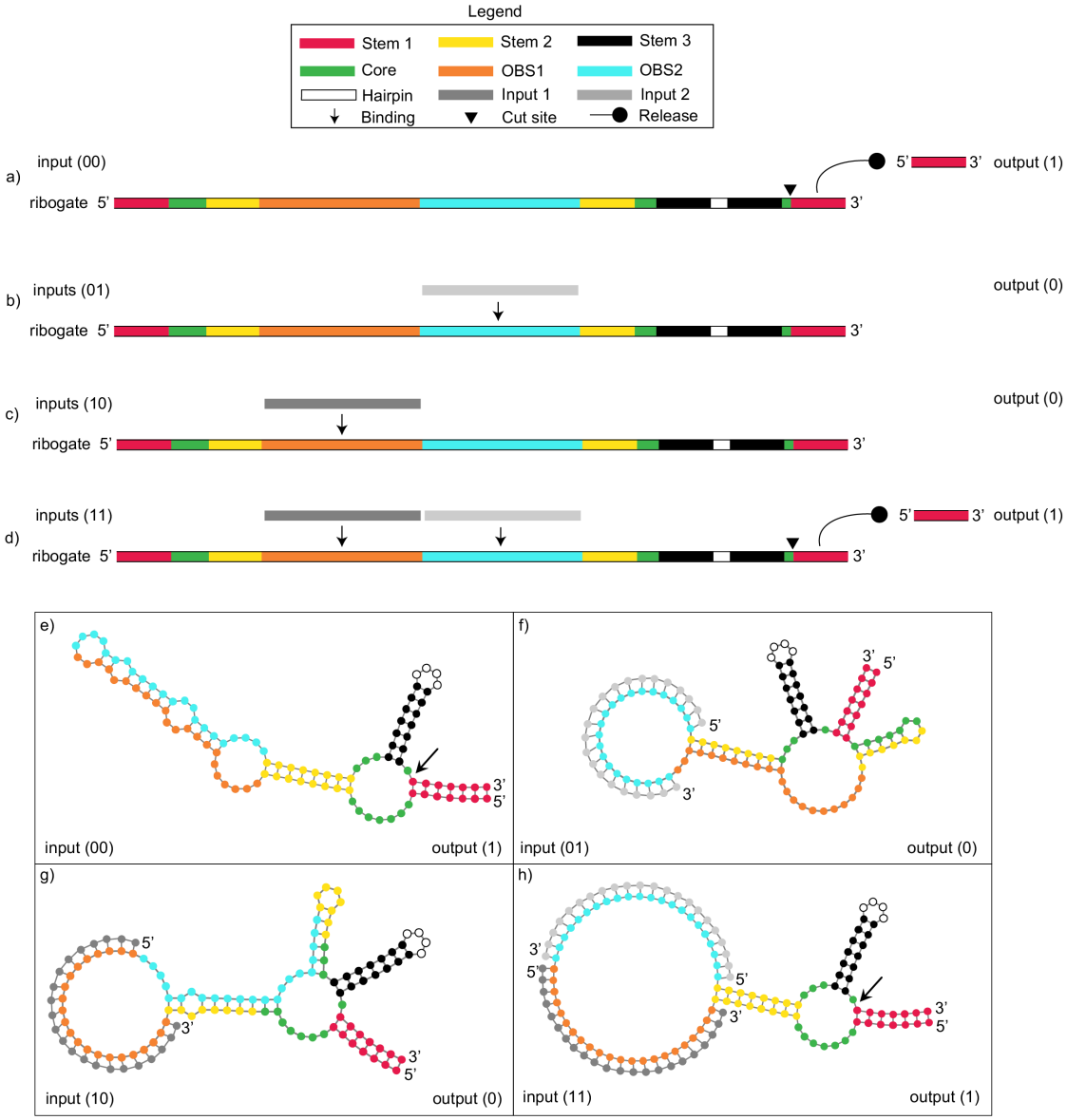


Figure 3: XNOR gate ribogate model. a-d) A ribogate contains all the segments necessary to fold into an active ribozyme: two halves of each stems and three parts of the catalytic core. However, also contains an OBS that modulate ribozyme activity by binding to short input strands. Each logical state is represented by a given combination of input strands. Some combinations cause the ribozyme to fold into an active conformation and release an output strand via self-cleavage. This represents an output value of 1. Other combinations render it inactive, so that no output strand is released. This represents an output value of 0. By changing the sequences of the inputs and the various ribogates segments, different logic functions can be implemented. e-h) show the actual structures that the XNOR ribogate adopts for each input combination. The ribozyme is active for states in which the three stems and catalytic core form.

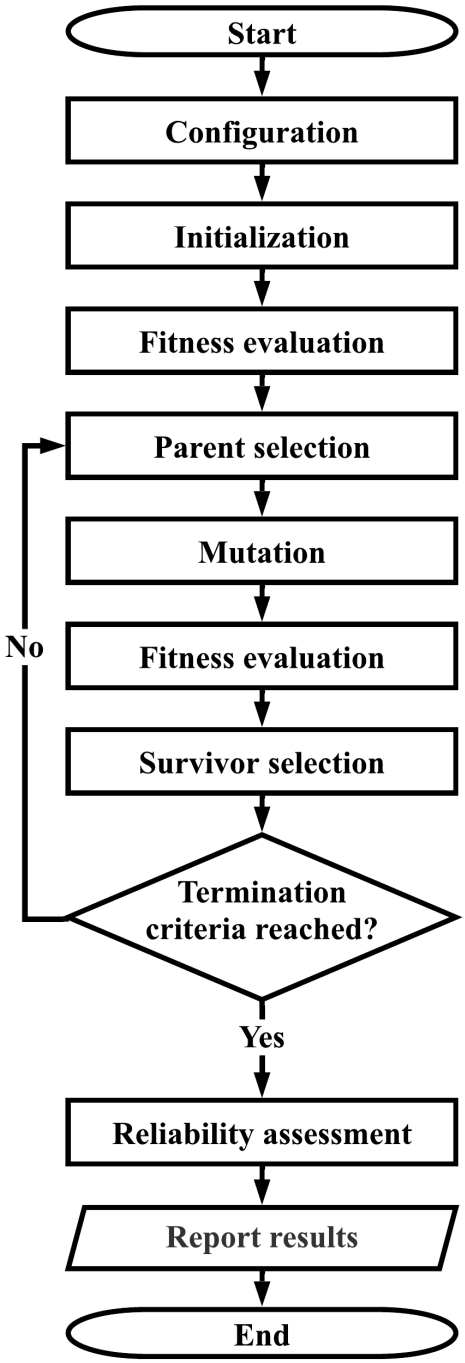


Figure 4: Flowchart of the TruthSeqEr evolutionary algorithm.

3.3.1 Configuration

Each individual consists of a gate strand and one or more input strands. The set of possible nucleotide sequences that these strands can assume is constrained by a *nucleotide dependency graph* (*NDG*). During the configuration stage, the NDG is first generated from a more abstract *segment dependency graph* (*SDG*). Then, a *mutation weight* and a set of *valid nucleotide assignments* are computed for each component of the NDG. The entire configuration stage is illustrated in Figure 6.

Segment dependency graph

The nodes of the segment dependency graph correspond to segments of the ribogate and the input strands. Edges represent two segments that must have the *potential* to bind together (i.e. they must be reverse complementary). Note that this does not mean that they *will* bind together. Each node is labeled with a string that constrains the sequence of nucleotides that the segment can assume. The SDG of an n-input ribogate contains the following segments:

1. n input segments, for one each input strand
2. 10 ribozyme segments
   (a) 1 pair of segments for each of the three stems
   (b) 3 segments for the catalytic core
   (c) 1 segment for the stem 3 hairpin
3. n or $n + 1$ extension region segments
   (a) n OBS segments
   (b) 1 negator segment if the target Boolean function outputs a 0 in the state when all inputs are 1

When traversing the gate strand from the 5' end to the 3' end, the segments are encountered in the following order:

1. First half of stem 1 (S1A)
2. First part of core (C1)
3. First part of stem 2 (S2A)
4. Negator (if applicable) (N)
5. OBS1 (O1)
6. OBS2 (if applicable) (O2)
7. OBS3 (if applicable) (O3)

8. Second half of stem 2 (S2B)
9. Second part of core (C2)
10. First half of stem 3 (S3A)
11. Hairpin of stem 3 (S3H)
12. Second half of stem 3 (S3B)
13. Third part of core (C3)
14. Second half of stem 1 (S1B)

We designate these segments as low-level or *L-segments*. Later in this work, we will use an alternative partitioning scheme to divide the ribogate into high-level or *H-segments*. In high level partitioning, the ribozyme is treated as a *single* segment. The other segments (inputs, extension region) are handled the same as way as in low-level partitioning. Examples of low-level and high-level partitioning are shown in Figure 5.

The topology of the SDG is simple: it consists of connected components of size 1 (isolated segments) or 2 (reverse complementary segments). The two segments of each stem must be reverse complementary to each other: their corresponding nodes in the SDG are therefore connected by an edge. The same is true for each OBS and its corresponding input segment. In addition to these structural constraints, each node is labeled with a constraint string. This string is composed of characters representing the nucleotides A, U, G, C, as well as the wild-card characters N and H. An N indicates the nucleotide can assume an A, U, G, or C, while an H indicates that the nucleotide can assume an A, U, or C [17]. The SDG for XNOR is shown in Figure 6 a).

Nucleotide dependency graph

The SDG has an equivalent *nucleotide dependency graph* (*NDG*) [36] which encodes constraints between individual nucleotides instead of segments. An NDG is generated from the SDG by:

- Splitting each unpaired segment of length L into L unpaired nodes (representing nucleotides).
- Splitting each pair of segments of length L into L pairs of nodes. Since paired segments must be reverse complementary, the i^{th} nucleotide of the first segment is paired with the $L-1-i^{th}$ nucleotide of the second segment.
- Splitting the constraint strings into individual characters and applying them to the appropriate nodes of the NDG.

The NDG for XNOR is shown in Figure 6 b).

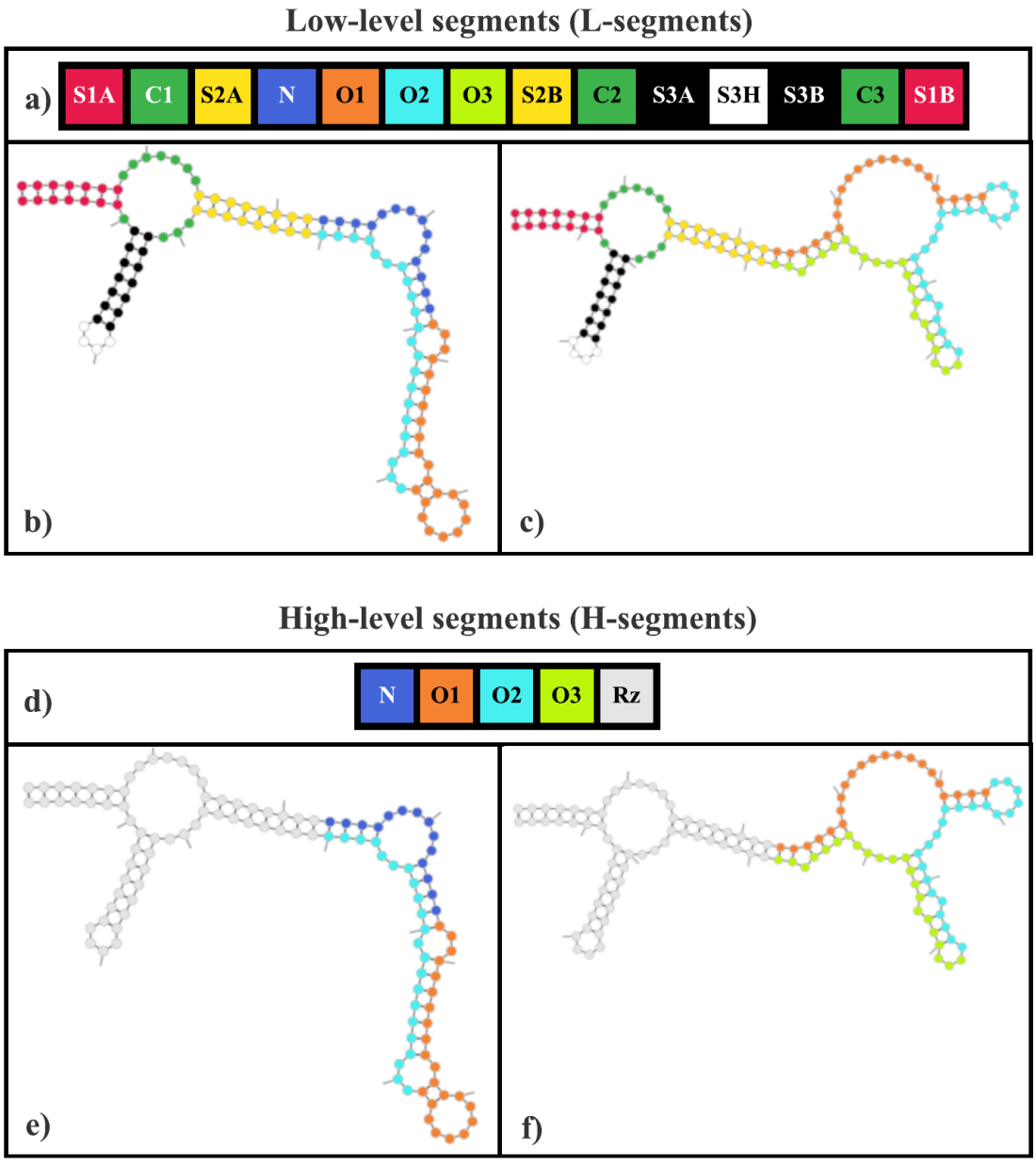


Figure 5: Two ways of partitioning ribogates into segments. At a low level (a-c), both the ribozyme and extension region are subdivided into segments. At a high level (d-f), only the extension region is subdivided: the ribozyme is treated as a single segment. In general, the extension region may contain up to three OBSs and a negator (a, d), but its specific composition depends on the target function. (b,e) show a 2-input gate with a negator while (c,f) depict a 3-input gate with no negator. List of abbreviations: Rz (ribozyme), O_n (OBS n), N (negator), C_n (segment n of catalytic core), S_nA (first half of stem n), S_nB (second half of stem n), S3H (hairpin loop of stem 3).

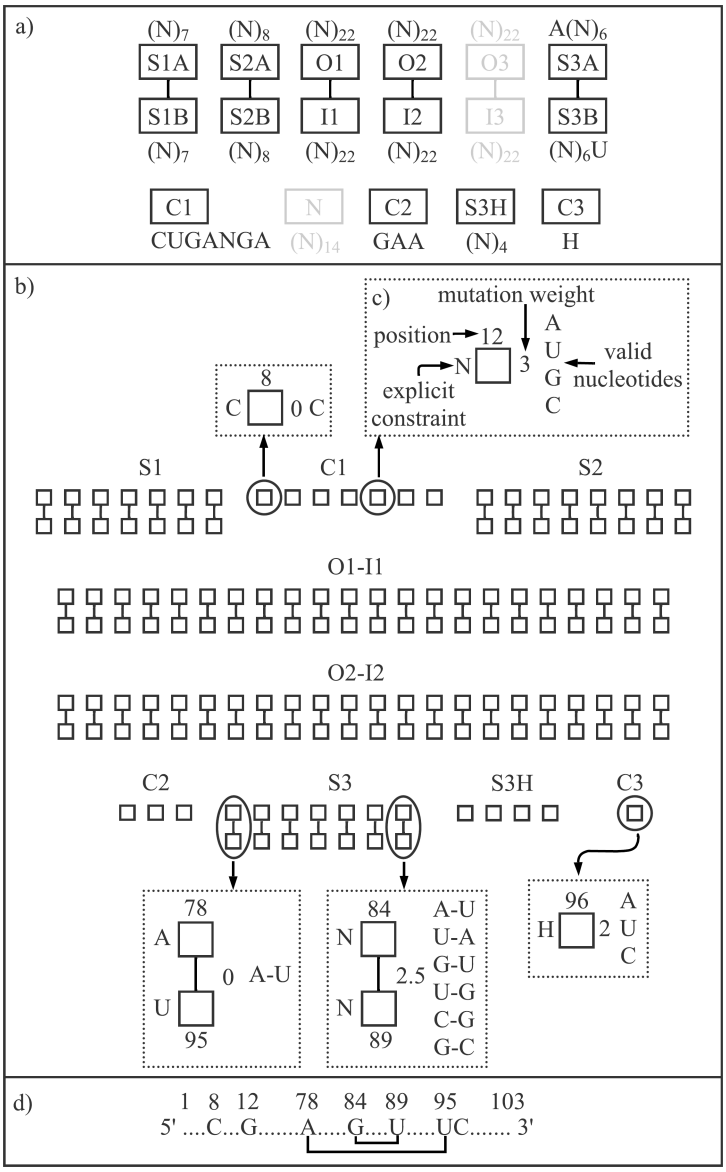


Figure 6: Configuration example. a) The SDG for an XNOR function (1001). List of abbreviations: I_n (input n), O_n (OBS n), N (negator), C_n (segment n of catalytic core), S_nA (first half of stem n), S_nB (second half of stem n), S3H (hairpin loop of stem 3). Since XNOR outputs a 1 when all inputs are 1, the negator segment is not included. This is represented by it being faded out in the figure. Also, since XNOR has two inputs, the OBS3 - input 3 pair is not included. Each component of the SDG is annotated with a nucleotide constraint string. The notation N_6 is equivalent to NNNNNN. b) The NDG is obtained by splitting the SDG into nucleotide components. c) Each component has an constraint character, a set of valid nucleotide assignments, and a mutation weight. These are shown for only certain components in the figure. d) The NDG constrains the possible sequences of the gate and inputs strands. A gate strand with a valid nucleotide assignment is shown.

Mutation weights and valid nucleotide assignments

During the initialization and mutation stages of TruthSeqEr, the gate and input strands are assigned nucleotide values that respect the NDG. In general, generating valid nucleotide assignments is non-trivial [36], especially when sequence constraints are imposed. Fortunately, due to the structural simplicity of the NDG (each connected component consists of at most two nodes), TruthSeqEr can generate valid nucleotide assignments using the following straightforward approach. Each component of the NDG is considered one at a time. Each component has a set of *valid* nucleotide assignments. An isolated nucleotide can assume any value permitted by its constraint character. A pair of nucleotides can assume any pair of complementary values that respect the constraint characters of both nucleotides. Components with a greater number of valid nucleotide assignments are more likely to be selected for mutation. Each connected component C has an associated *mutation weight* equal to:

$$\frac{\text{\# of valid nucleotide assignments of } C \text{ - 1}}{\text{\# of nucleotides in } C} \tag{1}$$

Valid nucleotide assignments and mutation weights for selected components of the NDG are shown in Figure 6 c).

3.3.2 Initialization

During initialization, each component of the NDG is selected and its corresponding nucleotides are assigned a random set of valid nucleotide values. An example of nucleotide assignment is shown in 6 d).

3.3.3 Fitness evaluation

TruthSeqEr's *primary* objective is to discover ribogates with a maximal *viability* score. Ribogates with high viability scores are ON in states where the target logical output is a 1 and OFF in states where the target logical output is a 0. However, directly optimizing a single viability score does not produce ideal results. Instead, TruthSeqEr is *guided* toward viable individuals by a set of *secondary* objectives. These secondary objectives also increase the *diversity* of the discovered logic gates. During fitness evaluation, the viability score is calculated as well as three secondary objective scores: *ON*, *OFF*, and *novelty* [58]. The ON score measures how *active* the ribozyme is in states where the target logical output is a 1, the OFF score measures how *inactive* the ribozyme is in states where the target logical output is a 0, and the novelty score measures how structurally *unique* the ribogate is compared to other ribogates in the population. To increase selection pressure, the secondary objectives are *nullified* if an individual's viability drops below a certain threshold.

Fitness evaluation is broken into five sub-stages. *Folding* predicts the structures that an individual's gate strand will adopt for each input combination. *Phenotype generation* processes the folding output into a more compact phenotype. *Novelty assessment* calculates a novelty score for each individual by measuring the distance between its phenotype

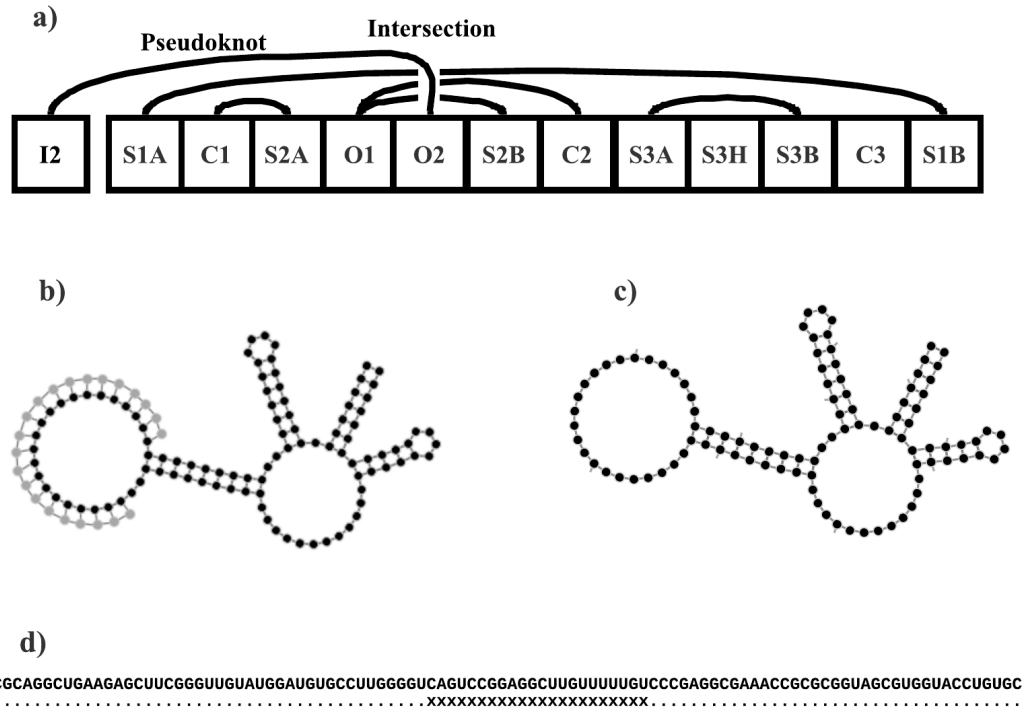


Figure 7: Folding constraints. a) In a ribogate, when an input binds to its OBS, a pseudoknot is formed because it crosses other base-pairs such as those in stem 1. b) We assume that the input always binds completely to its OBS. c) If the OBS is occupied, it cannot bind elsewhere on the gate strand. From the point of view of the other nucleotides, it is the same as if the OBS was constrained to be unpaired. d) This constraint is specified by means of a folding constraint string. Nucleotides above a dot are unconstrained. Nucleotides above an x are forced to be unpaired, even if they want to form base-pairs.

and those of the rest of the population. *Performance assessment* calculates the individual's viability, as well as its ON and OFF scores, from the predicted structures. Finally, *viability nullification* sets the secondary objectives scores to zero if the viability drops below a certain threshold.

Folding

Fitness evaluation begins by predicting the structure that the gate strand adopts for each input combination. Existing folding algorithms can predict the structure that the gate strand will adopt in isolation with reasonable accuracy and computational efficiency. However, they cannot do so when the input strands must be considered in addition to the gate strand. This is because a *pseudoknot* occurs when an input strand binds to the gate strand. A pseudoknot consists of two base-pairs that cross each other. This is shown in Figure 7 a).

Predicting the formation of pseudoknots accurately and efficiently is not possible in

general. For example, the algorithm presented in [23] runs in $O(n^5)$ time compared to the $O(n^3)$ time required for pseudoknot-free prediction. In addition, it can only predict a certain subset of pseudoknots. Fortunately, we can exploit *a priori* knowledge to circumvent this issue. We know that each input is reverse complementary to its corresponding OBS. In addition, since the input and OBS are long, many base-pairs form when they bind to each other. It is therefore reasonable to assume that each input will bind to its corresponding OBS, Since the OBS nucleotides are bound to the input nucleotides, they cannot bind to any other nucleotide on the gate strand. We can simulate this effect by means of a *folding constraint string* which *forces* these OBS nucleotides to remain unpaired. The effect of folding constraints is illustrated in Figure 7 b-d. For each state, the folding algorithm outputs a base-pairing probability matrix (BPPM) and an MFE structure. The MFE structure plays no role in the design of a ribogate, but it is useful for its analysis (Section 3.4).

Phenotype generation

An individual's *phenotype* is a vector that measures the amount of base-pairing between certain H-segments. The BPPM of each state is first coarse-grained into a *segment-pair magnitude matrix* (*SPMM*). The SPMMs are then processed, merged and flattened into a single vector (called the phenotype).

SPMMs. An entry of the SPMM encodes the amount of base-pairing between an extension region (ER) segment and another ER segment (possibly itself) or between an ER segment and the ribozyme (Rz). The ribozyme is treated as a black-box, meaning that base-pairs between any two nucleotides of the ribozyme are ignored. The amount of base-pairing between two segments is equal to the sum of the probabilities of all possible base-pairs that can form between these two segments. An example of coarse-graining a BPPM into a SPMM is shown in Figure 8 a) and b).

Phenotype. Next, the SPMMs of target ON states (i.e. states for which the output value of target Boolean function is 1) are selected for further processing. Specifically, the entries encoding the base-pairing between each extension region segment and the ribozyme are removed. The SPMMs of target OFF states are not modified. The processed SPMMs are then flattened and merged into a phenotype vector. This is shown in Figure 8 c). Excluding the ER-Rz entries of the target ON state SPMMs from the phenotype prevents novelty search from exploring solutions that are *guaranteed* to be non-viable.

Novelty assessment

Next, the novelty of an individual is assessed and a novelty score is calculated. Novelty assessment begins by calculating the *phenotypic distance* between each pair of individuals in the population. The phenotypic distance d between two individuals x and y with respective

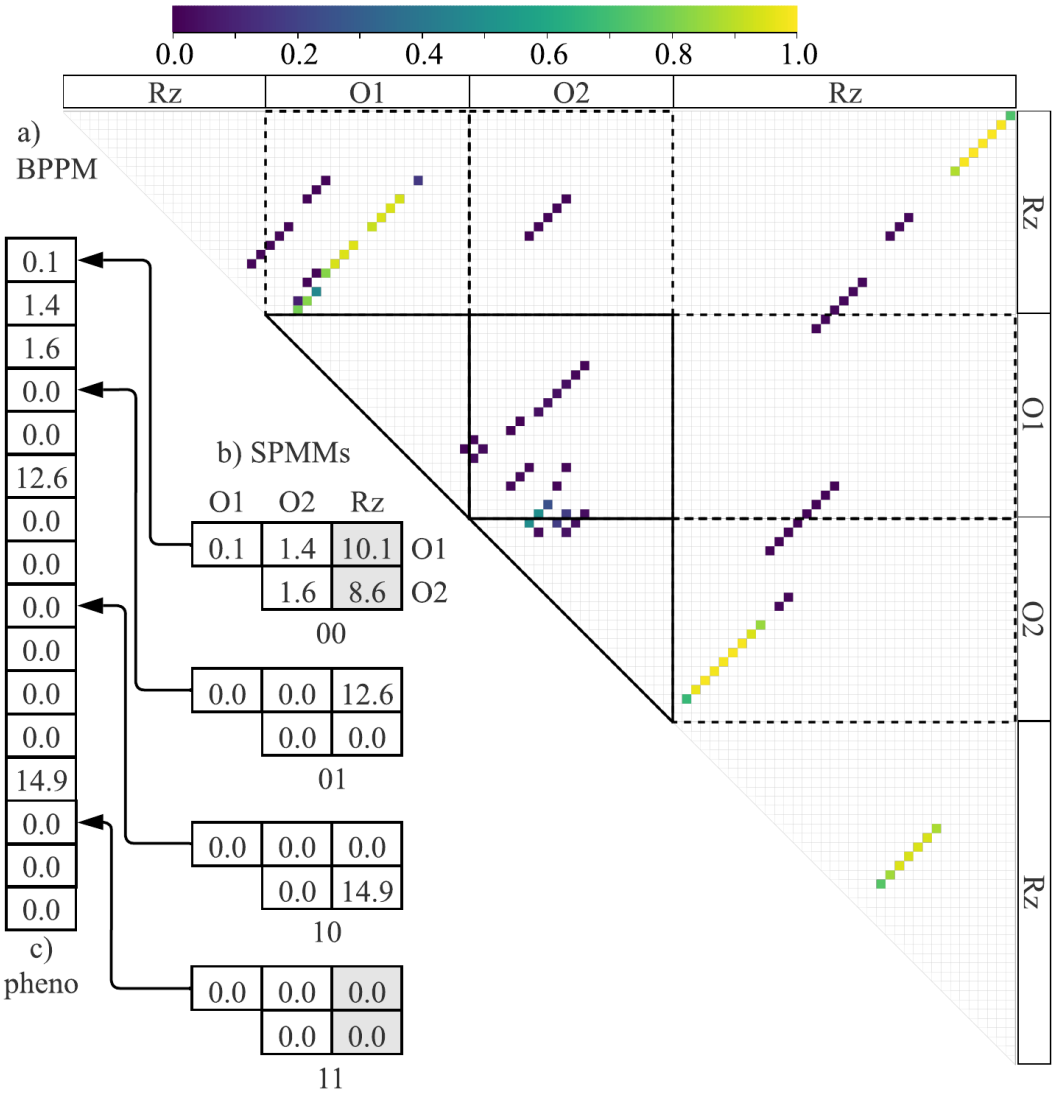


Figure 8: Phenotype generation example. a) The BPPM of state 00 of an individual targeting an XNOR gate. b) This BPPM is coarse-grained into a SPMM. We can see that there are significant interactions between the two OBSs (O1, O2) and the ribozyme (Rz). There are only weak interactions between the two OBSs and within each OBS. The BPPMs of the other three states are not shown, but their corresponding SPMMs are. c) The SPMMs are flattened and merged into a phenotype vector. Note that the OBS-ribozyme interactions are grayed out for states 00 and 11, and are excluded from the phenotype. The XNOR gate should be ON in those states and we do not want to reward individuals with OBS-ribozyme interactions. List of abbreviations: O_n (OBS n), Rz (ribozyme), BPPM (base-pair probability matrix), SPMM (segment-pair magnitude matrix).

phenotypes x^p and y^p is defined as:

$$d(X, Y) = \sum_{i=1}^{n} |x^p[i] - y^p[i]| \tag{2}$$

where n is the number of entries in the phenotype vector. Each individual x is then assigned a novelty score defined as

$$f^{nov} = \sum_{y \in K} d(x, y) \tag{3}$$

where K is the set of x's k-nearest neighbors. The novelty score rewards individuals that are in *sparse* regions of the phenotype space (i.e. that are far away from their nearest neighbors).

Performance assessment

Next, the performance of an individual is assessed and viability, ON, and OFF scores are calculated. An individual is considered high performing if its ribozyme is active (inactive) in each target ON (OFF) state.

Motif scores. So far we have considered ribozyme activity in binary terms: it is either ON or OFF. In this section, we quantify the ribozyme activity level more precisely. In Section 2.3.8, we explained that the ribozyme is active if four motifs form: the three stems and the core. Therefore, the activity level of a ribozyme can be predicted by measuring the degree to which these motifs are present in each state. TruthSeqEr calculates four *motif scores* from each BPPM. Three *stem scores* measure the average probability of a stem base-pair being present. A stem score of 1 indicates each base-pair in the stem has a 100% probability of being present. A single *core score* measures the average probability of a core nucleotide being unpaired. This score is 1 if every nucleotide has a 100% probability of being unpaired. The motif scores of a device with k states are stored in a k by 4 matrix. Each row corresponds to a state, and each column to a motif. Column 0 corresponds to the core motif and columns 1-3 to the stem motifs. Motif score calculation is illustrated in Figure 9 a-b).

ON and OFF matrices. The motif scores matrix is split into an n by 4 *ON matrix*, and a $(k - n)$ by 4 *OFF matrix*, where n is the number of target ON states. Note that $0 < n < k$. The ON (OFF) matrix stores the motif scores for target ON (OFF) states .

ON and OFF vectors. From the ON (OFF) matrix, TruthSeqEr calculates an ON (OFF) vector v^{ON} (v^{OFF}) that measures how close the ribozyme is to being active (inactive) for each target ON (OFF) state. The ON and OFF vectors are calculated slightly differently. In this work, we assume that the ribozyme is fully active if all four of its motifs form, and we assume that it is fully inactive if one of its stems is completely disrupted (i.e. none of the nucleotides in the stem are properly paired). The i^{th} entry of the ON vector is equal to

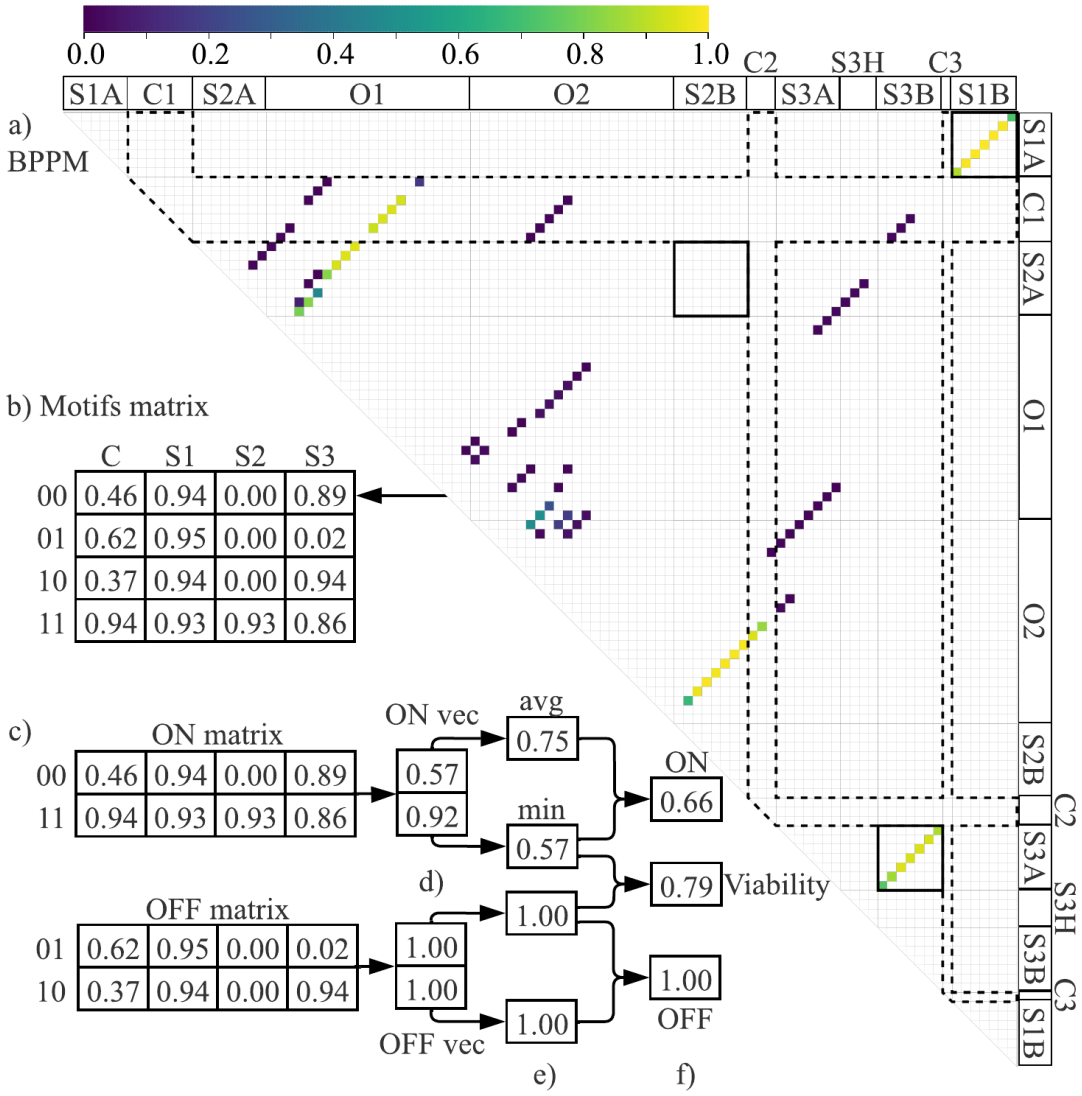


Figure 9: Performance assessment example. a) The BPPM of state 00 of an individual targeting an XNOR gate. The partners of the core nucleotides are shown inside the dashed polygon. For the core to form properly, this polygon must be empty. Since it is not, the core score for state 00 is low. The 3 stems are shown in bold boxes. For them to fold properly, there must be strong base-pairs along the diagonal. This is the case for stems 1 and 3, but not for stem 2. Therefore, stems 1 and 3 have high scores, and stem 2 has a low score. b) These scores are stored in the motifs matrix. c) The motifs scores matrix is split into two matrices representing the scores for target ON (00 and 11) and target OFF (01 and 10) states. d) The matrices are processed into vectors. e) The minimum and average values of these vectors are calculated. f) Pairs of averages and mins are themselves averaged into ON, OFF, and viability scores. Since stem 2 is disrupted for each target OFF state, the OFF score is high. However, stem 2 is also disrupted in state 00 when the ribozyme should be active. Therefore, the ON score is low.

the average of the motif scores of the i^{th} row of the ON matrix. The i^{th} entry of the OFF vector is equal to 1, the minimum stem score of the i^{th} row of the OFF matrix. ON and OFF vector calculation is illustrated in Figure 9 d).

Scores. From these two vectors, we calculate an ON score f^{ON} and an OFF score f^{OFF} as follows:

$$f^{ON} = \text{ama } v^{ON} \tag{4}$$

$$f^{OFF} = \text{ama } v^{OFF} \tag{5}$$

where we define ama as:

$$\text{ama } x = \text{avg } [\text{avg } x, \min x] \tag{6}$$

We also calculate the viability score as:

$$f^{via} = \frac{\min v^{ON} + \min v^{OFF}}{2} \tag{7}$$

ON, OFF, and viability score calculation are illustrated in Figure 9 e-f).

Primary vs secondary objectives. Maximizing the viability of the logic gates is our primary objective. The viability score is *stringent*: it judges the performance of a logic gate by its worst performing ON and OFF states. However, using viability to guide the search is not ideal since it does not reward improvement in states that are not the worst performing. On the other hand, the ON and OFF scores are less stringent. They take into account the performance of the gate across *all states*, although they apply a disproportionate weight to the worst performing state. Therefore, the ON and OFF secondary objectives are used to guide TruthSeqEr *instead* of viability. Also note that the ON and OFF scores are kept as two separate objectives and not summed together. This strategy is called *multi-objectivization* [101] and it has been shown to improve the performance of EAs for certain tasks.

Viability nullification

Finally, the ON, OFF, and novelty scores are *nullified* if the viability score of the individual drops below a certain *threshold*. The viability threshold is initially set low to give TruthSeqEr the opportunity for exploration. It is subsequently raised for every generation, following one of two possible schedules. The first schedule interpolates between an initial value v_0 and a final value v_f. The viability threshold $v_t(i)$ at the i^{th} generation (with i being 0-indexed) is equal to

$$v_t(i) = v_0 + i\frac{v_f - v_0}{N - 1} \tag{8}$$

where N is the total number of generations in the TruthSeqEr run. The second schedule interpolates between an initial value v_0 and a *breakpoint* value v_b prior to a breakpoint

generation b, and then interpolates between the breakpoint value and a final value v_f after the breakpoint generation. The viability threshold $v_t(i)$ at the i^{th} generation is equal to

$$\begin{cases} v_t(i) = v_0 + i\frac{v_b - v_0}{b} & i < b \\ v_t(i) = v_b + (i-b)\frac{v_f - v_b}{N-1-b} & i \geq b \end{cases} \tag{9}$$

An individual is considered viable at the end of a TruthSeqEr run if its viability score is greater than or equal to 0.95.

3.3.4 Parent selection and reproduction with mutation

First, each individual is selected as a parent. Next, each parent is copied and mutated R times, where R is the *mutation rate*. Mutation consists of selecting a component of the NDG and assigning its corresponding nucleotides a new set of valid values. The probability of a component being selected for mutation is proportional to its mutation weight.

3.3.5 Survivor selection and termination

The population and offspring are merged into a set of $2N$ individuals and the fittest N individuals are selected as survivors. Since each individual has multiple objective scores, NSGA-ii [21], a multi-objective sorting algorithm, is used to rank the individuals. NSGA-ii sorts the individuals into a set of *non-dominated fronts*. An individual x dominates another individual y if *all* of x's objective scores are *equal to or greater than* y's and *at least* one of x's is *strictly greater* than y's. All members of the same non-dominated front are dominated by the same number of individuals. The individuals of the first non-dominated front are not dominated by any other individual. Once they are sorted into non-dominated fronts, TruthSeqEr selects the N individuals dominated by the smallest number of other individuals as survivors. TruthSeqEr terminates after 200 generations.

3.3.6 Reliability assessment

If reliability assessment is enabled, the members of the final population of the EA are measured against additional criteria to determine their biological plausibility. Two new scores are calculated: *thermobalance* and *affinity*. An individual is eliminated if any of these scores falls outside its specified range.

Thermobalance

In previous steps of the algorithm, each state was considered in isolation. However, in reality the states are linked together through a dynamic process. In general, each state has multiple *successor* states. These are the next possible states that the gate can transition to upon the binding of specific input strands. For example, a 3-input ribogate in state 000 (no inputs bound to it) has three possible successor states: 001, 010, and 100, corresponding to the binding of the third, second, and first input strand, respectively. The proper functioning

of a ribogate depends on its ability to transition to the correct successor state when the relevant input strand binds, and to avoid unwanted transitions in the absence of that input. This requires the free energy gap between a state and its successor to be large enough to prevent unwanted transitions, but *not* so large as to prevent desired transitions when the correct input is present. The thermobalance score is equal to the average energy gap for every possible state transition. The acceptable range of this score is between 6 and 10.

Affinity

Recall that due to limits in pseudoknot prediction, we did not co-fold the gate and input strands directly. Instead, we assumed that each input was specific to its corresponding OBS, Unfortunately, in reality the input strand may bind elsewhere on the strand. It is therefore important to assess the validity of this original assumption. Although co-folding was too inaccurate to predict the *global* structure of the input-gate duplex, it is still useful for predicting *local* substructures involving the input strands. We therefore co-fold each input with the gate strand, and use the resulting BPPM to calculate the average probability of an input nucleotide binding to its corresponding OBS partner. The acceptable range for this score is 0.85 or above.

3.3.7 Experimental setup

Test cases

There are 2^{2^n} possible functions of n variables. This means that there are 256 possible 3-input functions. In order to reduce the computational cost of generating designs and the future experimental cost of validating them in a biological environment, we test our algorithm on a subset of the possible functions. First, we only consider n-input functions that depend *exactly* on n variables. For example, we do not consider the 3-input function a + b. We do however consider it as a 2-input function. Next, we observe that the set of possible functions can be partitioned into a set of *equivalence classes*. In this work we consider two types of equivalence classes: *P* and *NPN*. Two functions are part of the same P equivalence class if one can be obtained from the other by applying a *permutation* to its inputs [96]. Two functions are part of the same NPN equivalence class if one can be obtained from the other by applying a permutation to its inputs, *negating* any subset of its inputs (including possibly none or all of them), and/or negating its output [96].

With this in mind, we form *representative* sets of 1, 2, and 3-input functions. The 1-input set is simply the two 1-input functions of exactly 1 variable. The 2-input set consists of one member of each P equivalence class [26]. Finally, the 3-input set consists of one member of each NPN equivalence class [96]. A Boolean function can be denoted by its truth vector or its *function number*, the integer equivalent of its truth vector [96].

These functions exhibit varying degrees of complexity: some functions are *linearly separable* while others are linearly *inseparable*. In the case of a (Boolean) linearly separable function, states with an output value of 0 can be distinguished from states with an output value of 1 using a single *hyperplane* (e.g., a line in 2D or a plane in 3D). This hyperplane

Function number	Truth vector	Boolean expression	Description	Linearly separable?
f-1-1I	01	a	YES	YES
f-2-1I	10	a'	NOT	YES
f-1-2I	0001	ab	AND	YES
f-2-2I	0010	ab'		YES
f-6-2I	0110	ab' + a'b	XOR	NO
f-7-2I	0111	a + b	OR	YES
f-8-2I	1000	a'b'	NOR	YES
f-9-2I	1001	ab + a'b'	XNOR	NO
f-11-2I	1011	a + b'		YES
f-14-2I	1110	a' + b'	NAND	YES
f-1-3I	00000001	abc	AND	YES
f-7-3I	00000111	ab + ac	OR with enable	YES
f-9-3I	00001001	abc + ab'c'	XNOR with enable	NO
f-22-3I	00010110	abc' + ab'c + a'bc	ON for exactly 2 inputs	NO
f-23-3I	00010111	ab + ac + bc	ON for at least 2 inputs	YES
f-25-3I	00011001	bc + ab'c'		NO
f-27-3I	00011011	ac' + bc	Multiplexer	NO
f-105-3I	01101001	abc + ab'c' + a'bc' + a'b'c	Parity check	NO
f-129-3I	10000001	abc + a'b'c'	ON for 0 or 3 inputs	NO
f-135-3I	10000111	ab + ac + a'b'c'		NO

Table 1: Representative functions used as test cases for TruthSeqEr. The 1-input and 2-input functions are representatives from each P equivalence class. The 3-input functions are representatives from each NPN equivalence class.

serves as a *decision boundary* between the two classes of output values [8]. For linearly inseparable functions, a hyperplane is inadequate and more complex decision boundaries are required. Linear separable functions are generally considered easier to represent than linear inseparable functions. For instance, a single *artificial neuron* is sufficient to represent a linearly separable function, whereas a *neural network* comprised of multiple neurons is required to represent a linearly inseparable function [72]. The representative sets for 1, 2, and 3 inputs are shown in Table 1, with equivalent Boolean expressions and linearly separability clearly indicated [26] .

Parameters

We executed 4 runs of TruthSeqEr in addition to a random search run (to provide a comparison to [81]). All 5 runs performed 60,000 fitness evaluations. For the 4 TruthSeqEr runs, this was divided into 200 generations of 300 individuals. A mutation rate of 4 was used for all TruthSeqEr runs. Each TruthSeqEr run had a different set of objective scores: (ON,

	Run 1	Run 2	Run 3	Run 4
Number of generations	200	200	200	200
Population size	300	300	300	300
Mutation rate	4	4	4	4
Objective scores	ON, OFF	ON, OFF	ON, OFF, Novelty	ON, OFF, Novelty
Viability nullification?	No	Yes	No	Yes
Viability threshold (min, max)	N/A	(0, 0.9)	N/A	(0, 0.9)
Breakpoint (generation, value)	N/A	(49, 0.45)	N/A	(49, 0.45)
Novelty neighborhood size	N/A	N/A	30	30

Table 2: TruthSeqEr parameters.

Score	Min value	Max value
Thermobalance	6	10
Affinity	0.85	N/A

Table 3: Acceptable ranges for the two reliability scores.

OFF), (ON,OFF, viability nullification enabled), (ON, OFF, novelty), (ON, OFF, novelty, viability nullification enabled). The novelty score was calculated using a neighborhood size of 30. The EA parameters are summarized in Table 2. Folding was performed using the RNAFold and RNACofold programs from the ViennaRNA package [61]. The partition function option was enabled. Also, the *MaxLoop* parameter in the source code was changed from 30 to 300 before recompiling the package. This change prevented the folding software from incorrectly predicting the structure of larger ribogates. Experiments were performed on a system running Windows 10 Pro with an AMD Ryzen Threadripper 3970X 32-Core Processor and 64 GB of RAM. Each run used 12 processes in parallel and 4 runs were simultaneously executed. 1-input, 2-input, and 3-input runs took approximately 2, 4, and 8 hours, respectively. The acceptable ranges for the reliability scores are summarized in Table 3.

3.4 Results and discussion

TruthSeqEr successfully found diverse ribogates implementing *every* test function. We will first present results at a *population* level. We will discuss which combinations of objectives yielded the best results. We will visualize the phenotype space of different functions. We will also quantify and discuss the impact of reliability filtering. We will then present examples of *individual* ribogate designs. Three levels of abstraction will be considered, with each level revealing different information about ribogate operation.

3.4.1 Evaluating the performance of the TruthSeqEr EA

The results of random search and the 4 TruthSeqEr EA runs are summarized in Figure 10. For each (function, run) combination, the number of viable individuals at the end of the run was recorded and a diversity score was calculated (see Appendix for details). Figure 11 provides an in-depth look at the runs of functions f-1-1I, f-6-2I, and f-105. These functions are all instances of an n-input *parity check circuit* and make an informative case study in how different search methods scale. The phenotype space of these functions is visualized using a non-linear dimensionality reduction technique called *multidimensional scaling* (*MDS*) [52]. We now discuss these results in detail.

Rigobate design is non-trivial

In [81], random search was used to design simple 1 and 2-input ribogates. It was therefore an open question as to whether it could handle more complex designs. Our results indicate that this is not the case. Random search found many viable solutions for the 1-input functions, significantly fewer solutions for the 2-input functions, and barely any solutions for the 3-input functions. Indeed, it found no viable solutions for 8 of the 10 3-input functions. Although it found a very small number of viable solutions for functions 1-3I and 7-3I, these are both linearly separable. In all cases, the solutions found by random search had low diversity. The poor performance of random search across more complex test functions underscores the challenges of ribogate design.

Multi-objective EAs are suitable for ribogate design

All 4 TruthSeqEr runs found viable individuals for each test function. In all cases, its performance was consistent across test functions, regardless of their complexity or arity. Run 1 (ON, OFF) found few viable solutions and the solutions it did find had low diversity. Run 2 (ON, OFF, viability) found many viable solutions of low diversity. Run 3 (ON, OFF, novelty) found few viable solutions, but those that it did find were diverse. Finally, run 4 (ON, OFF, novelty, viability) struck the best balance between viability and diversity. With the exception of two functions (f-22-3I and f-135-3I) every member of the final population was viable and even in those two cases the majority of the population was viable. Most impressively, this viability did not come at the cost of diversity.

To better understand the impact of viability nullification, we consider run 1 in more detail. Since two objectives were enabled (ON and OFF), each generation of this run had not just one fittest solution, but rather a Pareto front of non-dominated solutions. Unfortunately, this front was large compared to the population size, and it contained many individuals with marginally higher ON scores, but very low OFF scores. For example, two individuals with respective (ON, OFF) scores of (0.95, 0.98) and (0.96, 0.05) do not dominate each other, but the first is significantly more viable than the second. The large size of this front hindered the convergence of TruthSeqEr. Viability nullification mitigated this by severely penalizing individuals below the current viability threshold, even if they were members of the non-dominated front. This prevented TruthSeqEr from spending significant resources

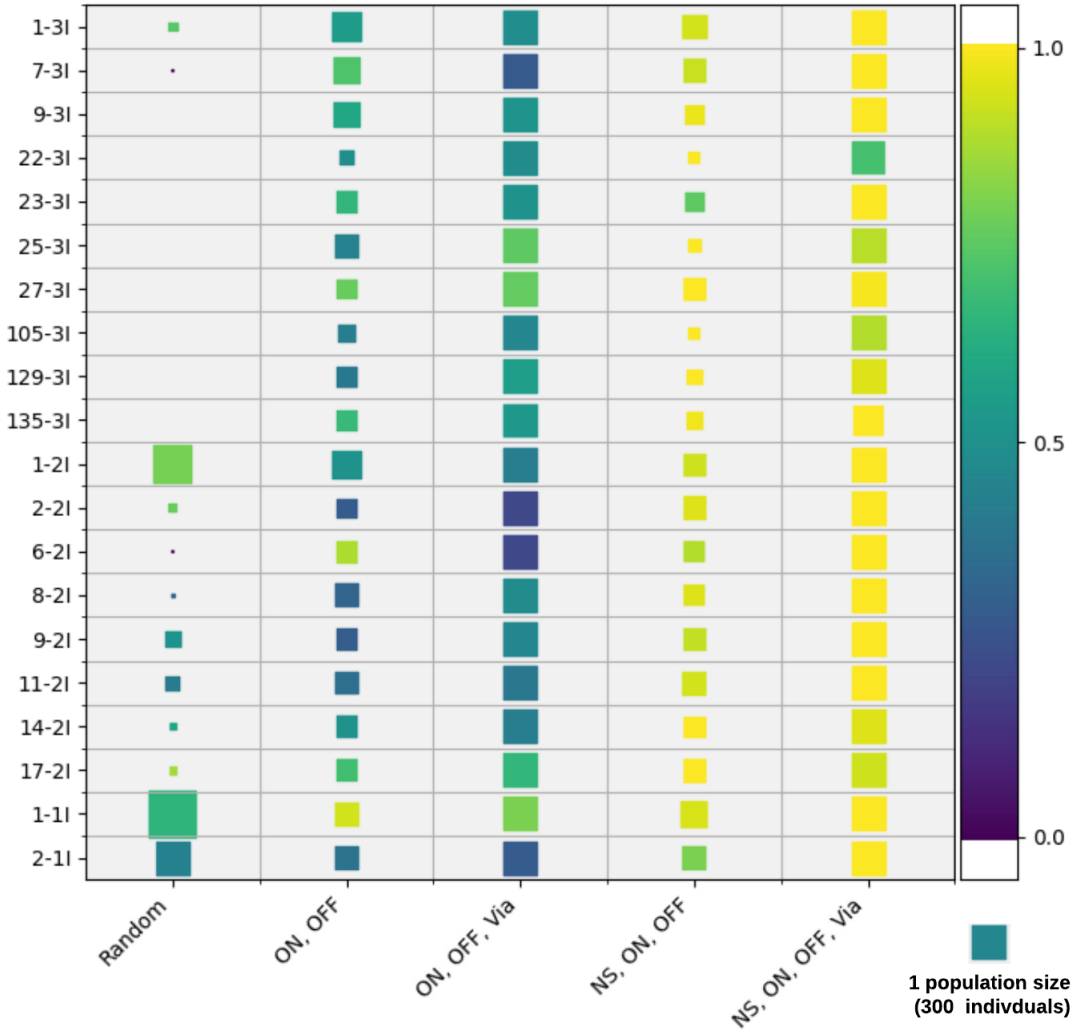


Figure 10: Summary results for random search and four Truth-Seq-Runs on the 1, 2, and 3-input representative functions. The size of a square represents the number of viable individuals and its color represents their diversity. Random search fails to find viable solutions to most of the 3-input representative functions. All four TruthSeqEr runs find viable solutions for each 1, 2, and 3-input representative function. The ON and OFF scores used in each run guide the search to solutions with active ribozymes in the ON states and inactive ribozymes in the OFF states. The number of viable solutions is increased by enabling viability nullification (via). The diversity of the solutions is increased by enabling novelty search (NS).

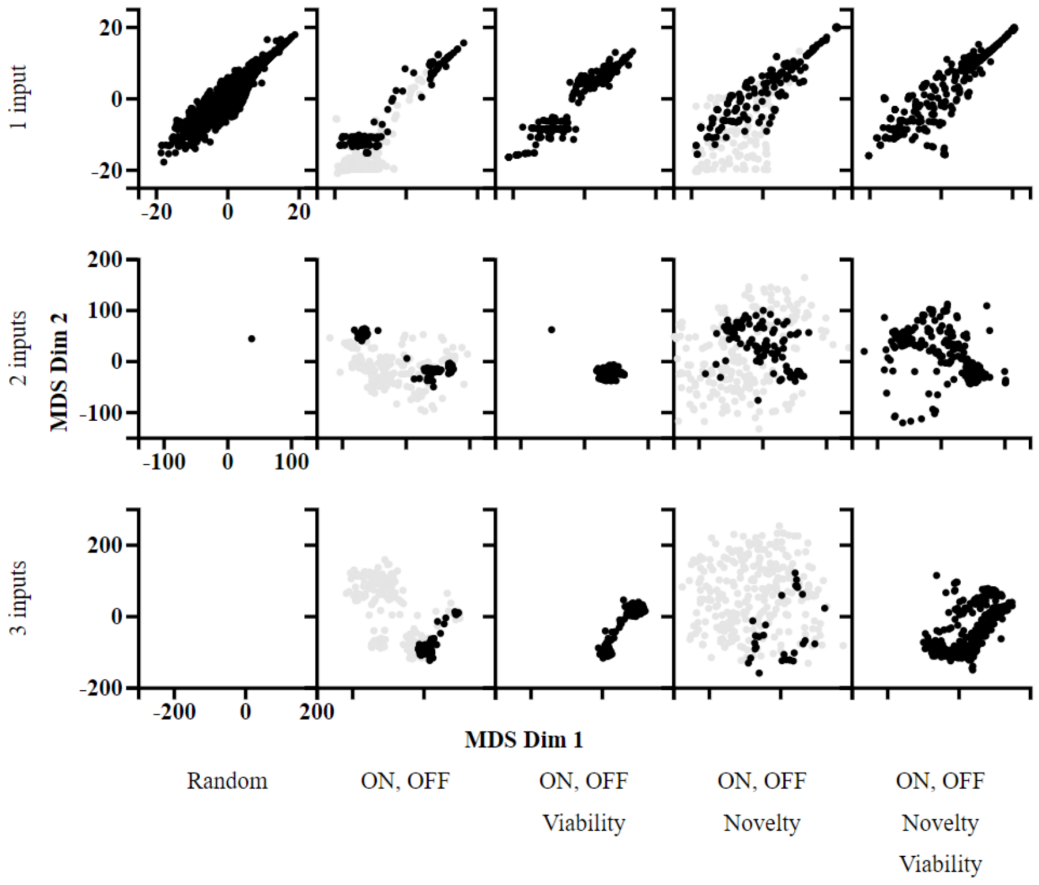


Figure 11: A matrix of scatter plots of the phenotype space of populations of ribozyme designs implementing 1, 2, and 3-input even parity checkers. Multidimensional scaling (MDS) was used to reduce the high dimensional phenotype space to a two dimensional space (Dim 1 and Dim 2). Each row represents a given number of inputs and each column represents a TruthSeqEr run. Viable individuals are shown in black and non-viable are shown in gray. Note that due to their much greater number, the non-viable individuals of the random search run are not shown. The number of viable solutions discovered by random search plummets as the size of the function increases. (ON, OFF) finds viable solutions for all three cases, but a significant portion of the population is non-viable. Enabling viability nullification results in the entire population being viable. Enabling novelty search without viability nullification increases the diversity of the population, but a significant portion of it is non-viable. Enabling both viability nullification and novelty search achieves the best of both worlds: a completely viable, diverse population.

exploring far-from-viable candidate solutions, as evidenced by the increased number of viable individuals generated in runs 2 and 4. Although performance-based multi-objective EAs generally promote more diversity than single-objective EAs, the results of runs 1 and 2 indicate that, in our case, this diversity was limited. Adding an explicit novelty objective significantly increased diversity, as demonstrated in runs 3 and 4.

3.4.2 Reliability assessment

The final population of run 4 (ON, OFF, novelty, viability) was evaluated against more stringent criteria (affinity and thermobalance) that aim to measure biological plausibility. Ribogates that fell outside the specified range for either criterion were eliminated. The results of this filtering are shown in Table 4. Multiple designs meeting the reliability criteria were found for each function, but their numbers were substantially reduced. There was a lot of variance in the data. At one extreme, only 9.7% the population met the criteria (f-7-2I) while at the other extreme, 74% of the population did (f-27-3I). This outcome is not too concerning since the designs were not explicitly optimized against these criteria in the EA loop. Indeed, this showcases the *robustness* that diversity grants a population. If all individuals were confined to a specific region of the phenotype space, then it is possible that that one region would be unreliable. By encouraging novelty, TruthSeqEr distributes the risk among various solutions, preventing over reliance on a single design strategy. This especially important since there is no well-established universal set of ribogate reliability criteria. As a matter of fact, TruthSeqEr could help biologists *devise* those criteria by providing them with a large, diverse, and unbiased dataset of ribogates that they could use for experimentation and characterization.

Figure 12 plots the reliable and non-reliable designs in phenotype space. Despite being fewer in number, the reliable individuals are still quite dispersed, suggesting that it is possible to meet these reliability criteria without severely compromising structural diversity.

3.4.3 Ribogate secondary structures

We now showcase some ribogates designed by TruthSeqEr. In this section, we view the secondary structures of ribogates that met the reliability criteria. In Figure 13 a) and b) we show an XNOR gate and in b) an XOR gate, both 2-input functions. These two gates are the inverse of each other: their truth vectors are 1001 and 0110, respectively. This opposition is also apparent in their secondary structures. In the XNOR gate, the two OBSs are initially bound to each other and do not disrupt the ribozyme. If either input is added, the corresponding OBS will be occupied and the other OBS will be free to disrupt the ribozyme. However, if both inputs are added, no OBS can disrupt the ribozyme. Conversely, the XOR gate contains a negator segment in its extension region. Initially, both OBSs are bound to each other and the negator is free to bind to the ribozyme. If either input is added, the unoccupied OBS will be free, and instead of binding to the ribozyme as in the XNOR case, it will bind to the negator instead, rendering the ribozyme active. However, if both inputs are present, there is no OBS to bind to the negator, allowing it to disrupt the ribozyme freely.

Function number	Affinity survivors (%)	Thermodynamics survivors (%)	Combined survivors (%)
f-1-1I	79.7	40.7	33.0
f-2-1I	86.7	18.0	17.3
f-1-2I	75.7	59.3	45.3
f-2-2I	49.0	48.7	24.7
f-6-2I	40.3	64.0	26.0
f-7-2I	28.3	32.0	9.7
f-8-2I	73.0	18.0	16.3
f-9-2I	58.3	27.0	17.7
f-11-2I	54.7	50.3	32.0
f-14-2I	77.0	28.3	24.7
f-1-3I	56.7	74.3	40.7
f-7-3I	20.0	61.0	14.7
f-9-3I	57.3	35.0	26.3
f-22-3I	76.3	89.0	70.3
f-23-3I	16.0	79.3	12.0
f-25-3I	45.7	88.3	38.3
f-27-3I	81.7	87.3	74.0
f-105-3I	51.3	60.0	33.0
f-129-3I	62.0	77.7	47.7
f-135-3I	33.3	78.7	24.3

Table 4: Percentage of run 4 (ON, OFF, via, NS) ribogates that met the affinity and thermobalance reliability criteria,

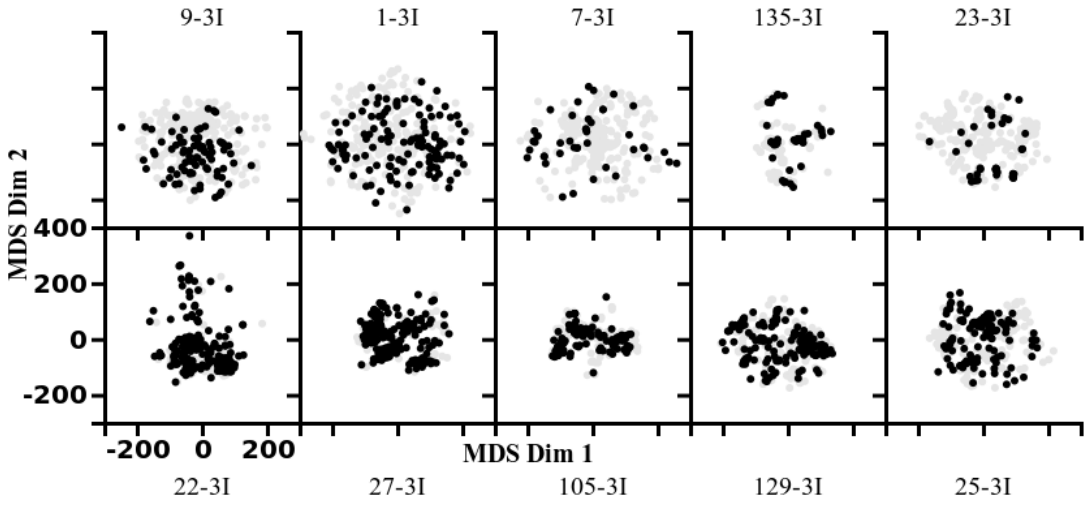


Figure 12: Scatter plot matrix of the phenotype space of the 10 3-input test functions. Dots represent viable individuals obtained at the end of run 4 (ON, OFF, novelty, viability). Gray dots fail to meet the reliability criteria while black dots succeed.

In Figure 14, we examine a ribogate implementing the 3-input function f-129-3I. This function has a truth vector of 10000001 and it can be seen as a generalization of the XNOR gate: it is active when either all or none of the inputs are present. It also has a mechanism of action similar to that of XNOR. In the initial state, all OBSs are bound to each other and the ribozyme is active. When all OBSs are occupied, none are available to disrupt it so it remains is active. If any single OBS is unoccupied, it will bind to the ribozyme. However, unlike the XNOR gate, if 2 OBSs are free, they will bind to the gate and not to each other.

3.5 Conclusions

Logic gates can be realized in biological matter instead of silicon. One promising approach is to use catalytic RNA that changes shape in response to different inputs signals. These ribogates are an attractive solution due to their low energy footprint, fast dynamics, versatility, ability to interface with cellular environments, and their suitability for computational design. Despite these advantages, designing ribogates is challenging for a multitude of reasons. Current approaches require extensive domain knowledge which limits their accessibility and generality. In addition, the ribogates that can be designed are of limited complexity and diversity. Furthermore, devices that are predicted to function *in silico* may fail in practice,

In response to these shortcomings, we presented a multi-objective evolutionary algorithm called TruthSeqEr. TruthSeqEr combines handcrafted performance-based objective functions, novelty search, and a novel technique called viability nullification, allowing it to generate diverse populations of ribogates implementing e*very* member of a representative set of 1, 2, and 3-input functions, including both linearly separable and inseparable

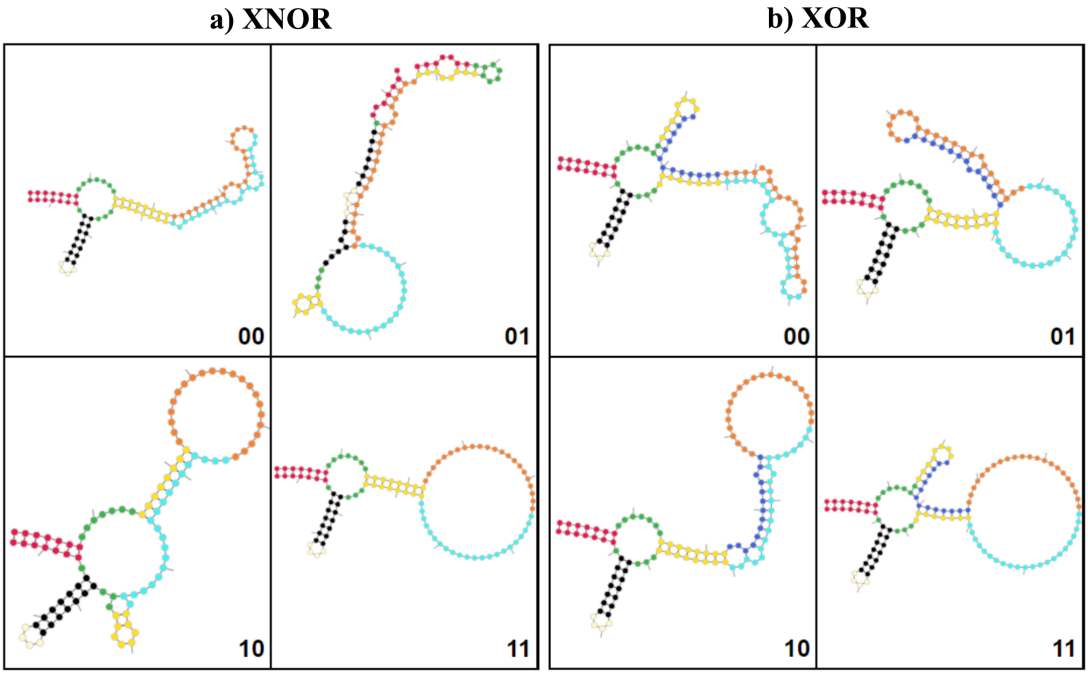


Figure 13: Secondary structures for each state of two 2-input ribogates. Input strands are not shown. a) XNOR. In states 00 and 11, the three stems and the catalytic core of the ribozyme form and the gate outputs a 1. In states 01 and 10, the unoccupied OBS disrupts the ribozyme active structure and the gate outputs a 0. b) XOR. The extension region contains a negator segment which disrupts the ribozyme in states 00 and 11. In states 01 and 10, the negator binds to the unoccupied OBSs instead. Refer to Figure 5 a) for segment color coding and main text for a more detailed description of these gates.

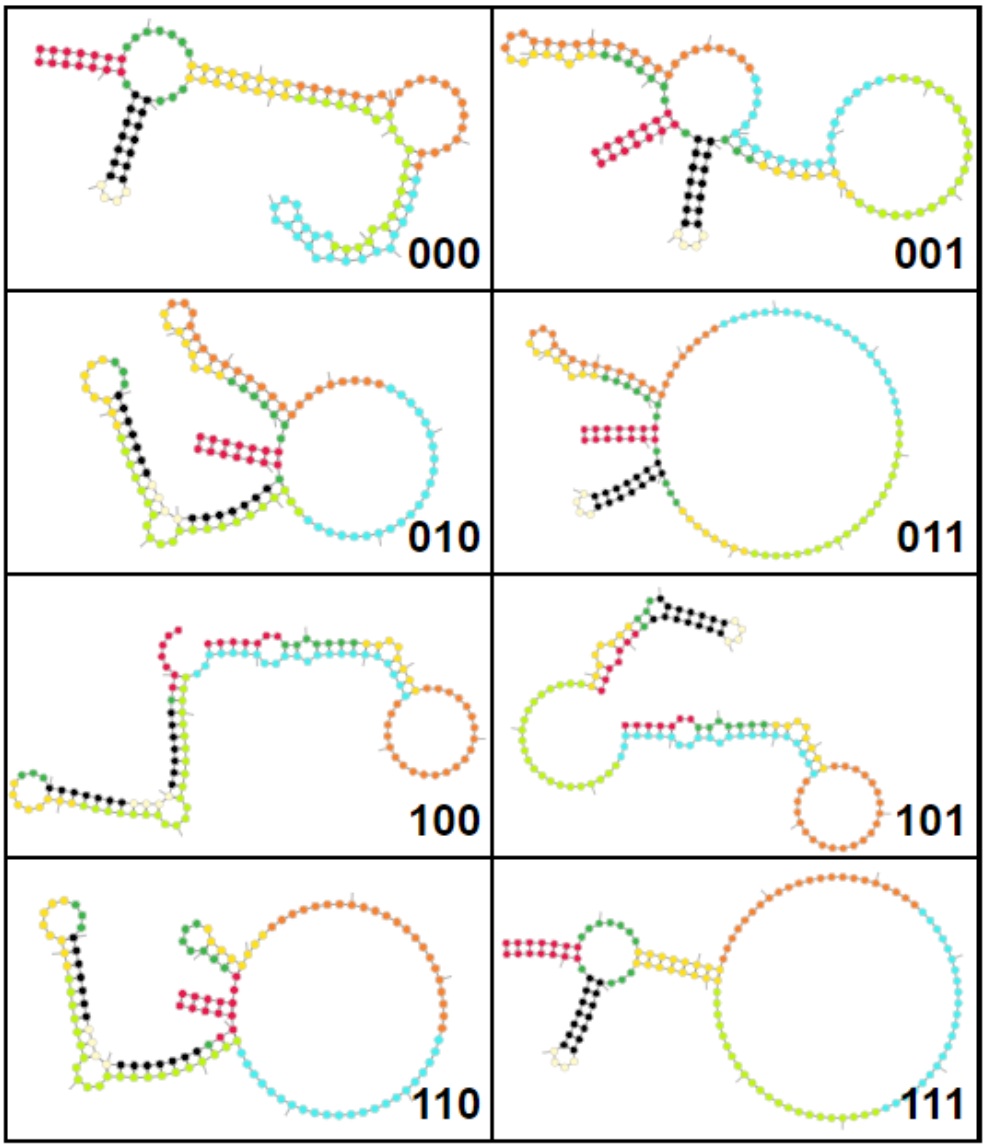


Figure 14: Secondary structures for each state of a 3-input ribogate implementing function f-129-3I. Input strands are not shown. The three stems and catalytic core of the ribozyme only form when all inputs are absent (state 000) or present (state 111). Refer to Figure 5 a) for segment color coding and main text for a more detailed description of this gate.

functions. Using TruthSeqEr, devices such as multiplexers and parity check circuits can be implemented by a *single* ribogate. Unlike existing approaches, *no* domain knowledge is required of the user: all they have to do is specify the target function.

One limitation of this work is that we cannot guarantee that ribogates designed by TruthSeqEr will function as expected in *in vitro* or *in vivo* biological conditions. This is due in part to limitations in RNA folding algorithms, which have difficulty predicting complex structural motifs such as pseudoknots and hybridized structures involving multiple strands. These algorithms also struggle to efficiently predict the dynamic behavior of RNA as well as the interactions between RNA and existing cellular machinery. Furthermore, the ribogates designed by TruthSeqEr exhibit many complex interactions between their segments, which makes them more likely to misfold. That being said, we have implemented measures to reduce the risk of ribogate failure. Reliability filtering ensures more stringent checks, while the diversity of generated designs helps minimize the impact of unforeseen issues. These diverse ribogates also serve as a rich dataset that can be explored by synthetic biologists and used to develop more accurate measures of ribogate fitness. In addition, the extracted mechanism graphs can serve as a roadmap for future experimentation in which devices with fewer interactions are tested before devices with more interactions.

To conclude, this work highlights the power of ribogates as a computational medium and the demonstrates the effectiveness of evolutionary algorithms such as TruthSeqEr in their design. We hope that our contributions encourage further research in ribogate design, modeling, and applications.

Chapter 4

Ribogate abstraction

4.1 Introduction

In the previous chapter, we presented an EA called TruthSeqEr, and used it to design ribozyme-based logic gates (ribogates) implementing a representative set of Boolean functions. A fascinating result was that a single ribogate could implement functions with a wide range of complexity. Consider the following two functions: a 3-input AND gate and a 3-input parity check circuit. These are donated by function numbers f-1-3I and f105-3-I, respectively. f-1-3I is linearly separable and can be realized by a single artificial neuron whereas f105 is not linearly separable and a network of multiple neurons is required to realize it. f-1-3I's Boolean expression is simple (abc) whereas f-105-3I's is much more sophisticated (abc + ab'c' + a'bc' + a'b'c), meaning that a much larger digital logic circuit is required to implement the latter than the former. And yet we showed that each function can be implemented by a single ribogate. This chapter is motivated by the following question: what distinguishes ribogates implementing simple functions from those implementing much more complex functions? To answer this question, we will gradually abstract away the details of a ribogate's set of secondary structures until we arrive at predictive graphical representation of its logical behavior. By analyzing these graphs, the source of a ribogate's computational power will be revealed.

4.2 Segment structures

The secondary structures presented in the previous chapter showcase the ribogate's characteristics in detail. However, their size and the number of ribogate states can make them somewhat cumbersome to interpret. While color-coding nucleotides based on L-segments improves visualization, it is not a perfect solution. Consequently, we introduce a coarse-grained alternative to secondary structures, referred to as *segment structures*. This approach simplifies the representation of ribogates, allowing for more accessible analysis and understanding of their properties.

The segment structure nodes correspond to H-segments, which include the extension region (ER) segments (OBSs and optional negator) and the single ribozyme (Rz) segment.

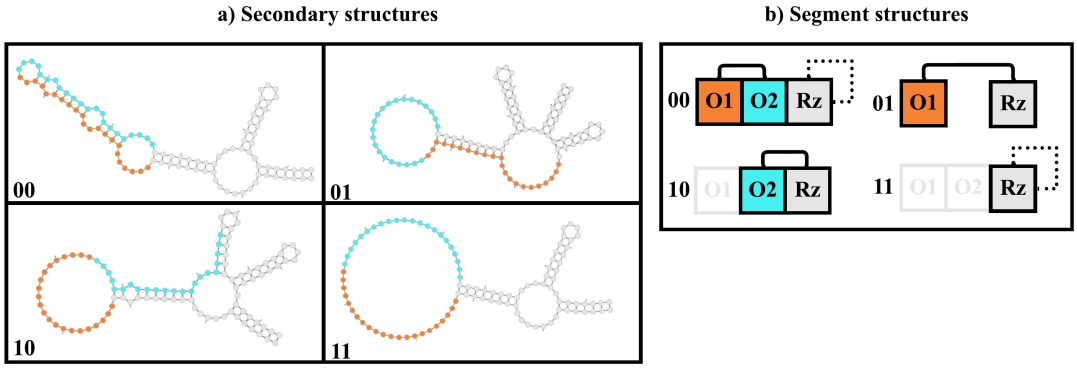


Figure 15: An XNOR gate viewed at two levels of abstraction. a) Secondary structures. Nodes represent nucleotides and edges represent base-pairs. Nucleotides that are bound to one of the input strands (not shown) are prevented from forming base-pairs with the rest of the ribogate. In states 00 and 11, the three stems and the catalytic core of the ribozyme form and the ribogate outputs a 1. In states 01 and 10 the ribozyme is disrupted and outputs a 0. b) Segment structures. Nodes represent H-segments: OBS1 (O1), OBS2 (O2) and the ribozyme (Rz). Structural edges (solid) represent segment pairs. In state 00, the two OBSs share base-pairs in the secondary structure; they are therefore paired together in the segment structure. In states 01 and 10 one OBS is paired with the ribozyme. When no structural edge is incident on the Rz node, the ribozyme is active (states 00 and 11). This is denoted by a dashed logical edge connecting the ribozyme to itself. Although the ribozyme contains base-pairs with itself in all four states, as a convention we do not draw a self-incident Rz structural edge. OBSs that are bound to their corresponding input strand (not shown) are grayed out and are unable to bind to other segments. Compared to secondary structures, segment structures have significantly fewer nodes and edges.

There are two types of segment structure edges: *structural* and *logical*. Structural edges may form between two ER segments or between an ER segment and the ribozyme, but *not* between the ribozyme and itself. A structural edge between two segments indicates that they share at least one base pair. The fact that a segment comprises multiple nucleotides introduces properties in segment structures that are forbidden in secondary structures (e.g. self-pairing, multiple partners). However, as a convention, we do not allow a structural edge to form between the ribozyme and itself. Instead, a logical edge is added between the ribozyme and itself, if and only if, it is in its active conformation. In each state, an input binding to the OBS is represented by labeling that OBS node as being *unavailable*. No edges may be incident on an unavailable node. Figure 15 shows the correspondence between secondary and segment structures for an XNOR gate. Equipped with a more convenient way of visualizing ribogates, we will now delve deeper in our analysis.

4.3 Canonical devices

We have previously seen that TruthSeqEr designs viable and diverse ribogates for an entire set of representative functions. In this section, we seek to uncover a *canonical* ribogate for each 3-input function: this ribogate is the *simplest* one capable of realizing a given function. Identifying these ribogates can help us discover design principles while avoiding excessive focus on details.

We define a ribogate's *complexity* as the total number of edges over *all* its segment structures. To determine the simplest ribogate, we start from the final population of run 4 (ON, OFF, novelty, viability) *without* subjecting it to reliability filtering. We set this population as the initial population of a new Truth-Seq-Run that utilizes an alternative set of objectives that reward simpler structures. The details of this run are explained in the Appendix. Figure 16 shows the segment structure of the canonical devices of fives functions which we will examine in detail:

1. f-1-3I. No OBS-OBS interactions are observed. However, whenever an OBS is available it will bind to the ribozyme. Therefore, the gate is only active when all inputs are present. This is the expected behavior of an AND gate.

2. f-23-3I. Similar to f-1-3I, but if exactly one OBS is unoccupied, it will remain unpaired instead of disrupting the ribozyme

3. f-9-3I. Also similar to f-1-3I, but in state 100, OBS2 and OBS3 will bind to each other instead of the ribozyme.

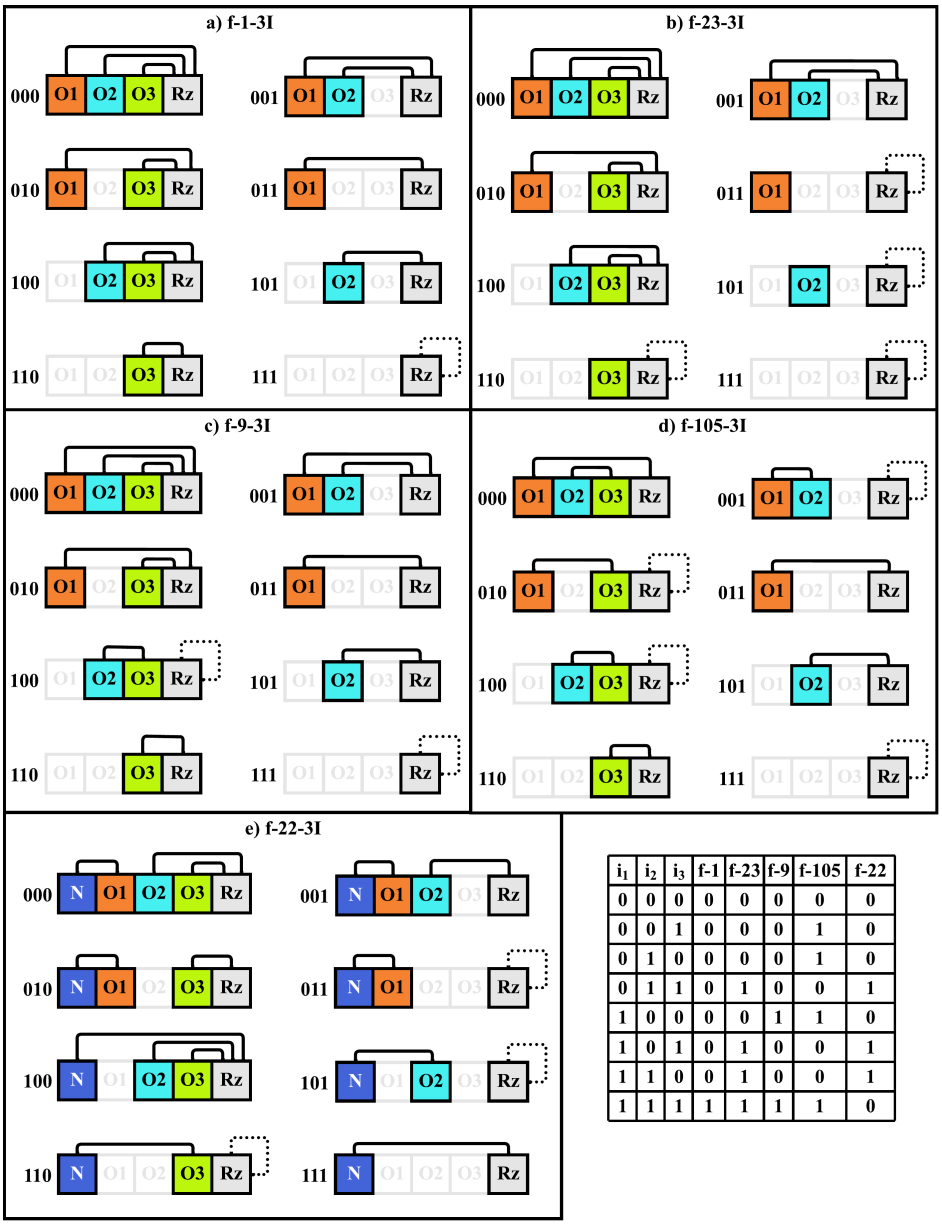
4. f-105-3I. OBS-OBS interactions are prominent: whenever two OBSs are free, they will bind to each other. If a single OBS is free, it will bind to the ribozyme. Therefore, the ribozyme is active only if an odd number of input strands are present. This is the expected behavior of a parity checker.

5. f-22-3I is the one 3-input test function that requires a negator. Its canonical device exhibits no OBS-OBS interactions, but plenty of plenty of Negator-OBS interactions. The negator plays two *opposing* roles. If a single OBS is free, the negator will bind to it and prevent it from disrupting the ribozyme. However, if no OBSs are free, the negator itself binds to the ribozyme and deactivates it.

These examples show the usefulness of canonical devices and segment structures for understanding ribogate functionality. The segment structures of the remaining 3-input NPN canonical devices (f-7-3I, f-25-3I, f-27-3I, f-129-3I, f-135-3I) are shown in the Appendix.

4.4 Mechanism graphs

Segment structures provide a more compact way of displaying structural information. However, they are still limited: they describe *what* structures a ribogate adopts, but not *why* it adopted those structures. In this section we hypothesize that these structures are generated



i_1	i_2	i_3	f-1	f-23	f-9	f-105	f-22
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	0	1	0
0	1	1	0	1	0	0	1
1	0	0	0	0	1	1	0
1	0	1	0	1	0	0	1
1	1	0	0	1	0	0	1
1	1	1	1	1	1	1	0

Figure 16: Segment structures of 5 canonical ribogates. There is one structure for each of the 8 states (000 to 111). Any segment binding (solid edge) to the ribozyme (Rz) disrupts it. Occupied OBSs (O1, O2, O3) are grayed out and are prevented from pairing with other segments. f-1-3I, f-23-3I and f-9-3I exhibit similar binding patterns involving many OBS-Rz interactions. Slight variations change the states in which the ribozyme is active (denoted by the dashed Rz self-loop). f-105-3I is characterized by extensive OBS-OBS interactions that cause the ribozyme to only be active for an odd number of inputs. f-22-3I is the only one of these five functions that contains a negator segment (N): this allows it to be inactive when all inputs are present. Unlike secondary structures, segment structure nodes may have multiple partners. This is observed for the Rz segment in all functions except f-105-3I. Refer to the main text for a detailed description of these devices.

through a process called *additive segment competition*. The key features of this model are listed below:

1. Segments *compete* for the opportunity to bind to each other. Unless otherwise specified (point 7), a segment may only have *one* partner per state.

2. Different pairs of segments have different *binding strengths*.

3. In a given state, there are multiple valid *candidate* structures that a ribogate may adopt. The structure that it actually adopts will be the one with the highest *stability*.

4. The stability of a segment structure is obtained by *adding* the binding strengths of all segments pairs present in the structure.

5. An input binding to an OBS removes all segment pairs involving that OBS from contention. This in general causes a change in the optimal (highest stability) structure.

6. A ribogate is represented as a weighted *mechanism graph*. The nodes of this graph correspond to H-segments, its edges denote segments pairs that are permitted to form, and the weights encode the strength of these pairs.

7. Segment competition can be relaxed by grouping certain edges into sets called *bundles*. Edges that are part of the same bundle may *concurrently* bind to the same segment in a given state. By default, *any* number of edges in a bundle may be simultaneously present in the same segment structure. However, some bundles are considered *partial*, meaning that only a *maximum* number of edges in a bundle may be simultaneously present in the same segment structure.

The mechanism graph for a 2-input OR gate is shown in Figure 17 a). It consists of three edges: one structural edge between each OBS and the ribozyme, and one logical self-incident ribozyme edge. The two OBS-Rz edges are grouped into a bundle. Figure 17 b) illustrates how this mechanism graph generates segment structures for each input state. We now consider these states one at a time.

1. State 11 (2 inputs present, 0 unoccupied OBSs). There is only one edge to consider: the Rz-Rz logical edge. The stability is maximized for the structure that includes this edge, and the ribozyme is active.

2. States 01 and 10 (1 input present, 1 unoccupied OBS). The Rz-Rz logical edge (weight: 3) competes with the one available OBS-Rz edge (weight 2). The Rz-Rz edge has the higher weight and wins: the ribozyme is active.

3. State 00 (0 inputs present, 2 unoccupied OBSs). Neither OBS-Rz edge is strong enough to displace the Rz-Rz edge on its own. However, since they are part of a bundle they can both be present. Since the sum of their weights (2 + 2 = 4) is greater than the Rz-Rz weight, the two OBS-Rz edges win, the Rz-Rz edge does not form, and the ribozyme is inactive.

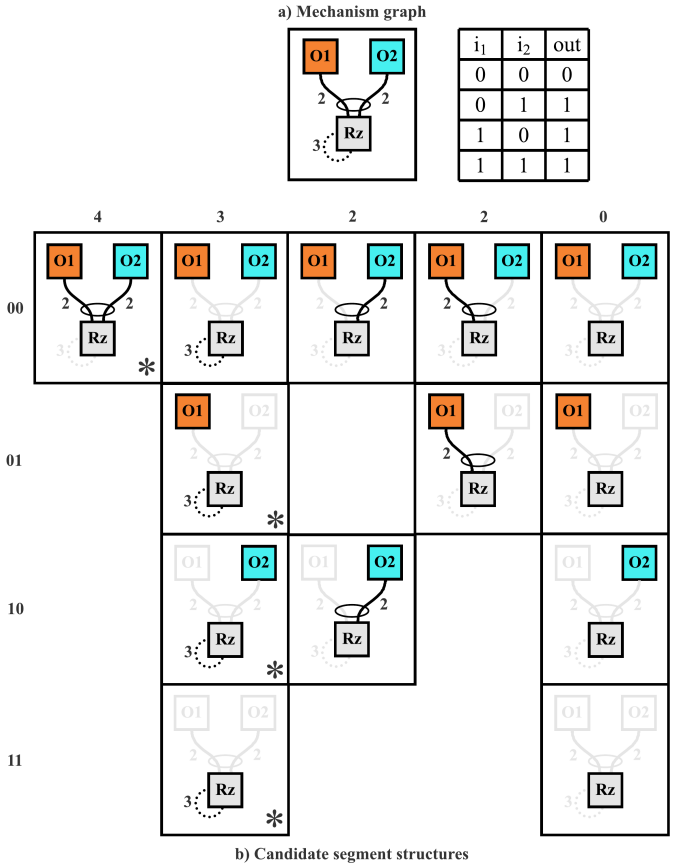


Figure 17: a) Mechanism graph of a 2-input ribogate implementing OR. The O1, O2, and Rz squares denote segments OBS1, OBS2, and the ribozyme, respectively. Solid lines denote structural edges, the dashed line denotes the logical edge, the edge weights denote segment binding strengths, and the ellipse denotes a bundle. b) Determining the structure with the highest stability. In each state, there are multiple candidate segment structures that are subgraphs of the mechanism graph. The rows represent the four different input combinations (00, 01, 10, and 11). Black lines depict mechanism graph edges that *are* present in a given candidate structure. The structure's stability is equal to the sum of these edges. Faded out lines depict mechanism graph edges that are *not* present in the structure. They have no effect on stability. In each row, the structures are sorted by descending stability. Candidate structures in the same column have the same stability (shown on top). Only certain candidate structures are valid. For example, edges OBS1-Rz and Rz-Rz cannot both be present in the same structure because they are both incident on Rz, but not part of the same bundle. However, OBS1-Rz and OBS2-Rz *are* part of the same bundle, allowing OBS1 and OBS2 to simultaneously bind to Rz. As more inputs are added, the mechanism graph has fewer candidate structures. This is because occupied OBSs (shown as faded out) are not permitted to bind to other segments. The most stable structure in each state is denoted by an asterisk. The logical Rz-Rz edge indicates ribozyme activity and it appears in the segment structure for all states where at least one input is present. This is the expected behavior of an OR gate. Refer to the main text for a more detailed description of this device.

4.5 Mechanism graph analysis

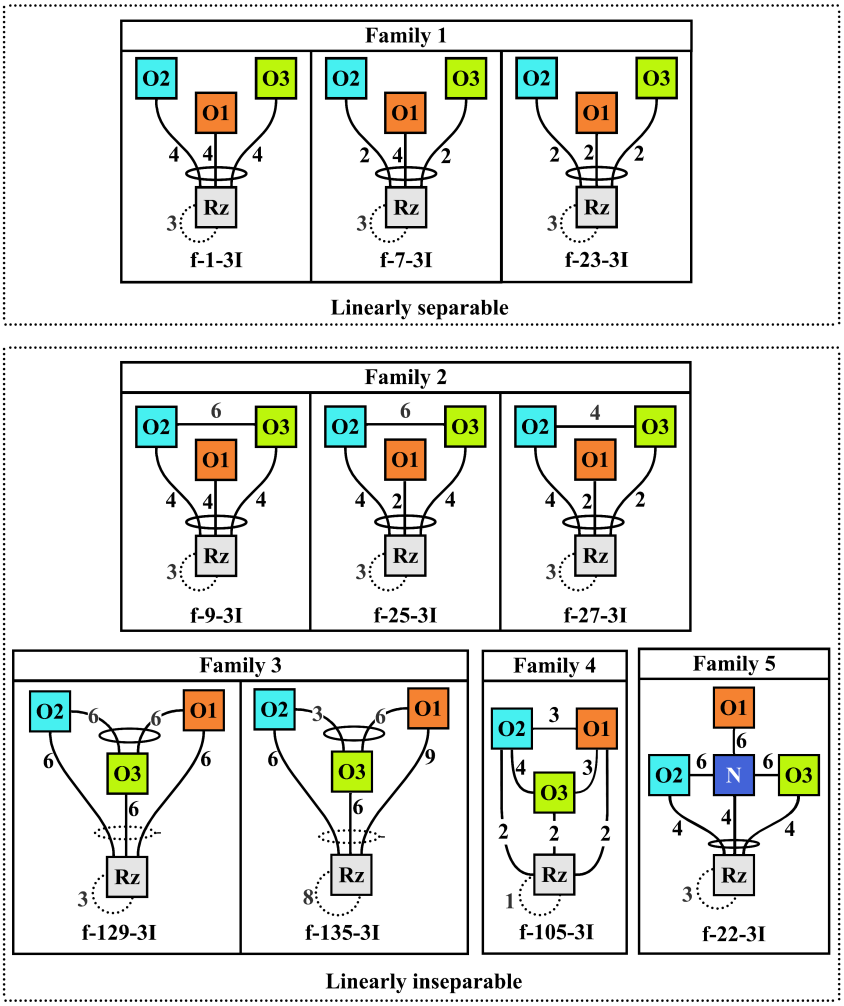
The mechanism graphs for the 10 3-input NPN functions are shown in Figure 18. They were obtained through a mostly-automated method described in the Appendix. We group these graphs into 5 families based on structure. Members of the same family have the same segment interactions; the only difference is the *strength* of these interactions. Members of different families have different segment interactions. Indeed, when comparing different families, we notice a clear progression in terms of structural *complexity*. In families 1-4, all three OBSs interact with the ribozyme. What distinguishes one family from the other is the number of *OBS-OBS interactions*. Family 1 has 0 OBS-OBS interactions, family 2 has 1, family 3 has 2, and family 4 has 3. Family 5 is a special case since it consists of the only function that requires a negator segment. It has 0 OBS-OBS interactions, but 3 OBS-Negator interactions. We now describe each family in detail:

1. Family 1. There are no OBS-OBS interactions. All three OBS-Rz edges are part of a bundle, allowing them to concurrently bind to the ribozyme and deactivate it. However, for this to happen in a given state, the cumulative weight of the available OBS-Rz edges must exceed the weight of the dashed ribozyme self-edge. In the f-1-3I graph, each OBS-Rz edge is strong enough to deactivate the ribozyme on its own, meaning that the ribogate outputs a 1 only when all OBSs are occupied by inputs. In the f-7-3I and f-23-3I graphs, certain OBS-Rz edges are *weakened*, allowing the ribogate to remain active in additional states.

2. Family 2. There is now a potential OBS2-OBS3 interaction. Since the OBS2-OBS3 and the Rz-Rz edges do not have any segments in common, they can both be present in the same structure. This makes it harder to deactivate the ribozyme, since the cumulative weight of the OBS-Rz edges must now exceed not just the Rz-Rz weight, but rather the *sum* of the Rz-Rz and OBS2-OBS3 edge weights (if OBS2 and OBS3 are both unoccupied). Weakening certain edges within this family increases the number of states in which the ribogate is active.

3. Family 3. OBS1 and OBS2 can non-competitively bind to OBS3 because the OBS1-OBS3 and OBS2-OBS3 edges are part of the same bundle. Since the three OBS-Rz edges are part of a partial bundle (indicated by the dashed oval), up to two of the three OBSs can concurrently bind to the ribozyme, rendering it inactive. For this to happen, the cumulative weight of these OBS-Rz edges must exceed that of the Rz-Rz edge plus the OBS1-Rz and OBS2-Rz edges (if they are available). Weakening certain edges within this family increases the number of states in which the ribogate is active.

4. Family 4. The one member of this family contains no bundles, meaning that each segment can have at most one partner. All OBSs have the potential to bind to each other, as well as to the ribozyme. The weight of each OBS-Rz edge exceeds that of the Rz-Rz edge, meaning that any single OBS can deactivate the ribozyme. However, the weight of each OBS-OBS edge exceeds that of every OBS-Rz edge. Consequently,



i_1	i_2	i_3	f-1	f-7	f-23
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	1
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	0	1	1
1	1	1	1	1	1

i_1	i_2	i_3	f-9	f-25	f-27
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	1	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	0	0	1
1	1	1	1	1	1

i_1	i_2	i_3	f-129	f-135	f-105	f-22
0	0	0	1	1	0	0
0	0	1	0	0	1	0
0	1	0	0	0	1	0
0	1	1	0	0	0	1
1	0	0	0	0	1	0
1	0	1	0	1	0	1
1	1	0	0	1	0	1
1	1	1	1	1	1	0

Figure 18: Mechanisms graphs for each of the 10 3-input NPN functions. They are grouped into 5 families: members have the same edges but different weights (binding strengths). Family 1 implements the simplest functions (linearly separable) and also has the simplest structure. Families 2-5 are more sophisticated, allowing them to implement linearly inseparable functions. A set of segment structures can be obtained for each mechanism graph using the procedure illustrated in Figure 17. Detailed explanations for each family are provided in the text. Solid ovals represent bundles with edges that can all be concurrently present in a segment structure. The dashed ovals in family 3 represent partial bundles, where only a maximum of 2 OBS-Rz edges can be simultaneously present.

two unoccupied OBSs will prefer to bind to each instead of the ribozyme. Therefore, the ribozyme is active if an even number (0 or 2) OBSs are unoccupied. This is the expected behavior of a parity check circuit.

5. Family 5. The one member of this family contains a negator segment that can interact with any of the OBSs as well as the ribozyme itself. In addition, OBS2 or OBS3 can interact with the ribozyme. Since the OBS2-Rz, OBS3-Rz, and N-Rz edge weights each exceed the Rz-Rz weight, they can deactivate the ribozyme on their own. However, whether they do so also depends on any available OBS-N edges.

We have seen that we can change the function implemented by a ribogate by changing its weights. This property is shared by another bio-inspired component: the *artificial neuron*. Loosely inspired by biological neurons, an artificial neuron computes a weighted sum of its inputs and passes this value through a non-linear activation function. Figure 19 a) shows a mechanism graph and an artificial neuron implementing a 2-input AND gate. In Figure 19 b), the mechanism graph and neuron weights are changed so that they both implement an OR gate instead. We have also seen that the addition of OBS-OBS and OBS-Negator interactions allow the ribozyme to implement new functions, specifically the linearly inseparable functions in families 2-5. However, adding more interactions *within* a single neuron is not possible: to compute a linearly inseparable function, a neural *network* of multiple neurons is required. This principle is illustrated in Figure 19 c). We now perform a detailed analysis and discussion of how a single ribogate is able to implement linearly inseparable functions.

4.6 Linear inseparability and OBS-OBS interactions

In this analysis, we consider *base* structures that are in competition with each other. In each state, the candidate segment structures are substructures of these base structures. The specific structure of a candidate (and by extension its stability) depends on which edges are available, which in turn depends on which OBSs are unoccupied by inputs. This allows us to express the stability of each candidate in terms of the input state.

In the case of family 1, two base structures compete with each other: 1) the three OBS-Rz edges and 2) the Rz-Rz edge. These are illustrated in Figure 20 b). The stabilities of these two base structures are expressed by Equations 13 and 14, respectively:

$$i'_1 w_{O_1 R} + i'_2 w_{O_2 R} + i'_3 w_{O_3 R} \tag{10}$$

$$w_{RR} \tag{11}$$

The i'_n term is equal to 0 when the n^{th} input is present and 1 when it is absent. Therefore, the $i'_n w_{O_n R}$ term indicates than an OBS-Rz edge only contributes to the stability when its corresponding input is absent. The ribogate is active if the candidate derived from the second base structure is more stable than the candidate derived from the first one. This occurs if the following inequality holds true:

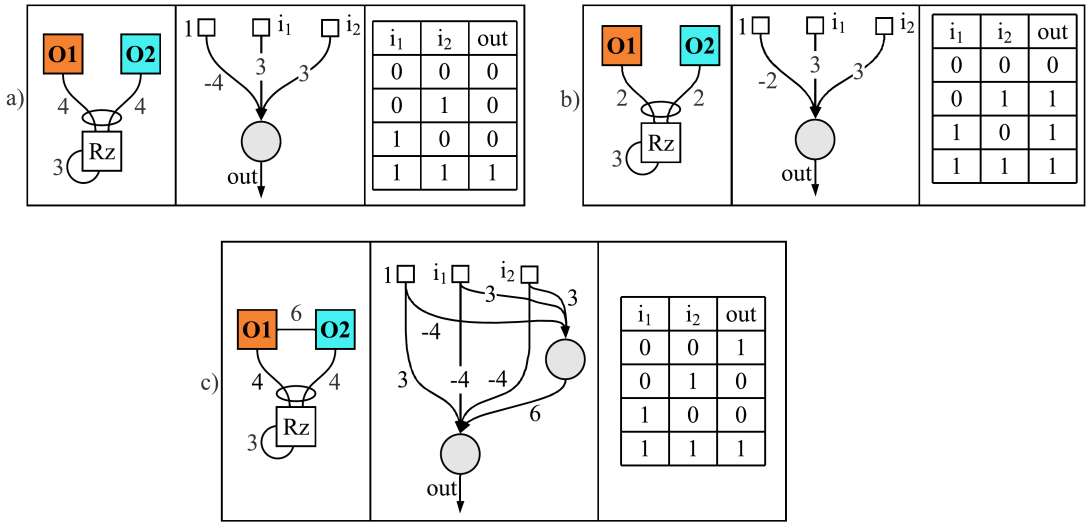


Figure 19: Comparison between ribogates and artificial neurons. a) A ribogate mechanism graph (left) and an artificial neuron (center) both implementing a linearly separable AND function (right). Each binary input to the neuron is multiplied by its corresponding edge weight. The constant 1 term allows for a *bias* to be applied to the neuron, regardless of the input state. The outputs of the multiplication operations are summed together and then passed through an activation function which returns 0 if its input is negative, and 1 otherwise. This is indicated by the edges flowing into the shaded circle. b) By changing the weights of the mechanism graph (left) and the neuron (center), a new linearly separable function, OR, is obtained (right). c) The addition of an OBS1-OBS2 edge allows a single ribogate (left) to implement a linearly inseparable function (XNOR, right). However, a single neuron is incapable of representing this function. Instead, a neural network of two neurons is required (center).

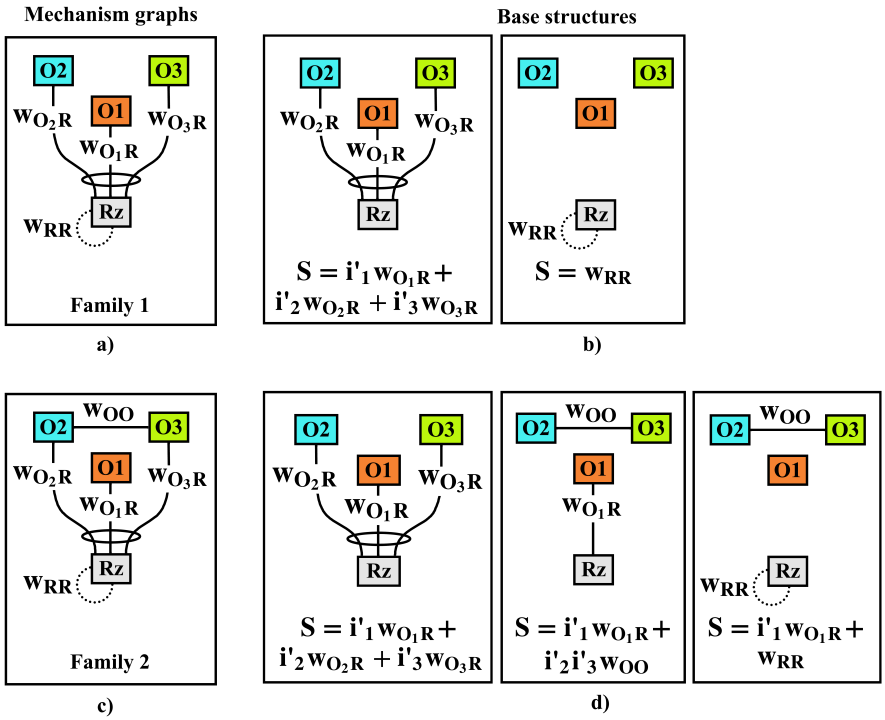


Figure 20: Stability equation for family 1 and family 2 ribogates. a) Mechanism graph of a family 1 ribogate. There are three OBS-Rz edges with weights w_{O_nR}. There is also an Rz-Rz edge with weight w_{RR}. c) Mechanism graph of a family 2 ribogate. It is the same as the family 1 graph, but it contains an addition OBS2-OBS3 segment with weight w_{OO}. b, d) In each state, the ribogate adopts a structure that is a subset of one of these base structures. The stability (S) of that structure is represented by the equation at the bottom of the box. Its value depends on which OBSs are unoccupied by inputs (i_n).

$$i'_1 w_{O_1 R} + i'_2 w_{O_2 R} + i'_3 w_{O_3 R} < w_{RR} \tag{12}$$

In the case of family 2, *three* base structures now compete with each other: 1) the three OBS-Rz edges, 2) the Rz-Rz edge and the OBS2-OBS3 edge, and 3) the OBS1-Rz edge and the OBS2-OBS3 edge. These are illustrated in Figure 20 d). Their stabilities are expressed by Equations 13, 14, and 15, respectively:

$$i'_1 w_{O_1 R} + i'_2 w_{O_2 R} + i'_3 w_{O_3 R} \tag{13}$$

$$w_{RR} + i'_2 i'_3 w_{OO} \tag{14}$$

$$i'_1 w_{O_1 R} + i'_2 i'_3 w_{OO} \tag{15}$$

The non-linear $i'_2 i'_3 w_{OO}$ term indicates that the OBS2-OBS3 edge only contributes to the stability when inputs 2 and 3 are both absent. The ribogate is active if the candidate derived from the second base structure is more stable than both the one derived from the first and third ones. This occurs if the following *two* inequalities holds true:

$$\begin{aligned} i'_1 w_{O_1 R} + i'_2 w_{O_2 R} + i'_3 w_{O_3 R} &< w_{RR} + i'_2 i'_3 w_{OO} \\ i'_1 w_{O_1 R} &< w_{RR} \end{aligned} \tag{16}$$

The above analysis shows that ribogates governed by the additive segment model *intrinsically* implement linear (Inequality 12) and non-linear (Inequalities 16) decision boundaries as they change shape in response to various inputs. Crucially, the factor that distinguishes these two types of decision boundaries is not the *number* of segments, but rather the *complexity* of the interactions between them. This enables a single ribogate to implement functions that would require multiple standard logic gates or artificial neurons.

4.7 Additive segment competition vs secondary structure prediction

In this work, we have used RNA secondary structure prediction to *design* ribogates and additive segment competition (ASC) to *analyze* them. We now take a moment to compare the two processes. ASC is an abstract version of RNA secondary structure folding and the two processes have many similarities. Both treat structure prediction as an optimization problem: folding uses dynamic programming [67] to find the secondary structure with the lowest free energy whereas ASC uses exhaustive search to find the segment structure with the highest stability. Both impose limits on the number of partners that their nodes may have. Finally, both apply constraints to certain nodes (OBS nucleotides in folding and OBS segments in ACS) to prevent them partnering with other nodes, thereby changing the optimal structure.

Despite these many similarities, there are some key differences. Segment structures generated by ACS may have nodes with self-loops and multiple partners. They are also

much more concise: they have a maximum of 5 nodes whereas their corresponding secondary structures have more than 100. Critically, ACS models a structure's stability as a sum of *independent* edge weights. This allows us to easily reason about the effect of adding or removing certain edges. This is not the case in folding: the free energy of an RNA secondary structure is not simply the sum of the independent contributions of its base-pairs. Rather, it is the result of many *non-additive* effects such as base-pair stacking and loop entropy [103].

4.8 Conclusion

In this chapter, we proposed a simple, graph-based model of ribogate operation called additive segment competition (ACS). For each representative 3-input function, we extracted a simple graph illustrating the mechanism of action of a canonical ribogate implementing that function. By examining these mechanism graphs, we were able to arrange ribogate gates into families and discover shared design principles and identify the structural motifs allowing certain ribogates to implement sophisticated linearly inseparable functions. *Despite* its simplicity, ACS appears to be a plausible model of ribogate behavior, being able to reproduce the observed segment structures of each canonical ribogate with virtually no error. *Because* of its simplicity, we have seen that ribogates can be grouped into families, and that OBS-OBS or OBS-Negator interactions are required for linear inseparability. We have also seen that like artificial neurons, ribogates can implement different functions by changing their weights, but that unlike neurons, they can solve entirely new classes of problems by changing their interactions. These insights suggest that ACS is not only a useful model of ribogate behavior, but a potential new form of unconventional computing that requires further investigation.

Chapter 5

TriCleaver

5.1 Introduction

Certain genes contain a region of DNA in which the same triplet of bases is repeated multiple times, one after another; such regions are called *trinucleotide repeats*. A key feature of these trinucleotide repeats is their ability to adopt non-canonical DNA structures such as DNA stem loops [104]. These structures have the potential to interfere with cellular processes such as DNA replication, DNA repair, and transcription in a way that causes the repeats to expand (i.e. increase in number) [63]. This expansion can occur in the germ line during gametogenesis, particularly spermatogenesis, resulting in offspring with a larger number of repeats than their parents [104]. The expansion can also occur in somatic cells, even post-mitotic (i.e. non-dividing) ones such as neurons [104]. Furthermore, as the number of repeats increases, they become less stable, meaning that they are more likely to expand in the germ line or somatic cells [68]. Below a certain threshold, trinucleotide repeats do not impede normal function. However, above that threshold, debilitating and life-threatening *trinucleotide repeat expansion disorders* (*TREDs*) can occur [68]. TREDs can cause disease via the expanded mRNA transcript itself, and/or the protein it is translated into, with the precise mechanism depending on which gene is expanded [63]. The majority of TREDs are autosomal dominant [63], meaning that they are inherited from one of the non-sex chromosomes, and that one copy of the gene, called the *mutant*, is sufficient for disease to manifest. The other copy of the gene, called the *wild-type* (*WT*, is unaffected and may play a key role in cellular function. A promising class of treatments for TREDs are *gene therapies* which aim to downregulate expression of the mutant allele [1]. However, the similarity between the mutant and wild-type (generally only differing in the number of repeats) poses a major challenge: treatments that downregulate the mutant can also inadvertently downregulate the WT.

In light of this, an ideal gene therapy will be *allele selective*, meaning that only the mutant allele is downregulated, not the wild-type. Several studies have reported allele selectivity using nucleic acid based approaches [66] such as antisense oligonucleotides (ASOs) [77] (see Section 2.3.4), RNA interference (RNAi), via the small interfering RNA (siRNA) (see Section 2.3.6) [71, 40] or microRNA (miRNA) [109] pathways (see Section

2.3.5), RNA-targeting CRISPR [73], and ribozymes [2, 80]. In order for selective silencing to occur, the target gene must exhibit some features that allow the therapeutic RNA or DNA strand to distinguish the mutant from the wild-type. The studies just cited exploit one (or more) of four possible features: *single nucleotide polymorphisms* (*SNPs*), *synonymous* codons, secondary structure differences, and repeat length differences.

Like all species, humans exhibit a certain amount of genetic variation. Most of this variation [4] is due to single nucleotides being mutated at certain locations in the genome. If this single nucleotide mutation is present in at least 1% of the population, it is called a single nucleotide polymorphism (SNP). Some SNPs are heterozygous, meaning that the maternal and paternal alleles have different nucleotides at the SNP's genomic location. Heterozygous SNPs that occur within genes associated with TREDs provide a marker for the mutant allele to selectively targeted [12, 77, 71]. A main drawback of this approach, however, is that different populations afflicted with a given TRED will have different SNPs, meaning that multiple therapies will have to be designed, each targeting a different SNP [44].

Another way to achieve allele selectivity is to take advantage of the degeneracy of the genetic code. As explained in Section 2.3.1, multiple codons map to the same amino acid. In [2], ribozymes were computationally designed to target the coding region of a gene associated with a TRED called OPMD. We note that on its own, this treatment would not be selective since the coding region between the wild-type and mutant was identical. However, the wild-type mRNA transcript was replaced with a synthetic one that coded for the same amino acid sequence as the original gene, but that used different codons [2]. This meant that the coding sequence of the synthetic wild-type mRNA was different than the one of the endogenous mutant, thereby protecting the synthetic mRNA from ribozyme cleavage. While effective, we note that this approach complicates therapeutic delivery, because not only does a smaller ribozyme strand have to be delivered, but so does a large mRNA one.

Another approach is to design small nucleic acid strands such as ASOs [77], siRNAs, [71, 40], microRNA [109], and CRISPR guide RNAs [73] that target the stretch of trinucleotide repeats. At a first glance, it may appear that this would not result in selective silencing, since both the wild-type and repeat contain repeats the same type of repeats. This is indeed true in some cases, where both the wild-type and mutant were downregulated [71, 40]. However, in other cases, a selective effect was observed [40, 109, 73]. The mechanisms behind this selectivity are not known with complete certainty, but the mutant mRNA strand containing more binding sites for the therapeutic DNA/RNA strands and the mutant repeats region adopting a secondary structure distinct from that of the wild-type repeats region are two plausible explanations [40, 109, 73]. A limitation of this approach is that the selectivity decreases as the number of mutant repeats approaches the number of wild-type repeats [44]. Another concern is that many other functional genes in the cell contain a certain number of repeats, potentially making them *off-targets* [44]. While such off-target affects have not been reported so far in cells and animal models [109, 73], it doesn't guarantee that such affects won't occur in human trials [44].

Finally, in [80], it was shown that an allosteric ribozyme could differentiate between two strands of different lengths, cleaving only the longer one. The starting point of this

work was a computationally designed AND gate that cleaved itself only when both of its OBSs were occupied by oligonucleotides. This *cis*-acting ribozyme was converted into *trans*-acting form by removing the stem 3 hairpin. This resulted in two strands: a trans-acting ribozyme and a substrate that binds to the ribozyme arms. The substrate was then extended by adding a variable length sequence of GCG repeats as well as effector regions complementary to the ribozyme's OBSs. Two versions of the substrate were synthesized: one contained 7 GCG repeats (typical of a wild-type PABPN1 transcript) while the other contained 11 (typical of a mutant PABPN1 transcript). It was shown that only the 11 repeat transcript was long enough to bind the OBSs and the ribozyme arms at the same time and activate the ribozyme. Consequently, the 11 repeat strand was cleaved but the 7 repeat strand was not. This selective cleavage only occurred in the presence of an additional antisense oligonucleotide which was required to unravel a hairpin that the GCG repeats would fold into. While this work is an interesting proof of concept, we believe its real-world applicability is limited. The target strands did not correspond to an actual TRED, but were instead designed to be cleaved by a modified version of a pre-existing ribozyme.

Despite exciting progress in the field of TRED therapeutics, it is clear that abundant challenges remain. To help address some of these, we designed *TriCleaver*, an EA that designs *selective ribozymes* (*sRzs*) that cleave the mRNA transcript of the mutant allele but not the mRNA transcript of the WT allele. In the next section, we will explain how sRzs function. Then, in Section 5.3, we will describe the TriCleaver evolutionary algorithm.

5.2 Selective ribozyme model

For each transcript we consider three regions: the repeats segment, the region upstream of the repeats segment (which we simply denote as the *upstream* segment), and the region downstream of the repeats segment (which we simply denote as the *downstream* segment). We consider the general case in which no SNPs are available to target and the wt and mutant transcripts have identical upstream and downstream segments. The only difference between them is that the repeats segment of the mutant is longer than the repeats segment of the wild-type. The sRz consists of an extension region attached to a *trans*-acting minimal hammerhead ribozyme. The arms of the sRz are designed to bind to a region of the mRNA transcript called the *ribozyme binding site* (*RzBS*). The RzBS is located in the downstream segment; it is therefore present on both the wild-type and mutant transcript. The extension region of the sRz consists of a *sensor* flanked by a *linker* segment on each side. The sensor is complementary to a certain number of repeats. Specifically, it is longer than the repeats segment of the wild-type transcript, and it is shorter than (or the same length as) the repeats segment of the mutant transcript. This means that the *entire* sensor will be bound to the mutant repeats segment, but only *part* of the sensor will be bound to the wild-type repeats segment. This discrepancy causes the sRz to adopt different structures depending on the number of repeats of the transcript. When it is bound to the wild-type, it adopts an inactive conformation, but when it is bound to the mutant, it adopts the active conformation. The sRz model is illustrated in Figure 21.

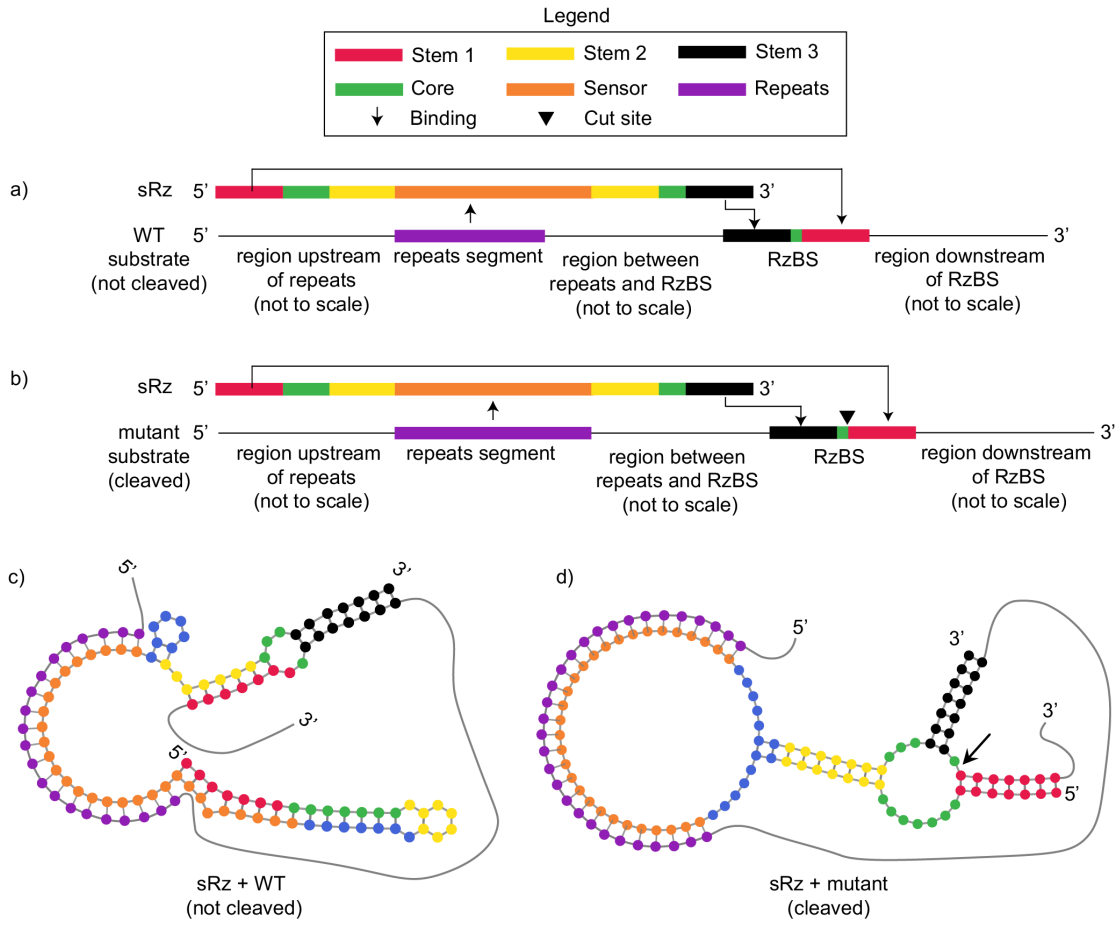


Figure 21: Selective ribozyme (sRz) model. a-b). A sRz is a *trans*-acting ribozyme that binds to a ribozyme binding site (RzBS) that is present on the both the WT and mutant transcripts associated with a TRED. In addition to the usual ribozyme segments, a sRz also contains a sensor that is complementary to the repeats on both transcripts. a) The WT repeats are shorter than the sensor and will only partial bind to it. The unbound portion of the sensor will interfere with ribozyme folding, thus preventing WT cleavage. b) The mutant repeats completely bind to the sensor, making it unable to deactivate the ribozyme. The mutant is thus cleaved. c) Secondary structure of an inactive sRz bound to the WT. d)) Secondary structure of an active sRz bound to the mutant.

5.3 Computational methodology

TriCleaver is a multi-objective EA that designs a *population* of selective ribozymes (sRzs) over several *generations*. The first generation proceeds as follows. First, the representation of candidate sRzs is *configured* based on the WT and mutant transcripts provided by the user. A population of random *individuals* is *initialized* and these individuals are each assigned a *fitness* measuring their performance on a set of objectives. Certain individuals are then selected as *parents*, who produce *offspring* through *mutation*. These offspring also have their fitness evaluated and a set of *survivors* is selected from the combined set of parents and offspring. This concludes the first generation. Subsequent generations cycle through parent selection, reproduction, fitness evaluation, and survivor selection. The survivors at the end of one generation become the population at the beginning of the next generation. This cycle *terminates* after a fixed number of generations. Finally, the results are reported to the user.

5.3.1 Configuration

The configuration stage of TriCleaver is similar to that of TruthSeqEr, but exhibits a few important differences since sRzs are *trans*-acting ribozymes, instead of *cis*-acting as in the case of ribogates. *Cis*-acting ribozymes contain all segments required for activity on the same strand (two halves of each stem and the three catalytic core segments). However, in the case of *trans*-acting ribozymes, they are split between the strands: the ribozyme (which contains S1A, S2A, S2B, S3, C1, and C2) and the substrate (which contains S1B, S3B, and C3). In addition, the substrate is not co-evolved with the sRz, but is instead provided beforehand by the user when they specify the target TRED. Therefore, the sequences of S1B, S3B, and C3 depend on the RzBS, the region of the substrate where the sRz ribozyme arms bind. This in turn constrains S1A and S3A, since they must be complementary to S1B and S3B, respectively. In TriCleaver, the location of the RzBS is part of each individual's representation and is evolved during the EA. The RzBS location is subject to two constraints. First, it must be located on a region downstream of the repeats stretch. Second, it must contain a specific sequence (NUH) at specific nucleotides in order to respect the sequence constraints of an active hammerhead ribozyme. Another important difference is that unlike ribogates, sRzs do not sense small oligonucleotide inputs. Instead, they have a single sensor segment that binds to both the wild-type and mutant repeats. As explained in Section 5.2, the length of the sensor is chosen so that it is longer than the length of the WT repeats, but shorter (or equal) to the length of the mutant repeats. It is important to note that in TREDs with many WT repeats, there will be extensive base-pairing between the sensor and the repeats. This could potentially trigger undesirable responses such as RNAi or the PKR pathway [25]. To mitigate this, the sensor is not perfectly complementary to the repeats: it contains periodic mismatches. These mismatches should also increase the flexibility of the sensor-repeats duplex, maximizing the probability that the sRz folds correctly when bound to the substrate. Flexibility is further increased by adding linker segments on each side of the sensor. Configuration begins by generating a set of valid RzBSs from the

downstream segment of the user-provided TRED transcripts. Next, the length of the sensor segment is determined, Then, a nucleotide dependency graph (NDG) is generated from a segment dependency graph (SDG). Finally, a mutation weight and a set of valid nucleotide assignments are computed for each component of the NDG.

Segment dependency graph

The nodes of the segment dependency graph correspond to segments of the sRz and the target substrate. In general, edges represent two segments that are reverse complementary. In TriCleaver, mismatches are allowed between the sensor and repeats segment. This is represented by a Boolean *mismatch vector* of the same length as the sensor. At 1 at a given index indicates that a mismatch is allowed between the corresponding nucleotides; a 0 indicates no mismatch is allowed. Each node in the SDG is labeled with a string that constrains the sequence of nucleotides that the segment can assume. The RzBS segments contain multiple possible constraint strings; an additional parameter called *RzBS location* determines which one is used. The SDG of an sRz contains the following segments:

1. 9 sRz segments
    (a) 6 ribozyme segments
        i. 1 stem 1 segment (S1A)
        ii. 2 stem 2 segments (S2A and S2B)
        iii. 1 stem 3 segment (S3A)
        iv. 2 catalytic core segments (C1 and C2)
    (b) 3 extension region segments
        i. 1 sensor segment
        ii. 2 linker segments
2. 4 substrate segments
    (a) 1 repeats segment
    (b) 3 RzBS segments
        i. 1 stem 1 segment (S1B)
        ii. 1 stem 3 segment (S3B)
        iii. 1 catalytic core segment (C3)

Nucleotide dependency graph

The SDG has an equivalent nucleotide dependency graph NDG which encodes constraints between individual nucleotides instead of segments. An NDG is generated from the SDG by:

- Splitting each unpaired segment of length L into L unpaired nodes (representing nucleotides).

- Splitting each pair of segments of length L into L pairs of nodes. Since paired segments must be reverse complementary, the i^{th} nucleotide of the first segment is paired with the $L - 1 - i^{th}$ nucleotide of the second segment.
- Splitting the constraint strings into individual characters and applying them to the appropriate nodes of the NDG.
- Splitting the sensor-repeats mismatch vector into individual bits and applying each bit to the corresponding pair of nucleotides in the NDG.

Mutation weights and valid nucleotide assignments

Each component of the NDG has a set of valid nucleotide assignments. An isolated nucleotide can assume any value permitted by its constraint character. In general, a pair of nucleotides can assume any pair of complementary values that respect the constraint characters of both nucleotides. However, if a mismatch is specified, this pair of nucleotides must assume a pair of *non-complementary* values that respect the constraint characters of both nucleotides Components with a greater number of valid nucleotide assignments are more likely to be selected for mutation. Each connected component C has an associated mutation weight equal to:

$$\frac{\text{\# of valid nucleotide assignments of } C \text{ - 1}}{\text{\# of nucleotides in } C} \tag{17}$$

5.3.2 Initialization

During initialization, a RzBS location is randomly selected. Next, each component of the NDG is selected and its corresponding nucleotides are assigned a random set of valid nucleotide values.

5.3.3 Fitness evaluation

During fitness evaluation, the structure that each candidate adopts in the wild type and mutant states is predicted using folding software. These structures are then assessed for performance (i.e. how inactive is the ribozyme in the WT state and how active is it in the mutant state) and novelty.

Folding

In principle, two structures must be predicted in order to evaluate the fitness of an individual: the sRz bound to the WT and the sRz bound to the mutant, However, we do not directly fold the sRz with the entire WT or mutant substrate. This is for two reasons. First, the substrate can be very long, resulting in a prohibitively long folding time since pseudoknot-free folding has a cubic time complexity. Second, when the sRz is bound to the RzBS off the substrate and the sensor is bound to the repeats segment of the substrate, a pseudoknot is

formed. As explained in Section 3.3.3, pseudoknot prediction possess significant difficulties. Therefore, we only fold the sRz with part of the substrate, specifically the RzBS, and possibly short segments flanking it on either side. The effect of the repeats binding to the sensor is simulated using a folding constraint. When the sRz is bound to the mutant, we force all nucleotides to be unpaired, with the exception of nucleotides that are specified to be mismatches. We denote this state as a target ON state, since we want the sRz to cleave the mutant. When the sRz is bound to the wild type, only a subset of the sensor nucleotides are forced to be unpaired. This is because the sensor is shorter than the mutant repeats, but longer than the wild-type repeats. Note that there are many potential subsets of sensor nucleotides that the WT repeats can bind to. We therefore consider multiple possible binding patterns and treat each one as a separate folding state. We denote each of these states as a target OFF state since we don't want the sRz to cleave the wild-type. Finally, we also consider the state where the sRz has bound to the RzBS but the repeats haven't had a chance to bind to the sensor yet. In this case, no folding constraint is specified. We denote this state as a target OFF state.

Phenotype generation

An individual's phenotype is a vector that measures the amount of base-pairing between certain segments. The BPPM of each state is first coarse-grained into a segment-pair magnitude matrix (SPMM). The SPMMs are then processed, merged and flattened into a single vector (called the phenotype). Note that TriCleaver uses lower level L-segments, in contrast to the higher level H-segments used by TruthSeqEr. This results in a more fine-grained phenotype.

SPMMs. An entry of the SPMM encodes the amount of base-pairing between two segments. The amount of base-pairing between two segments is equal to the sum of the probabilities of all possible base-pairs that can form between these two segments.

Phenotype. The SPMMs of target ON states (i.e. states for which the output value of target Boolean function is 1) are then selected for further processing. Specifically, the entries encoding the base-pairing between each extension region segments and the ribozyme segments are removed. The SPMMs of target OFF states are not modified. The processed SPMMs are then flattened and merged into a phenotype vector.

Novelty assessment

Next, the novelty of an individual is assessed and a novelty score is calculated. Novelty assessment begins by calculating the phenotypic distance between each pair of individuals in the population. The phenotypic distance d between two individuals x and y with respective phenotypes x^p and y^p is defined as:

$$d(X, Y) = \sum_{i=1}^{n} |x^p[i] - y^p[i]| \tag{18}$$

where n is the number of entries in the phenotype vector. Each individual x is then assigned a novelty score defined as

$$f^{nov} = \sum_{y \in K} d(x, y) \tag{19}$$

where K is the set of x's k-nearest neighbors.

Performance assessment

Next, the performance of an individual is assessed and viability, ON, and OFF scores are calculated. An individual is considered high performing if its ribozyme is active (inactive) in each target ON (OFF) state.

Motif scores. TriCleaver then calculates four *motif scores* from each BPPM. Three *stem scores* measure the average probability of a stem base-pair being present. A stem score of 1 indicates each base-pair in the stem has a 100% probability of being present. A single *core score* measures the average probability of a core nucleotide being unpaired. This score is 1 if every nucleotide has a 100% probability of being unpaired. The motif scores of a device with k states are stored in a k by 4 matrix. Each row corresponds to a state, and each column to a motif. Column 0 corresponds to the core motif and columns 1-3 to the stem motifs.

ON and OFF matrices. The motif scores matrix is split into an n by 4 *ON matrix*, and a $(k - n)$ by 4 *OFF matrix*, where n is the number of target ON states. Note that $0 < n < k$. The ON (OFF) matrix stores the motif scores for target ON (OFF) states .

ON and OFF vectors. From the ON (OFF) matrix, TriCleaver calculates an ON (OFF) vector v^{ON} (v^{OFF}) that measures how close the ribozyme is to being active (inactive) for each target ON (OFF) state. The ON and OFF vectors are calculated slightly differently. In this work, we assume that the ribozyme is fully active if all four of its motifs form, and we assume that it is fully inactive if one of its stems is completely disrupted (i.e. none of the nucleotides in the stem are properly paired). The i^{th} entry of the ON vector is equal to the average of the motif scores of the i^{th} row of the ON matrix. The i^{th} entry of the OFF vector is equal to 1 minus the minimum stem score of the i^{th} row of the OFF matrix. ON and OFF vector calculation is illustrated in Figure 9 d).

Scores. From these two vectors, TriCleaver calculates an ON score f^{ON} and an OFF score f^{OFF} as follows:

$$f^{ON} = \text{ama } v^{ON} \tag{20}$$

$$f^{OFF} = \text{ama } v^{OFF} \tag{21}$$

where we define ama as:

$$\text{ama } x = \text{avg } [\text{avg } x, \min x] \tag{22}$$

We also calculate the viability score as:

$$f^{via} = \frac{\min v^{ON} + \min v^{OFF}}{2} \tag{23}$$

Viability nullification

Finally, the ON, OFF, and novelty scores are *nullified* if the viability score of the individual drops below a certain *threshold*. The viability threshold is initially set low to give TriCleaver the opportunity for exploration. It is subsequently raised for every generation, following one of two possible schedules. The first schedule raises the threshold at a constant rate Δ, interpolating between an initial value v_0 and a final value v_f. Formally,

$$\Delta = \frac{v_f - v_0}{N} \tag{24}$$

where N is the total number of generations in the TriCleaver. The second schedule raises the threshold at a rate Δ_1 during the first N_b generations, interpolating between an initial value v_0 and a breakpoint value v_b, and at a rate Δ_2 during the remaining generations, interpolating between the breakpoint value and final value v_f. Formally,

$$\Delta = \begin{cases} \Delta_1 = \frac{v_b - v_0}{N_b} & i < N_b \\ \Delta_2 = \frac{v_f - v_b}{N - N_b} & i \geq N_b \end{cases} \tag{25}$$

where i is the current generation. An individual is considered viable at the end of a TriCleaverEr run if its viability score is greater than or equal to 0.90.

5.3.4 Parent selection and reproduction with mutation

First, each individual is selected as a parent. Next, each parent is copied and mutated R times, where R is the *mutation rate*. TriCleaver uses two types of mutation operators. The first, used 85% of the time, consists of selecting a component of the NDG and assigning its corresponding nucleotides a new set of valid values. The probability of a component being selected for mutation is proportional to its mutation weight. The second mutation operator, used 15% of the time, randomly changes the individual's RzBS location. Care must be taken when mutating the RzBS location, since the ribozymes arms (S1A and S3A) will in general no longer be complementary to the new RzBS. In this case, any conflicting S1A and S3A nucleotides are randomly assigned different nucleotides compatible with the new RzBS.

5.3.5 Survivor selection and termination

The population and offspring are merged into a set of $2N$ individuals and the fittest N individuals are selected as survivors. NSGA-ii is used to sort the individuals into a set of non-dominated fronts and TriCleaver selects the N individuals dominated by the smallest number of other individuals as survivors. TriCleaver terminates after 200 generations.

Disease	OPMD	HD
Target transcript	PABPN1	HTT
Repeat type	GGC	CAG
Number of WT repeats	7	19
Number of generations	10	109
Population size	300	300
Mutation rate	6	6
Objective scores	ON, OFF, Novelty	ON, OFF, Novelty
Viability nullification?	Yes	Yes
Viability threshold (min, max)	(0, 0.90)	(0, 0.90)
Breakpoint (generation, value)	(49, 0.20)	(49, 0.20)
Novelty neighborhood size	30	30

Table 5: TriCleaver parameters.

5.3.6 Experimental setup

We executed two runs of TriCleaver. The first run targeted PABPN1 transcripts associated OPMD, while the second one targeted HTT transcripts associated with Huntington's disease (HD). For each disease, we were provided with WT and mutant transcripts by Dr. Aida Abu-Baker of the Montreal Neurological Institute (MNI). For OPMD, the provided PAPBN1 wild-type and mutant transcripts had 6 and 9 GCG repeats, respectively. Note that the first repeat is proceeded by a G and the last one is followed by a GC. Therefore, we decided to target 7 and 10, GGC repeats, respectively, since we are not bound by the ribosome reading frame. For HD, the provided HTT wild-type and mutant transcripts had 19 and 109 CAG repeats, respectively. Each run used viability nullification and the same set of objectives scores: ON, OFF, and novelty (with a neighborhood size of 30). In both cases, 200 generations of 300 individuals were used. The EA parameters are summarized in Table 5. Folding was performed using the RNAFold and RNACofold programs from the ViennaRNA package [61]. The partition function option was enabled. Also, the *MaxLoop* parameter in the source code was changed from 30 to 300 before recompiling the package. Experiments were performed on a system running Windows 10 Pro with an AMD Ryzen Threadripper 3970X 32-Core Processor and 64 GB of RAM. Each run used 12 processes in parallel.

5.3.7 Biological methods

21 sRzs were selected from the final population of the OPMD run for further validation. They were constructed and tested in human embryonic kidney (HEK) cells by Dr. Aida Abu-Baker and Ms. Pegah Hadavi of the Montreal Neurological Institute. The experimental procedure they followed is described in detail [35]; here we provide an adapted, self-contained description of this procedure.

sRz plasmid construction

1. Each sRz selected for validation was packaged into a cassette containing a partial tRNA-val sequence, a complete constitutive transport element (CTE) sequence, and restriction enzyme cut sites [35].

    - The constitutive transport element (CTE) serves as a binding site for proteins called helicases, which unwind the target substrate, facilitating the binding of the selective ribozyme to the target cut region [74].
    - The tRNA-val allows the selective ribozyme to be efficiently exported from the nucleus (where the selective ribozyme is transcribed) into the cytoplasm (where the target substrate is located) [74]
    - The restriction enzyme cut sites are recognized by restriction enzymes which cut double-stranded DNA (dsDNA) [4]

2. dsDNA for each cassette was ordered from IDT [35]

3. Restriction nucleases (KpnI-HF and BstBI) were used to cut out a ribozyme from an existing pUC-KE-tRNA-CTE plasmid [35] provided by Dr. Barbara Nawrot [74]. The KpnI-HF and BstBI nucleases were also used to create staggered ends on the cassette dsDNA [35]

4. The cassette dsDNA was then inserted into the linearized plasmid "using the Quick Ligase kit (NEB) and the corresponding protocol" [35]

sRz cloning

1. The plasmid DNA was introduced into bacterial cells in a process called transformation. The transformed cells consisted of a strain called XL 10-Gold Ultracompetent from Agilent Technologies, which had an increased ability to uptake DNA [35]

2. The transformed cells were then incubated on an agar plate until visible colonies appeared [35]

3. Bacteria from these colonies were then transferred to a liquid broth which promoted replication [35]

4. Next, a miniprep procedure was performed, which involved lysing the cultured bacteria and extracting and purifying the plasmid DNA [35]. Specifically, the "QIAprep Spin Miniprep Kit (QIAGEN) [was used with] the [manufacturer's] protocol but eluted in lower volumes of 20 μL, instead of the recommended volume of 50 μL" [35]

5. Sanger sequencing was then performed on the plasmids to verify that the selective ribozyme sequences were correct [35]

PABPN1 gene cloning

1. In addition to the selective ribozyme plasmid, three types of PABPN1 plasmids were constructed and cloned [35]. The first contained the wild-type PABPN1 gene carrying 10 alanine repeats, the second contained a mutant PABPN1 gene carrying 13 alanine repeats, and the third contained a longer mutant PABPN1 gene carrying 17 repeats [35]. Note that the sRzs were designed to target the 13-alanine mutant and no selectivity was observed when testing them on the 17-alanine variant [35]. Consequently, the 17-alanine mutant will not be further discussed in this thesis.

2. "The plasmids, ... provided by [the] Rouleau lab [2, 3] ... were prepared by cloning cDNAs of PABPN1 wild type and mutant gene into [the] pEGFP-C2 vector (Clontech, Palo Alto, CA, USA) as described in [69]. This resulted in each plasmid coding a PABPN1-GFP fusion from which the fluorescent signal [could] be used to confirm transfection." [35]

Transfection

1. Different combinations of the sRz and wild-type / mutant PABPN1 plasmids were transfected into human embryonic kidney cells (HEK293E). See table 2 in [35] for more details.

2. The HEK cells "were cultured in DMEM (Invitrogen) containing 10% fetal bovine serum in cell culture incubator at 37 C" [35]

3. "The cells were seeded in 12-well plates and were transfected at 70% to 80% confluency using the jetPRIME transfection reagent (Polyplus) and the corresponding supplier's protocol" [35]

RNA extraction

1. 300 μL of TRIZOL was added to the cell samples, causing the cells to break (lyse) and their contents to become soluble [35]

2. Next, 70 μL of chloroform was added to the TRIZOL solution [35]

3. "The mixture was then vigorously vortexed for 15 seconds and incubated at room temperature for 2 to 3 minutes before being centrifuged for 15 minutes at 12000 RPM" [35]

4. These latter two steps caused three layers (phases) to form, each containing specific biomolecules. The organic phase contained proteins, the interphase contained DNA, and the aqueous phase contained the RNA that we wished to extract [89]

5. The aqueous phase "was then collected and 150 μL of isopropyl alcohol was added to precipate the RNA" [35]

6. After being "incubated at room temperature for 10 minutes ... the samples [were] then centrifuged for 10 minutes at 12000 RPM" [35]

7. This resulted in a dense RNA pellet underneath a liquid supernatant layer. The supernatant was then discarded [35]

8. The following two steps were then performed twice

   (a) The pellet was washed with 300 μL of 75% ethanol to remove impurities [35]

   (b) Following addition of the ethanol, "the samples were incubated in -80 degrees Celsius overnight, ... centrifuged at 12000 rpm for 5 minutes, [and the] ethanol was discarded" [35]

9. To ensure complete ethanol evaporation, "the caps were left open for about 10 minutes" [35]

10. "The RNA pellet was then re-suspended in 18 μL of nuclease-free water and the samples were stored in -80 degrees Celsius" [35]

mRNA PAPBN1 expression quantified via RT-PCR

1. Once purified, the RNA was used a template to synthesize complementary DNA (cDNA) [35]

2. This cDNA was then amplified in a PCR reaction that included two types of TaqMan probes [35]

   - The first was an RNA polymerase II probe that served as a baseline measure of fluorescence [35]
   - The second was a PABPN1 probe used to quantify the relative expression of PABPN1 mRNA by normalizing it with respect to the value of RNA polymerase II probe [35]

3. PCR was performed under the following conditions: "50°C for 2 min, 95°C for 2 min, and then 40 cycles of 95°C for 1s and 60°C for 20s" [35]

Protein extraction

1. Scraping and centrifugation (6000 RPM for 5 minutes) were used to collect the cells into a pellet, which was isolated by discarding the supernatant [35]

2. The pellet was then washed by "adding 300 μL of PBS and centrifuging at 6000 RPM for 5 minutes" [35]

3. The cells were lysed via the addition of 60 μL of buffer, followed by sonication [35]. The buffer consisted of 8M urea, 2% β-mercaptoethanol, and 0.5% SDS [35] which denatured and made the proteins more soluble by disrupting hydrogen bonds, disrupting hydrophobic interactions, and reducing disulfide linkages, respectively [62, 4]

4. Finally, a Bradford assay was performed in order to determine the protein concentration of each sample [35]

Western blot

1. The solubilized proteins were then loaded onto a polyacrylamide gel and separated by size via electrophoresis [35]

2. An electric current was then used to transfer the proteins from the gel onto a nitrocellulose membrane [35]

3. "The blots were incubated with PABPN1 antibodies (Abcam, ab75855) (1:2000) and milk (5%, w/v) overnight before being developed using the Clarity western Blotting Substrate (Bio-Rad) in the ChemiDoc System (Bio-Rad)" [35]. The role of the milk was to prevent the antibodies from binding non-specifically to the membrane [97].

A subset of sRzs from the HD run are also being tested; as of the time of writing, we are waiting for experimental results.

5.4 Results and Discussion

5.4.1 TriCleaver designs selective ribozymes that function *in silico*

Figure 22 shows a snapshot of the population of the OPMD run at five different time points (generations). We can see that the viability of the initial population is low. During the first half of the run, the fitness and diversity of the population increases. During the second half of the run, viability continues to increase, while the diversity slightly decreases. The low viability of the initial population indicates that sRz design is non-trivial; we can't just generate a set of random sequences. The high viability of the final population (all 300 sRzs were predicted to be selective) indicates that TriCleaver is an effective search algorithm. Furthermore, the fact that the final population did not lose diversity compared to the initial population underscores the effectiveness of novelty search.

5.4.2 TriCleaver designs selective ribozymes that function in cells

Of the 300 sRzs generated by TriCleaver during the OPMD run, 21 were tested in HEK cells and 2 (designated as Rzb5 and Rzb8) where determined to be selective. Figure 23 shows that the Rzb5 and Rzb8 downregulate expresion of the mutant allele, both at the mRNA and protein level, while minimally affecting the WT allele. These results show

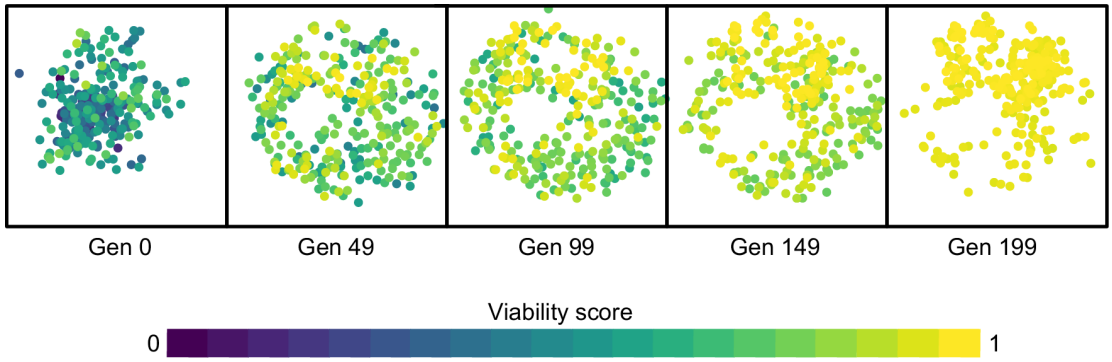


Figure 22: OPMD run timelapse. The population of sRzs at 5 generations during the OPMD run was visualized using multi-dimensional scaling (MDS). Each dot represents a candidate sRz. Dots that are close together have similar structures. An individual's viability is represented by its color; low viability is depicted as dark purple while high viability is depicted as bright yellow.

that sRzs are able to discriminate between strands with a very similar number of repeats (the mutant PABPN1 transcript is only 4 repeats longer than the WT). This is exciting since other approaches that target the repeats have been shown to be less effective when the WT and mutant repeats are close in length. Figure 24 shows the predicted structure of Rzb5 and Rzb8, as well as their location in phenotype space. We can see the secondary structures of both sRzs in the active state (when bound to the mutant) is nearly identical. This expected since the ribozyme must fold into a specific active structure leaving no room for variation. However, in the inactive state (when bound to the wild-type), they adopt noticeably different strutures. This shows that there is more than one way for an sRz to selectively silent the mutant.

5.4.3 Novelty search helps compensate for an incomplete model

All of the sRzs that were experimentally tested were predicted to be functional. However, only 2 out 21 (9.5 %) were functional in HEK cells. This indicates that our model is incomplete and factors are at play. This is not surprising, because as we saw in Chapter 1, it is impossible to model a cell's behavior with complete accuracy. Some of this uncertainty can be reduced by augmenting the fitness evaluation function with more stringent criteria, as was done for TruthSeqEr 3, but establishing these criteria would require much more experimentation and risks over-constraining the search space. Furthermore, while this approach can reduce uncertainty, it cannot eliminate it altogether. A key advantage of TriCleaver is that it generates functional sRzs without trying to account for every factor that could affect sRz function. This feature becomes even more important as more uncertainty is introduced into the biological environment. For example, as explained in 2.1.4, therapeutic RNA strands are often chemically modified to extend their half-life. However, these modifications can change how the strands interact with other molecules in the cell, sometimes in unpredictable ways [109]. TriCleaver is intrinsically designed to deal with

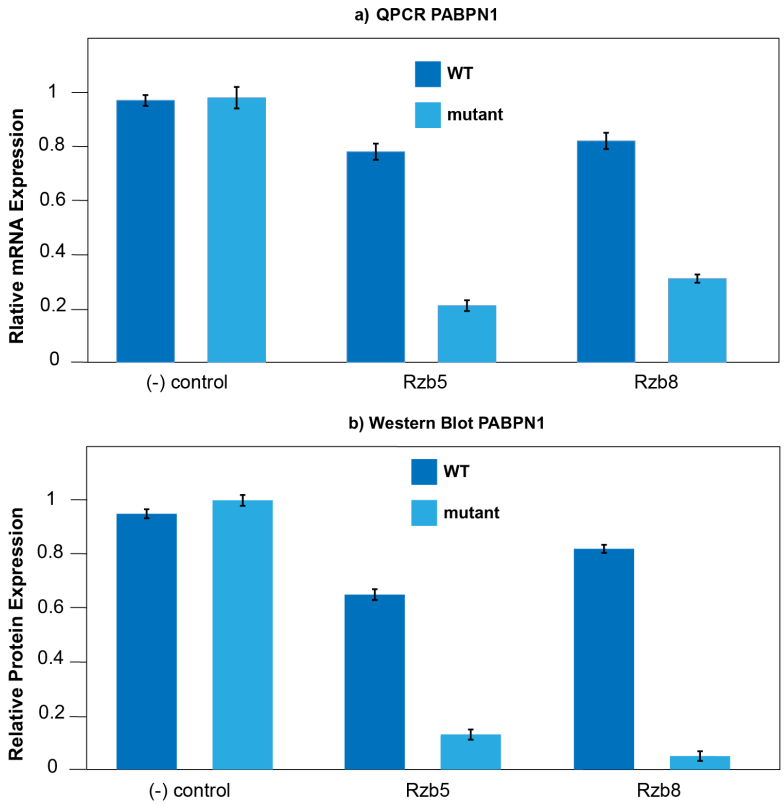


Figure 23: Experimental validation of sRzs. Rzb5 and Rzb8, two sRzs tageting the PABPN1 transcript associated with OPMD where shown to have a selective effect in HEK cells. a) qPCR results quantifying the relative expression levels of the WT and mutant mRNA when targeted with a negative control, Rzb5, and Rzb8. b) Western blot results quantifying the expression levels of the WT and mutant protein when targeted with a negative control, Rzb5, and Rzb8. Data provided courtersy of Dr. Aida Abu-Baker of Montreal Neurological Institute.

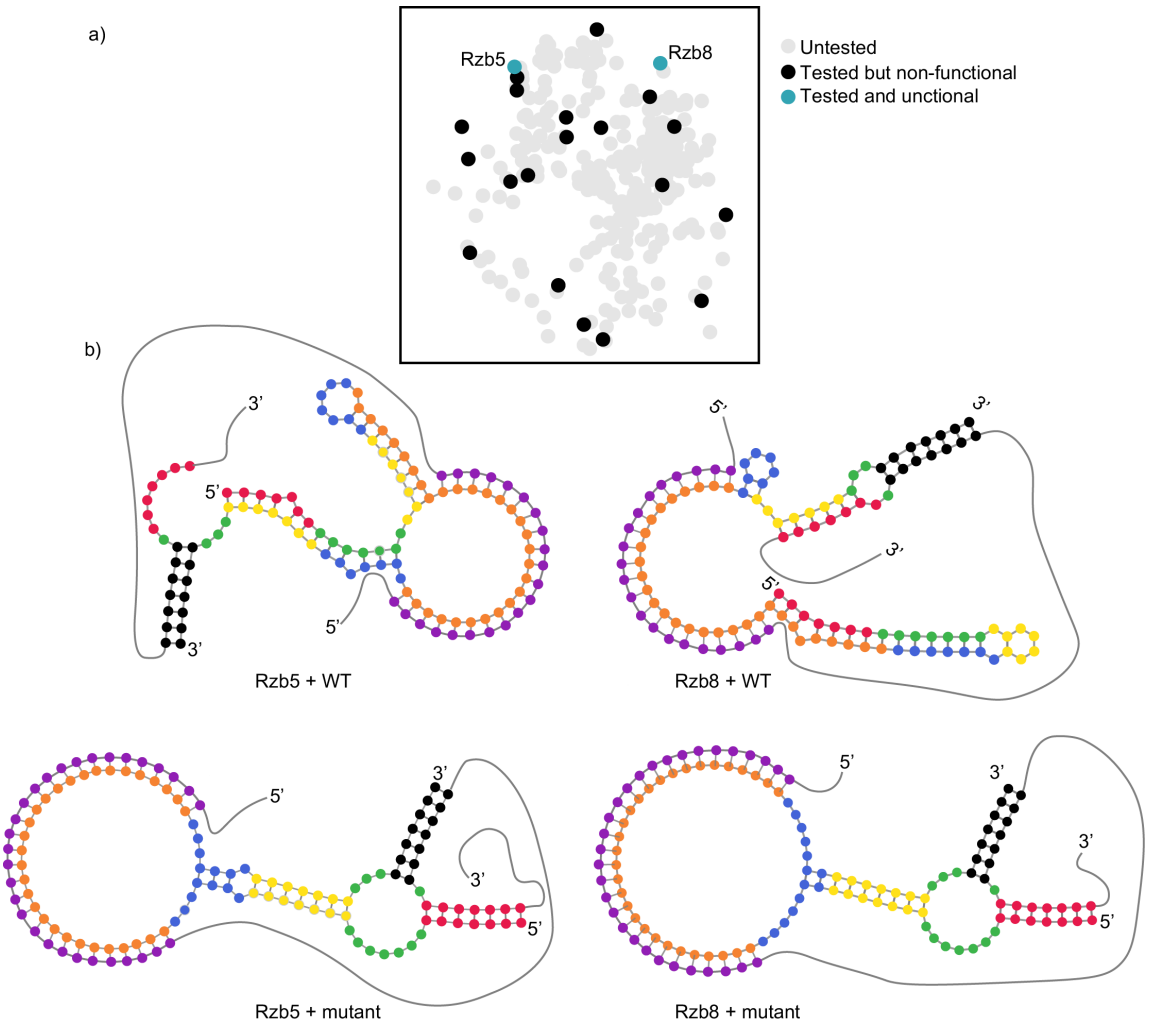


Figure 24: Visualization of the validated OPMD sRzs. a) MDS plot of the final population of the TriCleaver OPMD run. Each dot represents a sRz. Dots that are closer together have closer predicted structures. The dot's color indicates whether the sRz was untested in HEK cells, whether it was tested and found to be non-functional, or whether it was tested and demonstrated selectivity. b) Predicted secondary structures of the two experimentally validated sRzs, Rzb5 and Rzb8. Only one binding state of the WT is shown.

this unpredictability. We emphasize that TriCleaver does not just generate a population of random sequences. While this would maximize sequence diversity, the vast majority of the candidate sRzs would have no chance whatsoever of being functional, as can be seen by the low fitness of the initial random population in Figure 22. Instead, TriCleaver's performance objectives (ON and OFF) rely on a rational model of selective ribozyme behavior to generate plausible candidates, while its novelty objective prevents it from over-relying on it.

5.4.4 TriCleaver is general

TriCleaver also successfully designed a population of sRzs targeting Huntington's disease. Figure 25 shows the predicted secondary structures of sRzs targeting the HTT transcript. These structures highlight a major difference between the HTT and PABPN1 transcripts: the number of repeats. The WT and mutant PABPN1 transcripts are close in length: they only differ by 4 repeats. In contrast, the WT and mutant HTT transcripts differ by 90 repeats. This showcases another advantage of TriCleaver: its generality. The sRzs it designs are able to distinguish between alleles with similar repeat counts (as seen in OPMD) as well as alleles with very different repeat counts (as seen in Huntington's). Furthermore, TriCleaver does not require the identification of specific SNPs, allowing its designs to be used across an entire population afflicted by a given TRED. However, we must temper our excitement, since we currently only have *in silico* results for the sRzs targeting HTT (experiments in cells are currently being performed at the Montreal Neurological institute).

5.4.5 Selective ribozymes use a unique mechanism to achieve selectivity

Many other approaches require the recruitment of proteins in order to downregulate their targets. A key distinction between sRzs and many other selective approaches is that sRzs perform the silencing themselves and do not require the recruitment of endogenous proteins such as RNase H (in the case of ASOs) or AGO2 (in the case of miRNA). This prevents the sRzs from diverting these proteins from their usual functions, which could have devastating consequences [7]. Another distinguishing features of sRzs is that they need to bind to two regions in order to be active: the RzBS and the mutant repeats region. This theoretically increases their specificity compared to other approaches that only target repeats.

5.5 Conclusion

Trinucleotide repeat expansion disorders (TREDs) are serious genetic diseases caused by an expanded number of trinucleotide repeats in certain genes. Autosomal dominant cases are characterized by a healthy wild-type allele, and a pathogenic mutant allele. An emerging class of treatments are nucleic acid based gene therapies that selectively downregulate the mutant allele. While promising, these therapies currently suffer from limitations such

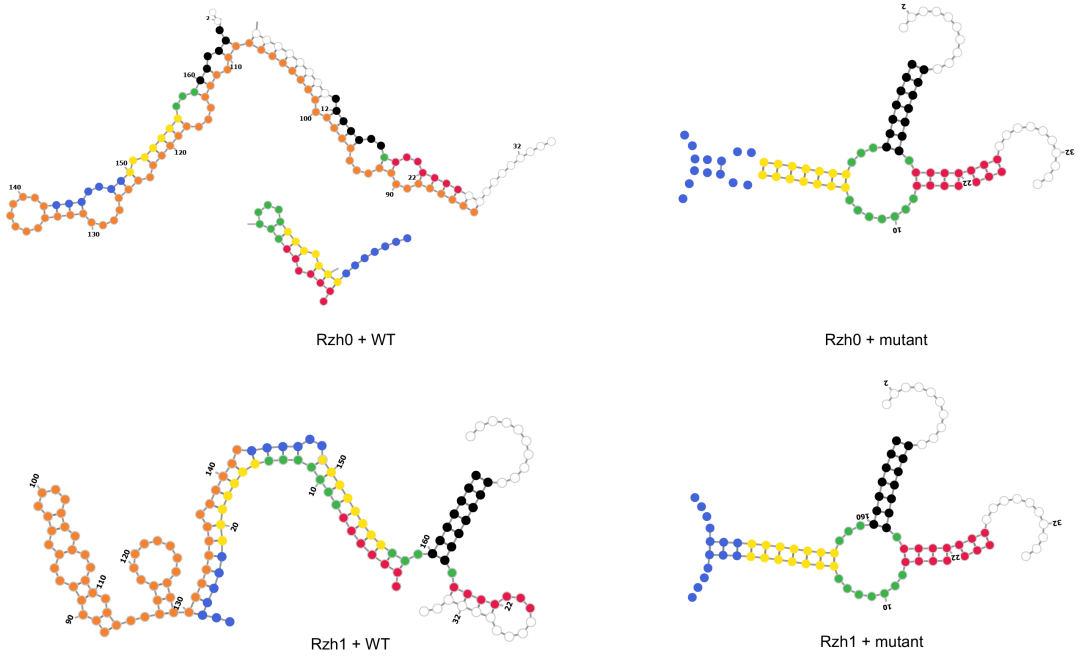


Figure 25: Predicted secondary structures of two sRzs targeting Huntington's disease. Due to the large number of repeats (and consequently large sensor size), the sensor and repeats are not shown.

as reduced selectivity when the wild-type and mutant are close in length, lack of generality, and potential off-target effects. In this chapter, we presented TriCleaver, an easy-to-use, customized evolutionary algorithm (EA) that designs selective ribozymes (sRzs) targeting TREDs provided by users. The user provides the sequences of the wild-type and mutant transcripts associated with a given TRED, and TriCleaver produces a diverse population of sRzs that are predicted to cleave the mutant, but not the wild-type. In this study, we ran TriCleaver against two TREDs: OPMD and Huntington's disease (HD). *In silico* results show that TriCleaver can design diverse populations of sRzs targeting each of these diseases. This suggests TriCleaver is a general solution for designing TRED therapies since OPMD and HD are characterized by different repeat types and lengths. In addition, *in vitro* results in mammalian cells show that two sRzs designed by TriCleaver selectively silenced the mutant allele of PABPN1, the gene associated with OPMD. Altogether, these results indicate that TriCleaver is a promising solution for designing TRED treatments.

Chapter 6

Conclusions and Future Work

This thesis explored the intersection of two techniques: evolutionary algorithms (EAs) and allosteric ribozymes (ARs). EAs are population-based search heuristics inspired by natural evolution and ARs are catalytic non-coding RNA (ncRNA) whose activity can be modulated via shape changes induced by external molecules. By combining EAs and ARs, we enabled the successful design of two promising classes of biological devices: ribogates and selective ribozymes (sRzs).

Ribogates are logic gates that take short RNA strands as inputs and produce a short RNA output strand as an output via a catalyzed self-cleavage reaction. The presence of a given input or output strand represents a HIGH logical state (1), while its absence represents a LOW logical state (0). AR-based logic gates have been around for quite some time, but their computational capacity and versatility is limited. Furthermore, the computational methods used to design them are not user-friendly, require biological domain knowledge, and cannot easily be adapted to create new ribogates. To address these limitations, we created TruthSeqEr, an EA that designs populations of ribogates. TruthSeqEr is easy to use and requires no biological domain knowledge: the user simply provides a truth table of a target function. TruthSeqEr is also versatile, it was able to successfully design (*in silico*) ribogates implementing all instances from representative sets of 1, 2, and 3-input functions. In addition, TruthSeqEr can design ribogates with a larger computational capacity than other methods: it is able to design 3-input ribogates implementing linearly inseparable functions. In order to analyze these ribogates at a deeper level, we developed a model called additive segment competition (ACS) which represents each ribogate as a small mechanism graph. Comparing the mechanism graphs of different ribogates revealed two exciting properties: ribogates can be classified into families of varying complexity based on shared structural motifs and ribogates act as more general version of an artificial neuron.

sRzs are ribozymes that selectively cleave the mutant mRNA transcript associated with a specific trinucleotide repeat expansion disorder (TRED). TREDs are a type of genetic disorder caused by an expanded number of trinucleotide repeats. The majority of TREDs are autosomal dominant, meaning they exhibit an expanded disease-causing mutant allele and a shorter functional wild-type (WT) allele. sRzs are an attractive option since they can downregulate the pathogenic allele while sparing the WT allele that might be essential to proper cell function. Other selective therapies for TREDs have been proposed, but they

suffer limitations such as lack of generality, failure to discriminate between WT and mutant alleles close in length, and potential off-target effects. To address these limitations, we created TriCleaver, an EA that designs populations of sRzs. TriCleaver is easy to use: the user simply provides the mutant and WT transcripts associated with a given TRED. TriCleaver is also a general method: *in silico* results show that it can design sRzs targeting both OPMD and Huntington's disease (HD), two TREDs characterized by different repeats and different WT and mutant lengths. In addition, two sRzs where shown to be effective against OPMD in mammalian cells. We also note that both TruthSeqEr and TriCleaver use a quality diversity approach to generate diverse ribogates and sRzs. This prevents the designs over-exploiting strategies that are incorrectly predicted to work due to limitations in modeling RNA and cellular behavior. In summary, the work in this thesis highlights the power of both EAs and ARs, as well as the utility of the TruthSeqEr and TriCleaver algorithms we developed.

6.1 Limitations

Although TruthSeqEr and TriCleaver show promise, they exhibit many limitations that could reduce the real-world impact of the ribogates and sRzs they design. We now highlight the most important of these limitations and suggest possible research avenues towards mitigating them.

6.1.1 Artificial model system

The selective ribozymes targeting the PABPN1 gene were tested on a relatively artificial system consisting of PABPN1 wild-type and mutant genes being cloned into plasmids and transfected into HEK cells. While such experiments were an important milestone, in order to assess the real-world therapeutic value of the sRzs, more realistic biological models will be required. To address this, our colleagues at the Montreal Neurological Institute (MNI) have begun testing sRzs targeting the PAPBN1 and HTT genes in patient-derived cells. In addition, we are finalizing the creation of an easy-to-use web service that hosts the TriCleaver and TruthSeqEr algorithms, making it easy for future researchers to generate designs for model organisms such as *c. elegans* and mice.

6.1.2 Slow testing rate

Both TruthSeqEr and TriCleaver can generate hundreds of diverse designs in a less than a day on a consumer workstation. However, it has taken years to test a relatively small number of ribogates and selective ribozymes in biological environments. This makes it difficult and time-consuming to evaluate the performance of the current version of the algorithms. This slow feedback loop also makes it impractical to use the gathered experimental data to improve the algorithms. There are two ways in which biological validation can be accelerated. The first is to automate the experimental protocols used to test ribogates and sRzs.

This is already been done to some extent: our partners at the Montreal Neurological Institute (MNI) have begun collaborating with Concordia's Genome Foundry to automate steps such as plasmid construction. However, the experimental protocol (described in Section 5.3.7), contains multiple steps, and not all of these steps can be cost-effectively automated. In the future, it is likely that automation will become more prevalent and cheaper, but currently, biological experimentation remains a bottleneck. Therefore, we have begun exploring a second avenue: amending the algorithms so that the designs they generate are more amenable to automation. In collaboration with Jonathan Perreault's lab at INRS, we have crafted an altered version TruthSeqEr that generates ribogates with special features such as barcode sequences and OBSs that respond to an identical set of inputs. This allows us to order hundreds of ribogates at once for low cost using an oligo pool [53, 51], simultaneously test these ribogates using a single test tube per input state (instead of one tube per ribogate per input state), and quantify ribozyme cleavage using high-throughput sequencing [87]. It is our hope that these techniques will greatly increase the rate at which ribogates and selective ribozymes can be tested.

6.1.3 Limited prediction accuracy

9.5% of the tested sRzs selectively down-regulated expression of the mutant PABPN1 gene. In qualitative terms this was a success since two sRzs with therapeutic potential were designed. However, in quantitative terms there is much room for improvement because a vast majority of the sRzs that were predicted to be functional *in silico* failed in cells. This lack of accuracy is caused in large part by limitations in the folding algorithm used to predict the ribogate and sRz secondary structures.

Recall from Section 3.3.3 that an input binding to its corresponding OBS (or the repeats binding to the sensor) resulted in the formation of an interactive pseudoknot. Although algorithms exist that can predict pseudoknots, they are limited to certain classes of pseudoknots and have an increased computational complexity [110]. We circumvented the pseudoknot problem by making certain assumptions on how the inputs (repeats) would bind to their corresponding OBS (sensor) and encoding these assumptions into folding constraints. However, the accuracy of our computational model depends on the validity of these assumptions. Also, it was only possible to make these assumptions because we knew *a priori* that interactive pseudoknots would occur. However, it is quite possible that certain ribogate and sRz designs contain additional pseudoknots whose existence cannot be reasoned about without actually folding the strands. Therefore, the lack of pseudoknot prediction is concerning. It may therefore be beneficial to incorporate pseudoknot-capable foldings algorithms, even if they can only predict a subset of pseudoknot classes. To mitigate the increased computational footprint of these algorithms, we might call them only once as a post-processing step instead of every generation during the EA loop.

In current version of TriCleaver, we did not co-fold the sRz with the entire substrate. Instead, we only co-folded it with the ribozyme binding site (RzBS), the region of the substrate targeted by the ribozyme arms. This significantly reduces computation time, since the shorter the strand, the faster its structure can be predicted by the folding software.

However, this approach may be too drastic and prediction accuracy might be increased by considering a longer region around the RzBS.

Note that even in the pseudoknot-free case, folding algorithms are not 100% accurate. One possible way to address this is to use multiple folding algorithms instead of a just one. In a given state, each folding algorithm would produce a base-pairing probability matrix (BPPM). These multiple BPPMs could then be aggregated into a single BPPM using a statistical measure such as the mean or distance-weighted mean.

Furthermore, we have only considered the secondary structures that would be observed once equilibrium has been reached. However, it might take an intractable amount of time to reach the equilibrium state. This means that in practice, the ribogates and sRzs might not adopt the structures predicted by the folding algorithm, but instead get trapped in a local minimum. It is plausible that the extensive OBS-OBS binding in advanced ribogates increases this possibility. To mitigate this, we could predict folding *trajectories* instead of just the equilibrium structure by using algorithms such as *Kinfold* [28]. Indeed, this approach was used in [81] as part of the algorithm that designed simple 1 and 2-input logic gates. Unfortunately, we found that using it as part of the EA loop when designing more sophisticated devices was too time-consuming. That being said, it could be worth investigating using it as filter during a post-processing step.

In addition to the aforementioned folding limitations, there are other factors that make it difficult to predict the functionality of ribogates and sRzs. Indeed, even if we were able to predict with complete accuracy the ensemble of structures that a ribozyme adopts in a given state, mapping this ensemble to single real number quantifying the ribozyme's activity level is not trivial. In this work, we calculated a set of motif scores from the base-pairing probability matrices and aggregated them into activity and inactivity scores, but it is possible that better metrics exist.

Machine learning is an exciting approach that could potentially solve both structural prediction and ribozyme activity quantification. A model could be trained that predicts ribozyme activity from the sRz/ribogate and the substrate/input sequences. However, machine learning methods typically require copious data, and as was previously explained, gathering experimental data is time-consuming.

6.1.4 Utilization of only a single type of ribozyme

Currently, TriCleaver and TruthSeqEr utilize only one type of ribozyme: the minimal hammerhead. However, as discussed in Section 2.3.8, there are many more types cleavage-catalyzing ribozymes. These ribozymes exhibit different properties that may be beneficial to synthetic biologists. For example, the extended hammerhead ribozyme [92] has a faster cleavage rate the minimal variant. One possible avenue of improvement is to modify TriCleaver so that it may support multiple types of ribozyme. However, this is not trivial. For a ribozyme to be used as an sRz or ribogate, it must meet certain criteria such as having well-defined active structure and a location to place the extension region that doesn't interfere with the active structure,

6.1.5 Palindromic repeats of lengths other than 3 are not considered

TriCleaver currently only designs sRzs that target substrates with a repeat unit of length 3 (e.g. GCG, CAG, etc.). However, although less common, some diseases have transcripts with repeats of lengths other 3 nucleotides. For example, some cases of amyotrophic lateral sclerosis (ALS) are caused by a hexanucleotide repeat expansion (GGGGCC) [48]. It may therefore be worthwhile to update TriCleaver so that it can handle arbitrary repeat unit lengths.

6.1.6 Cut site diversity may be limited

By using novelty search, TriCleaver successfully designs sRzs that are structurally diverse. However, we have noticed that certain RzBSs are over-represented in the sRz population. This may be problematic, since certain RzBSs might be sequestered by other RNAs or proteins. Therefore, we may want to explicitly promote RzBS diversity by modifiying to the phenotypic distance calculation so that it considers RzBS locations in additional to secondary structures.

Bibliography

[1] A. Abu-Baker, N. Kharma, C. Neri, S. Rasheed, P. A. Dion, L. Varin, and G. A. Rouleau. Current targeted therapeutic strategies for oculopharyngeal muscular dystrophy: from pharmacological to rna replacement and gene editing therapies. *International Journal of Clinical Neurosciences and Mental Health*, (3(Suppl. 1)), 2016.

[2] A. Abu-Baker, N. Kharma, J. Perreault, A. Grant, M. Shekarabi, C. Maios, M. Dona, C. Neri, P. A. Dion, A. Parker, L. Varin, and G. A. Rouleau. Rna-based therapy utilizing oculopharyngeal muscular dystrophy transcript knockdown and replacement. *Mol Ther Nucleic Acids*, 15:12–25, 2019.

[3] A. Abu-Baker, J. Laganiere, R. Gaudet, D. Rochefort, B. Brais, C. Neri, P. Dion, and G. Rouleau. Lithium chloride attenuates cell death in oculopharyngeal muscular dystrophy by perturbing wnt/-catenin pathway. *Cell death disease*, 4(10):e821–e821, 2013.

[4] B. Alberts, R. Heald, A. Johnson, D. Morgan, M. Raff, K. Roberts, P. Walter, J. Wilson, and T. Hunt. *Molecular Biology of the Cell*. W. W. Norton & Company, New York, 7 edition, 2022.

[5] B. L. Allen and D. J. Taatjes. The mediator complex: a central integrator of transcription. *Nat Rev Mol Cell Biol*, 16(3):155–66, 2015.

[6] L. Armstrong. *Epigenetics*. Garland Science, New York, 2014.

[7] R. Batra, D. A. Nelles, D. M. Roth, F. Krach, C. A. Nutter, T. Tadokoro, J. D. Thomas, L. J. Sznajder, S. M. Blue, and H. L. Gutierrez. The sustained expression of cas9 targeting toxic rnas reverses disease phenotypes in mouse models of myotonic dystrophy type 1. *Nature biomedical engineering*, 5(2):157–168, 2021.

[8] C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[9] R. Bose, I. Saleem, and A. M. Mustoe. Causes, functions, and therapeutic possibilities of rna secondary structure ensembles and alternative states. *Cell Chemical Biology*, 2024.

[10] R. R. Breaker. Imaginary ribozymes, 2020.

[11] D. H. Burke, N. D. Ozerova, and M. Nilsen-Hamilton. Allosteric hammerhead ribozyme traps. *Biochemistry*, 41(21):6588–6594, 2002.

[12] J. B. Carroll, S. C. Warby, A. L. Southwell, C. N. Doty, S. Greenlee, N. Skotte, G. Hung, C. F. Bennett, S. M. Freier, and M. R. Hayden. Potent and selective antisense oligonucleotides targeting single-nucleotide polymorphisms in the huntington disease gene/allele-specific silencing of mutant huntingtin. *Molecular Therapy*, 19(12):2178–2185, 2011.

[13] L.-L. Chen. The expanding regulatory mechanisms and cellular functions of circular rnas. *Nature reviews Molecular cell biology*, 21(8):475–490, 2020.

[14] Y. Chen, S. Zhang, E. M. Young, T. S. Jones, D. Densmore, and C. A. Voigt. Genetic circuit design automation for yeast. *Nature Microbiology*, 5(11):1349–1360, 2020.

[15] Z. Chen, R. D. Kibler, A. Hunt, F. Busch, J. Pearl, M. Jia, Z. L. VanAernum, B. I. Wicky, G. Dods, and H. Liao. De novo design of protein logic gates. *Science*, 368(6486):78–84, 2020.

[16] C. R. Clapier, J. Iwasa, B. R. Cairns, and C. L. Peterson. Mechanisms of action and regulation of atp-dependent chromatin-remodelling complexes. *Nature reviews Molecular cell biology*, 18(7):407–422, 2017.

[17] A. Cornish-Bowden. Nomenclature for incompletely specified bases in nucleic acid sequences: recommendations 1984. *Nucleic acids research*, 13(9):3021, 1985.

[18] A. Cubillos-Ruiz, T. Guo, A. Sokolovska, P. F. Miller, J. J. Collins, T. K. Lu, and J. M. Lora. Engineering living therapeutics with synthetic biology. *Nature Reviews Drug Discovery*, 20(12):941–960, 2021.

[19] B. R. Cullen and S. Cherry. Is rna interference a physiologically relevant innate antiviral immune response in mammals? *Cell host & microbe*, 14(4):374–378, 2013.

[20] C. Davidovich, L. Zheng, K. J. Goodrich, and T. R. Cech. Promiscuous rna binding by polycomb repressive complex 2. *Nature structural & molecular biology*, 20(11):1250–1257, 2013.

[21] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[22] E. Decroly, F. Ferron, J. Lescar, and B. Canard. Conventional and unconventional mechanisms for capping viral mrna. *Nat Rev Microbiol*, 10(1):51–65, 2011.

[23] R. M. Dirks and N. A. Pierce. A partition function algorithm for nucleic acid secondary structure including pseudoknots. *Journal of computational chemistry*, 24(13):1664–1677, 2003.

[24] J. A. Doudna and J. R. Lorsch. Ribozyme catalysis: not different, just worse. *Nat Struct Mol Biol*, 12(5):395–402, 2005.

[25] D. Elliott and M. Ladomery. *Molecular Biology of RNA*. Oxford University Press, 2016.

[26] E. A. Ernst. *Optimal combinational multi-level logic synthesis*. University of Michigan, 2009.

[27] C. R. Faehnle, J. Walleshauser, and L. Joshua-Tor. Mechanism of dis3l2 substrate recognition in the lin28–let-7 pathway. *Nature*, 514(7521):252–256, 2014.

[28] C. Flamm, W. Fontana, I. L. Hofacker, and P. Schuster. Rna folding at elementary step resolution. *Rna*, 6(3):325–338, 2000.

[29] A. H. Fox, S. Nakagawa, T. Hirose, and C. S. Bond. Paraspeckles: where long noncoding rna meets phase separation. *Trends in biochemical sciences*, 43(2):124–135, 2018.

[30] M. W. Gander, J. D. Vrana, W. E. Voje, J. M. Carothers, and E. Klavins. Digital logic circuits in yeast with crispr-dcas9 nor gates. *Nature communications*, 8(1):15459, 2017.

[31] N. Gil and I. Ulitsky. Regulation of gene expression by cis-acting long non-coding rnas. *Nat Rev Genet*, 21(2):102–117, 2020.

[32] G. J. Goodall and V. O. Wickramasinghe. Rna in cancer. *Nature Reviews Cancer*, 21(1):22–36, 2021.

[33] A. A. Green, J. Kim, D. Ma, P. A. Silver, J. J. Collins, and P. Yin. Complex cellular logic computation using ribocomputing devices. *Nature*, 548(7665):117–121, 2017.

[34] V. Haberle and A. Stark. Eukaryotic core promoters and the functional basis of transcription initiation. *Nat Rev Mol Cell Biol*, 19(10):621–637, 2018.

[35] P. Hadavi. *Testing and Characterization of Selective Trans-Acting Hammerhead Ribozymes that Cleave Two Disease-Causing Mutant Transcripts of the PABPN1 Gene*. Thesis, 2022.

[36] S. Hammer, B. Tschiatschek, C. Flamm, I. L. Hofacker, and S. Findeiß. Rnablueprint: flexible multiple target nucleic acid sequence design. *Bioinformatics*, 33(18):2850–2858, 2017.

[37] A. K. Henras, C. Plisson-Chastang, M. F. O'Donohue, A. Chakraborty, and P. E. Gleizes. An overview of pre-ribosomal rna processing in eukaryotes. *Wiley Interdiscip Rev RNA*, 6(2):225–42, 2015.

[38] M. W. Hentze, A. Castello, T. Schwarzl, and T. Preiss. A brave new world of rna-binding proteins. *Nature reviews Molecular cell biology*, 19(5):327–341, 2018.

[39] I. L. Hofacker, W. Fontana, P. F. Stadler, L. S. Bonhoeffer, M. Tacker, and P. Schuster. Fast folding and comparison of rna secondary structures. *Monatshefte fur chemie*, 125:167–167, 1994.

[40] J. Hu, M. Matsui, K. T. Gagnon, J. C. Schwartz, S. Gabillet, K. Arar, J. Wu, I. Bezprozvanny, and D. R. Corey. Allele-specific silencing of mutant huntingtin and ataxin-3 genes by targeting expanded cag repeats in mrnas. *Nature biotechnology*, 27(5):478–484, 2009.

[41] R. L. Jernigan, I. Bahar, and K. Dill. *Protein actions: Principles and modeling*. Garland Science, New York, 2017.

[42] L. Jiang, C. Shao, Q.-J. Wu, G. Chen, J. Zhou, B. Yang, H. Li, L.-T. Gou, Y. Zhang, and Y. Wang. Neat1 scaffolds rna-binding proteins and the microprocessor to globally enhance pri-mirna processing. *Nature structural & molecular biology*, 24(10):816–824, 2017.

[43] S. Jonas and E. Izaurralde. Towards a molecular understanding of microrna-mediated gene silencing. *Nature reviews genetics*, 16(7):421–433, 2015.

[44] C. Kay, N. H. Skotte, A. L. Southwell, and M. R. Hayden. Personalized gene silencing therapeutics for huntington disease. *Clin Genet*, 86(1):29–36, 2014.

[45] A. D. Keefe, S. Pai, and A. Ellington. Aptamers as therapeutics. *Nature reviews Drug discovery*, 9(7):537–550, 2010.

[46] P. Kerpedjiev, S. Hammer, and I. L. Hofacker. Forna (force-directed rna): Simple and effective online rna secondary structure diagrams. *Bioinformatics*, 31(20):3377–9, 2015.

[47] C. Kilchert, S. Wittmann, and L. Vasiljeva. The regulation and functions of the nuclear rna exosome complex. *Nature Reviews Molecular Cell Biology*, 17(4):227–239, 2016.

[48] G. Kim, O. Gautier, E. Tassoni-Tsuchida, X. R. Ma, and A. D. Gitler. Als genetics: gains, losses, and implications for future therapies. *Neuron*, 108(5):822–842, 2020.

[49] J. Kim, Y. Zhou, P. D. Carlson, M. Teichmann, S. Chaudhary, F. C. Simmel, P. A. Silver, J. J. Collins, J. B. Lucks, and P. Yin. De novo-designed translation-repressing riboregulators for multi-input cellular logic. *Nature chemical biology*, 15(12):1173–1182, 2019.

[50] S. Kim and J. Wysocka. Deciphering the multi-scale, quantitative cis-regulatory code. *Mol Cell*, 83(3):373–392, 2023.

[51] S. Kosuri and G. M. Church. Large-scale de novo dna synthesis: technologies and applications. *Nature methods*, 11(5):499–507, 2014.

[52] J. B. Kruskal and M. Wish. *Multidimensional scaling*. Sage, 1978.

[53] B. P. Kuiper, R. C. Prins, and S. Billerbeck. Oligo pools as an affordable source of synthetic dna for cost-effective library construction in protein-and metabolic pathway engineering. *ChemBioChem*, 23(7):e202100507, 2022.

[54] J. Kuriyan, B. Konforti, and D. Wemmer. *The molecules of life: Physical and chemical principles*. Garland Science, New York, 2013.

[55] T. Kuwabara, M. Warashina, T. Tanabe, K. Tani, S. Asano, and K. Taira. A novel allosterically trans-activated ribozyme, the maxizyme, with exceptional specificity in vitro and in vivo. *Molecular Cell*, 2(5):617–627, 1998.

[56] F. Lai, U. A. Orom, M. Cesaroni, M. Beringer, D. J. Taatjes, G. A. Blobel, and R. Shiekhattar. Activating rnas associate with mediator to enhance chromatin architecture and transcription. *Nature*, 494(7438):497–501, 2013.

[57] S. M. Lauberth, T. Nakayama, X. Wu, A. L. Ferris, Z. Tang, S. H. Hughes, and R. G. Roeder. H3k4me3 interactions with taf3 regulate preinitiation complex assembly and selective gene activation. *Cell*, 152(5):1021–1036, 2013.

[58] J. Lehman and K. O. Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, pages 329–336.

[59] D. M. Lilley and F. Eckstein. Ribozymes and rna catalysis: introduction and primer. 2007.

[60] J. Lim, M. Ha, H. Chang, S. C. Kwon, D. K. Simanshu, D. J. Patel, and V. N. Kim. Uridylation by tut4 and tut7 marks mrna for degradation. *Cell*, 159(6):1365–1376, 2014.

[61] R. Lorenz, S. H. Bernhart, C. Höner zu Siederdissen, H. Tafer, C. Flamm, P. F. Stadler, and I. L. Hofacker. Viennarna package 2.0. *Algorithms for molecular biology*, 6:1–14, 2011.

[62] C. B. Malafaia, M. L. Guerra, T. D. Silva, P. M. Paiva, E. B. Souza, M. T. Correia, and M. V. Silva. Selection of a protein solubilization method suitable for phytopathogenic bacteria: a proteomics approach. *Proteome Science*, 13:1–7, 2015.

[63] I. Malik, C. P. Kelley, E. T. Wang, and P. K. Todd. Molecular mechanisms underlying nucleotide repeat expansion disorders. *Nat Rev Mol Cell Biol*, 22(9):589–607, 2021.

[64] S. Malik and R. G. Roeder. Regulation of the rna polymerase ii pre-initiation complex by its associated coactivators. *Nature Reviews Genetics*, 24(11):767–782, 2023.

[65] A. G. Matera, R. M. Terns, and M. P. Terns. Non-coding rnas: lessons from the small nuclear and small nucleolar rnas. *Nature reviews Molecular cell biology*, 8(3):209–220, 2007.

[66] M. Matsui and D. R. Corey. Allele-selective inhibition of trinucleotide repeat genes. *Drug discovery today*, 17(9-10):443–450, 2012.

[67] J. S. McCaskill. The equilibrium partition function and base pair binding probabilities for rna secondary structure. *Biopolymers: Original Research on Biomolecules*, 29(6-7):1105–1119, 1990.

[68] C. T. McMurray. Mechanisms of trinucleotide repeat instability during human development. *Nat Rev Genet*, 11(11):786–99, 2010.

[69] C. Messaed, P. A. Dion, A. Abu-Baker, D. Rochefort, J. Laganiere, B. Brais, and G. A. Rouleau. Soluble expanded pabpn1 promotes cell death in oculopharyngeal muscular dystrophy. *Neurobiology of disease*, 26(3):546–557, 2007.

[70] R. Micura and C. Höbartner. Fundamental studies of functional nucleic acids: aptamers, riboswitches, ribozymes and dnazymes. *Chemical Society Reviews*, 49(20):7331–7353, 2020.

[71] V. M. Miller, H. Xia, G. L. Marrs, C. M. Gouvion, G. Lee, B. L. Davidson, and H. L. Paulson. Allele-specific silencing of dominant disease genes. *Proceedings of the National Academy of Sciences*, 100(12):7195–7200, 2003.

[72] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational*. MIT press, 1988.

[73] K. H. Morelli, Q. Wu, M. L. Gosztyla, H. Liu, M. Yao, C. Zhang, J. Chen, R. J. Marina, K. Lee, K. L. Jones, M. Y. Huang, A. Li, C. Smith-Geater, L. M. Thompson, W. Duan, and G. W. Yeo. An rna-targeting crispr–cas13d system alleviates disease-related phenotypes in huntington's disease models. *Nature Neuroscience*, 26(1):27–38, 2023.

[74] B. Nawrot, S. Antoszczyk, M. Maszewska, T. Kuwabara, M. Warashina, K. Taira, and W. J. Stec. Efficient inhibition of -secretase gene expression in hek293 cells by trnaval-driven and cte-helicase associated hammerhead ribozymes. *European journal of biochemistry*, 270(19):3962–3970, 2003.

[75] A. A. Nielsen, B. S. Der, J. Shin, P. Vaidyanathan, V. Paralanov, E. A. Strychalski, D. Ross, D. Densmore, and C. A. Voigt. Genetic circuit design automation. *Science*, 352(6281):aac7341, 2016.

[76] L. Nissim, M.-R. Wu, E. Pery, A. Binder-Nissim, H. I. Suzuki, D. Stupp, C. Wehrspaun, Y. Tabach, P. A. Sharp, and T. K. Lu. Synthetic rna-based immunomodulatory gene circuits for cancer immunotherapy. *Cell*, 171(5):1138–1150. e15, 2017.

[77] M. E. Ostergaard, A. L. Southwell, H. Kordasiewicz, A. T. Watt, N. H. Skotte, C. N. Doty, K. Vaid, E. B. Villanueva, E. E. Swayze, C. F. Bennett, M. R. Hayden, and P. P. Seth. Rational design of antisense oligonucleotides targeting single nucleotide polymorphisms for potent and allele selective suppression of mutant huntingtin in the cns. *Nucleic Acids Res*, 41(21):9634–50, 2013.

[78] S. S. Panchapakesan and R. R. Breaker. The case of the missing allosteric ribozymes. *Nature chemical biology*, 17(4):375–382, 2021.

[79] L. A. Passmore and J. Coller. Roles of mrna poly (a) tails in regulation of eukaryotic gene expression. *Nature Reviews Molecular Cell Biology*, 23(2):93–106, 2022.

[80] R. Penchovsky. Engineering integrated digital circuits with allosteric ribozymes for scaling up molecular computation and diagnostics. *ACS Synthetic Biology*, 1(10):471–482, 2012. doi: 10.1021/sb300053s.

[81] R. Penchovsky and R. R. Breaker. Computational design and experimental validation of oligonucleotide-sensing allosteric ribozymes. *Nature biotechnology*, 23(11):1424–1433, 2005.

[82] J. Perreault, Z. Weinberg, A. Roth, O. Popescu, P. Chartrand, G. Ferbeyre, and R. R. Breaker. Identification of hammerhead ribozymes in all domains of life reveals novel structural variations. *PLoS computational biology*, 7(5):e1002031, 2011.

[83] E. Z. Poirier, M. D. Buck, P. Chakravarty, J. Carvalho, B. Frederico, A. Cardoso, L. Healy, R. Ulferts, R. Beale, and C. Reis e Sousa. An isoform of dicer protects mammalian stem cells against multiple rna viruses. *Science*, 373(6551):231–236, 2021.

[84] H. Porta and P. M. Lizardi. An allosteric hammerhead ribozyme. *Bio/Technology*, 13(2):161–164, 1995.

[85] J. J. Quinn and H. Y. Chang. Unique features of long non-coding rna biogenesis and function. *Nature reviews genetics*, 17(1):47–62, 2016.

[86] E. I. Ramlan and K.-P. Zauner. In-silico design of computational nucleic acids for molecular information processing. *Journal of Cheminformatics*, 5:1–15, 2013.

[87] J. A. Reuter, D. V. Spacek, and M. P. Snyder. High-throughput sequencing technologies. *Molecular cell*, 58(4):586–597, 2015.

[88] C. Rinaldi and M. J. Wood. Antisense oligonucleotides: the next frontier for treatment of neurological disorders. *Nature Reviews Neurology*, 14(1):9–21, 2018.

[89] D. C. Rio, M. Ares, G. J. Hannon, and T. W. Nilsen. Purification of rna using trizol (tri reagent). *Cold Spring Harbor Protocols*, 2010(6):pdb. prot5439, 2010.

[90] G. Rodrigo, T. E. Landrain, S. Shen, and A. Jaramillo. A new frontier in synthetic biology: automated design of small rna devices in bacteria. *Trends in Genetics*, 29(9):529–536, 2013.

[91] E. Rohner, R. Yang, K. S. Foo, A. Goedel, and K. R. Chien. Unlocking the promise of mrna therapeutics. *Nature biotechnology*, 40(11):1586–1600, 2022.

[92] V. Saksmerprome, M. Roychowdhury-Saha, S. Jayasena, A. Khvorova, and D. H. Burke. Artificial tertiary motifs stabilize trans-cleaving hammerhead ribozymes under conditions of submillimolar divalent ions and high temperatures. *Rna*, 10(12):1916–1924, 2004.

[93] R. L. Setten, J. J. Rossi, and S. P. Han. The current state and future directions of rnai-based therapeutics. *Nat Rev Drug Discov*, 18(6):421–446, 2019.

[94] R. Shang, S. Lee, G. Senavirathne, and E. C. Lai. micrornas in action: biogenesis, function and regulation. *Nature Reviews Genetics*, 24(12):816–833, 2023.

[95] J. Sollier and K. A. Cimprich. Breaking bad: R-loops and genome integrity. *Trends in cell biology*, 25(9):514–522, 2015.

[96] R. S. Stanković, S. Stanković, H. Astola, and J. T. Astola. *Equivalence classes of Boolean functions*, pages 1–31. Springer, 2010.

[97] R. Sule, G. Rivera, and A. V. Gomes. Western blotting (immunoblotting): history, theory, uses, protocol and problems. *Biotechniques*, 75(3):99–114, 2023.

[98] K. Tatosyan, I. Ustyantsev, and D. Kramerov. Rna degradation in eukaryotic cells. *Molecular Biology*, 54:485–502, 2020.

[99] I. Tinoco Jr and C. Bustamante. How rna folds. *Journal of molecular biology*, 293(2):271–281, 1999.

[100] T. Treiber, N. Treiber, and G. Meister. Regulation of microrna biogenesis and its crosstalk with other cellular pathways. *Nature reviews Molecular cell biology*, 20(1):5–20, 2019.

[101] V. Trianni and M. López-Ibáñez. Advantages of task-specific multi-objective optimisation in evolutionary robotics. *PloS one*, 10(8):e0136406, 2015.

[102] V. Tripathi, J. D. Ellis, Z. Shen, D. Y. Song, Q. Pan, A. T. Watt, S. M. Freier, C. F. Bennett, A. Sharma, and P. A. Bubulya. The nuclear-retained noncoding rna malat1 regulates alternative splicing by modulating sr splicing factor phosphorylation. *Molecular cell*, 39(6):925–938, 2010.

[103] D. H. Turner and D. H. Mathews. Nndb: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure. *Nucleic acids research*, 38(suppl1):D280–D282, 2010.

[104] K. Usdin, N. C. House, and C. H. Freudenreich. Repeat instability during dna repair: Insights from model systems. *Crit Rev Biochem Mol Biol*, 50(2):142–67, 2015.

[105] S. Valadkhan. The role of snrnas in spliceosomal catalysis. *Progress in Molecular Biology and Translational Science*, 120:195–228, 2013.

[106] D. Van Vranken and G. A. Weiss. *Introduction to bioorganic chemistry and chemical biology*. Garland Science, 2018.

[107] T. J. Vos and U. Kothe. snr30/u17 small nucleolar ribonucleoprotein: a critical player during ribosome biogenesis. *Cells*, 9(10):2195, 2020.

[108] R.-W. Yao, Y. Wang, and L.-L. Chen. Cellular functions of long noncoding rnas. *Nature cell biology*, 21(5):542–551, 2019.

[109] D. Yu, H. Pendergraff, J. Liu, H. B. Kordasiewicz, D. W. Cleveland, E. E. Swayze, W. F. Lima, S. T. Crooke, T. P. Prakash, and D. R. Corey. Single-stranded rnas use rnai to potently and allele-selectively inhibit mutant huntingtin expression. *Cell*, 150(5):895–908, 2012.

[110] K. Zandi. *Computational Design and Experimental Validation of Functional Ribonucleic Acid Nanostructures.* Thesis, 2018.

[111] P. Zhu, J. Wu, Y. Wang, X. Zhu, T. Lu, B. Liu, L. He, B. Ye, S. Wang, and S. Meng. Lncgata6 maintains stemness of intestinal stem cells and promotes intestinal tumorigenesis. *Nature cell biology*, 20(10):1134–1144, 2018.

[112] P. Zhu, X. Zhu, J. Wu, L. He, T. Lu, Y. Wang, B. Liu, B. Ye, L. Sun, and D. Fan. Il-13 secreted by ilc2s promotes the self-renewal of intestinal stem cells through circular rna circpan3. *Nature immunology*, 20(2):183–194, 2019.

Appendix A

Appendix

A.1 Diversity score

We calculated a diversity score equal to the average phenotypic distance between each pair of viable individuals in the final population. For each function, this diversity is normalized by dividing it by the largest diversity measured for a run on that function. Normalization is performed because some functions have more potential for diversity than others.

A.2 Finding canonical devices

Simplicity search is a Truth-Seq-Er run with an additional objective: minimize the number of H-segments that bind to each other. In exploratory work (not shown), this new objective considered *all* possible H-segments interactions. However, this did not produce acceptable results. Results were significantly improved when only OBS and negator *self-interactions* were considered. It appears that these interactions are largely *non-essential* unlike the interactions between two *different* segments. These self-interactions are represented by a new fitness term, f^{SI}, which is equal to sum of the diagonal entries of the SPMMS.

In addition to the self-interactions fitness term, three other minor changes are made to Truth-Seq-Er. Instead of initializing a random population, simplicity search begins with the *final population* of a Truth-Seq-Er run. Furthermore, in order to compensate for a new fitness term being added, the two performance scores (f^{ON}, f^{OFF}) are averaged into a single *switch* score: f^{sw}. Finally, the viability nullification parameters are set more aggressively. The fitness parameters of simplicity search are shown in table A.1. For each 3-input function, the canonical device is the viable individual with the least number of segment interactions from the final simplified population.

Number of generations	200
Population size	300
Mutation rate	4
Objective scores	Switch, self-interactions, novelty
Viability nullification?	Yes
(Start, End) threshold values	(0.9, 0.95)
Threshold Breakpoint (generation, value)	N/A
Novelty neighborhood size	30

Table A.1: Simplicity search fitness parameters.

A.3 Canonical devices for functions f-7-3I, f-25-3I, f-27-3I, f-129-3I, f-135-3I

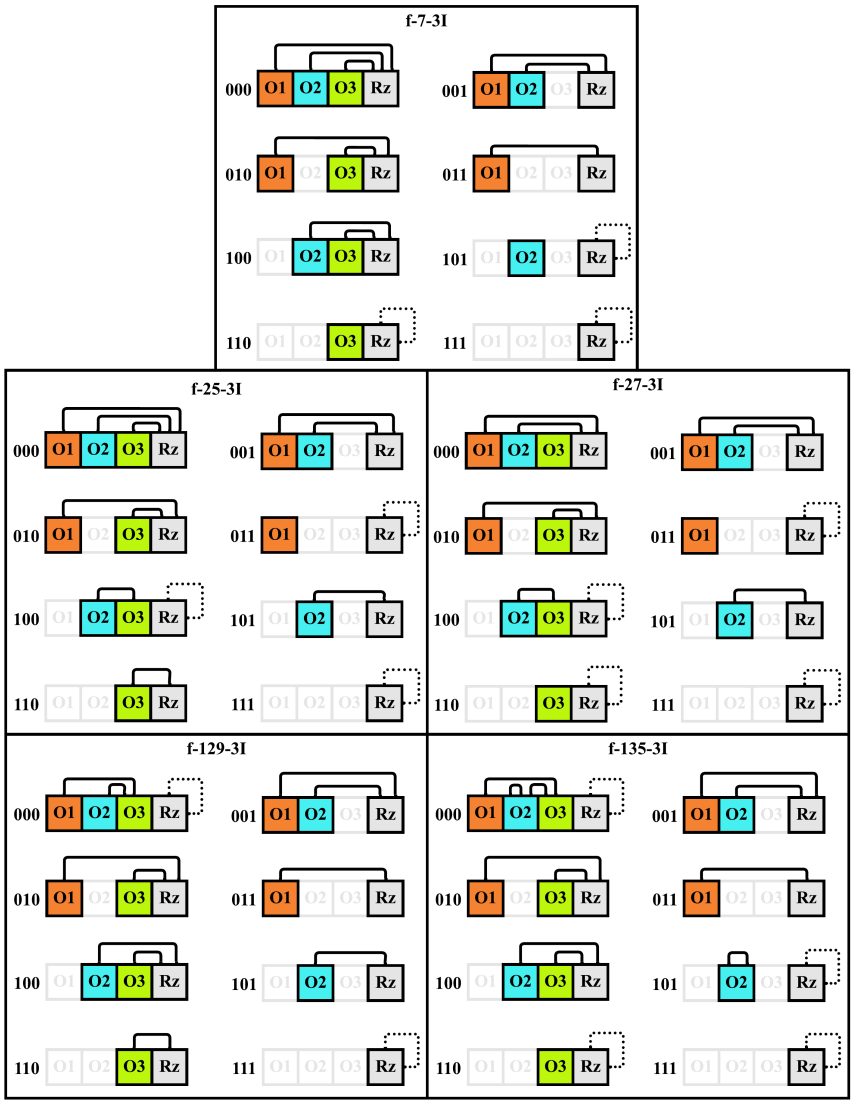


Figure A.1: Segment structures of canonical devices.

A.4 Simulation

In the main text, we generated segment structures from a given mechanism graph by hand. Here we present an algorithm called *simulation* which automates the process. It uses exhaustive search to calculate the stability of all candidate structures and it returns the most stable one. The pseudocode for simulation is shown in Algorithm 1.

Algorithm 1: Simulate mechanism graph

Inputs: (E, V, B) // The edges, nodes, and bundles of the mechanism graph
        W // A dictionary storing the weight assigned to each edge
        V_O // The set of occupied nodes (OBSs bound to their respective input)

// Generate the set of candidate structures
$edges \leftarrow$ the set of all $e \in E$ such that e is not incident on any $node \in V_O$
$provisional\_candidates \leftarrow$ powerset($edges$) // All possible subsets of $edges$

// Remove invalid structures from the set of candidates
$candidates \leftarrow$ copy $provisional\_candidates$
for each c in $provisional\_candidates$ **do**
    for each $node$ in V **do**
        $node\_bundles \leftarrow \emptyset$
        $node\_edges \leftarrow$ the set of all edges $\in edges$ that are incident on $node$
        // If multiple edges are incident on a node, they must share a bundle
        // If at least one node violates this condition, the structure is invalid
        if $|node\_edges| > 1$ **then**
            for each $edge$ in $node\_edges$ **do**
                $edge\_bundles \leftarrow$ all bundles that include edge
                $node\_bundles \leftarrow node\_bundles \cup \{edge\_bundles\}$
            end for
            $common\_bundles \leftarrow \bigcap node\_bundles$
            if $|common\_bundles| == 0$ **then**
                remove c from $candidates$
                break
            end if
        end if
    end for
end for

// Calculate the stability of each candidate
$stabilities \leftarrow$ empty list
for each c in $candidates$ **do**
    $stability = \sum_{e \in c} W[e]$ // Sum the edge weights
    append $stability$ to $stabilities$
end for

// Select the candidate with the highest stability
$indices \leftarrow$ argsort $stabilities$ (by descending value)
$L \leftarrow$ length of $indices$
$index\_1 \leftarrow indices[0]$ //index of candidate with the highest stability
$max\_stability\_structure \leftarrow candidates[index\_1]$

// Determine whether the highest stability structure is unique
// This is necessary for mechanism extraction (discussed in Section A.5)
$bool\_degenerate \leftarrow 0$
// If there was more than one candidate
if $L > 1$ **then**
    $index\_2 \leftarrow indices[1]$ //Index of the candidate with the second highest stability
    if $stabilities[index\_1] == stabilities[index\_2]$ **then**
        $bool\_degenerate \leftarrow 1$ // The highest stability structure is not unique
    end if
end if
return ($max\_stability\_structure$, $stabilities[index\_1]$, $bool\_degenerate$)

A.5 Mechanism extraction

Mechanism extraction is an algorithm that takes as input a list of *target* segment structures and it outputs a mechanism that can *reproduce* these structures *without error* when it is simulated. Mechanism extraction consists of two steps. First, the target segment structures are *merged* into an unweighted graph. Then, the edges of this graph are *assigned weights* using an iterative procedure similar to simulated annealing. A full example of mechanism extraction is shown in Figure A.2.

A.5.1 Merging

The *unweighted* mechanism graph represents TO segments that interact with each other in at least one state. The unweighted mechanism graph is equal to the union of the target TO structures. In addition, certain edges of the unweighted mechanism may be grouped together into sets called bundles. Edges are part of the same bundle if they are incident on the *same node* in the *same segment structure*. Merging is detailed in Algorithm 3. An example unweighted mechanism graph is shown in Figure A.2 a).

Algorithm 2: Merge target segment structures

Inputs: $segment\_structures$ // There is one segment structure per state

$E \leftarrow \emptyset$
$V \leftarrow \emptyset$
$provisional\_bundles \leftarrow \emptyset$
for each $structure$ in $segment\_structures$ **do**
    $(e, v) \leftarrow$ (edges, nodes) of $structure$
    //Take the union of the segment structures
    $E \leftarrow E \cup e$
    $V \leftarrow V \cup v$
    for each $node$ in v **do**
        $node\_edges \leftarrow$ the set of all edges in e that are incident on $node$
        if $|node\_edges| > 1$ **then**
            $bundle \leftarrow node\_edges$
            $provisional\_bundles \leftarrow provisional\_bundles \cup bundle$
        end if
    end for
end for

// Discard any bundles that are strict subsets of other bundles
$bundles \leftarrow$ copy $provisional\_bundles$
for each $bundle\_i$ in $provisional\_bundles$ **do**
    for each $bundle\_j$ in $provisional\_bundles$ **do**
        if $bundle\_i \subset bundle\_j$ **then**
            remove $bundle\_i$ from $bundles$
            break
        end if
    end for
end for
return $(E, V, bundles)$ // Unweighted mechanism graph

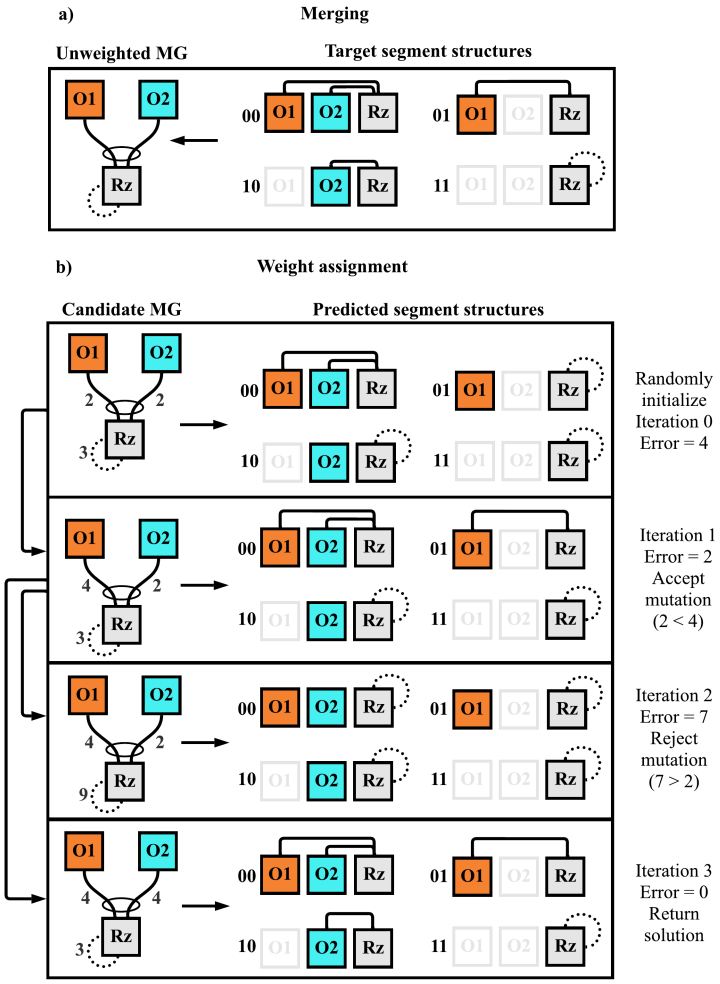


Figure A.2: Mechanism extraction example. a) The target segment structures are merged into an unweighted mechanism graph (UMG). The UMG has the same nodes as the target segment structures. Its edges are equal to the union of the edge sets of the target segment structures. Since O1 and O2 are both incident on Rz in the same state (00), they are grouped into a bundle. b) The UMG is assigned a set of random weights. Each iteration, the mechanism graph is simulated with its current weights. An error is calculated based on how dissimilar the target and predicted segment structures are. Then, a random weight of the mechanism graph is mutated. If this mutation results an error reduction, it is always accepted. Otherwise, it is only accepted with a certain probability. If the mutation is rejected, a new one is applied to the parent graph. If the target and predicted structures match, the current candidate mechanism graph is returned as the solution. List of abbreviations: O_n (OBS n), Rz (ribozyme).

A.5.2 Weight assignment

Once the unweighted mechanism graph has been obtained, weights are assigned to it. In general, mechanism graphs having identical edges, nodes, and bundles, but different weights, will produce different segment structures when simulated. The *weight assignment* algorithm searches for weights such that the mechanism reproduces the target segment structures. Each edge of the unweighted mechanism is initially assigned a *random* weight, and these weights are *mutated* through an iterative process until a solution is found. The weights at iteration i are denoted by $w(i)$. Each iteration, the current mechanism graph is simulated to generate a list of *predicted* segment structures. The *similarity* between the predicted and target segment structures is assessed, and an error is calculated. The higher the similarity, the lower the error. If the error of $w(i)$ is smaller than or equal to the error of $w(i-1)$ (i.e. the current iteration is an improvement over the previous one), then $w(i)$ is *always accepted*. Otherwise, $w(i)$ is only accepted with a certain *probability*. As explained in the main text, the segment structure of each state is the candidate structure with the highest stability. Note that for certain weight assignments, multiple candidate structures of a state may have maximum stability. In this case, the candidate mechanism graph is considered *invalid* and it is assigned a very high error as a penalty. The reason for this is that we want the segment structures generated by simulation to be non-ambiguous. Weight assignment is detailed in Algorithms 4 and 5, and is illustrated in Figure A.2 b). Note that the graphs generated by mechanism extraction are not unique. Two mechanism graphs with different weights and/or bundles can generate the same segment structures when simulated.

A.5.3 Experimental setup

We performed mechanism extraction for each 3-input NPN function. For each function, the input to the mechanism extraction algorithm was the segment structures of that function's canonical ribogate. Before running the algorihtm, we made slight changes to the second segment structures of two functions. For f-135-3I, we removed the OBS2-OBS2 self-loop in states 000 and 101. For f-27-3I, we added an edge an OBS3-Rz edge in state 000. None of these modifications affect the output state of the ribogate and they result in cleaner mechanism graphs.

A.5.4 Manual post-processing

The mechanism extraction algorithm does not natively handle partial bundles. Instead, some graphs will have partially overlapping bundles. In this step, we manually remove these bundles and replace them with a partial bundle that allows a maximum of two concurrent edges. Specifically, for f-127 and f-135, we remove the {OBS1-RZ, OBS2-Rz}, {OBS1-Rz, OBS3-Rz}, and {OBS2=Rz, OBS3-Rz} bundles and replace them with a {OBS1-Rz, OBS2-Rz, OBS3-Rz} (max 2) partial bundle.

Algorithm 3: Assign mechanism weights

Inputs: (E, V, B) // The edges, nodes, and bundles of the mechanism graph
        $target\_structures$ // The segment structures we wish for the mechanism graph to reproduce when simulated

// Score a random initial weight assignment
$min\_weight \leftarrow 1$
$max\_weight \leftarrow 8$
$prev\_weights \leftarrow$ empty dictionary
for each $edge$ in E **do**
    $w \leftarrow$ random integer between min_weight and max_weight
    $prev\_weights[edge] \leftarrow w$
end for
$prev\_error \leftarrow$ $score\_mechanism\_graph(E, V, B, prev\_weights, target\_structures)$

// Iterate until a solution is found or the max # of iterations is exceeded
$num\_iterations \leftarrow 2000$
for i from 0 to $num\_iterations - 1$ **do**
    $weights \leftarrow prev\_weights$
    // Mutate weight
    $edge \leftarrow$ random edge from E
    $weights[edge] \leftarrow$ random integer between min_weight and max_weight
    // Score mechanism graph
    $error \leftarrow score\_mechanism\_graph(E, V, B, weights, target\_structures)$
    // Accept or reject mutation
    if $error == 0$ **then**
        return $(E, V, B, weights)$ // Return the complete mechanism graph
    else if $error <= prev\_error$ **then**
        $prev\_error \leftarrow error$
        $prev\_weights \leftarrow weights$
    else
        $jump\_probability \leftarrow exp(prev\_error - error)$
        $jump\_sample \leftarrow$ random floating point number between 0 and 1
        if $jump\_probability > jump\_sample$ **then**
            $prev\_error \leftarrow error$
            $prev\_weights \leftarrow weights$
        end if
end for
return NULL // No solution

Algorithm 4: Score mechanism graph

Inputs: (E, V, B) // The edges, nodes, and bundles of the mechanism graph
        W // A dictionary storing the weight assigned to each edge
        $target\_structures$ // The segment structures we wish for the mechanism graph to reproduce when simulated

$num\_states \leftarrow$ length of $target\_structures$
$occupied\_nodes\_all \leftarrow$ the set of occupied nodes for each state
$error \leftarrow 0$
for i from 0 to num_states - 1 **do**
    $occupied\_nodes \leftarrow occupied\_nodes\_all[i]$
    $predicted\_structure, stability, bool\_degenerate \leftarrow simulate\_mechanism\_graph(E, V, B, W, occupied\_nodes)$
    if $bool\_degenerate == 1$ **then**
        // Severely penalize mechanism graphs that generate non-unique structures
        $error \leftarrow 1000$
        break
    else
        $E_p \leftarrow$ edges of predicted_structure
        $E_t \leftarrow$ edges of target_structures[i]
        // The error is equal to the # of edges that are in the predicted,
        // but not the target structure (and vice versa)
        $state\_error \leftarrow |(E_p \cup E_t) \backslash (E_p \cap E_t)|$
        $error \leftarrow error + state\_error$
    end if
end for
return $error$