RNA Generative Modeling With Tree Search

1st Stephen Obonyo
LIASD
Paris 8 University
Paris, France
sobonyo@up8.edu

2nd Nicolas Jouandeau
LIASD
Paris 8 University
Paris, France
n@up8.edu

3rd Dickson Owuor
SCES
Strathmore University
Nairobi, Kenya
dowuor@strathmore.edu

***Abstract*—Ribonucleic acid (RNA) molecules are key in many biological processes. The ability to generate RNA sequences that fold according to a given target structure while satisfying constraints such as minimum free energy (MFE) and the GC content is important in many applications such as synthetic biology and drug design. In this work, we propose a novel method for designing RNA sequences that satisfy these constraints. Our method uses an RNA-based Language Model (LLM) to generate RNA sequences while guiding the RNA sequence generation process using Monte Carlo Tree Search (MCTS). The MCTS ensures that the RNA-based LLM sequence generation process leads to a valid RNA sequence that folds according to the target structure. Instead of performing random rollout during the simulation phase of the MCTS, we sample the next RNA sequence from the RNA-based LLM. By design, our method can control LLM issues such as hallucinations where the generated sequences are inconsistent with the training data, and generation of invalid sequences as only rollouts that lead to correct RNA design are considered. We show that our method, without user-defined RNA design rules, can generate valid RNA sequences that can fold accordingly while outperforming the existing models on 50% of the test datasets while also recording competitive results on the remaining test datasets.**

***Index Terms*—RNA Sequence Design, RNA Inverse Folding, Monte Carlo Tree Search, Tree search, Generative Modeling, Large Language Models, Deep Learning, Machine Learning.**

I. Introduction

RNA is a biological sequence present in the body of all living organisms. The sequence is responsible for key biological functions such as gene regulation and expression, protein translation, protein synthesis and genetic coding. An RNA sequence is composed of a set of bases e.g. Cytosine(C), Guanine (G), Uracil (U) and Adenine (A). According to [1], GC and AU can form a base pair due to special molecular bonding properties. These base pairs are commonly referred to as Watson-Crick pairs. In addition to the Watson-Crick pairs, UG can also form a base pair in some RNA sequences.

The distribution of the bases and base pairs in an RNA sequence can vary depending on the complexity of the loops and junctions. Despite the variation, however, the general distribution of the base is 93%, 1%, 5% and 1% for A, U, G and C respectively. This is in contrast to the base pairs is distributed as 60%, 33% and 7% for GC or CG, AU or UA and GU or UG respectively [2].

In RNA sequence design, also referred to as RNA Inverse Folding, the goal is to design an RNA sequence that folds according to a given target structure. While this is the primary underlying objective, other key constraints that need to be satisfied include: (i) sequence should have desirable minimum free energy (MFE) and, (ii) the expected GC content. The MFE is the minimum amount of energy required by the RNA sequence to fold into its most stable conformational structure. On the other hand, the GC content is the percentage of GC base pairs in an RNA sequence. These two constraints are key as they affect the stability of the RNA sequence. For instance, RNA sequences with lower MFE values and higher GC content values are considered more stable [3], [4].

In the worst-case scenario, RNA sequence design for most targets is considered an NP-hard problem [6]–[8]. The RNA design of simple sequence, however, can be solved in quadratic or polynomial time. [6], [7].

Despite RNA sequence design being a challenging problem, it has a wide range of applications in biology and medicine. For instance, the development of new drugs and vaccines is contingent on the development of effective RNA sequence design methods. Furthermore, RNA sequence design also has a wide range of applications in nanotechnology and synthetic biology [9] and the design of RNA sequences for self-assembly and nanostructures [10].

Based on the significance of the RNA sequence design, it has been one of the active research areas in computational biology. Several methods have been proposed to solve the RNA sequence design including (i) exploring the whole search space through dynamic programming methods [11], (ii) conducting a local search to find the solution from candidate solutions [12], (iii) using genetic and evolutionary algorithms to find the optima which is as close as possible to the global solution [13], [14], (iv) physics-based modelling [15]–[17], (v) constraint programming with GC and energy requirement constraints [18], (vi) tree search methods such as MCTS [19] and nested MCTS [2] to find the optimal solution by iteratively sampling the search space by selecting nodes based on the Upper Confidence Bound (UCB) then expanding the selected node and sampling the next base or base pair until the the terminal state is reached, (vii) combining tree search with Deep Learning model as policy or value function [20] and, (viii) Reinforcement Learning (RL) where the agent (model) learns by trial and error by interacting with the environment to find the optimal policy [21], [22].

Tree search, e.g MCTS, RL and recently LLM-based methods are state-of-the-art methods for solving complex RNA

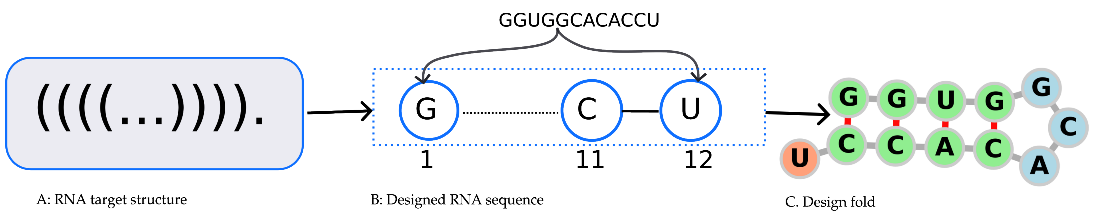

Fig. 1. *RNA Sequence Design Pipeline. (A) Target structure: secondary structure of the target encoded in Dot-bracket notation; (B) designed RNA sequence: the base pairs and bases are assigned to the design positions leading to the sequence* ***GGUGGCACACCU****. (C) Folding the designed RNA sequence: the sequence is folded and the base pair and base positions are compared to the target's matching parenthesis and dots respectively. The matching parenthesis in the target structure accepts base pairs GC/CG, AU/UA or GU/UG while the dots accept bases A, U, G or C. The base pairs in (C) are compared to the matching parenthesis in (A) and the bases are compared to the dots in (A). If the comparison leads to a hamming distance of 0, the RNA design (B) is considered successful.* Source: [5].

sequence design problems. We highlight some of the existing works in these three areas in the following paragraph.

In the first example, MCTS-RNA [19] uses classical MCTS with random policy rollout. After the selection of a node and expanding it according to the MCTS algorithm, a random policy is used to randomly sample the next base or base pair up to the terminal state. For the second example, [2], is an MCTS-based solution that uses nested MCTS with adaptive user-defined RNA design rules. In nested MCTS, the solutions from the lower-level MCTS are used to guide the search in the upper-level MCTS. For the third instance, [20] introduces a novel method of combining tree search (MCTS) with the DL model. Guiding search with ML/DL [23], [24] is a powerful optimization method for solving problems with large search spaces like RNA sequence design. The fourth instance, [21], is the RL-based method which uses policy gradient [25] to learn the optimal policy for RNA sequence design. In RL, a policy is a function that maps the state to the action e.g the policy can take an intermediate RNA sequence (state) and predict the next base or base pair (action). The policy gradient method learns the policy by following the gradient of the expected reward e.g. the hamming distance between the target structure and the predicted structure. [26] is an RNA sequence design that is based on the LLM model. In the work, the LLM learns by predicting the randomly masked bases and base pairs in the RNA sequence. This process is then followed by fine-tuning the LLM model on the RNA sequence design as a downstream task.

MCTS-RNA and nested MCTS are designed based on random policy rollout. Such policy design is prone to sub-optimality and can be computationally intensive in large action spaces [24]. On the other hand, RL-based methods such as LEARNA [21] are based on policy gradient and thus inherit the convergence and stability issues associated with the RL algorithms. LLM-based methods are prone to hallucinations [27], [28] and generating invalid sequences that do not fold according to the target structure. In our work, we leverage the tree search (MCTS) which guides the RNA-based LLM such that hallucination and generation of invalid sequences are controlled. Furthermore, while some methods such as dynamic programming, constraint programming and genetic algorithms have also recorded very competitive results, these algorithms are either computationally intensive, require user-defined rules or converge to the local optimum.

In natural language processing (NLP), LLM has significantly improved with the introduction of attention-based models [29], [30]. These methods are key in modeling long-range semantic and syntactic dependencies in sequences. Such relationships are key in solving downstream tasks such as machine translation, question answering and text summarization. LLM unlocked the potential of Generative Modeling (GM) in NLP and by extension in other domains such as computational biology e.g. protein inverse folding [31], [32], and RNA-based applications [33]–[37] and Retrosynthetic Planning [38]–[40].

LLMs can be trained on a large set of RNA sequences and learn the distribution of the bases and base pairs in the RNA sequences. Subsequently, the trained LLMs can be used to predict new sequences that are consistent with the training data (generative) [41], [42].

RNA-based LLMs, just like other LLMs, are prone to generating inconsistent sequences due to hallucinations [27], [28]. In addition, to this limitation, evaluating the quality of the generated sequences is hard. While several methods have been proposed to solve the two aforementioned issues, existing proposals are not universally applicable. For instance, in RNA sequence design, the generated sequences are evaluated according to their ability to fold according to the target structure, thus, classical LLMs evaluation methods such as perplexity, BLEU and ROUGE scores are not applicable.

Combining tree search with DL is an effective optimization method for solving non-deterministic problems [23]. In this work, we propose a novel method that combines two key methods: LLMs and MCTS. Our method involves generating RNA sequences using an RNA-based language model while guiding the sequence generation process using MCTS. By design, our method can control LLM issues such as hallucination and generation of invalid sequences as only the paths that lead to valid RNA design are considered.

Contribution. In this work, we propose a novel method for designing RNA sequences using an RNA-based language model. Our method uses MCTS to guide the RNA sequence generation process whereby the intermediate sequence in the tree is used as a prompt to the language model during the simulation phase. MCTS builds a search tree by adding a new node corresponding to a new base or base pair. The nodes in the path leading to the addition of a new node are selected using the Upper Confidence Bound (UCB) algorithm [43]. During the simulation phase of the MCTS, the algorithm samples the next base pair or base from the LLM up to the terminal state. The resulting sequence is then folded and the reward (hamming distance [44]) is computed between the target structure and the predicted structure. The reward is then backpropagated to the root node of the tree. The reward equation is shown in Equation 1. $\mathbb{I}$ is the indicator function that returns 1 if the condition is true and 0 otherwise while p_i and t_i are the tokens in the predicted and target structure respectively.

$$\mathcal{H}(p,t) = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(p_i \neq t_i) \qquad (1)$$

The rest of the Sections in this paper are organized as follows: Section II presents some of the definitions used in this work, Section III covers the RNA sequence design problem, Section IV describes our proposed method, Section V describes the experimental setup including the datasets used, Section VI covers the discussion of the result, and Section VII covers the related work, Section VIII covers the future work and Section IX concludes the paper.

II. Definitions

In this section, we present some of the definitions used in this work.

i) **Base and base pair position**: A base is a single nucleotide in an RNA sequence e.g **A, U, G, C**. A base pair is a pair of bases that bond together to form a base pair e.g. **AU, UA, GC, CG, GU, UG**. The base pair position is the position in the RNA sequence design which accepts a base pair while the base position is the position in the RNA sequence design which accepts a base.

ii) **Target and predicted structure**: An RNA target structure is denoted using Dot Bracket notation standard where the matching parenthesis corresponds to the positions that can accept a base pair while the dots correspond to the positions that can accept a base. For instance, **((((...)))).** is a valid target structure. The predicted structure is the structure (also in Dot Bracket notation) obtained from folding an RNA sequence from the design process e.g. **GGUGGCACACCU** RNA sequence can be folded using off-the-shelf tools such as the ViennaRNA package [45] to obtain the predicted structure such as **.(((...)))..**, which can be compared to the target structure.

iii) **Valid RNA design**: If the hamming distance between the target structure and the predicted structure is 0, the RNA design is considered successful. We refer to such a design as a valid RNA design. The hamming distance is computed using Equation 1.

iv) **Accuracy**: In the experiments section of this work we used the accuracy metric which is computed as $1-\mathcal{H}(p,t)$ to evaluate the validity of the RNA design. In the results section, the reported accuracy is scaled to range from 0 to 100.

v) **Evaluation Function**: This is the function that accepts the predicted structure and the target structure and computes the hamming distance or accuracy between them. The result corresponds to the quality of each MCTS rollout.

vi) **Rollout Policy**: A policy is a function that maps the state to the action. A rollout policy is a function that is used to select the next base or base pair in the MCTS simulation phase. In classical MCTS, the rollout policy is random e.g. the valid actions - bases or base pairs - are selected randomly. In our work, the rollout policy is the RNA-based language model.

vii) **RNA user-defined rules**: During the RNA sequence design, the user can define rules that guide the RNA sequence design process. For instance, the user can define rules that guide the selection of the next base or base pair with a custom or adaptive probability distribution in left and right junctions and loops. While in our work, we do not use user-defined rules, some state-of-the-art methods such as nested MCTS [2] use them.

viii) **Desing Position**: During the design process, any RNA design position that does not have a base or base pair is considered a design position.

III. RNA Sequence Design

There are three RNA sequence structures: (i) the primary structure, (ii) the secondary structure and (iii) the tertiary structure. The primary structure is the sequence of bases and base pairs e.g. **GGUGGCACACCU**. The secondary structure includes bases and base pairs encoded as matching parentheses and dots (Dot Bracket) respectively e.g. **((((...)))).** (from folding **GGUGGCACACCU**). The tertiary structure is the three-dimensional structure of the RNA sequence. In this work, we only work with the primary and secondary structures.

Designing RNA or RNA Inverse Folding involves assigning bases and base pairs to the design positions according to the target structure such that the resulting RNA sequence folds according to the target structure. Importantly, while any base pair {GC, CG, AU, UA, GU, UG} can be accepted in the base pair design position, not all bases will lead to a valid RNA design. The same applies to the base position. This behavior is due to the (i) molecular bonding properties associated with the base and base pairs, and (ii) the favorable stable conformational structure of the resulting RNA sequence.

While the primary objective of the RNA sequence design is to design an RNA sequence that folds according to the target structure, other constraints such as sequence with the minimum free energy (MFE) and GC content need to be satisfied. The GC content is the percentage of G/C base pairs in an RNA sequence. The MFE is the minimum amount of energy required by the RNA sequence to fold into its most stable conformational structure. There are no widely accepted ranges for the MFE and GC content. However, lower MFE values and higher GC content values are considered desirable as they result in more stable RNA sequences [3], [4].

IV. Methods

In this section, we describe our proposed method for designing RNA sequences. Our approach involves generating RNA sequences using an RNA-based language model while guiding the sequence generation process using MCTS. We also describe the design of the RNA-based language model, its integration with the MCTS and the method for programming the RNA sequences into sentences which are used to train the language model. A description of the dataset used in this work is also presented in this section.

A. Problem Formulation

RNA sequence design can be formulated as a sequential decision problem composed of state, action, transition function, reward and discount parameter in the form of a set $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}$. The state $\mathcal{S}$ is the all possible RNA sequences that can be generated with $\{A, U, G, C\}$, the action $\mathcal{A}$ is the set of all possible bases and base pairs that can be assigned to the RNA sequence e.g. $\{A, U, G, C, AU, UA, GC, CG, GU, UG\}$ - for base positions the valid actions are $\{A, U, G, C\}$ while for base pair positions it's $\{AU, UA, GC, CG, GU, UG\}$, the transition function $\mathcal{P}$ which adds a new base or base pair to the intermediate RNA sequence (the LLM) and the reward $\mathcal{R}$ is the hamming distance (see equation 1) between the target structure and the predicted structure. The discount parameter γ controls the significance of the future rewards. Here, the value of γ is set to 1.0 throughout the experiments.

B. Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search is an any-time search optimization algorithm that has been successfully applied to many problems in playing computer games, scheduling and RNA sequence design [46]. The algorithm is composed of four key phases: (i) selection: the best node to expand is selected using the UCB algorithm, (ii) expansion: the selected node is expanded by adding a new node to the tree, (iii) simulation: the algorithm randomly selects the valid actions up to the terminal state, (iv) backpropagation: the reward value is backpropagated from the terminal state to the root node of the tree. The reward value is then used to update the Q values of the nodes in the tree. Figure 2 shows the MCTS phases.

In this work, during the selection phase, the algorithm selects the best base pair or base based on the modified UCB equation 2 where $Q(s,\ b)$ is the Q value of the current node, s is the current node, b is the base pair or base, C is a constant set to $\sqrt{2}$, $P(b)$ is the prior probability of the base pair or base 9 (we use the general distributions described are described in Section I), and N_i and n_i are the number of times the parent node and the child node have been visited respectively. In expansion, the algorithm expands the current node by adding a new node to the tree. Each new node added corresponds to a new base pair or base added to the RNA design pipeline.

$$U(s, b) = Q(s,\ b) + C \times P(b) \times \sqrt{\frac{\ln(N_i)}{n_i}} \tag{2}$$

In the simulation phase, instead of selecting the next base pair or base to assign randomly up to the terminal state, we sample the next base and base pair from the RNA-based language model - the intermediate sequence in the tree is used as a prompt for the language model. In the backpropagation phase, the reward value is backpropagated from the terminal state to the root node of the tree. The reward is the hamming distance (shown in Equation 1) between the target structure and the predicted structure. This value is then used to update the Q values of the nodes in the tree. The algorithm terminates when the maximum number of iterations is reached. The MCTS Algorithm used is shown in Algorithm 1. The algorithm is run for a maximum of 1000 simulations.

```
Data: RNAState, Iterations $N$
Result: Base or base pair $b$
Node = Tree(RNAState)
i = 1
while $i \leq N$ do
    if Node is not fully expanded then
        b = selectAction(Node)
        Node = expand(Node, b)
        break
    end
    else
        Node = selectNextNode(Node)
    end
    reward = LLM_rollout(Node)
    backpropagate(Node, reward)
end
return bestAction(Node)
```

Algorithm 1: MCTS with LLM Rollout

C. Language Modelling

For the last three decades, NLP, tasks such as machine translation, text summarization, question answering and text generation have been actively researched. The recent advances in Deep Learning (DL) [33], [47] and novel architectures such as attention mechanisms and Transformers [29], [30] have significantly improved the performance of these tasks. In addition, the introduction of Large Language Models (LLMs) [41], [42], [48], [49] has unlocked the potential of transfer learning in NLP with pre-trained LLMs. This technique is useful in solving downstream tasks such as sentiment analysis, text classification, text summarization and question answering.

LLMs are modeled to learn the distribution of the tokens (words) in a sequence. Theoretically, this can be conceptualized as a Markov chain where the probability of word w_i is conditioned on the previous n words $w_{i-n}, w_{i-n+1}, ..., w_{i-1}$ or, precisely just the previous word w_{i-1}. Currently, there are two main classes of LLMs: (i) autoregressive models and (ii) masked language models. Autoregressive (decoder) models such as GPT [41] and GPT-2 [42] are based on the Transformer architecture [30]. The models are trained to predict the next token in a sequence given the previous tokens.

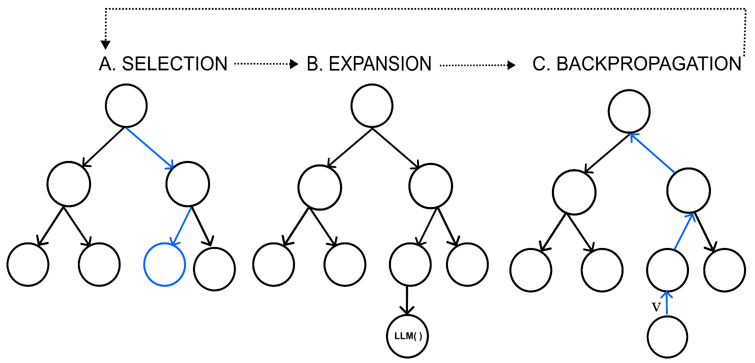

Fig. 2. *MCTS phases. Selection, expansion and backpropagation. During the rollout (simulation) phase, the algorithm samples the next base pair or base from the LLM up to the terminal state. The resulting sequence is then folded and the reward (hamming distance) is computed between the target structure and the predicted structure. The reward is then backpropagated to the root node of the tree.*

Masked (encoder) language models such as BERT [49] and Roberta [50] are also based on the Transformer architecture, however, the models are trained to predict the masked tokens in a sequence e.g. given the sequence *I went to the* ***[MASK]***, the model is trained to predict the masked token *swimming pool*. Apart from the success of NLP, LLMs have also been applied to computational biology tasks such as protein inverse folding [31], [32], and RNA-based applications [26], [34] and Retrosynthetic Planning [38]–[40], [51].

While the classical MCTS simulation phase is based on the random policy, in this work we sampled valid actions (bases and base pairs) from the RNA-based language model. Our algorithmic design is motivated by the increasing sub-optimality associated with the random policy [24]. This step has also been argued to be computationally intensive in some problem contexts such as molecular modeling where the action space corresponds to all valid chemicals in organic chemistry.

We train an RNA-based language model on 2M RNA sequences obtained from the *RNACentral Database* [1]. The RNA sequences from that database are in their primary structure. In RNA Sequence Design, however, the input is the secondary structure. Accordingly, we generate the secondary structure (the target) from the primary structure (the input) using the ViennaRNA package [45]. Subsequently, we program the RNA sequences into sentences according to the target structure and then train the language model. The procedure is discussed in Section IV-D.

The model architecture was based on a smaller version of the autoregressive GPT-2 model called Distilled-GPT2 [52]. Distilled-GPT2 was designed by distilling knowledge from the GPT model. Knowledge Distillation (KD) is a technique used to transfer knowledge from a large model to a smaller model [53] as such the smaller model can be trained faster and requires less computational resources yet still maintains the performance of the close or equal to the larger model. Instead of training the model from scratch, we fine-tuned a model already trained on a large corpus of RNA sentences. The information about the Distilled-GPT2 pre-training can be found on the HuggingFace page [2].

The Adam optimizer [54] with a cosine annealed learning rate starting at 5e-5 was used to train the model. During training, we used a batch size of 16 while running the training session for a total of 50 epochs. RNA Model was trained on a single Tesla T4 GPU (16GB memory). The training took 5 days. The RNA-based language model was trained by using The model is implemented using the PyTorch V.1.9.0 and framework [55] and HuggingFace Transformers library V.2.6.0 [56]. The context length was set to 1024. We have included the training script in the supplementary material.

D. Programming RNA Sequences

The programming makes it easy to map the intermediate sequence during the design to the target structure. The RNA-based model is trained on the RNA sentence as well as its reversed copy. Figure 3 shows the RNA sentence generation process.

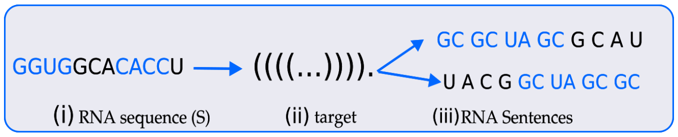

Fig. 3. figure
Programming RNA Language. In (iii) top-most: the RNA sequence index of the base pair positions are concatenated e.g. positions 1 and 11 are base pair positions leading to the first GC, and position 12 is a base position leading to U (the last token). In (iii) bottom-most: reversed RNA sentence.

RNA sequences (sentences) used to train the Language Model were generated as follows: (i) Given an RNA sequence from the database, we used the ViennaRNA package [45] to fold the sequences into their target structures, (ii) match the base pairs and bases to their respective positions in the target structure, and (iii) index the base pairs and bases according to

[1]https://ftp.ebi.ac.uk/pub/databases/RNAcentral/releases/

[2]https://huggingface.co/distilgpt2

the target. The procedure used to program the RNA sequences into a set of sentences is shown in the Algorithm 2.

```
Data: RNA Sequence seq
Result: RNA Sentence sent
sent = [ ]
base, base_pair = Target_Structure(seq)
for pos in base + base_pair do
    if isPaired(pos) then
        sent += seq[pos[0]] + sent[pos[1]]
    end
    else
        sent += seq[pos]
    end
end
return sent
```

Algorithm 2: Programming RNA Sequences

V. Experiments

We tested our algorithm four datasets: [14], [13], [21], [57]. We refer to these datasets as Kauf, Modena, Runge and Eterna respectively. Kauf contained 146 and 103.3 RNA sequences and average length respectively, Modena: 29 and 191.9, Runge: 100 and 240.7 and, Eterna: 100 and 159.2. We compare our method to four other methods: (i) MODENA [13]: genetic algorithm, (ii) NEMO [2]: nested MCTS, (iii) MCTS-RNA [19]: MCTS with local search enhancements (iv) LEARNA [21]: policy gradient algorithm and (v) RNAInverse [57]: dynamic programming. The results are shown in Table I.

VI. Discussion

In this section, we discuss the results of the experiments. The discussion is based on the validity of the RNA sequence design, the GC content and the MFE. Note, however, that the accuracy values are normalized in the range 0 to 100 i.e. 1-$\mathcal{H}(p, t) \times 100$. We also show some sample solutions found by our algorithm.

On valid RNA sequence design. We measured the validity of the RNA sequence design by computing the accuracy between the target structure and the predicted structure. Our method outperformed the comparative methods on half of the test datasets except for the Eterna and Runge datasets. NEMO design is based on nested MCTS where the solutions from the lower-level MCTS are used to guide the search in the upper-level MCTS [2]. While the algorithm is quite competitive, it includes several RNA user-defined rules that are not universally applicable to all RNA sequence design problems and thus impede large-scale RNA sequence design. For instance, the NEMO algorithm uses hand-designed base and base pair probability distributions for the left and right junctions as well as the loops. Furthermore, due to the nested recursive nature of the algorithm, it is computationally intensive compared to the classical MCTS algorithm. Our method, however, does not include any RNA user-defined rules and thus can be applied to any RNA sequence design problem at any scale. In addition, the MCTS phase of our method is computationally effective as it is based on the LLM prediction which can be achieved on $\mathcal{O}(1)$ time complexity. This is further supported by the comparison with the MCTS-RNA method, a model that is similar in design to our method. MCTS-RNA is based on the classical MCTS algorithm where in the simulation phase the actions are sampled randomly up to the terminal state. As previously mentioned, this algorithmic design is prone to sub-optimality and computationally intensive in large action spaces or non-terminating episodes. This is reflected in Table I where our method outperformed MCTS-RNA on all the test datasets a difference of more than 5 (on average). We argue that this variation is because our LLM rollout policy was stronger than the random policy used in the MCTS-RNA. While the design of the rollout policy in our method and MCTS-RNA is the primary difference, another key difference is that our UCB method includes a bias term $P(b)$ which is the prior probability of the base pair or base. In this work, we used the values of the general distributions described in Section I.

While LEARNA, a policy gradient algorithm, recorded the best performance on the Runge dataset, our method outperformed it on the remaining datasets. The ability of our model to outperform LEARNA can be attributed to the inheritance of policy gradient algorithm issues such as convergence and stability. MODENA is a genetic-based algorithm as such it is prone to local optima. Our method outperformed MODENA on all test datasets with a difference of more than 35 on average. Similar observations were noted while comparing the performance of RNAInverse [57] to our work.

Overall, our method outperformed the comparative methods on half the test datasets. However, it did not record very competitive results on the Eterna dataset. This dataset is challenging as the target structures are complex. Such complexity is not adequately reflected in the training dataset thus the model is not able to generalize well on that dataset. We believe that by augmenting the training dataset with more complex sequences, our method can generalize better on the Eterna dataset. However, this is a challenge as the current RNA databases do not contain a sequence of such complexities. This lays an interesting direction for future work.

On GC content and MFE. The GC content and MFE are important factors in RNA sequence design [9]. RNA sequences with higher GC content and lower MFE values are considered more stable. There are no widely agreed expected values of the GC content and MFE. However, the GC content values above 50% and less than 60% are considered desirable in many instances while for MFE, lower values are considered more desirable. In Figure 5 and Figure 6, we compare our MFE and GC content values to some of the state-of-the-art methods. We compute the MFE values using the ViennaRNA package [45]. Our method records GC content value that is within the expected ranges - based on the comparative state-of-the-art methods such as NEMO and MCTS-RNA. On average (Figure 5), however, our method records a slightly higher GC content value compared to the other methods. The GC content value

TABLE I

Accuracy between target structure and RNA sequence fold. All the models were run 4 times. The average values are reported.

	RNAInverse	**MODENA**	**NEMO**	**MCTS-RNA**	**LEARNA**	**Ours**
Runge	19.05	36.51	90.01	89.65	94.30	90.88
Modena	19.05	47.62	90.80	90.48	90.33	**95.91**
Kauf	45.55	44.16	92.21	89.61	80.20	**93.89**
Eterna	18.99	16.46	81.01	58.23	60.70	70.95

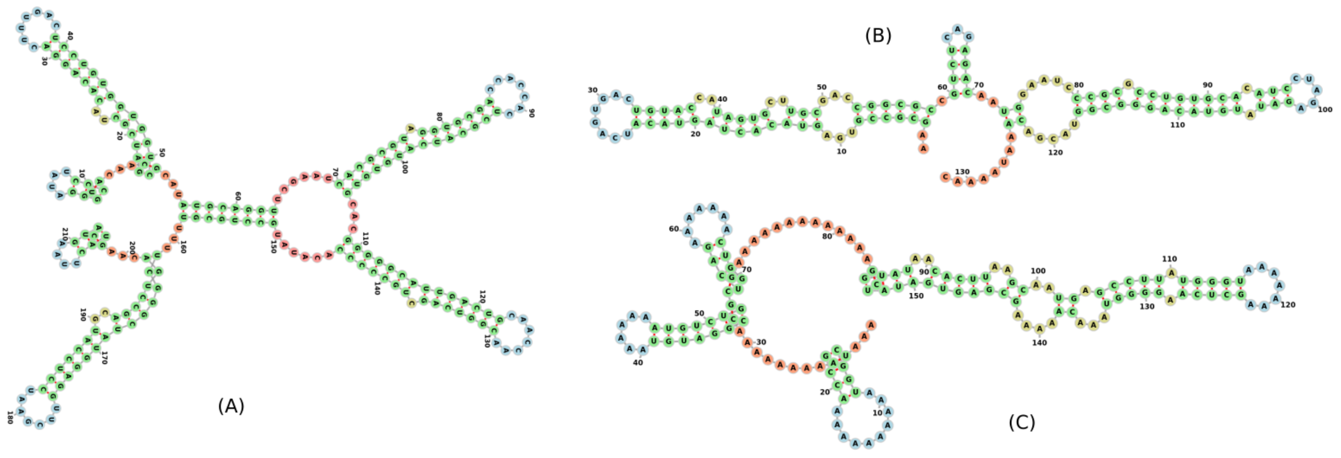

Fig. 4. *Sample correct folds found in our algorithm. The comparative models were not able to accurately fold these examples. In (A); a sample from the Eterna dataset, (B); sampled from the Kauf dataset (C); a Modena dataset sample. The proposed the model learned to balance base and base pair distribution loops and left and right junctions.*

associated with the MCTS-RNA includes very small variation. However, any RNA sequence design method should be able to generate sequences with a wide range of GC content values that are within the expected ranges.

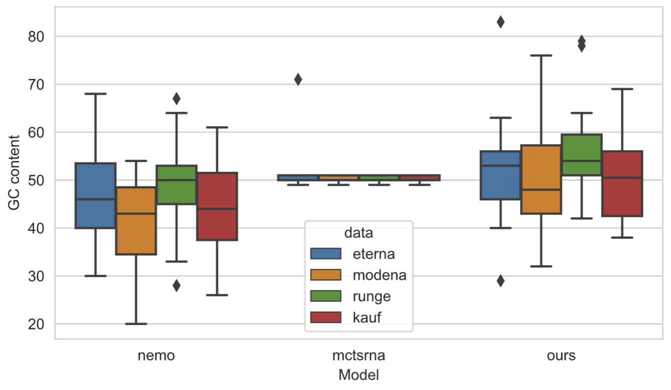

Fig. 5. *GC Content Comparison*

Sample solutions. Some unique solutions were found by our algorithm. The comparative models were not able to accurately fold these examples. In (A); a sample from the Eterna dataset, (B); sampled from the Kauf dataset (C); a Modena dataset sample. As shown the proposed model learned to balance base and base pair distribution loops and left and right junctions without the need for user-defined rules.

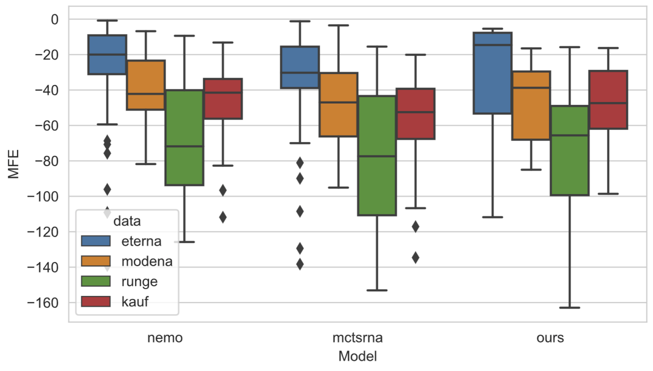

Fig. 6. *MFE Content Comparison*

VII. Related Work

[22] introduced a Reinforcement Learning (RL) approach utilizing Graph Convolution [58] to model RNA Inverse Folding. This method closely resembles the approach by [59], employing undirected graphs and atomic-level sampling. Since RNA sequences inherently exhibit graph-like structures, both methods uphold this structural constraint throughout the modeling process. [21] proposed an RL-based method that learns to select the best action A_t at each time step. Specifically, their approach is based on the Proximal Policy Optimization (PPO) algorithms [60], utilizing gradient descent methods to optimize action selection based on expected reward signals.

Various tree search methods have been proposed for RNA Inverse Folding. [19] proposed a based method with local search enhancements, considering factors such as mismatches and constraints like the expected GC content. The NEMO model, introduced by [2], demonstrated the ability to design more complex sequences with improved performance, particularly when coupled with varying base or base pair distributions. [2] was further extended by [20], allowing for trajectory adaptation using a deep neural network as a policy function. Tree search methods, a more recent development, are compared to some existing Dynamic Programming (DP) approaches. INFO-RNA [61] employed DP to design initial sequences, refined by a one-step look-ahead search function. RNAinverse [11] modeled RNA Inverse Folding using adaptive random walks, minimizing the difference between the target and the designed structure at paired locations. The model proposed by [62] drew inspiration from random walks.

RNAiFold [18] utilized constraint programming, specifying constraints associated with the designed sequence before initiating the design process. These constraints include the desired Minimum MFE, undesired MFE, and ranges for base and base pair distributions. AntaRNA [14] employed swarm intelligence to learn the local sequence features crucial for achieving the correct fold. This the method was further extended to design more complex RNA structures, particularly pseudoknots. Genetic and evolutionary algorithmic strategies have also been explored in several RNA Inverse Folding studies, including MODENA [13], ERD [63], and works by [64], [65].

VIII. Future Work

In this work, we proposed a novel method for designing RNA sequences using an RNA-based language model. Our method uses MCTS to guide the RNA sequence generation process whereby the intermediate sequence in the tree is used as a prompt to the language model during the simulation phase. In the future, we plan to extend our method to handle longer and more complex RNA structures (pseudoknots). In addition, we plan to design larger RNA-based language models that can model longer context lengths. Finally, we plan to experiment with iterative deepening search as an alternative tree search algorithm.

IX. Conclusion

In this paper, we proposed a novel method for designing RNA sequences using an RNA-based language model which guides the sequence generation process using MCTS. The tree search ensures that only valid sequences are generated and as such, common issues associated with the LLMs such as hallucinations and invalid sequences generation are controlled. We showed that our method outperformed the comparative methods on half of the test datasets. We also showed that our method can generate valid sequences that match the expected ranges of the MFE and GC content. As part of future work, designing larger RNA-based language models that can model longer context lengths and experimenting with an alternative tree search method would be interesting.

References

[1] K. F. Schaffner, "The watson-crick model and reductionism," *The British Journal for the Philosophy of Science*, vol. 20, no. 4, pp. 325–348, 1969.

[2] F. Portela, "An unexpectedly effective monte carlo technique for the rna inverse folding problem," *BioRxiv*, p. 345587, 2018.

[3] E. Trotta, "On the normalization of the minimum free energy of rnas by sequence length," *PloS one*, vol. 9, no. 11, p. e113380, 2014.

[4] H. J. J. Cleaves *et al.*, "Watson-crick pairing," *Encyclopedia of Astrobiology*, pp. 2650–2650, 2015.

[5] S. Obonyo, N. Jouandeau, and D. Owuor, "Self-playing rna inverse folding," *SN Computer Science*, vol. 5, no. 4, p. 403, 2024.

[6] T. Akutsu, "Dynamic programming algorithms for rna secondary structure prediction with pseudoknots," *Discrete Applied Mathematics*, vol. 104, no. 1-3, pp. 45–62, 2000.

[7] R. B. Lyngsø and C. N. Pedersen, "Rna pseudoknot prediction in energy-based models," *Journal of computational biology*, vol. 7, no. 3-4, pp. 409–427, 2000.

[8] S. Ieong, M.-Y. Kao, T.-W. Lam, W.-K. Sung, and S.-M. Yiu, "Predicting rna secondary structures with arbitrary pseudoknots by maximizing the number of stacking pairs," *Journal of Computational biology*, vol. 10, no. 6, pp. 981–995, 2003.

[9] N. C. Seeman, "At the crossroads of chemistry, biology, and materials: structural dna nanotechnology," *Chemistry & Biology*, vol. 10, no. 12, pp. 1151–1159, 2003.

[10] P. W. Rothemund, "Folding dna to create nanoscale shapes and patterns," *Nature*, vol. 440, no. 7082, pp. 297–302, 2006.

[11] I. L. Hofacker, "Vienna rna secondary structure server," *Nucleic acids research*, vol. 31, no. 13, pp. 3429–3431, 2003.

[12] M. Andronescu, A. P. Fejes, F. Hutter, H. H. Hoos, and A. Condon, "A new algorithm for rna secondary structure design," *Journal of molecular biology*, vol. 336, no. 3, pp. 607–624, 2004.

[13] A. Taneda, "Multi-objective optimization for rna design with multiple target secondary structures," *BMC bioinformatics*, vol. 16, no. 1, pp. 1–20, 2015.

[14] R. Kleinkauf, M. Mann, and R. Backofen, "antarna: ant colony-based rna sequence design," *Bioinformatics*, vol. 31, no. 19, pp. 3114–3121, 2015.

[15] X. Xu and S.-J. Chen, "Physics-based rna structure prediction," *Biophysics reports*, vol. 1, pp. 2–13, 2015.

[16] M. Zuker and P. Stiegler, "Optimal computer folding of large rna sequences using thermodynamics and auxiliary information," *Nucleic acids research*, vol. 9, no. 1, pp. 133–148, 1981.

[17] M. Zuker and D. Sankoff, "Rna secondary structures and their prediction," *Bulletin of mathematical biology*, vol. 46, pp. 591–621, 1984.

[18] J. A. Garcia-Martin, P. Clote, and I. Dotu, "Rnaifold: a constraint programming algorithm for rna inverse folding and molecular design," *Journal of bioinformatics and computational biology*, vol. 11, no. 02, p. 1350001, 2013.

[19] X. Yang, K. Yoshizoe, A. Taneda, and K. Tsuda, "Rna inverse folding using monte carlo tree search," *BMC bioinformatics*, vol. 18, no. 1, pp. 1–12, 2017.

[20] T. Cazenave and T. Fournier, "Monte carlo inverse folding," in *Monte Carlo Search International Workshop*. Springer, 2020, pp. 84–99.

[21] F. Runge, D. Stoll, S. Falkner, and F. Hutter, "Learning to design rna," *arXiv preprint arXiv:1812.11951*, 2018.

[22] P. Eastman, J. Shi, B. Ramsundar, and V. S. Pande, "Solving the rna design problem with reinforcement learning," *PLoS computational biology*, vol. 14, no. 6, p. e1006176, 2018.

[23] T. Anthony, Z. Tian, and D. Barber, "Thinking fast and slow with deep learning and tree search," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[24] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[25] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[26] Y. Zhang, M. Lang, J. Jiang, Z. Gao, F. Xu, T. Litfin, K. Chen, J. Singh, X. Huang, G. Song *et al.*, "Multiple sequence-alignment-based rna language model and its application to structural inference," *bioRxiv*, pp. 2023–03, 2023.

[27] J. Maynez, S. Narayan, B. Bohnet, and R. McDonald, "On faithfulness and factuality in abstractive summarization," *arXiv preprint arXiv:2005.00661*, 2020.

[28] A. P. Parikh, X. Wang, S. Gehrmann, M. Faruqui, B. Dhingra, D. Yang, and D. Das, "Totto: A controlled table-to-text generation dataset," *arXiv preprint arXiv:2004.14373*, 2020.

[29] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[31] Z. Zheng, Y. Deng, D. Xue, Y. Zhou, F. Ye, and Q. Gu, "Structure-informed language models are protein designers," *bioRxiv*, pp. 2023–02, 2023.

[32] J. Dauparas, I. Anishchenko, N. Bennett, H. Bai, R. J. Ragotte, L. F. Milles, B. I. Wicky, A. Courbet, R. J. de Haas, N. Bethel *et al.*, "Robust deep learning–based protein sequence design using proteinmpnn," *Science*, vol. 378, no. 6615, pp. 49–56, 2022.

[33] J. Ding and A. Regev, "Deep generative model embedding of single-cell rna-seq profiles on hyperspheres and hyperbolic spaces," *Nature communications*, vol. 12, no. 1, p. 2554, 2021.

[34] F. Ozden, S. Barazandeh, D. Akboga, U. O. S. Seker, and A. E. Cicek, "Rnagen: A generative adversarial network-based model to generate synthetic rna sequences to target proteins," *bioRxiv*, pp. 2023–07, 2023.

[35] Z. Yan, W. L. Hamilton, and M. Blanchette, "Neural representation and generation for rna secondary structures," *arXiv preprint arXiv:2102.00925*, 2021.

[36] S. Li, S. Moayedpour, R. Li, M. Bailey, S. Riahi, M. Miladi, J. Miner, D. Zheng, J. Wang, A. Balsubramani *et al.*, "Codonbert: Large language models for mrna design and optimization," *bioRxiv*, pp. 2023–09, 2023.

[37] R. J. Penić, T. Vlašić, R. G. Huber, Y. Wan, and M. Šikić, "Rinalmo: General-purpose rna language models can generalize well on structure prediction tasks," *arXiv preprint arXiv:2403.00043*, 2024.

[38] P. Schwaller, T. Gaudin, D. Lanyi, C. Bekas, and T. Laino, ""found in translation": predicting outcomes of complex organic chemistry reactions using neural sequence-to-sequence models," *Chemical science*, vol. 9, no. 28, pp. 6091–6098, 2018.

[39] P. Karpov, G. Godin, and I. V. Tetko, "A transformer model for retrosynthesis," in *International Conference on Artificial Neural Networks*. Springer, 2019, pp. 817–830.

[40] Y. Wan, C.-Y. Hsieh, B. Liao, and S. Zhang, "Retroformer: Pushing the limits of end-to-end retrosynthesis transformer," in *International Conference on Machine Learning*. PMLR, 2022, pp. 22 475–22 490.

[41] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[42] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[43] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*. Springer, 2006, pp. 282–293.

[44] R. W. Hamming, "Error detecting and error correcting codes," *The Bell system technical journal*, vol. 29, no. 2, pp. 147–160, 1950.

[45] 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*, vol. 6, no. 1, pp. 1–14, 2011.

[46] M. Świechowski, K. Godlewski, B. Sawicki, and J. Mańdziuk, "Monte carlo tree search: A review of recent modifications and applications," *Artificial Intelligence Review*, vol. 56, no. 3, pp. 2497–2562, 2023.

[47] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[48] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv preprint arXiv:1801.06146*, 2018.

[49] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[50] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[51] B. Liu, B. Ramsundar, P. Kawthekar, J. Shi, J. Gomes, Q. Luu Nguyen, S. Ho, J. Sloane, P. Wender, and V. Pande, "Retrosynthetic reaction prediction using neural sequence-to-sequence models," *ACS central science*, vol. 3, no. 10, pp. 1103–1113, 2017.

[52] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[53] G. Hinton, O. Vinyals, J. Dean *et al.*, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[55] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[56] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, 2020, pp. 38–45.

[57] J. Anderson-Lee, E. Fisker, V. Kosaraju, M. Wu, J. Kong, J. Lee, M. Lee, M. Zada, A. Treuille, and R. Das, "Principles for predicting rna secondary structure design difficulty," *Journal of molecular biology*, vol. 428, no. 5, pp. 748–757, 2016.

[58] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[59] G. Meng, M. Tariq, S. Jain, S. Elmetwaly, and T. Schlick, "Rag-web: Rna structure prediction/design using rna-as-graphs," *Bioinformatics*, vol. 36, no. 2, pp. 647–648, 2020.

[60] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[61] A. Busch and R. Backofen, "Info-rna—a fast approach to inverse rna folding," *Bioinformatics*, vol. 22, no. 15, pp. 1823–1831, 2006.

[62] N. S. Merleau and M. Smerlak, "An evolutionary algorithm for inverse rna folding inspired by lévy flights," *bioRxiv*, 2022.

[63] A. Esmaili-Taheri and M. Ganjtabesh, "Erd: a fast and reliable tool for rna design including constraints," *BMC bioinformatics*, vol. 16, no. 1, pp. 1–11, 2015.

[64] N. Dromi, A. Avihoo, and D. Barash, "Reconstruction of natural rna sequences from rna shape, thermodynamic stability, mutational robustness, and linguistic complexity by evolutionary computation," *Journal of Biomolecular Structure and Dynamics*, vol. 26, no. 1, pp. 147–161, 2008.

[65] R. B. Lyngsø, J. W. Anderson, E. Sizikova, A. Badugu, T. Hyland, and J. Hein, "Frnakenstein: multiple target inverse rna folding," *BMC bioinformatics*, vol. 13, no. 1, pp. 1–12, 2012.