A NOVEL REDUCTION OF RNA DESIGN TO SAT

Apoorv Malik , Ameyassh Nagarajan

Oregon State University
Crovallis, Oregon
{malikap, ameyass}@oregonstate.edu

ABSTRACT

Ribonucleic acid (RNA) sequences consist of nucleotide bases: Adenine (A), Cytosine (C), Guanine (G), and Uracil (U). These bases form sequences that can fold into multiple structures. A sequence is considered a design of a given target structure, if it uniquely folds into that structure with the highest number of base pairings. The RNA design problem, which seeks to find such a sequence for a given target structure, is NP-complete, as shown by Edouard et al. [1]. This paper presents a novel method to reduce any RNA design instance in the simple energy model to a SAT formula in polynomial time, a reduction not previously demonstrated in the literature. Our work establishes this polynomial-time reduction from RNA design to SAT, providing a new approach to solving the RNA design problem.

***K*eywords** Computational Complexity · Computational Biology · RNA Design

1 Introduction

RNA, or ribonucleic acid, acts as a messenger between DNA and proteins. It is made up of four bases: Adenine (A), Cytosine (C), Guanine (G), and Uracil (U). These bases pair specifically (A with U, and G with C) to form secondary structures.

RNA folding is the process by which an RNA sequence $\mathbf{x}$ adapts a two-dimensional secondary structure $\mathbf{y}$, based on the pairing of its bases. This process is driven by the formation of hydrogen bonds between complementary bases (A-U and G-C). The goal of RNA folding is to predict the structure $\mathbf{y}$ given a sequence $\mathbf{x}$. This is computationally feasible and can be done in polynomial time using dynamic programming algorithms.

RNA design, on the other hand, is the inverse problem of RNA folding. It involves finding a sequence $\mathbf{x}$ that will **uniquely** fold into a specific target structure $\mathbf{y}$ with maximal base pairings. This is known as the inverse folding problem. The challenge here is to ensure that the designed sequence $\mathbf{x}$ uniquely folds into the target structure $\mathbf{y}$ and not into any other structure with the same number of base pair. This problem is computationally difficult and is known to be NP-complete.

Solving the RNA design problem is important for several reasons:

- **Vaccine Development**: RNA vaccines, such as those developed for COVID-19, use RNA sequences to produce proteins that stimulate an immune response. Efficiently designing these sequences is essential for rapid vaccine development.
- **Gene Therapy**: RNA sequences can be designed to fix or replace faulty genes, offering potential treatments for genetic disorders.
- **Biosensing**: RNA molecules can act as sensors to detect pathogens or environmental changes, useful in diagnostics and monitoring.

Despite its importance, the RNA design problem is challenging because of the complexity of the folding process and the vast number of possible sequences. Efficiently solving this problem could lead to major advancements in the fields mentioned above.

From a computational perspective, the RNA design problem is highly relevant to areas such as algorithm design, computational complexity, and bioinformatics. The problem requires innovative approaches to manage the computational challenges and achieve practical solutions.

Edouard et al. [1] introduced a method to reduce any SAT formula to an RNA design problem in polynomial time, proving that the RNA design problem is NP-complete. However, there has been no research showing the reverse reduction. In this work, we present a novel method to reduce any RNA design instance to a SAT formula in polynomial time. This method leverages efficient SAT solvers to find potential solutions to a given RNA design instance. Our approach fills a crucial gap in the existing research and offers new tools for addressing this challenging problem.

In this paper, we demonstrate that RNA Design $\leq_p$ SAT, meaning that the RNA design problem can be reduced to a SAT problem in polynomial time. Specifically, we focus on the simplest version of the RNA design problem which uses the simplest energy model, where the energy of an RNA structure is determined solely by the number of base pairs (without considering more complex interactions such as stacking or loop penalties). In this version, more base pairing results in lower energy structures, leading to more optimal and stable configurations. Conversely, fewer base pairs result in higher energy and less stable structures.

By reducing RNA design to SAT, we can leverage powerful SAT solvers to find valid RNA sequences $\mathbf{x}$ that fold into a given target structure $\mathbf{y}$.

2 Background and preliminaries

2.1 Understanding RNA Sequences & Structures

An RNA secondary structure $\mathbf{y}$ is represented as a string over the alphabet $\Sigma = \{$ `'('`, `')'`, `'.'` $\}$, and an RNA sequence $\mathbf{x}$ is represented over the alphabet $\Delta = \{$`'A','U','G','C'`$\}$.

The dot-bracket notation is a common way to represent RNA secondary structures. In this notation:

- An opening parenthesis '(' at position i indicates that the nucleotide at position i is paired with a nucleotide at a later position j.
- A closing parenthesis ')' at position j indicates that the nucleotide at position j is paired with an earlier position i.
- A dot '.' indicates that the nucleotide at that position is unpaired.

For example, the dot-bracket notation $\mathbf{y} =$ `..((..))..` represents an RNA structure where the nucleotides at positions 3 and 8, and positions 4 and 7 are paired, while the nucleotides at positions 1, 2, 5, 6, 9, and 10 are unpaired. This can be visualized as:

.	.	(	(	.	.	)	)	.	.
1	2	3	4	5	6	7	8	9	10

In this notation, pairs are indicated by matching parentheses, and each pair (i, j) corresponds to the positions of the matching '(' and ')' in the string. For instance, in the above example, the pairs are $(3, 8)$ and $(4, 7)$.

An RNA sequence $\mathbf{x}$, represented over the alphabet $\Delta = \{$`'A','U','G','C'`$\}$, specifies the sequence of nucleotides that make up the RNA molecule. For example, $\mathbf{x} =$ `AUGCGUAAGC` is a sequence of length 10 where:

- 'A' denotes Adenine,
- 'U' denotes Uracil,
- 'G' denotes Guanine,
- 'C' denotes Cytosine.

2.1.1 Pseudoknots in RNA Structures

In RNA secondary structures, pseudoknots (crossing base pairs) are not allowed. Let $\mathbf{y}$ be any valid RNA secondary structure. A pseudoknot occurs when two pairs $(i, j) \in pairs(\mathbf{y})$ and $(i', j') \in pairs(\mathbf{y})$ cross each other. This crossing can be represented mathematically as:

$$i < i' < j < j' \quad \text{or} \quad i' < i < j' < j$$

For $\mathbf{y}$ to be a valid structure, any two pairs $(i, j) \in \text{pairs}(\mathbf{y})$ and $(p, q) \in pairs(\mathbf{y})$ must satisfy the non-crossing property of pairs. This non crossing property can be represented mathematically as:

$$i < p < q < j \quad \text{or} \quad p < i < j < q$$

When pairs in our sequence satisfy the above constraint, they are called nested pairings. These are the types of pairings allowed in the RNA design problem.

2.1.2 Simple Energy Model using Watson-Crick Pairs

We focus on the simplest energy model, where the energy of an RNA structure is determined solely by the number of base pairs, without considering complex interactions such as stacking or loop penalties. According to the Watson-Crick model:

- **Adenine (A)** pairs with **Uracil (U)**
- **Guanine (G)** pairs with **Cytosine (C)**

In this model, a higher number of base pairs results in lower energy and more optimal structures, while a lower number of base pairs leads to higher energy and less optimal structures. This approach simplifies the RNA design and RNA folding problems, making them more manageable to solve computationally.

2.2 Compatible Sequences

A sequence $\mathbf{x}$ is compatible with a secondary structure $\mathbf{y}$ if it can form valid base pairs under the Watson-Crick model. Specifically, this means that for each base pair position $(\mathbf{y}_i, \mathbf{y}_j)$ in the structure $\mathbf{y}$, the corresponding nucleotides in $\mathbf{x}$ (i.e., $\mathbf{x}_i$ and $\mathbf{x}_j$) must form a valid Watson-Crick pair (A-U or G-C).

2.3 Design Sequences

A sequence $\mathbf{x}$ is considered a design for a secondary structure $\mathbf{y}$ if it is the **unique sequence** that folds into $\mathbf{y}$ with the maximum number of base pairings. This means that among all sequences that can fold into $\mathbf{y}$, $\mathbf{x}$ achieves the highest number of valid base pairs, and no other sequence can achieve the same number of base pairings. If multiple sequences can achieve this maximum, none of them are considered design sequences. It is interesting to note that for a given structure $\mathbf{y}$, all design sequences are compatible, but not all compatible sequences qualify as designs!

2.4 Folding

Folding refers to the process by which an RNA sequence $\mathbf{x}$ adopts a specific secondary structure $\mathbf{y}$ through the formation of base pairs. This process is driven by the Watson-Crick base pairing rules, where A pairs with U and G pairs with C. The goal in RNA design is to find a sequence that folds into a desired structure $\mathbf{y}$ with maximum stability.

2.5 Formal Definitions

Let $\Sigma =$ `{ '(', ')', '.' }` be the set of symbols representing paired and unpaired nucleotides in a secondary structure $\mathbf{y}$ ($\mathbf{y}_i \in \Sigma$).
$\Delta =$ `{'A','U','G','C'}` be the set of symbols representening nucleotides in a sequence $\mathbf{x}(\mathbf{x}_i \in \Delta)$

1. **Structure** $\mathbf{y}$: A string $\mathbf{y} \in \Sigma^*$ representing the secondary structure.
2. **Sequence** $\mathbf{x}$: A string $\mathbf{x}$ over the nucleotide alphabet $\{A, U, G, C\}$.
3. **Compatibility**: A sequence $\mathbf{x}$ is compatible with a structure $\mathbf{y}$ if $\mathbf{x}$ can fold into $\mathbf{y}$ forming valid Watson-Crick base pairs.
4. **Design**: A compatible sequence $\mathbf{x}$ is a design if it is the unique sequence achieving the maximum number of base pairings for the structure $\mathbf{y}$.

2.6 Example of RNA Design

Consider an RNA secondary structure represented by the base pairings:

```
().()
```

In this notation, parentheses represent paired bases and dots represent unpaired bases.

Let's examine two compatible sequences for this structure:

Example 1: Sequence is a Design Consider the sequence `GCACG`. Below are all the possible structures compatible with `GCACG`:

1 ().()
2 ()...
3 ...()
4 (..).
5 .(..)
6 ....

This sequence can optimally fold into a unique structure `().()`, resulting in the maximal base pairings of `G-C` and `C-G`. Hence, `GCACG` is a **Design** for the given structure.

Example 2: Sequence is not a Design Consider the sequence `GCAGC`. The possible base pairings are:

$$C - G$$
$$G - C$$

However, this sequence can fold into multiple structures:

1 `().()` with base pairings `G-C` and `C-G`
2 `((.))` with the same base pairings

Since `GCAGC` does not uniquely maximize the base pairings and can form multiple structures with the same number of (maximum) pairings, it is not a **Design** for the given structure.

2.7 SAT Problem

The Boolean Satisfiability Problem (SAT) seeks an assignment of variables such that the given formula ϕ evaluates to true. The formula is expressed in conjunctive normal form (CNF), consisting of:

- A conjunction (AND) of clauses.
- Each clause is a disjunction (OR) of literals.
- A literal is a variable or its negation.

Mathematically, a SAT formula is:

$$\phi = (l_{1,1} \vee l_{1,2} \vee \ldots \vee l_{1,k}) \wedge (l_{2,1} \vee l_{2,2} \vee \ldots \vee l_{2,k}) \wedge \ldots \wedge (l_{m,1} \vee l_{m,2} \vee \ldots \vee l_{m,k})$$

Example of a SAT Instance

Consider:

$$\phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (x_2 \vee \neg x_3 \vee \neg x_4)$$

This formula is satisfiable with the assignment $x_1 = \text{true}, x_2 = \text{false}, x_3 = \text{true}, x_4 = \text{false}$.

SAT is one of the first known NP-complete problems. Formally:

$$L = \{s \mid \exists w,\ |w| \leq p(|x|) \text{ and } M(s, w) = 1\}$$

where:

- s is an instance of the problem,
- w is a witness,
- p is a polynomial function,
- $M(s, w)$ returns 1 if w is a valid witness proving s is a yes-instance of L.

Definition of NP-Complete

A decision problem L is NP-complete if:

- $L \in$ NP.
- Every problem in NP can be reduced to L in polynomial time. Formally, for every problem $L' \in$ NP, there exists a polynomial-time computable function f such that:

$$x \in L' \iff f(x) \in L$$

Polynomial-Time Reduction

A problem L_1 is polynomial-time reducible to a problem L_2 (denoted $L_1 \leq_p L_2$) if there exists a polynomial-time computable function $f : \Sigma^* \to \Sigma^*$ such that for all $x \in \Sigma^*$:

$$x \in L_1 \iff f(x) \in L_2$$

This type of reduction is known as a Karp reduction.

NP-Completeness of SAT

SAT is a canonical NP-complete problem because:

- **SAT is in NP**: Given a SAT formula ϕ, we can verify in polynomial time if an assignment satisfies ϕ.
- **NP-hardness of SAT**: For every problem $L \in$ NP, there is a polynomial-time reduction to SAT. Thus, any NP problem can be transformed into a SAT instance in polynomial time.

Formally, if L is any problem in NP, then there exists a polynomial-time computable function f such that:

$$x \in L \iff f(x) \in \text{SAT}$$

3 Reducing RNA to SAT

Edouard et al. [1] introduced a novel method to reduce any instance of a SAT formula to an instance of RNA design in polynomial time, thereby establishing the NP-completeness of the RNA design problem.

We now present a novel method to reduce any RNA design instance to a SAT formula in polynomial time. The previous work by Edouard et al. [1] did not demonstrate this reduction direction, and to the best of our knowledge, there is no existing literature or academic paper that shows this reduction. Our work provides a novel finding by establishing a polynomial-time reduction from RNA design to SAT.

We developed a verifier that verifies a design $\mathbf{x}$ for a given structure $\mathbf{y}$ by modifying the Nussinov algorithm [2] to compute the k most optimal structures of $\mathbf{x}$ (where optimality is defined as maximizing the number of paired positions in $\mathbf{x}$). We call this verifier k-best folding, and it runs in polynomial time $O(n^3 + n^2 k \log k)$.

Given the target structure $\mathbf{y}$ and the design sequence $\mathbf{x}$ (output from the SAT solver), the k-best folding verifier checks if $\mathbf{x}$ uniquely folds into $\mathbf{y}$ with the maximum number of base pairings in polynomial time. If the k-best folding verifier for sequence $\mathbf{x}$ returns multiple structures with the maximum number of base pairings, we conclude that the design is invalid. However, if it returns only one structure $\mathbf{y}^*$ (such that $\mathbf{y}^* = \mathbf{y}$) with the maximum number of base pairings, then the design is valid, and the output of the SAT solver is correct.

By transforming RNA design into a SAT problem, we can utilize efficient SAT solvers to explore potential solutions, thus providing a robust computational approach to address the RNA design problem.

3.1 Reduction from RNA Design to SAT

The reduction from the RNA design problem to a SAT formula involves encoding the constraints of the RNA design into a Conjunctive Normal Form (CNF) formula. This section outlines the process of creating the SAT formula.

3.1.1 Variables

We define variables to represent the possible bases (or nucleotides) at each position i in the RNA sequence $\mathbf{x}$:

$$A_i, U_i, G_i, C_i$$

where A_i is true if position i is Adenine, U_i is true if position i is Uracil, G_i is true if position i is Guanine, and C_i is true if position i is Cytosine.

3.1.2 Single Nucleotide Positional Constraint

To ensure each position in the sequence $\mathbf{x}$ has exactly one base/nucleotide, we add the following clauses:

- At least one base:
$$A_i \vee U_i \vee G_i \vee C_i$$
- At most one base:
$$\neg A_i \vee \neg U_i, \quad \neg A_i \vee \neg G_i, \quad \neg A_i \vee \neg C_i$$
$$\neg U_i \vee \neg G_i, \quad \neg U_i \vee \neg C_i, \quad \neg G_i \vee \neg C_i$$

3.1.3 Target Structure Pairing Constraint

In the target structure $\mathbf{y}$, for each pair $(i, j) \in pairs(\mathbf{y})$, we add Watson-Crick base pairing constraints to the sequence positions x_i and x_j to ensure that the designed sequence x can form a valid base pair at (i, j). To ensure this, we add the following clauses:

$$\neg A_i \vee U_j, \quad \neg U_i \vee A_j, \quad \neg G_i \vee C_j, \quad \neg C_i \vee G_j$$

3.1.4 Pairing Constraints for Undesirable Pairs

In RNA secondary structures, pseudoknots (crossing base pairs) are not allowed. This means that if two pairs (i, j) and (i', j') exist in the structure $\mathbf{y}$ such that $i < i' < j < j'$ or $i' < i < j' < j$, they would create a crossing pattern, resulting in a pseudoknot. This would render the structure $\mathbf{y}$ invalid. Therefore, if (i, j) and (i', j') cross, they must be mutually exclusive, meaning at most one of them can exist in $\mathbf{y}$.

To ensure that the designed sequence $\mathbf{x}$ does not form any undesirable pair $(i', j') \notin \mathbf{y}$, we add constraints that either prevent the formation of (i', j') or ensure that if (i', j') does form, it leads to a sub-optimal structure with fewer base pairs. For each base pair (i', j') that is not present in the structure $\mathbf{y}$, i.e., $(i', j') \notin pairs(\mathbf{y})$, we add one of the following constraints in the sequence design:

- **Ensure Breaking of Desirable Pairs:** If the undesirable pairing of positions (i', j') occurs in $\mathbf{y}$, it should result in breaking at least two desirable base pairs $(p, q) \in pairs(\mathbf{y})$ and $(i, j) \in pairs(\mathbf{y})$, leading to a sub-optimal structure with fewer base pairs. This involves:
  - Calculating the set of broken nucleotide positions for each undesirable pair (i', j') by identifying other desirable pairs $(i, j) \in \mathbf{y}$ that would be disrupted if (i', j') were to pair.
  - Ensuring that the formation of an undesirable pair (i', j') would create a pseudoknot, thereby disrupting/breaking at least two desirable pairs (i, j) and (p, q).
  - Ensuring that the broken positions $\{i, j, p, q\}$ do not pair with each other (for example, (i, p) or (q, j)). Forming new pairs would restore the number of paired positions and allow the sequence $\mathbf{x}$ to fold into a new rival structure $\mathbf{y}'$ with the same maximal number of base pairs as $\mathbf{y}$, rendering the design sequence $\mathbf{x}$ invalid. This non-pairing constraint can be achieved by adding the following SAT clauses:
$$\neg A_i \vee \neg U_p, \quad \neg U_i \vee \neg A_p, \quad \neg G_i \vee \neg C_p, \quad \neg C_i \vee \neg G_p$$
$$\neg A_q \vee \neg U_j, \quad \neg U_q \vee \neg A_j, \quad \neg G_q \vee \neg C_j, \quad \neg C_q \vee \neg G_j$$
- **Prevent Formation of (i', j') Base Pair:** If the undesirable pairing of positions (i', j') in $\mathbf{y}$ does not result in the breaking of at least two desirable pairs present in $\mathbf{y}$, then we do not allow $\mathbf{x}_{i'}$ and $\mathbf{x}_{j'}$ to pair by adding the following constraint:
$$(\mathbf{x}_{i'}, \mathbf{x}_{j'}) \notin \{(A, U), (U, A), (C, G), (G, C)\}$$
This ensures that the bases at positions i' and j' in the design sequence $\mathbf{x}$ do not form a valid Watson-Crick pair, thereby preventing the formation of the undesired pair (i', j') in $\mathbf{y}$. This constraint can be encoded by the following SAT clause:
$$\neg A_{i'} \vee \neg U_{j'}, \quad \neg U_{i'} \vee \neg A_{j'}, \quad \neg G_{i'} \vee \neg C_{j'}, \quad \neg C_{i'} \vee \neg G_{j'}$$

By encoding these constraints into a CNF formula, we create a SAT instance that captures the requirements of the RNA design problem. Solving this SAT instance will yield a sequence $\mathbf{x}$ that folds into the desired structure $\mathbf{y}$ with the maximum number of valid base pairings, ensuring no alternative structure can achieve the same or higher number of pairings.

The number of variables in our SAT formula is $4n$, where n is the length of the structure. The number of clauses in our SAT formula is of $O(n^3)$.

The entire reduction from the RNA design problem to a SAT formula can be achieved in $O(n^3)$ time, where n is the length of the target structure $\mathbf{y}$. This is because there are n^2 possible position pairs (i, j), and each pair requires $O(n)$ time to add the necessary clauses representing different constraints. Therefore, the complexity of the reduction process is polynomial, making it feasible to convert the RNA design problem into a SAT instance efficiently.

This reduction from RNA design to SAT is a novel approach, as previous work has not demonstrated this direction of reduction. Our work fills this gap, providing a new perspective and methodology for analyzing the computational complexity of RNA design problems.

4 Experiments and Results

To validate our reduction, we generated $10{,}000$ random valid structures, each with a length between 10 and 100. Using the SAT solver, we identified designable structures and verified the designed sequences (output of our SAT solver) with our k-best folding verifier. The verifier confirmed that all sequences produced by the SAT solver were correct, indicating that each sequence $\mathbf{x}$ uniquely folded into its target structure $\mathbf{y}$ with maximal base pairing. We then randomly selected $1{,}000$ designable and $1{,}000$ undesignable structures, reran the SAT solver, and analyzed the results to assess the scalability of our SAT solver with increasing structure length. This evaluation provided insights into the performance of our SAT solver with respect to the size of the input structure.

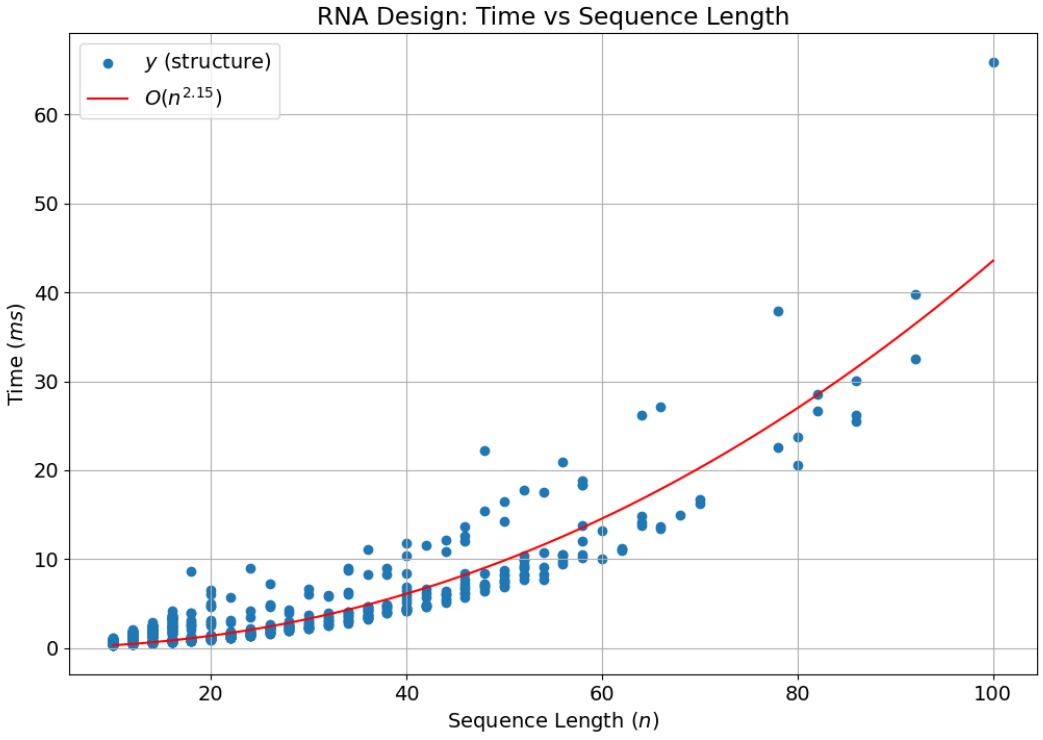

Figure 1: RNA Design: Time vs Sequence Length

As shown in Figure 1, the plot gives us insight on how the performance of the RNA Design SAT solution scales with respect to the the input size n, where n is the length of the input RNA structures. Given that n is the only dimension of this problem, the time complexity of our reduction is $\Theta(n^3)$ and the performance of the SAT solver scales quadratically by a factor of $O(n^2)$.

While our method reliably produces accurate designs for a given target structure, we identified a limitation in its ability to always find a design. Specifically, the algorithm may sometimes incorrectly determine that a target structure is undesignable when it is, in fact, designable. This issue arises particularly in cases where the RNA sequence has more than two unpaired positions. The constraints encoded in the SAT clauses can be overly restrictive, leading to this false negative result.

5 Conclusion and Future Work

In this paper, we introduced a novel method to reduce the RNA design problem to a SAT problem, enabling the use of SAT solvers to find optimal RNA sequences. We validated our approach through the following steps:

- Generated 10,000 random RNA structures.
- Used the SAT solver to find RNA sequences that fold uniquely and optimally into these structures.
- Verified the correctness of these designs (sequences) using the k-best folding verifier.
- Analyzed the scalability and efficiency of our solution to demonstrate its practical applicability.

Our experimental results show that the solution scales with $O(n^2)$ time complexity for structures of length ≤ 100, and containing at most 8 unpaired positions. This robust solution for the RNA design problem establishes a foundation for further research with more complex energy models.

Our solution exhibits 100% precision, as it accurately verifies designs when they are found. However, the recall is not 100%, meaning that it fails to identify design for some designable structures. Evaluating recall is challenging because it requires enumerating and verifying every possible compatible sequence for a given structure, a task that grows exponentially ($O(4^n)$) with structure length.

Future work will focus on several key areas to enhance our approach:

- **Refining Constraints:** We will work on refining the constraints encoded in the SAT clauses to improve recall while maintaining precision. This involves optimizing the reduction process to make it less restrictive and broadening its applicability.
- **Evaluation Methods:** We aim to develop more effective methods to evaluate the performance of the algorithm, ensuring a comprehensive assessment of its capabilities.
- **Complex RNA Structures:** We plan to explore applications in more complex RNA structures and related problems, extending the utility of our method beyond simple structures.
- **Advanced SAT Techniques:** Leveraging advanced SAT solving techniques, we aspire to enhance the efficiency and practical use of our approach in real-world scenarios. This will involve refining the reduction to accommodate a wider range of RNA structures and improving solver performance to handle larger datasets effectively.
- **Complex Energy Models:** We will explore more complex energy models that consider interactions beyond simple Watson-Crick base pairing, such as stacking energies, loop penalties, and other thermodynamic factors. This will allow us to design RNA sequences that are not only structurally accurate but also thermodynamically stable.

By addressing these areas, we aim to make our approach more robust, versatile, and applicable to a broader spectrum of RNA design challenges.

6 Code

The code for our solver and verifier can be accessed through the following link https://github.com/1998apoorvmalik/rna-design-sat-reduction. You are welcome to explore and run the solver and verifier by following the instructions in the README file. You may also clone the repository and execute the code on your local machine. We encourage you to experiment with the tool and provide feedback.

References

[1] Édouard Bonnet, Paweł Rzążewski, and Florian Sikora. Designing rna secondary structures is hard. *Journal of Computational Biology*, 27(3):302–316, 2020. PMID: 32160034.

[2] Ruth Nussinov, George Pieczenik, Jerrold R. Griggs, and Daniel J. Kleitman. Algorithms for loop matchings. *SIAM Journal on Applied Mathematics*, 35(1):68–82, 1978.