Briefings in Bioinformatics, 2025, **26(2)**, bbaf106
https://doi.org/10.1093/bib/bbaf106
Problem Solving Protocol

DRAG: design RNAs as hierarchical graphs with reinforcement learning

Yichong Li[1], Xiaoyong Pan[2,3,*], Hongbin Shen[2,3], Yang Yang[1,4,*]

[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University, 800 Dong Chuan Rd., Minhang District, Shanghai 200240, China
[2]Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, 800 Dong Chuan Rd., Minhang District, Shanghai 200240, China
[3]Key Laboratory of System Control and Information Processing, Ministry of Education of China, 800 Dong Chuan Rd., Minhang District, Shanghai 200240, China
[4]Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering, Shanghai Jiao Tong University, 800 Dong Chuan Rd., Minhang District, Shanghai 200240, China

[*]Corresponding authors. Xiaoyong Pan. E-mail: 2008xypan@sjtu.edu.cn; Yang Yang. E-mail: yangyang@cs.sjtu.edu.cn.

Abstract

The rapid development of RNA vaccines and therapeutics puts forward intensive requirements on the sequence design of RNAs. RNA sequence design, or RNA inverse folding, aims to generate RNA sequences that can fold into specific target structures. To date, efficient and high-accuracy prediction models for secondary structures of RNAs have been developed. They provide a basis for computational RNA sequence design methods. Especially, reinforcement learning (RL) has emerged as a promising approach for RNA design due to its ability to learn from trial and error in generation tasks and work without ground truth data. However, existing RL methods are limited in considering complex hierarchical structures in RNA design environments. To address the above limitation, we propose DRAG, an RL method that builds design environments for target secondary structures with hierarchical division based on graph neural networks. Through extensive experiments on benchmark datasets, DRAG exhibits remarkable performance compared with current machine-learning approaches for RNA sequence design. This advantage is particularly evident in long and intricate tasks involving structures with significant depth.

Keywords: RNA sequence design; hierarchical division; reinforcement learning; graph neural networks

Introduction

With the increasing understanding of RNA's role in biological processes, the functions of RNAs involved in regulatory processes have been revealed. The design of RNA sequences has become a pivotal focus in medicine and synthetic biology [1, 2]. It has important implications for various research and clinical applications, including disease detection, drug synthesis, and gene therapy [3–6]. During the severe acute respiratory syndrome coronavirus-2 pandemic, RNA-based drugs and vaccines have been rapidly developed [7, 8], highlighting the need for efficient RNA sequence design methods to advance RNA therapies and vaccines [9].

RNA sequence design, also known as inverse folding, involves finding nucleotide sequences that can fold into a secondary structure that matches a given target structure. The rapid advancement of highly efficient and accurate methods for predicting RNA secondary structures has laid the foundation for RNA sequence design [10–14]. Additionally, various RNA design puzzles have been created to validate these design methods. The Eterna research group [15, 16] established an open online lab that combines structure prediction algorithms with interactive games. Through the collective efforts of human players, RNA design problems of varying complexities are addressed. Generated sequences and player data are used to build the Eterna 100 benchmark [17]. This benchmark includes 100 RNA design tasks and provides a standard metric for evaluating RNA design methods [18]. Despite significant progress, RNA sequence design remains challenging due to the complexity of RNA structures and the vast search space of feasible sequences. Experimental studies by Varani *et al.* [19] highlight the intricate molecular mechanisms of RNA folding. Meanwhile, Das *et al.* [20] demonstrated the complexity of designing RNA sequences using advanced computational tools like R3Design. These studies underscore the need for innovative approaches to address the inherent complexities of RNA design.

Computational RNA design techniques began with search algorithms like RNAinverse [21] and Info-RNA [22]. These methods have evolved into more efficient methods [23, 24]. Recent search-based methods attempt to optimize the search process to enhance efficiency and explore a more diverse range of feasible solutions. MoiRNAiFold [25] uses constraint programming to narrow the search space, while aRNAque [26, 27] employs evolutionary algorithms (EAs) to iteratively optimize RNA sequences. SAMFEO [28] introduces diverse mutation strategies and uses Boltzmann distribution to select candidate sequences. DesiRNA [29] utilizes the Monte Carlo algorithm for the selection and acceptance of mutation sites. These approaches yield impressive success rates in RNA design. Nevertheless, search-based methods for RNA design often require domain knowledge to establish guiding principles for the search process. This requirement may limit the applicability and scalability of these methods. In the absence of such principles,

Received: November 29, 2024. **Revised:** February 10, 2025. **Accepted:** February 25, 2025

the methods tend to depend heavily on stochastic processes. As a result, the design outcomes can be significantly influenced by the initial conditions and various random factors. As the complexity of RNA hierarchical structures increases, the search space expands exponentially, presenting a notable challenge for these methods.

Machine learning (ML)-based methods have consistently achieved excellent results in various RNA design objectives [30–33]. RNA design does not neatly fit into conventional supervised learning, as multiple sequences can fold into the same structure. Several ML frameworks have been explored, including supervised learning methods like SentRNA [34], imitation learning like EternaBrain [35], and reinforcement learning (RL) approaches like Eastman *et al.* [36], LEARNA[37], RNASP [38], and libLEARNA [39]. SentRNA considers both RNA bases and secondary structures as inputs and introduces “remote action” sites to analyze relationships among disconnected sites within the secondary structure. EternaBrain learns from player design trajectories collected by the Eterna online lab, predicting mutation locations and types, with manual paradigms as corrections for improved performance. LEARNA encodes local dot-bracket sequences and uses a long short-term memory (LSTM) [40] network combined with one-dimensional convolution to infer base types. Overall, ML has the potential to capture the complex relationships between nucleotides in RNA molecules and effectively model the intricate secondary and tertiary structures of RNAs. While RL is effective for combinatorial optimization problems, it encounters several challenges in RNA design: (1) representing states in RNA design challenges. Sequence-based representations may overlook structural information, and RNA’s topological structure is sparse. (2) The number of feasible sequences on complex RNA molecules is vast, and the relationship between sequences and structures is intricate, hindering the agent’s exploration efficiency. (3) Leveraging hierarchical relationships in RNA structures proves challenging for the agents.

To tackle more intricate tasks and enable efficient exploration of feasible regions, we present DRAG. DRAG is an RL-based Design method that represents RNAs as hierarchical Graphs to capture the intricate topological complexities inherent in RNA structures. By leveraging hierarchical graph representation (HGR), DRAG overcomes the limitations of existing methods that often fail to account for the hierarchical nature of RNA structures, especially in complex and deeply nested structures. Additionally, DRAG employs a task division mechanism that further enhances its performance by breaking down complex RNA design tasks into manageable subtasks. This approach not only simplifies the design process but also improves the efficiency and accuracy of RNA sequence generation. To further enhance the performance of the RL approach, we specially craft the reward function and the environment. The main contributions are summarized below.

- We propose an HGR of RNAs for RNA sequence. This method design offers distinct advantages by effectively capturing abundant node and topological information encoded within RNA structures and uncovering the topological arrangement of stem-loop substructures, enabling exploration of the interplay between substructures and the overall RNA structure.
- We develop two strategies to enhance the performance of RL: (1) a composite reward that considers both energy and structural discrepancy; and (2) task decomposition based on HGR, which facilitates the creation of a hierarchical task pool for RL and the development of an RNA environment that is aligned with the division of structural hierarchy.
- On widely used benchmark datasets, DRAG outperforms other ML-based approaches for RNA design, achieving the highest success rate.

Materials and methods

RNA design problem

Given an RNA sequence x and a specific secondary structure τ, the folding free energy $G(x,\tau)$ serves as an indicator of the RNA’s stability in that structure. A lower folding free energy corresponds to increased stability.

Within the set of all possible secondary structures $\mathcal{T}(x)$, there exist one or multiple structures with the lowest folding free energy. Under the minimum free energy mode of RNAfold [11], there is only one structure with the lowest folding free energy of a sequence. Therefore, in this work, we assume the structure with the minimum free energy of the RNA sequence to be unique, referred to as the Minimum Folding Free Energy (MFE) structure and defined as

$$\tau_{\mathrm{MFE}}(x) = \underset{\tau\in\mathcal{T}(x)}{\arg\min}\, G(x,\tau). \tag{1}$$

The goal of RNA design is to generate a sequence x with the MFE structure that aligns with the desired target structure τ^*. This generation task can be transformed into a search problem. Starting from an initial sequence, the objective is to minimize the distance between τ^* and $\tau_{\mathrm{MFE}}(x)$, where $x \in X$ and X denotes the entire space of RNA sequences (X can be further refined to include only those sequences that adhere to the base pairing regulations consonant with τ^*). This optimization process entails the repeated introduction of mutations (including single-node mutations on loops and node-pair mutations on stems) within a finite number of steps.

The RNA design process can be defined as a Markov decision process (MDP), which requires that the state transition satisfies the Markov property: $p(x^{t+1}|x^t, ..., x^0) = p(x^{t+1}|x^t)$. The state x is an RNA sequence. All possible states constitute the state space X. The action space is bifurcated into two distinct components: A_{loc} and A_{mut}, where $A_{\mathrm{loc}} = \{\mathrm{site}_1, \mathrm{site}_2, \ldots, \mathrm{site}_n\}$ represents the selection of n mutation sites on the RNA sequences and $A_{\mathrm{mut}} = \{\mathrm{A,U,C,G}\}$ involves the choice of nucleotide type for mutation. The state transition in this mutation process is deterministic, represented as $P(x'|x,a) = 1$. The reward function $R(x,a)$ is predicated on the variation in distance and energy after mutation. This will be elucidated in detail in the subsequent sections. The discount factor γ serves as a hyperparameter that quantifies the present value of future rewards within this framework. Therefore, the MDP for the RNA design problem can be represented as a quintuple, i.e. $(X, A = (A_{\mathrm{loc}}, A_{\mathrm{mut}}), P, R, \gamma)$.

The DRAG model overview

DRAG comprises an RNA design environment and a graph neural network (GNN)-based agent. The environment breaks down target RNA secondary structures into smaller tasks, creating a task pool. The associated sub-structures are converted into graphs and processed by the GNN-based agent. With a multi-layer GNN, the agent extracts features from these graphs and calculates probability distributions for mutation sites and types. These probabilities guide mutation commands, which are communicated back to the environment. The environment evaluates optimization progress toward the target, generating rewards for the agent. The agent’s

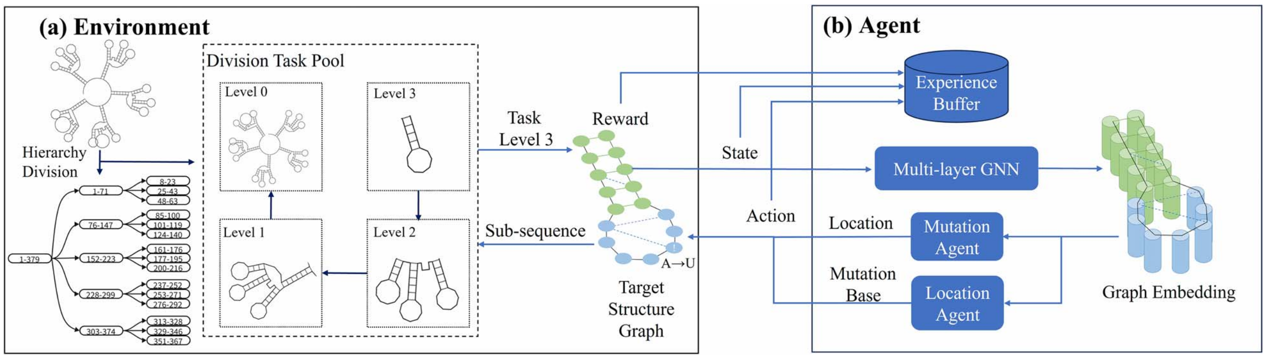

Figure 1. The DRAG framework: (a) design environment with task division decomposing target RNA structures into a task pool; (b) GNN-based agent transforming sub-structures into graph states processed through multi-layer networks to generate mutation actions, with experience buffer storing states, rewards, and actions for training.

experience buffer accumulates all observations, rewards, actions, and relevant data, facilitating the model training. The flowchart of DRAG is shown in Fig. 1.

Graph representation of RNAs

In DRAG, we construct an undirected graph based on the predicted secondary structure obtained from RNAfold. This graph includes the adjacency matrix, node features, and edge features as its components.

Node features for RNA graph

The node features represent embeddings of bases. To capture the nucleotide-specific information, we utilize the GloVe algorithm [41] that provides pre-trained embeddings for the RNA bases. To ensure the quality of these base embeddings, we use the entire human genome as the corpus, transforming DNA sequences into RNA sequences by replacing all occurrences of "T" with "U."

The adjacency matrix and edge features

The adjacency matrix encompasses three types of edges, i.e. edges along the backbone of RNA molecules formed through chemical bonds, edges connecting base pairs on the target structure that are established via hydrogen bonds, and site pairs that exhibit substantial pairing potential as predicted by RNAfold. For the representation of edge features, we employ 4D vectors, where the first three dimensions constitute a one-hot vector that signifies the edge type. In contrast, the fourth dimension quantifies the pairing probability between the connected sites. Specifically, the probability of the chemical bond edge is set to 1, while the probabilities associated with the other two edges are derived from the results of RNAfold predictions.

Secondary structure division

Inspired by the structural analysis employed in ERD [42] and eM2dRNAs [43], we recognize that the secondary structure of RNA can be deconstructed into nested, hierarchically arranged substructures. Multibranched RNA structures are split into stems and inner loop blocks, treated as generalized nodes, resulting in a tree-like RNA topology with varying substructure complexities. To capture this hierarchical organization, we employ the depth-first search algorithm, which traverses along the RNA's main chain from the 5' end to the 3' end. We consider both isolated nodes on loops and stems as nodes, adding them to the list of nodes in the external loop. When encountering a stem, it and its enclosed substructures are treated as a new branch. The search then recursively identifies stem-loop structures and incorporates new branches as generalized nodes into the external loop list. Details of the algorithms are present in Supplementary Algorithms 1 and 2.

An example of RNA secondary structure hierarchical division is shown in Fig. 2(a). This strategy facilitates task decomposition that is well suited for RL.

Key elements of the RL model

State

The state is what the agent perceives. Instead of generating a state from the entire sequence and target structure, the environment divides the target structure into substructures of different levels. Using the RNA hierarchical division approach, these substructures range from low-level to high-level, serving as design subtasks. The current task's sub-sequence and sub-structure are transformed into sub-graphs, presented sequentially to the agent as observation states. We retain the sequences of the designed substructures and combine them as a part of the initial sequence of the subsequent task, breaking down complex RNA design tasks into multiple sub-tasks, from shallow to deep, for efficient design and learning. Figure 1(a) shows an example of a hierarchical task pool.

To mitigate the possibility of encountering sub-structures without feasible sub-sequences, we introduce a slack threshold for the distance between secondary structures within the sub-tasks, allowing the environment to transition to higher-level tasks when necessary.

Rewards

DRAG adopts a composite reward, taking into account both the distance between the predicted structure of the RNA sequence and the target structure, and the folding free energy of the RNA sequence on the target structure. To more precisely characterize the distance between the currently designed sequence and the target sequence, we use the adjacency matrix corresponding to the two structures. We calculate the difference and count the number of rows that are not all zero as the distance D. Equation (2) defines the distance score, S_d, between the MFE structure of current sequence x^t and the target structure τ^*:

$$S_d^t = S_d(x^t, \tau^*) = \frac{1}{1 + D(\tau_{\mathrm{MFE}}(x^t), \tau^*)}. \tag{2}$$

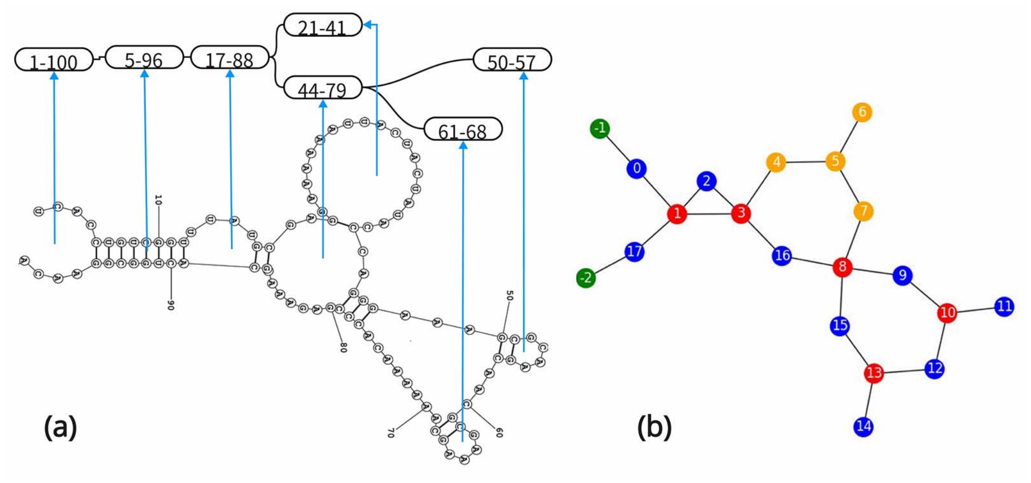

Figure 2. Abstract structure graph of target secondary structure with a hierarchical tree: (a) target secondary structure of an RNA; (b) abstract structure graph of the target secondary structure, where the green nodes stand for the 5' end and 3' end, respectively, red nodes stand for the stem, blue nodes stand for the loop and the yellow nodes represent the regions containing mutatable sites.

The energy score S_e is calculated by Equation (3):

$$S_e^t = S_e(x^t, \tau^*) = 1 - \frac{G(x^t, \tau^*) - G(x^t, \tau_{\mathrm{MFE}}(x^t))}{1 + |G(x^t, \tau_{\mathrm{MFE}}(x^t))|}, \tag{3}$$

where G is the free energy of folding a given sequence into a given structure.

Then the total reward at tth step, R_{all}^t, is the weighted sum of the changes between the pre- and post-action scores, i.e.

$$R_{all}^t = \alpha(S_d^t - S_d^{t-1}) + \beta(S_e^t - S_e^{t-1}), \tag{4}$$

where α and β are weighting factors for the changes in energy score and distance score, respectively.

Additionally, to further accentuate the distinct impacts of location and mutation actions in the design process, we devise separate reward components for each type of action. Equation (5) details the reward attributed to the location agent's outcomes.

$$R_l^t = \max_{x \in N(x^{t-1}, i)} S_d(x, \tau^*) - S_d(x^{t-1}, \tau^*), \tag{5}$$

where R_l^t is the reward for the location agent at tth step. $N(x^{t-1}, i)$ is the set of possible mutated sequences given the sequence x^{t-1} and mutation site i. The final reward of the **location agent** is a weighted sum of R_{all}^t and R_l^t.

Equation (6) quantifies the change in the target structure's formation likelihood upon site mutation.

$$R_m^t = 2(\mathbf{A}_{\tau*}[i, \cdot] - 0.5)(\mathbf{B}_{x^t}[i, \cdot] - \mathbf{B}_{x^{t-1}}[i, \cdot])^\top, \tag{6}$$

where R_m^t is the private reward for the mutation agent at tth step with RNA sequence mutates from x^{t-1} to x^t where the mutation site is i. $\mathbf{A}_{\tau*}$ is the adjacency matrix of the target structure τ^*. $\mathbf{B}_{x^t}$ is the base pairing probability matrix of the current sequence x^t. $\mathbf{A}[i, \cdot]$ and $\mathbf{B}[i, \cdot]$ represent the ith row vector of $\mathbf{A}$ and $\mathbf{B}$, respectively. The final reward of the **mutation agent** is a weighted sum of R_{all}^t and R_m^t.

Actions

As mentioned in Section RNA design problem, the agent performs two types of actions: identifying mutation-susceptible sites and specifying the base to replace at those sites. In RNA design, accurately identifying mutable sites is crucial for improving the design's success and efficiency. To achieve this, we first analyze sites where the current target secondary structure differs from the predicted structure (these sites are called difference sites) and examine RNA sub-structures that could impact these sites. We use an abstract structure graph, treating "stems" and "loops" as nodes to display adjacency relationships. Figure 2(b) shows an example of these graphs for the target secondary structures. In this graph, each node represents a local stem or loop and includes the positions occupied by that region within the RNA sequence. By considering the location of the different sites and their adjacent nodes, we determine the sites based on the following principles:

- If the difference site is located on a stem node, the mutatable site encompasses both the positions within the stem node itself and the neighboring ring nodes;
- If the difference site is positioned on a ring node, the mutatable site is the loop node itself.

These principles serve as a systematic and effective means of identifying mutation-susceptible sites that facilitate the achievement of the desired target secondary structure.

Furthermore, we apply action restrictions based on the analysis of mutatable sites, confining the sampling process to the set of mutatable sites alone. The probability of action restrictions is positively correlated with the distance between the current sequence's MFE structure and the target structure. This constraint ensures that the sampling is exclusively carried out within the sites that have been identified as candidates for inducing the desired changes.

Agents

The GNN model underpins the agent implementation. The model comprises the backbone network for feature extraction, the location agent, and the mutation agent. The two agents consist of an actor network and a critic network, respectively. The backbone network uses multiple layers of Graph Isomorphism Networks [44], as shown in Fig. 1(b).

The location agent's actor comprises fully connected (FC) layers and a softmax readout layer, outputting the site selection probability on RNA sequences. The critic includes FC layers and a summed readout layer, predicting the value for the location

Table 1. Accuracy comparison between DRAG and other ML-based methods on Rfam-taneda, Rfam-learn-test, Eterna 100 v1, and Eterna 100 v2

Method	**# Solved puzzles / # All**			
	Rfam-taneda	**Rfam-learn-test**	**Eterna 100 v1**	**Eterna 100 v2**
EternaBrain-SAP	23/29	98/100	61/100	59/100
LEARNA	23/29	97/100	67/100	/
Meta-LEARNA	24/29	**100**/100	68/100	/
Meta-LEARNA-Adapt	24/29	99/100	68/100	68/100
SentRNA w/o full moveset	/	/	47/100	/
SentRNA	24/29	**100**/100	78/100	69/100
DRAG w/o full moveset	**25**/29	**100**/100	77/100	75/100
DRAG	**25**/29	**100**/100	**79**/100	**77**/100

agent. The location agent selects the mutation site based on this probability. Mutation site embedding serves as input for the mutation agent. The mutation agent's actor uses FC layers and softmax to generate mutation probabilities for bases or base pairs. Similarly, the critic consists of FC layers and predicts the value associated with the mutation agent.

To distinguish between single-node and node-pair mutations, we define two agent groups: one for single-node mutations and one for node-pair mutations. The single-node group takes the single-node position embedding as input, while the node-pair group uses the node-pair position embedding. These agent groups operate alternately during the RNA sequence design process. Each group predicts and selects mutation sites based on their embeddings, ensuring both single-node and node-pair mutations are effectively incorporated into the design.

Training strategy and sequence refinement

We employ the Proximal Policy Optimization algorithm [45] due to its strong ability to deal with large states and action spaces. During training, the agent interacts with the RNA design environment, evaluating and sampling mutation site and base type probabilities based on the RNA graph. The agent then executes mutations, and the environment calculates state changes, assigning rewards to the agents. These rewards are fed back to the mutation site and type prediction modules. As the agent interacts with the environment, state changes and rewards are stored in the experience pool, and the model is trained using these interactions after each round. To improve design outcomes, we integrate sequence refinement from SentRNA [34]. After each design attempt, the best sequence undergoes minor random mutations and greedy algorithm iterations.

Training and evaluation

The training set used is derived from LEARNA's Rfam-learn dataset [37], including 65 000 RNA secondary structures from Rfam. For model evaluation, we use diverse challenges from Eterna 100 v1 and v2 [46], each consisting of 100 structures. These benchmarks are designed for different RNAfold versions, with 81 target structures being identical across both versions. However, due to varying RNAfold prediction results, the The RNA sequence design problem's difficulty differs. Additionally, we include two datasets from LEARNA: the Rfam test set (29 structures) and the Rfam-taneda test set (100 structures).

During training, each structure undergoes 80 mutation opportunities, with 40 for isolated nodes and node pairs per round. The design and model training for all secondary structures are performed within a single epoch. In testing, we set an upper limit of 600 design rounds per RNA structure and allow up to 80 mutations per round. Separate models are trained for Eterna 100 v1 and v2 due to differing RNAfold versions.

As Koodli *et al.* (2021) [46] evaluated ML-based methods on Eterna 100 v2, we compare DRAG with leading ML methods and their variants across all four test sets:

(1) EternaBrain-SAP: employs imitation learning and single-action-playout strategy to optimize sequences.
(2) LEARNA: uses RL. Different from DRAG, it solves a generation task by starting with a blank sequence and gradually filling in bases. We also consider two other variants of LEARNA, i.e. Meta-LEARNA (trained with the meta-learning framework) and Meta-LEARNA-Adapt (trained using empirical data).
(3) SentRNA: adopts supervised learning. We consider two versions, one with the full moveset and one without.

Experimental results

DRAG outperforms current ML-based methods

Table 1 presents results where baseline methods on Eterna 100 v1 are from their original publications, and results on Eterna 100 v2 are from [46]. DRAG outperforms all methods across datasets, consistently solving the most puzzles. On the Rfam-learn-test dataset, all methods show similar performance, with DRAG and two baseline models achieving perfect success due to the dataset's simplicity.

For more complex datasets, DRAG excels. On Rfam-taneda, it solves 25 puzzles; on Eterna 100 v1, it solves 79 puzzles; and on Eterna 100 v2, it solves 77 puzzles, surpassing other ML-based methods. Notably, SentRNA resolves only 47 puzzles in Eterna 100 v1 without full-moveset optimization, while DRAG achieves 77 without this refinement. Design details can be found in Supplementary Fig. S1. In comparison with ML methods using sliding window-based state sampling, DRAG shows superior performance, particularly on puzzles with complex hierarchical structures. These include Puzzle 10 in Rfam-taneda and Puzzles 52 and 79 in Eterna 100. Notably, Puzzle 79 in Eterna 100 is characterized by a nested stem-loop structure with six layers of depth. Puzzle 52 features a multibranched structure with nested interactions among the branches. Additionally, Puzzle 10 in Rfam-taneda combines both multibranch and deeply layered structural elements. It features notably long stems that enhance the substructural stability. This characteristic is particularly well suited to DRAG's task-specific approach. DRAG excels in breaking down complex topologies into hierarchical RNA structures. In contrast, sliding window methods struggle with intricate designs

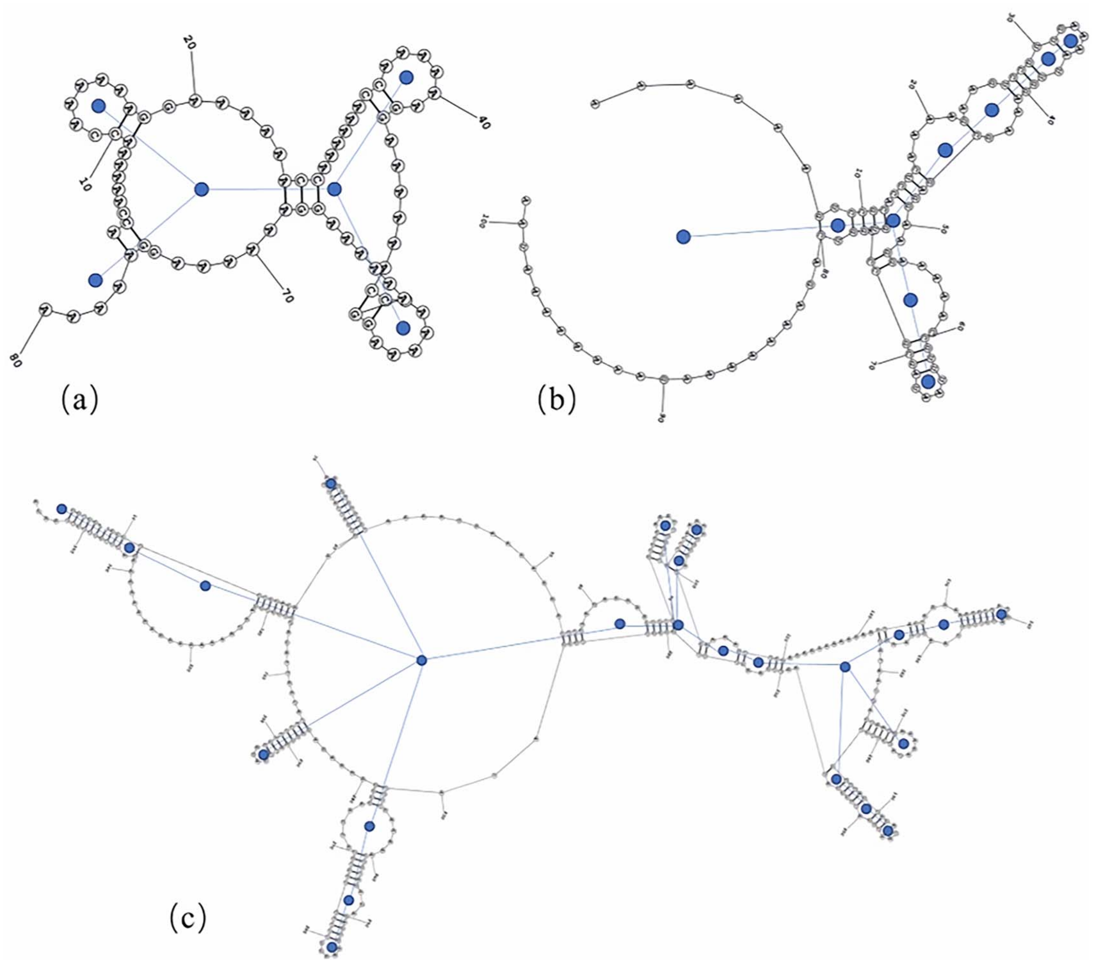

Figure 3. Three intricate hierarchical structures: (a) Puzzle 52 in Eterna 100; (b) Puzzle 79 in Eterna 100; (c) Puzzle 10 in Rfam-taneda.

but perform better on simpler tasks. Figure 3 illustrates the target structures and hierarchical representations for Puzzles 52 and 79 in Eterna 100, and Puzzles 10 in Rfam-taneda.

Integration of DRAG with search-based methods

Direct comparisons with search-based methods are challenging due to their dependence on initial population (seeds) and multiple runs for successful designs. Nonetheless, we evaluated their performance on Eterna 100 v2. Due to server constraints, we were unable to replicate the 24-h experimental results presented in MoiRNAiFold's paper [25]. Instead, each puzzle was attempted 10 times, with a 3600-s limit per attempt using MoiRNAFold's server. For aRNAque, we used default parameters, performing 10 iterations, each with 600 generations and a population size of 100. This setup led to 73 successful designs for MoiRNAiFold and 78 for aRNAque, both lower than reported results.

Considering the complementary nature of RL and search-based methods, we integrate the EA from aRNAque into DRAG, creating a variant called DRAG-EVO. DRAG-EVO solves two more puzzles than DRAG, outperforming it among search-based methods. This suggests that combining EA with RL could expedite the evolutionary process. The trained agent predicts mutation probabilities, helping to select mutations and reduce randomness, improving efficiency. Future research can explore the integration of these methods further.

Ablation studies

To assess the impact of crucial components in DRAG, we perform ablation studies focusing on three key aspects: graph representations, hierarchical task division, and action restriction.

Ablation on graph representations of RNAs

The graph representation of RNA molecules is a pivotal feature of DRAG. As shown in Table 1, when compared with LEARNA, which adopts LSTM to model RNAs, DRAG exhibits significant advantages. In this experiment, we delve deeper into the influence of the refined graph representation utilized in DRAG. In this context, we train a modified version of DRAG that employs a simple graph representation, which uses one-hot encoding of the nucleotide type at RNA sites as node information. The edges in this graph solely correspond to the target structure, with equal edge weights uniformly set to 1. After each training epoch, we record the number of solved puzzles from the Eterna 100 v2 dataset, encompassing 100 rounds of design attempts. As indicated by the results shown in Fig. 4(a), the model with fine-grained representation outperforms its counterpart with simple topological graph representation. This signifies that the inclusion of detailed information through the fine-grained graph representation is beneficial for RNA design, empowering the model to learn and tackle more complex tasks effectively.

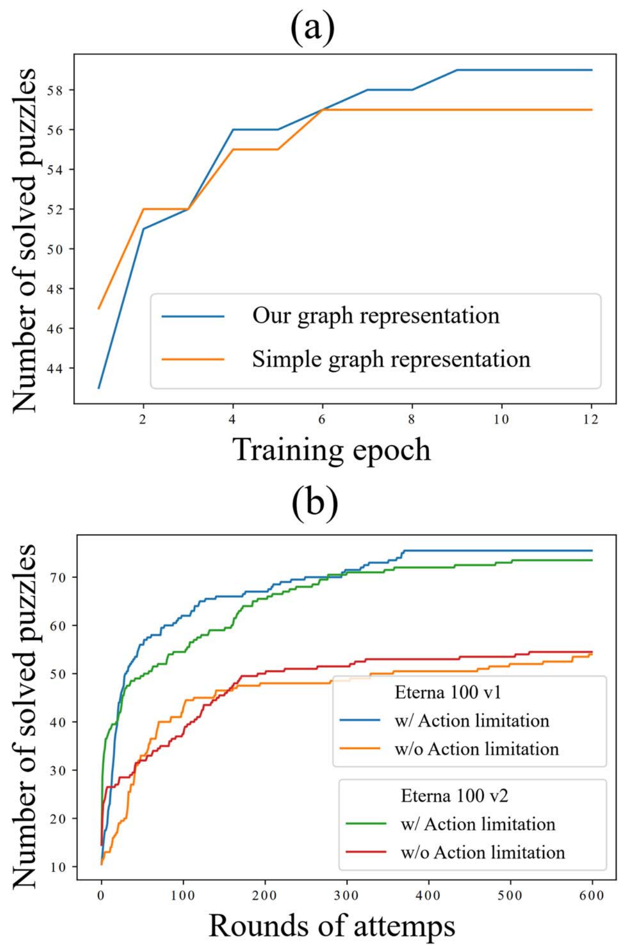

Figure 4. Results of ablation studies: (a) comparison between simple and refined graph representations, where x-axis denotes the number of training epochs and the y-axis indicates the number of puzzles resolved; (b) Comparison between with and without action restriction on two Eterna sets, where the x-axis denotes the rounds of design attempted and the y-axis reflects the count of puzzles resolved.

Ablation on action restrictions

In the process of designing RNA sequences, we impose constraints on the actions depending on their scope of influence. To investigate the influence of this mechanism on the efficacy of DRAG, we conduct an ablation experiment and the results are shown in Fig. 4(b).

The outcomes demonstrate that the agent equipped with action restrictions surpasses its counterpart lacking such restrictions in terms of success rates on both Eterna 100 v1 and v2. The influence of action restrictions is particularly evident in design tasks featuring intricate structures with multiple branches. By allowing agents to focus on pivotal sites that are more likely to align the MFE structure of the current sequence closer to the target structure, these restrictions not only elevate success rates but also reduce the number of rounds required.

Ablation on hierarchical task division

To illustrate the benefits of task division in RNA design, we generate a bar chart (Fig. 5) that displays the difference in the number of rounds to solve the puzzles both with and without task division. The chart reveals that task division consistently leads to a reduction in the rounds needed to solve most puzzles. Notably, three puzzles (77, 79, 80) are resolved exclusively by using the method that incorporates task division, highlighting its critical role. Additionally, for four other puzzles (16, 69, 70, 75), task division considerably decreases the number of rounds required. The secondary structure diagrams of these two groups of puzzles, which exhibit significant hierarchical depth, are shown in Supplementary Figs S3 and S4.

In simpler or shorter puzzles, no notable difference in performance is observed between the two methods. This occurs because simpler puzzles often yield better initial sequences that are already close to the target sequences. In the case of shorter structures, the initial subsequences may readily meet the requirements of the subtasks due to the slack threshold. Consequently, many actions within the subtasks become superfluous, diminishing the need for task division and making a direct design of the entire structure more effective. In certain instances, dividing simpler tasks leads to an increased number of rounds required to solve the puzzles. For example, puzzles 50, 51, and 83, which are highlighted by red bars in Fig. 5, show an increase in the number of rounds when task division is utilized. This can be attributed to the presence of multiple short and symmetric substructures within these puzzles (Supplementary Fig. S5), and there is competition among closely located substructures. The agent only detects this competition at higher level tasks, thus increasing the number of iteration rounds required.

Case study

To validate the biological significance of RNA sequences designed by DRAG, we conduct a case study focusing on phage tRNA design. Phage tRNA design is important in medicine and biotechnology, especially in the design of delivery systems [47, 48]. Using the T5 phage tRNA as a template, we target the secondary structure of Ec-tRNA-Tyr-GTA-1. We fix key motifs and conserved positions based on the findings of Azam *et al.* [49]. We design five tRNAs with distinct anticodons, specifically selected because these anticodons are rarely used in the host but are common in phage genomes. For each tRNA, we perform 100 epochs of design using DRAG and evaluate the sequences using tRNAscan-SE 2.0 [50]. The top 10 sequences from each design are further assessed for their folding free energy using RNAfold, and the maximum similarity among these 10 sequences is calculated. The average evaluation scores of the top 10 sequences for each tRNA variant are summarized in Table 2.

The tRNA sequences generated by DRAG exhibit high quality across multiple evaluation metrics. All sequences achieve tRNAscan-SE scores exceeding 50, indicating that DRAG successfully produces tRNA sequences with correct structural features and potential functionality. According to Chan *et al.* [50], tRNAscan-SE 2.0 considers sequences with scores above 50 to represent high-confidence, functional tRNA genes. The secondary structure of the generated sequences, as evaluated by RNAfold, aligns with that of the template Ec-tRNA-Tyr-GTA-1, including complete key domains such as the acceptor stem, D-loop, anticodon loop, and TΨC loop. Additionally, the sequences exhibit low folding free energy, reflecting their structural stability and proper folding. The GC content of the generated sequences falls within the reasonable range of 40%–60%, suggesting appropriate thermal stability and transcriptional efficiency. Furthermore, despite the constraints imposed by fixed motifs and conserved regions, the sequences demonstrate a notable degree of diversity. These results highlight DRAG's ability to generate biologically meaningful tRNA sequences, making them suitable for further experimental validation and potential applications in biotechnology.

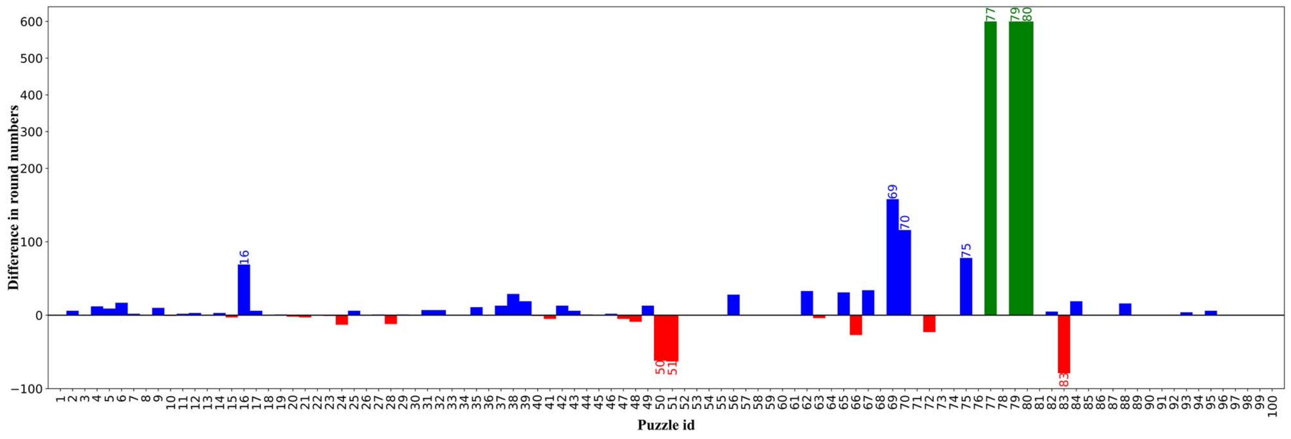

Figure 5. Comparison of average rounds per puzzle between methods with/without task division, where each bar represents a specific puzzle with height indicating rounds difference, with positive values (blue and green bars) denoting fewer rounds using task division.

Table 2. Average evaluation scores of the top 10 sequences for each tRNA variant

tRNA variant (Anticodon)	Average tRNAscan-SE score	Average free energy (kcal/mol)	Average GC content	Maximum pairwise similarity
GCU	56.19	−27.77	55.53%	61.18%
UCU	55.71	−27.49	54.47%	58.82%
UAU	56.56	−28.04	54.00%	57.65%
GUU	55.50	−27.37	54.82%	60.00%
CAU	53.16	−26.89	53.88%	60.00%

Discussion

We propose a novel approach for RNA design tasks, called DRAG, which is based on GNNs and RL. DRAG trains a deep learning model based on GNNs to predict the probability distribution of RNA mutation sites and mutation types using the RL framework. To our knowledge, DRAG is the first method combining GNN with RL in RNA design. It enhances RNA representation, strategy formulation for action selection, and the organization of complex tasks. Distinct from previous methods that generate or optimize sequences linearly along the RNA backbone, DRAG employs a mutation selection strategy that enhances both flexibility and efficiency. Additionally, it incorporates task division to further augment the design process, thereby improving computational efficiency and increasing success rates. We evaluate DRAG using established test sets for RNA sequence design. The results show that DRAG surpasses existing ML-based approaches, particularly displaying higher success rates in tackling lengthy and complex tasks.

Compared with other ML methods, the graph representation of RNAs and structure decomposition in DRAG demand increased memory allocation. Regarding time complexity, a substantial portion is consumed by the RNAfold functions employed during the design process. Our experiments indicate that RNAfold functions contribute to over 80% of the total design time. The frequent invocation of RNAfold is decisive in other RNA design methodologies as well, whether for evaluating designed RNA sequences or for their refinement. DRAG calls upon RNAfold more often to monitor the changes in the MFE structures of sequences at each iteration. This results in increased space complexity and extended runtime compared with alternatives like EternaBrain, LEARNA, and SentRNA during individual design attempts. However, over multiple iterations of design efforts for complex tasks, DRAG's time efficiency aligns closely with these baseline methods. This is largely due to our task division strategy, which effectively reduces the RNA length considered in RNAfold calculations. Additionally, the mutation-based design process, guided by intelligent agents, not only provides higher success rates but also offers greater flexibility in RNA design.

While DRAG is primarily designed for RNA, its framework holds the potential for adaptation to other biomolecules, such as DNA and proteins. In the case of DNA, single-stranded DNA (ssDNA) can form secondary structures similar to RNA, and DRAG could be adapted for designing ssDNA sequences by integrating DNA-specific secondary structure prediction tools. This adaptation might involve retraining or fine-tuning DRAG through transfer learning to account for the distinct structural properties of DNA. For proteins, the secondary structures are fundamentally different from RNA structures. Although DRAG's current framework is not directly applicable to designing protein sequences, its underlying principles, such as hierarchical representations and RL, could potentially be adapted. Such adaptations would require careful consideration of the unique characteristics and constraints of protein folding and function.

ML techniques offer significant benefits in terms of adaptability in RNA design. However, they do not necessarily outperform the most advanced search-based algorithms in terms of success rates. It is important to note that current assessments of design methodologies are somewhat limited, as they focus on the number of problems solved without taking into account the complexity of these problems. Moreover, certain strategies employed in search-based approaches can lead to considerable biases [51], and these methods are highly sensitive to initial conditions and the choice of seed sequences. Given the rapid evolution of ML, RNA design stands to gain from more sophisticated representation learning models. Additionally, the integration of EAs with ML is an exciting prospect that could markedly enhance the effectiveness of iterative sequence design procedures.

Key Points

- **Hierarchical graph representation (HGR) of RNAs:** we propose a novel HGR for RNA sequence design that captures node and topological information within RNA structures, revealing the arrangement of stem-loop substructures and facilitating the exploration of their interplay with the overall RNA structure.
- **Enhanced reinforcement learning strategies:** we develop two strategies to boost RL performance: (1) a composite reward function accounting for energy and structural discrepancy; and (2) task decomposition based on HGR, creating a hierarchical task pool and an RNA environment aligned with structural hierarchy.
- **Superior performance on benchmark datasets:** DRAG outperforms other ML-based approaches in RNA design on benchmark datasets, achieving the highest success rate.

Supplementary data

Supplementary data is available at *Briefings in Bioinformatics* online.

Funding

This work is supported by the National Natural Science Foundation of China (No. 62272300 and No. 62473257).

Data availability

https://github.com/LiYichong1996/DRAG.

References

1. Schmidt CM, Smolke CD. RNA switches for synthetic biology. *Cold Spring Harb Perspect Biol* 2019;**11**:a032532. https://doi.org/10.1101/cshperspect.a032532
2. Reynolds A, Leake D, Boese Q. *et al.* Rational siRNA design for RNA interference. *Nat Biotechnol* 2004;**22**:326–30. https://doi.org/10.1038/nbt936
3. Sahin U, Karikó K, Türeci Ö. mRNA-based therapeutics—developing a new class of drugs. *Nat Rev Drug Discov* 2014;**13**: 759–80. https://doi.org/10.1038/nrd4278
4. Chung YH, Beiss V, Fiering SN. *et al.* Covid-19 vaccine frontrunners and their nanotechnology design. *ACS Nano* 2020;**14**: 12522–37. https://doi.org/10.1021/acsnano.0c07197
5. Paunovska K, Loughrey D, Dahlman JE. Drug delivery systems for RNA therapeutics. *Nat Rev Genet* 2022;**23**:265–80. https://doi.org/10.1038/s41576-021-00439-4
6. Nguyen LT, Rananaware SR, Pizzano BLM. *et al.* Clinical validation of engineered CRISPR/Cas12a for rapid SARS-CoV-2 detection. *Commun Med* 2022;**2**:7. https://doi.org/10.1038/s43856-021-00066-4
7. Chappell J, Watters KE, Takahashi MK. *et al.* A renaissance in RNA synthetic biology: new mechanisms, applications and tools for the future. *Curr Opin Chem Biol* 2015;**28**:47–56. https://doi.org/10.1016/j.cbpa.2015.05.018
8. Bettini E, Locci M. SARS-CoV-2 mRNA vaccines: immunological mechanism and beyond. *Vaccine* 2021;**9**:147. https://doi.org/10.3390/vaccines9020147
9. Zhang H, Zhang L, Lin A. *et al.* Algorithm for optimized mRNA design improves stability and immunogenicity. *Nature* 2023;**621**: 396–403. https://doi.org/10.1038/s41586-023-06127-z
10. Reuter JS, Mathews DH. RNAstructure: software for RNA secondary structure prediction and analysis. *BMC Bioinformatics* 2010;**11**:1–9.
11. Lorenz R, Bernhart SH, Siederdissen CHZ. *et al.* ViennaRNA package 2.0. *Algorithms for molecular biology* 2011;**6**:1–14. https://doi.org/10.1186/1748-7188-6-26
12. Zhao Q, Zhao Z, Fan X. *et al.* Review of machine learning methods for RNA secondary structure prediction. *PLoS Comput Biol* 2021;**17**:e1009291. https://doi.org/10.1371/journal.pcbi.1009291
13. Chen C-C, Chan Y-M. Redfold: accurate RNA secondary structure prediction using residual encoder-decoder network. *BMC Bioinformatics* 2023;**24**:122. https://doi.org/10.1186/s12859-023-05238-8
14. Yang E, Zhang H, Zang Z. *et al.* GCNfold: a novel lightweight model with valid extractors for RNA secondary structure prediction. *Comput Biol Med* 2023;**164**:107246. https://doi.org/10.1016/j.compbiomed.2023.107246
15. Lee J, Kladwang W, Lee M. *et al.* RNA design rules from a massive open laboratory. *Proc Natl Acad Sci* 2014;**111**:2122–7. https://doi.org/10.1073/pnas.1313039111
16. Lee J, Kladwang W, Lee M. *et al.* RNA design rules from a massive open laboratory (vol 111, pg 2122, 2014). *Proc Natl Acad Sci* 2016;**111**:2122–0, 2127. https://doi.org/10.1073/pnas.1313039111
17. Anderson-Lee J, Fisker E, Kosaraju V. *et al.* Principles for predicting RNA secondary structure design difficulty. *J Mol Biol* 2016;**428**: 748–57. https://doi.org/10.1016/j.jmb.2015.11.013
18. Yao H-T, Ponty Y, Will S. Developing complex RNA design applications in the infrared framework. In: Lorenz R (ed.), *RNA Folding, Volume 2726 of Methods in Molecular Biology*. New York, NY: Humana, 2024. https://doi.org/10.1007/978-1-0716-3519-3_12.
19. Barnwal RP, Loh E, Godin KS. *et al.* Structure and mechanism of a molecular rheostat, an RNA thermometer that modulates immune evasion by neisseria meningitidis. *Nucleic Acids Res* 2016;**44**:9426–37. https://doi.org/10.1093/nar/gkw584
20. Barnwal RP, Yang F, Varani G. Applications of NMR to structure determination of RNAs large and small. *Arch Biochem Biophys* 2017 Nuclear Magnetic Resonance;**628**:42–56. https://doi.org/10.1016/j.abb.2017.06.003
21. Hofacker IL, Fontana W, Stadler PF. *et al.* Fast folding and comparison of RNA secondary structures. *Monatshefte fur chemie* 1994;**125**:167–88. https://doi.org/10.1007/BF00818163
22. Busch A, Backofen R. INFO-RNA—a fast approach to inverse RNA folding. *Bioinformatics* 2006;**22**:1823–31. https://doi.org/10.1093/bioinformatics/btl194
23. Weinbrand L, Avihoo A, Barash D. RNAfbinv: an interactive java application for fragment-based design of RNA sequences. *Bioinformatics* 2013;**29**:2938–40. https://doi.org/10.1093/bioinformatics/btt494
24. Kleinkauf R, Mann M, Backofen R. antaRNA: ant colony-based RNA sequence design. *Bioinformatics* 2015;**31**:3114–21. https://doi.org/10.1093/bioinformatics/btv319
25. Minuesa G, Alsina C, Garcia-Martin JA. *et al.* MoiRNAiFold: a novel tool for complex in silico RNA design. *Nucleic Acids Res* 2021;**49**: 4934–43. https://doi.org/10.1093/nar/gkab331

26. Merleau NSC, Smerlak M. A simple evolutionary algorithm guided by local mutations for an efficient RNA design. *Association for Computing Machinery* 2021;1027–34. https://doi.org/10.1145/3449639.3459280
27. Merleau NSC, Smerlak M. aRNAque: an evolutionary algorithm for inverse pseudoknotted RNA folding inspired by lévy flights. *BMC Bioinformatics* 2022;**23**:335. https://doi.org/10.1186/s12859-022-04866-w
28. Zhou T, Dai N, Li S. *et al.* RNA design via structure-aware multifrontier ensemble optimization. *Bioinformatics* 2023;**39**:i563–71. https://doi.org/10.1093/bioinformatics/btad252
29. Wirecki T, Lach G, Jaryani F. *et al.* DesiRNA: structure-based design of RNA sequences with a Monte Carlo approach. *Nucleic Acids Research* 2025;**53**:gkae1306. https://doi.org/10.1093/nar/gkae1306
30. Sumi S, Hamada M, Saito H. Deep generative design of RNA family sequences. *Nat Methods* 2024;**21**:435–43. https://doi.org/10.1038/s41592-023-02148-8
31. Joshi CK, Jamasb AR, Viñas R. *et al.* gRNAde: geometric deep learning for 3d RNA inverse design. *Methods Mol Biol* 2025;**2847**:121–35. https://doi.org/10.1007/978-1-0716-4079-1_8
32. Cheng T, Zhang Y, Gao Z. *et al.* Rdesign: hierarchical data-efficient representation learning for tertiary structure-based RNA design. *Briefings in Bioinformatics* 2025;**26**:bbae682. https://doi.org/10.1093/bib/bbae682
33. Huang H, Lin Z, He D. *et al.* Ribodiffusion: tertiary structure-based RNA inverse folding with generative diffusion models. *Bioinformatics* 2024;**40**:i347–56.
34. Shi J, Das R, Pande VS. SentRNA: Improving computational RNA design by incorporating a prior of human design strategies. *arXiv preprint* 2018;arXiv:1803.03146. https://arxiv.org/abs/1803.03146
35. Koodli RV, Keep B, Coppess KR. *et al.* EternaBrain: automated RNA design through move sets and strategies from an internet-scale RNA videogame. *PLoS Comput Biol* 2019;**15**:e1007059. https://doi.org/10.1371/journal.pcbi.1007059
36. Eastman P, Shi J, Ramsundar B. *et al.* Solving the RNA design problem with reinforcement learning. *PLoS Comput Biol* 2018;**14**:e1006176. https://doi.org/10.1371/journal.pcbi.1006176
37. Runge F, Stoll D, Falkner S. *et al.* Learning to design RNA. In: Bengio Y, Larochelle H. (eds.), *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*. New Orleans, LA, USA: OpenReview.net; 2019. Available from: https://openreview.net/forum?id=ByldsAqtwQ.
38. Obonyo S, Jouandeau N, Owuor D. Designing RNA sequences by self-play. In: Merelo JJ, Lam HK. (eds.), *Proceedings of the 14th International Joint Conference on Computational Intelligence (IJCCI 2022)*. Setúbal, Portugal: SciTePress; 2022, 305–12. https://doi.org/10.5220/0011550300003332
39. Runge F, Franke J, Fertmann D. *et al.* Partial RNA design. *Bioinformatics* 2024;**40**:i437–45. https://doi.org/10.1093/bioinformatics/btae222
40. Graves A, Graves A. Long short-term memory. *Neural Computation* 2012;**9**:1735–80. https://doi.org/10.1007/978-3-642-24797-2_4
41. Pennington J, Socher R, Manning CD. Glove: global vectors for word representation. In: Moschitti A, Pang B, Yih SW. (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014, 1532–43. https://aclanthology.org/D14-1162
42. Esmaili-Taheri A, Ganjtabesh M, Mohammad-Noori M. Evolutionary solution for the RNA design problem. *Bioinformatics* 2014;**30**:1250–8. https://doi.org/10.1093/bioinformatics/btu001
43. Rubio-Largo Á, Escobar-Encinas L, Lozano-García N. *et al.* Evolutionary strategy to enhance an RNA design tool performance. *IEEE. Access* 2024;**12**:15582–93. https://doi.org/10.1109/ACCESS.2024.3358426
44. Xu K, Hu W, Leskovec J. How powerful are graph neural networks? In: Bengio Y, Larochelle H. (eds.), *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*. New Orleans, LA, USA: OpenReview.net, 2019, 1–17. https://openreview.net/forum?id=ryGs6iA5Km
45. Schulman J, Wolski F, Dhariwal P. *et al.* Proximal policy optimization algorithms. *arXiv preprint* 2017;arXiv:1707.06347. https://doi.org/10.48550/arXiv.1707.06347
46. Koodli RV, Rudolfs B, Wayment-Steele HK. *et al.* Redesigning the eterna100 for the Vienna 2 folding engine. *BioRxiv preprint* 2021. https://doi.org/10.1101/2021.08.26.457839
47. Guerrero-Bustamante CA, Hatfull GF. Bacteriophage tRNA-dependent lysogeny: requirement of phage-encoded tRNA genes for establishment of lysogeny. *MBio* 2024;**15**:e03260–23.
48. Cui L, Watanabe S, Miyanaga K. *et al.* A comprehensive review on phage therapy and phage-based drug development. *Antibiotics* 2024;**13**:870. https://doi.org/10.3390/antibiotics13090870
49. Azam AH, Kondo K, Chihara K. *et al.* Evasion of antiviral bacterial immunity by phage tRNAs. *Nat Commun* 2024;**15**:9586. https://doi.org/10.1038/s41467-024-53789-y
50. Chan PP, Lin BY, Mak AJ. *et al.* tRNAscan-SE 2.0: improved detection and functional classification of transfer RNA genes. *Nucleic Acids Res* 2021;**49**:9077–96. https://doi.org/10.1093/nar/gkab688
51. Bangyal WH, Nisar K, Ag AAB. *et al.* Comparative analysis of low discrepancy sequence-based initialization approaches using population-based algorithms for solving the global optimization problems. *Appl Sci* 2021;**11**:7591. https://doi.org/10.3390/app11167591