CoCoRNA: Collective RNA Design with Cooperative Multi-agent Reinforcement Learning

Tianmeng Hu [1 2] **Biao Luo** [3] **Ke Li** [1 2]

Abstract

Designing RNA sequences that reliably fold into specific secondary structures is essential for understanding their biological functions but remains a challenging computational problem. We propose CoCoRNA, a cooperative multi-agent reinforcement learning framework for RNA inverse design. CoCoRNA simplifies the design task by decomposing it into smaller sub-problems, each solved collaboratively by multiple agents. This approach reduces the complexity of the problem and improves the exploration of design policies. During training, a centralized critic uses global structural information to guide the agents, enabling them to jointly optimize their design strategies. As a result, CoCoRNA learns high-quality RNA design policies that generalize effectively to unseen structures without additional training. Experiments on the `Rfam` dataset demonstrate that CoCoRNA substantially outperforms state-of-the-art methods in both success rate and design speed. Further experiments on other biological sequence design tasks highlight the effectiveness and broad potential of CoCoRNA for complex design tasks.

1. Introduction

RNA is a polymeric molecule composed of nucleotides. Each nucleotide contains one of four bases: adenine (`A`), guanine (`G`), cytosine (`C`), or uracil (`U`). These bases carry genetic information. Nucleotides fold into complex secondary structures following Watson-Crick and Wobble base-pairing rules. Common structural elements include stems, loops, bulges, and junctions. For RNA, secondary structures are often more critical for function than they are in proteins (Kwon, 2025). These structures participate directly in catalysis, ligand recognition, and molecular scaffolding across diverse RNA classes (Vandivier et al., 2016; Bose et al., 2024). Examples include ribosomal RNAs (Ramakrishnan, 2014), messenger RNAs (Rouskin et al., 2014), and small nuclear RNAs (Fica et al., 2013). Secondary structures also regulate translation initiation and influence RNA polymerase II activity, thus controlling gene translation and splicing (Georgakopoulos-Soares et al., 2022). However, the precise roles of RNA secondary structures in many biological processes remain unclear (Georgakopoulos-Soares et al., 2022). To fully understand RNA functions, researchers need reliable methods to investigate and manipulate these structures (Vandivier et al., 2016). Computational inverse design can address this need, guiding experiments and reducing costs associated with testing structural hypotheses (Dotu et al., 2013; Morandi et al., 2022).

Computational RNA secondary structure design is essentially an inverse problem. The goal is to find RNA sequences that fold into a specific secondary structure with minimum free energy. This problem was first introduced in the early 1990s (Hofacker et al., 1994). It is mathematically ill-posed in the Hadamard sense because multiple distinct sequences can fold into the same structure, as illustrated in Figure 1. Solving this inverse design problem is critical for advancing functional RNA research (Taft et al., 2010; Serganov & Nudler, 2013). Key applications include synthetic RNA biology (Dykstra et al., 2022) and RNA nanostructure design (Guo, 2010).

A major challenge in computational RNA design is the exponential growth of the sequence space. For an RNA sequence of length ℓ, there are 4^ℓ possible sequences. However, only a small fraction of these sequences will fold into the desired structure. For example, a 74-nucleotide tRNA with 40 paired and 34 unpaired positions results in a search space of approximately 10^{36} sequences. This enormous search space makes brute-force approaches computationally infeasible.

Stochastic optimization is a widely used approach for computational RNA design. These methods typically rely on RNA folding prediction tools, such as ViennaRNA's RNAfold (Lorenz et al., 2011), mfold/UNAFold (Markham

[1]Living Systems Institute, University of Exeter, United Kingdom [2]Department of Computer Science, University of Exeter, United Kingdom [3]School of Automation, Central South University, China. Correspondence to: Biao Luo <biao.luo@hotmail.com>, Ke Li <k.li@exeter.ac.uk>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

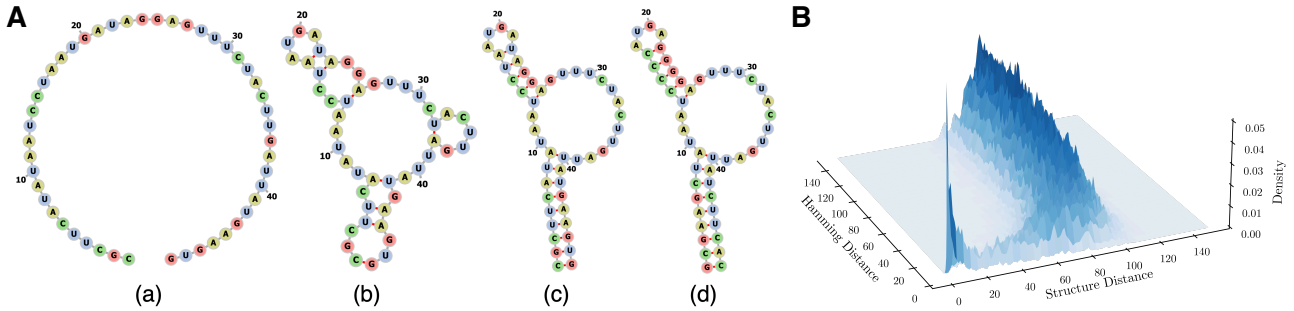

Figure 1. **A**. Examples of RNA sequences and structures visualized using Forna (Kerpedjiev et al., 2015). (a) is an unfolded RNA sequence. Structures (b) and (c) show two different folded forms of this sequence. Structure (c) is specifically generated by RNAfold (Lorenz et al., 2011), which uses a minimum free energy algorithm. Structure (d) is folded from a different sequence, yet it shares the same secondary structure as (c). **B**. Structure density surface for RNA sequences of length 150. The plot illustrates the distribution of Hamming distances between RNA sequences and their corresponding secondary structure distances.

& Zuker, 2008), and RNAstructure (Ali et al., 2023). These tools predict RNA structures by minimizing free energy according to established thermodynamic parameters (Mathews et al., 1999; 2004). The design process begins with an initial seed sequence. Techniques such as adaptive random walks (Hofacker et al., 1994) or stochastic local search iteratively mutate and evaluate candidate sequences until a suitable solution emerges. Well-known algorithms in this domain are RNA-SSD (Andronescu et al., 2004) and INFORNA (Busch & Backofen, 2006). Other effective strategies draw from evolutionary algorithms (Taneda, 2012; Esmaili-Taheri et al., 2014; Esmaili-Taheri & Ganjtabesh, 2015), constraint programming (Garcia-Martin et al., 2013; 2015), ant colony optimization (Kleinkauf et al., 2015), and ensemble optimization (Zhou et al., 2023).

Neural combinatorial optimization (NCO) has recently emerged as a powerful approach for combinatorial problems. Examples include the traveling salesman problem (Bello et al., 2016), maximum cut (Barrett et al., 2020), and bin packing (Jiang et al., 2023). NCO combines the representational strength of deep neural networks with the search efficiency of reinforcement learning (RL). In computational RNA design, Eastman et al. (2018) first introduced a pretrained RL policy. Their policy learned RNA folding rules to guide the search process. Experiments on the Eterna100 benchmark (Anderson-Lee et al., 2016) demonstrated encouraging performance compared to stochastic optimization methods. Later, Runge et al. (2019) proposed LEARNA, an RL-based approach that directly generates RNA sequences, enhancing success rates and computational efficiency.

Despite these advances, RL-based RNA inverse design still faces three major challenges:

- **Combinatorial explosion and sample inefficiency:** The RNA design space grows exponentially with sequence length, requiring enormous sample sizes. As a result, RL models struggle to accurately estimate value functions from limited data.
- **Sparse rewards and local optima:** Very few sequences exactly match the target structure, creating sparse rewards. Additionally, the RNA design landscape contains numerous local optima (see Figure 1). This combination makes policy improvement challenging.
- **Delayed rewards and misleading auxiliaries:** Strict success feedback in RNA inverse design is only available when a correct sequence is found. Random exploration rarely finds these sequences, causing most attempts to yield no useful feedback. Long task horizons further complicate intermediate evaluations. Some methods (Angermueller et al., 2020; Runge et al., 2019) use auxiliary rewards to guide exploration. However, these signals can overshadow the primary objective in long tasks, causing policies to overfit short-term gains and become trapped in local optima.

These challenges significantly limit the effectiveness of current RL methods (Eastman et al., 2018; Runge et al., 2019). As demonstrated by our experiments in Section 4, existing approaches typically require substantially longer computation time, yet achieve only modest success rates. Furthermore, once algorithms become trapped in local optima, additional search efforts yield diminishing returns.

Conceptually, navigating the RNA sequence space resembles exploring a complex maze. Multiple explorers, each starting from different points and sharing their discoveries, can map this maze more efficiently than a single explorer. Inspired by this analogy, we propose using multi-agent reinforcement learning (MARL) to tackle computational RNA design.

We formulate computational RNA design as a collec-

tive task, where multiple agents cooperate to generate RNA sequences. To our knowledge, this is the first study to frame biological sequence design explicitly as a multi-agent decision-making problem and the first to apply MARL in this domain.

- We develop CoCoRNA, a MARL framework designed for this collective RNA design task. CoCoRNA instantiates cooperative MARL for RNA inverse design through a Dec-POMDP formulation over joint sequence–structure states, RNA-aware decomposition schemes, and a search-augmented exploration mechanism to enhance learning efficiency.
- We rigorously evaluate CoCoRNA on the `Rfam` dataset, along with extensive ablation studies. Under identical time constraints, CoCoRNA achieves a 97.75% success rate, significantly surpassing the best baseline by 27.25 percentage points (38.7% relative improvement). Even when the strongest baseline receives 20 times more computational time, CoCoRNA still maintains an 8.21 percentage point lead (9.2% relative advantage). Importantly, CoCoRNA achieves an average solution time of only 3.24 seconds, which is over 70 times faster than the best-performing baseline.

2. Preliminaries

RNA secondary structure representation In computational biology, RNA secondary structures are commonly represented using dot-bracket notation. Unpaired nucleotides appear as dots (`.`), while base pairs are indicated by matching parentheses. An opening parenthesis (`(`) marks the $5'$ end of a base pair, and a closing parenthesis (`)`) marks the $3'$ end. For example, `((...))` represents a hairpin loop, with paired bases at both ends and unpaired nucleotides in the middle.

Problem formulation In RNA inverse design, the input is a target secondary structure rather than a source RNA sequence, and the goal is to generate RNA sequences whose predicted folded structures match the target. This setting is motivated by practical applications, where researchers often start from a desired structural motif and seek sequences that can realize it. Moreover, because multiple distinct RNA sequences can fold into the same secondary structure, inverse design also enables optimization of additional properties such as GC content and minimum free energy while preserving the target fold.

Decentralized partially observable Markov decision process We model the fully cooperative MARL problem as a decentralized partially observable Markov decision process (Oliehoek et al., 2008). Formally, this is defined by the tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{O}^i, \mathcal{A}^i, P, R, \gamma \rangle$, where:

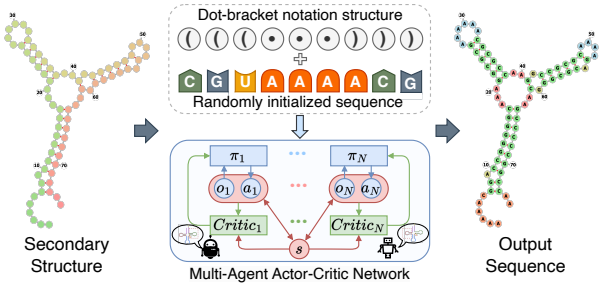

Figure 2. Overview of the CoCoRNA workflow.

- $\mathcal{N} = \{1, \ldots, N\}$: The set of N agents.
- $\mathcal{S}$: The global state space.
- $\mathcal{O}^i$: Local observation space of agent i.
- $\mathcal{A}^i$: The action space for agent i.
- $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$: State transition probability.
- $R : \mathcal{S} \times \mathcal{A} \to [R_{\min}, R_{\max}]$: Shared reward function.
- $\gamma \in [0, 1]$: The discount factor.

At each time step t, the environment is in state $s_t \in \mathcal{S}$, and each agent $i \in \mathcal{N}$ observes $o_t^i \in \mathcal{O}^i$ and selects $a_t^i \in \mathcal{A}^i$ according to $\pi^i(a_t^i \mid o_t^i)$. The joint action is $\boldsymbol{a}_t = (a_t^1, \ldots, a_t^N)$. The environment transitions to s_{t+1} with probability $P(s_{t+1} \mid s_t, \boldsymbol{a}_t)$, and all agents receive the shared reward $R(s_t, \boldsymbol{a}_t)$. The goal is to find $\boldsymbol{\pi} = (\pi^1, \ldots, \pi^N)$ maximizing $G = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \boldsymbol{a}_t)\right]$.

Centralized training with decentralized execution (CTDE) CTDE is a widely used paradigm in MARL (Rashid et al., 2018; Yu et al., 2022; Hu et al., 2026b). During training, agents access global information to evaluate and improve their policies; during execution, each agent acts only from local observations. This mitigates non-stationarity and improves scalability.

Actor-Critic methods with CTDE In the standard Actor-Critic framework (Barto et al., 1983), the Critic estimates the value function, while the Actor learns and updates its policy using the Critic's value estimates. Under CTDE, this naturally extends to centralized Critics and decentralized Actors (Lowe et al., 2017; Iqbal & Sha, 2019; Yu et al., 2022): during training, the Critic leverages global information to guide each agent's Actor; at execution time, agents act independently from local observations.

3. Proposed Method

The workflow of CoCoRNA is presented in Figure 2 and its pseudocode is provided in Appendix A. In the following

paragraphs, we will delineate its key algorithmic components step by step.

3.1. Problem decomposition

In a canonical MDP, a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ maps states to actions. Ideally, the state space $\mathcal{S}$ should capture all information relevant for decision-making. RNA design decisions depend primarily on two factors: the target RNA secondary structure, and the current RNA sequence. These factors matter because nucleotide positions have interdependencies due to base-pairing constraints. Thus, the state space $\mathcal{S}$ must combine RNA sequence and structural information:

$$\mathcal{S} = \Big\{ (s_{\texttt{seq}}, s_{\texttt{struct}}) \;\Big|\; s_{\texttt{seq}} \in \{\texttt{A}, \texttt{U}, \texttt{G}, \texttt{C}\}^{\ell}, \; s_{\texttt{struct}} \in \{\texttt{.}, \texttt{(}, \texttt{)}\}^{\ell} \Big\}. \tag{1}$$

where $s_{\texttt{seq}}$ is the nucleotide sequence, and $s_{\texttt{struct}}$ is the target secondary structure in dot-bracket notation. For sequences of length ℓ, the total number of possible states is $|\mathcal{S}| = (4 \times 3)^{\ell}$, covering all possible sequence-structure combinations. Such exponential growth results in a high-dimensional policy space, complicating policy learning and exploration.

To disentangle this complexity, we propose using MARL within the CTDE framework. We decompose the RNA design problem into multiple subproblems, assigning them to different agents who solve these tasks cooperatively. Each agent then focuses only on specific positions or substructures. This targeted approach reduces the dimensionality of each agent's state and policy spaces. In this work, we propose two decomposition schemes:

- **Position-based decomposition**: The RNA sequence is split into n segments, each assigned to a distinct agent. Each agent decides nucleotides exclusively for its segment.
- **Structure-type-based decomposition**: Each agent is assigned to a particular structural type (e.g., stems, loops). Agents design nucleotides solely at positions corresponding to their structural assignment.

Each agent receives a lower-dimensional local observation containing only a fragment of the RNA sequence and structure around its current position. Formally, an agent's observation space $\mathcal{O}$ is:

$$\mathcal{O} = \Big\{ (o_{\text{seq}}, o_{\text{struct}}) \;\Big|\; o_{\text{seq}} \in (\{\texttt{A}, \texttt{U}, \texttt{G}, \texttt{C}, \emptyset\})^{m}, \; o_{\text{struct}} \in (\{\texttt{.}, \texttt{(}, \texttt{)}, \emptyset\})^{m} \Big\}, \tag{2}$$

where m is the length of the observation window (an odd integer). The symbol $\emptyset$ serves as a placeholder for positions beyond sequence boundaries. For agent i, its local observation o_t^i is:

$$o_t^i = \left(s_{\texttt{seq}}^{[i-\kappa,\, i+\kappa]},\; s_{\texttt{struct}}^{[i-\kappa,\, i+\kappa]} \right), \tag{3}$$

where $\kappa = \frac{m-1}{2}$. o_t^i consists of the local nucleotide and structural subsequences centered at position i. To handle variable-length inputs, sequences are padded with $\emptyset$ to a maximum length $l_{\max}$. This local observation provides sufficient context for the agent's decision.

Although our primary implementation uses dot-bracket notation, the CoCoRNA framework only requires a computable state representation and can therefore be instantiated with alternative encodings (e.g., a base-pairing matrix). In Section 4, we further implement a matrix-based variant that uses a three-channel input (Structure, Sequence, Position) processed by a CNN, enabling CoCoRNA to naturally handle pseudoknots.

By decomposing the complex RNA design task into smaller subproblems, we leverage the collective strengths of MARL. Each agent specializes either in a sequence segment or a specific structural type, enabling more focused and efficient policy learning.

3.2. Policy optimization

We adopt multi-agent proximal policy optimization (MAPPO) (Yu et al., 2022) as our base algorithm due to its robust performance in various MARL tasks. Our framework provides each agent with two distinct networks: an Actor for decision-making and a Critic for policy evaluation. Specifically, the Actor is implemented as an MLP, while the Critic uses a CNN-based architecture to process the global sequence–structure state. Importantly, each agent maintains its own network parameters, enabling independent learning and specialized adaptation to its specific design role.

Actor network Each agent's Actor network receives a local observation o_t^i as input, which provides relevant context around the agent's assigned position. The network then outputs a probability distribution over the discrete action space:

$$\mathcal{A} = \{\texttt{A},\ \texttt{U},\ \texttt{G},\ \texttt{C}\}, \tag{4}$$

where each action corresponds to selecting a nucleotide type. We denote agent i's policy as $\pi_{\theta^i}(a_t^i \mid o_t^i)$, where θ^i denotes the parameters of its Actor.

Base-pair compatibility is enforced by the environment as a hard constraint. When a position whose paired partner has already been assigned is to be filled, the environment applies a validity mask so that only nucleotides compatible with the existing assignment remain admissible according to Watson-Crick and wobble pairing rules (A–U, G–C, and G–U). For example, if one side of a pair has already been assigned

G, the paired position can only take C or U. The Actor still outputs a probability distribution over all four nucleotides, but invalid actions are masked out before execution.

During interaction with the environment, each agent selects an action a_t^i from the valid action set and executes it by placing the corresponding nucleotide.

Critic network The Critic network takes the global state s_t as input, including both the current RNA sequence and its target secondary structure. We process these two components separately using two convolutional modules. Their outputs are concatenated and fed into subsequent Critic layers to produce a scalar value estimate $V(s_t)$. This estimate reflects the expected cumulative future rewards under the joint policy $\boldsymbol{\pi}$: $V^{\boldsymbol{\pi}}(s_t) = \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k} \,\middle|\, s_t\right]$. Using the Critic's value estimates, we calculate the temporal difference (TD) error $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$. Next, we apply generalized advantage estimation (GAE) (Schulman et al., 2015) to compute advantage estimates $A_t = \sum_{k=0}^{\infty}(\gamma\lambda)^k\delta_{t+k}$, where $\lambda \in [0, 1]$. These advantage estimates guide updates to each agent's Actor parameters θ^i. We follow the standard MAPPO update rule. The convergence of our algorithm is established by the following lemma, assuming standard conditions for policy gradient convergence are satisfied (Konda & Tsitsiklis, 1999; Sutton et al., 1999). The detailed proof is in Appendix B.

Lemma 3.1. *Consider* CoCoRNA*'s multi-agent actor-critic algorithm, where a centralized Critic estimates the joint advantage function, and decentralized policies are updated according to the gradient:*

$$g = \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_i \nabla_{\theta^i}\log\pi^i(a^i \mid o^i)\, A(s, \boldsymbol{a})\right]. \tag{5}$$

Suppose the joint policy $\boldsymbol{\pi}$ is differentiable, uses an appropriate learning rate, and the Critic is compatible with $\boldsymbol{\pi}$ (Konda & Tsitsiklis, 1999; Sutton et al., 1999). Then, with probability 1, the algorithm converges to a local maximum of the expected joint return $J^{\boldsymbol{\pi}}$. That is:

$$\liminf_{k\to\infty}\left\|\nabla_\theta J^{\boldsymbol{\pi}}(\theta_k)\right\| = 0 \quad \textit{w.p. 1.} \tag{6}$$

The centralized Critic uses global information to estimate values, guiding local policy updates of each agent. This strategy ensures individual actions align effectively with the team objective. By combining centralized value estimation and decentralized decision-making, we promote cooperative behavior despite agents acting independently based on local observations.

3.3. Reward function

We formulate RNA inverse design as a MDP which treats RNA design as a navigation task through the large design space. Our reward function is inspired by standard navigation problems, explicitly encouraging both exploration and optimization.

First, we use the ViennaRNA (Lorenz et al., 2011), a minimum free energy (MFE) folding algorithm, to predict the secondary structure $\mathbf{x}_f$ of the current RNA sequence. We then compute the Hamming distance, $H(\mathbf{x}_f, \mathbf{x}_t)$, between this predicted structure and the target structure $\mathbf{x}_t$ as: $H(\mathbf{x}_f, \mathbf{x}_t) = \sum_{i=1}^{\ell}\mathbb{1}(\mathbf{x}_f^i \neq \mathbf{x}_t^i)$, where $\mathbb{1}(\cdot)$ is an indicator function returning 1 if $\mathbf{x}_f^i \neq \mathbf{x}_t^i$, and 0 otherwise.

Reward function consists of two components: an intermediate reward and a final reward. At each step t, the multi-agent team selects actions, and the environment provides an intermediate reward reflecting the normalized reduction in Hamming distance from the previous time step. If the Hamming distance reaches 0, it means the predicted structure exactly matches the target structure. We then assign a substantial final reward C, and the episode terminates. Formally, the reward R_t is defined as:

$$R_t = \begin{cases} (H_{t-1} - H_t)/\ell, & \text{if } H_t > 0, \\ C, & \text{if } H_t = 0, \end{cases} \tag{7}$$

where $C > 0$ is a constant. The intermediate reward encourages stepwise improvements, rewarding incremental reductions in structural differences. The substantial final reward C strongly incentivizes agents to achieve a perfect match, thereby aiding them in overcoming local optima.

We further design multi-objective variants of the reward function. Specifically, we add an auxiliary term r_{aux} that is only activated when the structural design is successful (i.e., $H_t = 0$); otherwise, $r_{\text{aux}} = 0$. For example, to steer the GC content toward a target c^t, we use $r_{\text{GC}} = -\alpha|c_{\text{GC}} - c^t|$, where $c_{\text{GC}} \in [0, 1]$ is the normalized GC content and $\alpha > 0$ is a scaling factor. To optimize minimum free energy, we use $r_{\text{MFE}} = -\beta E$, where E is the MFE predicted by a folding engine and $\beta > 0$ is a scaling factor. We provide empirical results of these variants in Section 4.

3.4. Search-augmented exploration

The enormous RNA design space often causes cold-start issues for learning-based methods (Wang et al., 2024). In the early training stages, random policies often generate low-quality experiences, which limit learning efficiency. To address this, we introduce a heuristic called search-augmented exploration (SAE). SAE employs a limited greedy search early in training to enhance the quality of experiences stored in the replay buffer. Specifically, after an agent selects an action a based on its current policy π and receives reward r, we perform a brief local search. This search checks whether an alternative action a' can yield a higher reward r'. If such an improved action is found, we replace the orig-

inal experience tuple (o, a, r) with (o, a', r'). By improving data quality early on, SAE accelerates the agents' learning progress.

In practice, CoCoRNA applies SAE during the first 30% of training, effectively mitigating cold-start challenges by exposing agents to more rewarding experiences earlier. Importantly, SAE does not introduce additional learning objectives or modify the original optimization goals. Therefore, SAE preserves MAPPO's convergence guarantees, as detailed in Section 3.2 and Appendix B.

4. Empirical Study

4.1. Experiments on RNA inverse design

Dataset We evaluate CoCoRNA on RNA secondary structures from the `Rfam` dataset (Runge et al., 2024a). We compute pairwise edit distances among $65,000$ structures and apply mini-batch K-means to partition them into 100 clusters. One cluster (650 structures) serves as the test set, while the remaining data are filtered to remove structures highly similar to the test set, yielding $60,000$ training structures. The minimum edit distance between training and test structures is 111, with an average of 188.42. To ensure robustness and prevent data leakage, we also construct six alternative splits by randomly selecting different clusters as test sets (details in Appendix C.1).

Experimental setup We use position-based decomposition with $n = 4$ agents, where each RNA structure is divided into four subsequences of approximately equal length, each assigned to one agent. An episode has at most 400 steps and is considered successful if the agents collectively generate a sequence that folds into the target structure. We train the model for a total of 25 million steps. During evaluation, we generate sequences for each test structure with up to 15 attempts under a 30-second time limit. Network architectures and detailed hyperparameters are provided in Appendix F.

Metrics & baselines We use success rate and diversity as primary metrics. Success is an exact structural match (Hamming distance $= 0$), and diversity is the average pairwise normalized Hamming distance among generated sequences. We compare CoCoRNA with five state-of-the-art baselines: `antaRNA` (Kleinkauf et al., 2015), `LEARNA` and `Meta-LEARNA-Adapt` (Runge et al., 2019), `SAMFEO` (Zhou et al., 2023), `libLEARNA` (Runge et al., 2024b) and `OmniGenome` (Yang et al., 2025). `OmniGenome` is a recent large-scale foundation model with 186M parameters, functioning as a generative model assisting genetic algorithms for RNA inverse design. `antaRNA` and `SAMFEO` are stochastic optimization methods, both requiring significant optimization time. `LEARNA` and its meta-learning variant, `Meta-LEARNA-Adapt`, perform online learning per sequence, demanding instance-specific policy training and additional computational resources. To ensure fair efficiency comparisons, we evaluate these baselines under multiple time constraints. Note that we give CoCoRNA the most strict time limit 30 seconds in all experiments.

CoCoRNA achieves the highest success rate Table 1 summarizes the results over five runs. Under a strict 30-second limit, CoCoRNA achieves 97.75% success, improving by 27.25 absolute percentage points (38.7% relative) over the best baseline at the same budget (`SAMFEO-30s`). Even with 600s budgets (20× longer), CoCoRNA still outperforms `Meta-LEARNA-Adapt-600s` by 8.21 points (9.2% relative), establishing CoCoRNA as a highly reliable method.

CoCoRNA is highly efficient Beyond accuracy, CoCoRNA is highly efficient: it finds solutions in 3.24 seconds on average, over 70× faster than `Meta-LEARNA-Adapt` (234.2 seconds at the 600-second limit). Figure 3 shows the time distribution and its dependence on sequence length, confirming that CoCoRNA is consistently faster than all baselines. Notably, `OmniGenome` (186M parameters) does not outperform classical stochastic optimizers, and `antaRNA` performs poorly (success rate 26.37% at $1,200$s), often getting stuck in local optima.

CoCoRNA maintains good diversity In addition to high success and efficiency, CoCoRNA produces diverse solutions. As shown in Table 1, CoCoRNA achieves the highest average pairwise normalized Hamming distance (0.405), indicating broader exploration of the sequence space.

Long RNA design and family-wise results CoCoRNA is not inherently limited by sequence length. While our implementation uses a fixed input dimension (450 nt), longer sequences can be handled via a sliding-window representation. To demonstrate scalability, we tested the original CoCoRNA model (without retraining) on 500 RNA sequences from Rfam ranging from 500 nt to 971 nt, using a sliding-window approach (a 600s time limit, repeated 5 times). Despite being evaluated on sequences significantly longer (up to $\sim 2\times$) than those seen during training, CoCoRNA achieves a 92.4% success rate, highlighting strong zero-shot generalization.

Matrix-based CoCoRNA and pseudoknot design The dot-bracket representation used in our primary implementation cannot naturally express pseudoknots; however, the CoCoRNA framework is representation-agnostic and can be instantiated with alternative encodings. As shown in Figure 4 (middle), our matrix-based variant achieves a 96.95% success rate, which is comparable to the dot-bracket implementation. We further evaluate the matrix-based variant

Table 1. Evaluation of CoCoRNA and baseline methods on the RNA inverse design task. The time limit for CoCoRNA is set to 30 seconds. The suffixes in baseline names indicate different time constraints; for example, "−600s" denotes a time limit of 600 seconds per structure.

Method	Number of solved samples	Mean success rate	Average diversity
CoCoRNA	**635.40 ± 2.58**	**97.75%**	**0.405**
LEARNA-30s	63.80 ± 3.03	9.82%	0.311
LEARNA-300s	368.20 ± 6.18	56.65%	0.339
LEARNA-600s	382.20 ± 4.82	58.80%	0.336
Meta-LEARNA-Adapt-30s	381.20 ± 5.63	58.65%	0.377
Meta-LEARNA-Adapt-300s	564.20 ± 4.09	86.80%	0.392
Meta-LEARNA-Adapt-600s	582.00 ± 4.98	89.54%	0.400
libLEARNA-30s	440.20 ± 7.16	67.72%	0.385
libLEARNA-300s	554.60 ± 3.58	85.32%	0.351
libLEARNA-600s	583.60 ± 2.97	89.79%	0.371
SAMFEO-30s	458.20 ± 6.10	70.50%	0.361
SAMFEO-60s	488.60 ± 7.64	75.17%	0.388
SAMFEO-600s	560.00 ± 2.76	86.15%	0.394
antaRNA-60s	87.80 ± 2.70	13.51%	0.361
antaRNA-600s	110.00 ± 6.67	16.92%	0.390
antaRNA-1200s	171.40 ± 5.03	26.37%	0.387
OmniGenome-60s	261.4 ± 4.04	40.22%	0.389
OmniGenome-600s	502.20 ± 4.21	77.26%	0.380
OmniGenome-1200s	532.40 ± 4.51	81.91%	0.388

Table 2. Multi-objective design with CoCoRNA.

Objectives	Success rate	GC	MFE
Struct.	97.75%	0.62	−125.1
Struct. + GC	97.20%	0.51	–
Struct. + MFE	97.07%	–	−158.5
Struct. + GC + MFE	97.26%	0.53	−151.2

on a pseudoknotted dataset (Kleinkauf et al., 2015), with IPknot (Sato et al., 2011) as the oracle and fine-tune it using only 10% of the original training budget. On three held-out test sets, CoCoRNA achieves mean success rate 92.67%, outperforming antaRNA by over 28 percentage points.

Multi-objective design We further study multi-objective reward variants that control GC content (r_{GC}, target $c^t = 0.5$) and minimize free energy (r_{MFE}), where auxiliary rewards are applied only when the structural design is successful (see Section 3.3). Table 2 shows that CoCoRNA can steer GC content and improve MFE without sacrificing structural accuracy.

CoCoRNA generalizes across different oracles We train CoCoRNA with ViennaRNA and evaluate using RibonanzaNet (He et al., 2024), a deep learning–based structure predictor trained on large-scale experimental data. As shown in Figure 4, CoCoRNA achieves 94.90% success; the slight drop vs. ViennaRNA is expected, but the strong performance indicates robust generalization beyond a single oracle.

Family-wise performance We construct four family-specific test sets (500 structures each, deduplicated from Rfam seed). Figure 4 (right) shows results on five families: CoCoRNA achieves $> 99\%$ success on three, while ribozymes are harder (94.0%). When extending the time limit to 600s, the ribozyme success rate recovers to 98.8%, indicating that certain ribozyme structures present a more rugged optimization landscape and require extended search time for CoCoRNA to locate the global optimum.

Other biological sequence design tasks We apply CoCoRNA to solve other biological sequence design problems in the widely-used benchmark (Trabucco et al., 2022), including TF Bind 8, TF Bind 10 (Barrera et al., 2016), ChEMBL (Gaulton et al., 2012), and UTR (Sample et al., 2019). Experimental results show strong generalization. Details are provided in Appendix D.

4.2. Ablation studies

We perform ablation studies to dissect the contributions of different design choices in CoCoRNA.

Multi-agent architecture yields clear gains To assess the benefit of collective design, we compare CoCoRNA with a Single-agent variant that controls all positions. As shown in Figure 5 (left), the single-agent variant consistently underperforms CoCoRNA during training, indicating that multi-agent decomposition improves exploration and accelerates convergence.

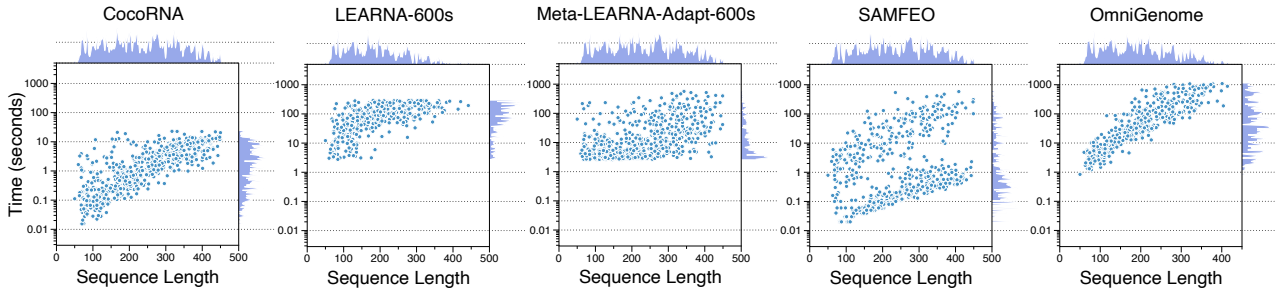

Figure 3. CPU time distribution of CocoRNA and four baselines over successfully solved cases.

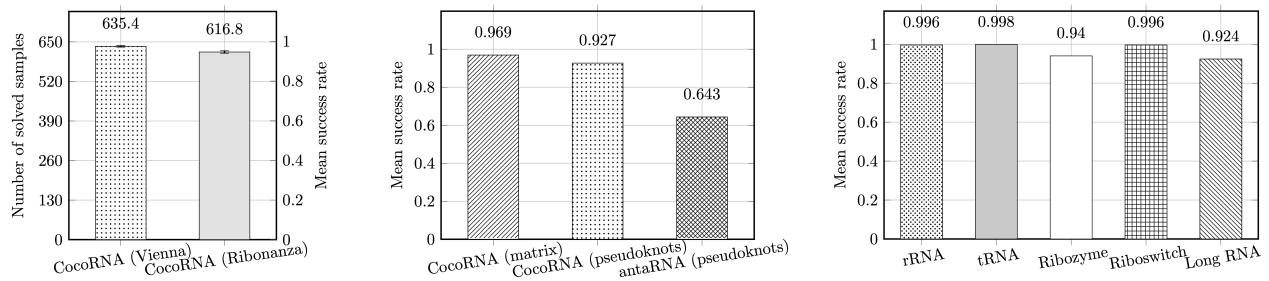

Figure 4. (**Left**) Cross-oracle validation experiment using RibonanzaNet. (**Middle**) Matrix representation and pseudoknot design results. (**Right**) Success rate breakdown across RNA families.

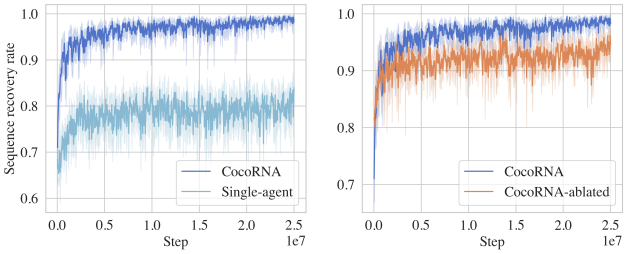

Figure 5. (**Left**) Comparison of sequence recovery rates during training between the `Single-agent` variant and CocoRNA. (**Right**) Effect of removing the SAE heuristic from CocoRNA.

SAE stabilizes and accelerates training We further evaluate a variant of CocoRNA without the SAE heuristic (`CocoRNA-ablated`). The learning curves in Figure 5 (right panel) show that this ablated model converges more slowly and exhibits higher variance. This indicates that SAE provides high-quality experiences early in training, significantly improving stability and efficiency. More detailed SAE-related analyses are deferred to Appendix C.3.

Additional robustness analyses To comprehensively evaluate CocoRNA's robustness, we also investigated alternative decomposition schemes, reward functions, numbers of agents, parameter-sharing strategies, dataset splits, and hyperparameter settings. Results consistently confirm the robustness and versatility of our framework. Full details and discussions are presented in Appendix C.

5. Related Works

The RNA inverse design problem was first introduced by Hofacker et al. (1994) in the Vienna RNA package, which uses an adaptive random walk algorithm. There are various follow-up approaches. For example, RNA-SSD (Andronescu et al., 2004) enhanced sequence initialization and reduced complexity by dividing sequences into smaller substructures. INFO-RNA (Busch & Backofen, 2006) further combined dynamic programming for sequence initialization with stochastic local search. Later approaches explored larger search spaces and incorporated additional constraints using evolutionary algorithms (Taneda, 2012; Esmaili-Taheri et al., 2014; Esmaili-Taheri & Ganjtabesh, 2015), Monte Carlo tree search (Yang et al., 2017; Portela, 2018), and simulated annealing (Matthies et al., 2012). Other notable methods utilized constraint programming (Garcia-Martin et al., 2013; 2015), ant colony optimization (Kleinkauf et al., 2015), or ensemble optimization (Zhou et al., 2023). However, these methods primarily rely on stochastic optimization guided by handcrafted heuristics. Such iterative search approaches are computationally expensive, with complexity rapidly increasing as sequence length grows. Moreover, partially due to the *no free lunch* theorem (Wolpert & Macready, 1997), these heuristics hardly generalize to new problems.

In contrast, machine learning methods can automatically discover effective heuristics from data. Recent learning-based methods have gained prominence, particularly gener-

ative models (Ingraham et al., 2019), large language models (Lin et al., 2023; Subramanian et al., 2024), and RL approaches (Angermueller et al., 2020; Hu et al., 2026a). These methods are successfully applied in various biological design tasks, such as protein function optimization (Lutz et al., 2023; Wang et al., 2023), protein sequence design (Dauparas et al., 2022; Gao et al., 2023), and drug repurposing (Huang et al., 2024). For RNA secondary structure design, Eastman et al. (2018) first utilized RL with pre-trained policies, effectively replacing handcrafted heuristics. Later, Runge et al. (2019) combined RL with neural architecture search to simultaneously optimize network architectures and hyperparameters. They further refined generative policies through instance-specific fine-tuning. Recent generative methods like RNAinformer (Patil et al., 2024) show promising results but are computationally intensive, limiting practical applicability to sequences under 100 nucleotides and lacking open-source availability. Similarly, OmniGenome (Yang et al., 2025), a large-scale 186M-parameter generative model, tackles RNA structure design by learning direct structure-to-sequence mappings.

We clarify key distinctions between CoCoRNA and related methods. CoCoRNA employs a MARL framework, decomposing the RNA design problem into manageable subtasks solved cooperatively. While RNA-SSD (Andronescu et al., 2004) also divides sequences into independently optimized subsequences, it does not effectively incorporate global structural information and thus often fails to achieve globally optimal solutions by combining local solutions. In contrast, CoCoRNA uses a centralized Critic with complete global context, guiding agents to jointly reach optimal solutions. Another related method, GameOpt (Bal et al., 2023), frames combinatorial Bayesian optimization as a cooperative game among optimization variables. Although both CoCoRNA and GameOpt adopt decomposition strategies, their approaches differ fundamentally: CoCoRNA trains dynamic policies for cooperative adaptation, whereas GameOpt searches for a fixed solution over predefined optimization variables.

6. Conclusion

We propose CoCoRNA, a collective design method that employs cooperative MARL to address the challenges of RNA secondary structure design. By formulating the RNA design task as a collective problem and decomposing it into multiple sub-tasks assigned to individual agents, CoCoRNA reduces the complexity faced by each agent, enhances policy exploration, and improves learning efficiency. Experimental results fully demonstrate the effectiveness and generalizability of CoCoRNA. Future work could explore more accurate reward models, adaptive decomposition strategies, and scalable credit-assignment mechanisms for larger multi-agent biological design problems.

Acknowledgments

This work was supported by the UKRI Future Leaders Fellowship under Grant MR/S017062/1 and MR/X011135/1; in part by National Natural Science Foundation of China under Grant 62376056 and 62076056; in part by the Science and Technology Innovation Program of Hunan Province under Grant 2024RC1011 and the Major Basic Research Projects in Hunan Province under Grant 2026JC0001; in part by the Isambard-AI, Royal Society Faraday Discovery Fellowship (FDF/S2/251014), BBSRC Transformative Research Technologies (UKRI1875), Royal Society International Exchanges Award (IES/R3/243136), Kan Tong Po Fellowship (KTP/R1/231017); and the Amazon Research Award and Alan Turing Fellowship.

Impact Statement

The proposed method may potentially accelerate advances in synthetic biology, RNA-based drug development, and the study of gene regulatory mechanisms, thereby contributing to the development of novel therapeutic strategies. However, applying AI models to biological sequence design may introduce risks related to interpretability and reliability. Enhancing the interpretability of AI models could help mitigate such concerns.

References

Abramson, J., Adler, J., Dunger, J., Evans, R., Green, T., Pritzel, A., Ronneberger, O., Willmore, L., Ballard, A. J., Bambrick, J., Bodenstein, S. W., Evans, D. A., Hung, C.-C., O'Neill, M., Reiman, D., Tunyasuvunakool, K., Wu, Z., Žemgulytė, A., Arvaniti, E., Beattie, C., Bertolli, O., Bridgland, A., Cherepanov, A., Congreve, M., Cowen-Rivers, A. I., Cowie, A., Figurnov, M., Fuchs, F. B., Gladman, H., Jain, R., Khan, Y. A., Low, C. M. R., Perlin, K., Potapenko, A., Savy, P., Singh, S., Stecula, A., Thillaisundaram, A., Tong, C., Yakneen, S., Zhong, E. D., Zielinski, M., Žídek, A., Bapst, V., Kohli, P., Jaderberg, M., Hassabis, D., and Jumper, J. M. Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature*, 630(8016):493–500, 2024.

Ali, S. E., Mittal, A., and Mathews, D. H. RNA secondary structure analysis using RNAstructure. *Curr. Protoc.*, 3 (7):e846, 2023.

Anderson-Lee, J., Fisker, E., Kosaraju, V., Wu, M., Kong, J., Lee, J., Lee, M., Zada, M., Treuille, A., and Das, R. Principles for predicting RNA secondary structure design difficulty. *J. Mol. Biol.*, 428(5, Part A):748–757, 2016.

Andronescu, M., Fejes, A. P., Hutter, F., Hoos, H. H., and Condon, A. A new algorithm for RNA secondary structure design. *Journal of molecular biology*, 336(3):607–624, 2004.

Angermueller, C., Dohan, D., Belanger, D., Deshpande, R., Murphy, K., and Colwell, L. Model-based reinforcement learning for biological sequence design. In *ICLR'20: Proc. of the 8th International Conference on Learning Representations*, 2020.

Bal, M. I., Sessa, P. G., Mutny, M., and Krause, A. Optimistic games for combinatorial bayesian optimization with applications to protein design. In *NeurIPS 2023 Workshop on Adaptive Experimental Design and Active Learning in the Real World*, 2023.

Barrera, L. A., Vedenko, A., Kurland, J. V., Rogers, J. M., Gisselbrecht, S. S., Rossin, E. J., Woodard, J., Mariani, L., Kock, K. H., Inukai, S., et al. Survey of variation in human transcription factors reveals prevalent dna binding changes. *Science*, 351(6280):1450–1454, 2016.

Barrett, T., Clements, W., Foerster, J., and Lvovsky, A. Exploratory combinatorial optimization with reinforcement learning. In *AAAI'20: Proc. of the 34th AAAI Conference on Artificial Intelligence*, pp. 3243–3250, 2020.

Barto, A. G., Sutton, R. S., and Anderson, C. W. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, SMC-13(5):834–846, 1983.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S. Neural combinatorial optimization with reinforcement learning. *CoRR*, abs/1611.09940, 2016.

Bose, R., Saleem, I., and Mustoe, A. M. Causes, functions, and therapeutic possibilities of RNA secondary structure ensembles and alternative states. *Cell chemical biology*, 31(1):17–35, 2024.

Busch, A. and Backofen, R. INFO-RNA: a fast approach to inverse RNA folding. *Bioinformatics*, 22(15):1823–1831, 2006.

Dauparas, J., Anishchenko, I., Bennett, N., Bai, H., Ragotte, R. J., Milles, L. F., Wicky, B. I., Courbet, A., de Haas, R. J., Bethel, N., et al. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378 (6615):49–56, 2022.

Dotu, I., Lozano, G., Clote, P., and Martinez-Salas, E. Using RNA inverse folding to identify ires-like structural subdomains. *RNA Biology*, 10(12):1842–1852, 2013.

Dykstra, P. B., Kaplan, M., and Smolke, C. D. Engineering synthetic RNA devices for cell control. *Nat. Rev. Genet.*, 23(4):215–228, 2022.

Eastman, P., Shi, J., Ramsundar, B., and Pande, V. S. Solving the RNA design problem with reinforcement learning. *PLoS computational biology*, 14(6):e1006176, 2018.

Esmaili-Taheri, A. and Ganjtabesh, M. ERD: a fast and reliable tool for RNA design including constraints. *BMC Bioinformatics*, 16(1):20, 2015.

Esmaili-Taheri, A., Ganjtabesh, M., and Mohammad-Noori, M. Evolutionary solution for the RNA design problem. *Bioinformatics*, 30(9):1250–1258, 2014.

Fica, S. M., Tuttle, N., Novak, T., Li, N.-S., Lu, J., Koodathingal, P., Dai, Q., Staley, J. P., and Piccirilli, J. A. RNA catalyses nuclear pre-mRNA splicing. *Nature*, 503 (7475):229–234, 2013.

Gao, Z., Tan, C., Zhang, Y., Chen, X., Wu, L., and Li, S. Z. ProteinInvBench: Benchmarking protein inverse folding on diverse tasks, models, and metrics. In *NeurIPS'23: Proc. of the 37th Annual Conference on Neural Information Processing Systems*, 2023.

Garcia-Martin, J. A., Clote, P., and Dotu, I. RNAiFOLD: a constraint programming algorithm for RNA inverse folding and molecular design. *J. Bioinform. Comput. Biol.*, 11(2):1350001, 2013.

Garcia-Martin, J. A., Clote, P., and Dotu, I. RNAiFold 2.0: a web server and software to design custom and Rfam-based RNA molecules. *Nucleic Acids Res.*, 43 (W1):W513–W521, 2015.

Gaulton, A., Bellis, L. J., Bento, A. P., Chambers, J., Davies, M., Hersey, A., Light, Y., McGlinchey, S., Michalovich, D., Al-Lazikani, B., et al. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(D1):D1100–D1107, 2012.

Georgakopoulos-Soares, I., Parada, G. E., and Hemberg, M. Secondary structures in RNA synthesis, splicing and translation. *Computational and structural biotechnology journal*, 20:2871–2884, 2022.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *ICML'17: Proc. of the 34th International Conference on Machine Learning*, pp. 1263–1272, 2017.

Guo, P. The emerging field of RNA nanotechnology. *Nat. Nanotechnol.*, 5(12):833–842, 2010.

He, S., Huang, R., Townley, J., Kretsch, R. C., Karagianes, T. G., Cox, D. B., Blair, H., Penzar, D., Vyaltsev, V., Aristova, E., et al. Ribonanza: Deep learning of RNA structure through dual crowdsourcing. *bioRxiv*, 2024.

Hofacker, I. L., Fontana, W., Stadler, P. F., Bonhoeffer, L. S., Tacker, M., and Schuster, P. Fast folding and comparison of RNA secondary structures. *Monatsh. Chem.*, 125(2): 167–188, 1994.

Hu, T., Luo, B., Yang, C., and Huang, T. MO-MIX: Multi-objective multi-agent cooperative decision-making with deep reinforcement learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(10):12098–12112, 2023.

Hu, T., Cui, Y., Luo, B., and Li, K. RIDER: 3D RNA inverse design with reinforcement learning-guided diffusion. In *ICLR'26: Proc. of the 14th International Conference on Learning Representations*, 2026a.

Hu, T., Cui, Y., Tang, R., Luo, B., and Li, K. Beyond monotonicity: Revisiting factorization principles in multi-agent q-learning. In *AAAI'26: Proc. of the 30th AAAI Conference on Artificial Intelligence*, pp. 21876–21884, 2026b.

Huang, K., Chandak, P., Wang, Q., Havaldar, S., Vaid, A., Leskovec, J., Nadkarni, G. N., Glicksberg, B. S., Gehlenborg, N., and Zitnik, M. A foundation model for clinician-centered drug repurposing. *Nature Medicine*, 30(12): 3601–3613, 2024.

Ingraham, J., Garg, V., Barzilay, R., and Jaakkola, T. Generative models for graph-based protein design. In *NeurIPS'19: Proc. of the 33rd Annual Conference on Neural Information Processing Systems*, 2019.

Iqbal, S. and Sha, F. Actor-attention-critic for multi-agent reinforcement learning. In *ICML'19: Proc. of the 36th International Conference on Machine Learning*, pp. 2961–2970, 2019.

Jiang, Y., Cao, Z., and Zhang, J. Learning to solve 3-d bin packing problem via deep reinforcement learning and constraint programming. *IEEE transactions on cybernetics*, 53(5):2864–2875, 2023.

Kalvari, I., Nawrocki, E. P., Ontiveros-Palacios, N., Argasinska, J., Lamkiewicz, K., Marz, M., Griffiths-Jones, S., Toffano-Nioche, C., Gautheret, D., Weinberg, Z., et al. Rfam 14: expanded coverage of metagenomic, viral and microRNA families. *Nucleic Acids Research*, 49(D1): D192–D200, 2021.

Kerpedjiev, P., Hammer, S., and Hofacker, I. L. Forna (force-directed RNA): Simple and effective online RNA secondary structure diagrams. *Bioinformatics*, 31(20): 3377–3379, 2015.

Kleinkauf, R., Mann, M., and Backofen, R. antaRNA: ant colony-based RNA sequence design. *Bioinformatics*, 31 (19):3114–3121, 2015.

Konda, V. and Tsitsiklis, J. Actor-critic algorithms. In *NIPS'99: Proc. of the 12th Annual Conference on Neural Information Processing Systems*, pp. 1008–1014, 1999.

Kulkarni, T. D., Narasimhan, K., Saeedi, A., and Tenenbaum, J. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *NeurIPS'16: Proc. of the Annual Conference on Neural Information Processing Systems*, pp. 3675–3683, 2016.

Kwon, D. RNA function follows form – why is it so hard to predict? *Nature*, 639(8056):1106–1108, 2025.

Lin, Z., Akin, H., Rao, R., Hie, B., Zhu, Z., Lu, W., Smetanin, N., Verkuil, R., Kabeli, O., Shmueli, Y., et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637): 1123–1130, 2023.

Lorenz, R., Bernhart, S. H., Höner zu Siederdissen, C., Tafer, H., Flamm, C., Stadler, P. F., and Hofacker, I. L. ViennaRNA package 2.0. *Algorithms for Molecular Biology*, 6(1):26, 2011.

Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *NeurIPS'17: Proc. of the 31st Annual Conference on Neural Information Processing Systems*, pp. 6382–6393, 2017.

Lutz, I. D., Wang, S., Norn, C., Courbet, A., Borst, A. J., Zhao, Y. T., Dosey, A., Cao, L., Xu, J., Leaf, E. M., et al. Top-down design of protein architectures with reinforcement learning. *Science*, 380(6642):266–273, 2023.

Markham, N. R. and Zuker, M. UNAFold: software for nucleic acid folding and hybridization. *Methods Mol. Biol.*, 453:3–31, 2008.

Mathews, D. H., Zuker, M., and Turner, D. H. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *J. Mol. Biol.*, 288(5):911–940, 1999.

Mathews, D. H., Disney, M. D., Childs, J. L., Schroeder, S. J., Zuker, M., and Turner, D. H. Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proc. Natl. Acad. Sci. USA*, 101(19):7287–7292, 2004.

Matthies, M. C., Bienert, S., and Torda, A. E. Dynamics in sequence space for RNA secondary structure design. *Journal of chemical theory and computation*, 8(10):3663–3670, 2012.

Morandi, E., van Hemert, M. J., and Incarnato, D. SHAPE-guided RNA structure homology search and motif discovery. *Nature Communications*, 13(1):1722, 2022.

Oliehoek, F. A., Spaan, M. T., and Vlassis, N. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

Patil, S., Runge, F., Franke, J. K., and Hutter, F. Towards generative RNA design with tertiary interactions. In *GEM: ICLR 2024 Workshop on Generative and Experimental Perspectives for Biomolecular Design*, 2024.

Portela, F. An unexpectedly effective monte carlo technique for the rna inverse folding problem. *BioRxiv*, pp. 345587, 2018.

Ramakrishnan, V. The ribosome emerges from a black box. *Cell*, 159(5):979–984, 2014.

Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., and Whiteson, S. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *ICML'18: Proc. of the 35th International Conference on Machine Learning*, pp. 4295–4304, 2018.

Rouskin, S., Zubradt, M., Washietl, S., Kellis, M., and Weissman, J. S. Genome-wide probing of RNA structure reveals active unfolding of mRNA structures in vivo. *Nature*, 505(7485):701–705, 2014.

Runge, F., Stoll, D., Falkner, S., and Hutter, F. Learning to design RNA. In *ICLR'19: Proc. of the 7th International Conference on Learning Representations*, 2019.

Runge, F., Farid, K., Franke, J. K., and Hutter, F. RnaBench: A comprehensive library for in silico RNA modelling. *bioRxiv*, pp. 1–24, 2024a.

Runge, F., Franke, J., Fertmann, D., Backofen, R., and Hutter, F. Partial RNA design. *Bioinformatics*, 40 (Supplement_1):i437–i445, 2024b.

Sample, P. J., Wang, B., Reid, D. W., Presnyak, V., McFadyen, I. J., Morris, D. R., and Seelig, G. Human 5' UTR design and variant effect prediction from a massively parallel translation assay. *Nature biotechnology*, 37(7):803–809, 2019.

Sato, K., Kato, Y., Hamada, M., Akutsu, T., and Asai, K. Ipknot: fast and accurate prediction of rna secondary structures with pseudoknots using integer programming. *Bioinformatics*, 27(13):i85–i93, 2011.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *CoRR*, abs/1506.02438, 2015.

Serganov, A. and Nudler, E. A decade of riboswitches. *Cell*, 152(1–2):17–24, 2013.

Subramanian, J., Sujit, S., Irtisam, N., Sain, U., Nowrouzezahrai, D., Kahou, S. E., and Islam, R. Reinforcement learning for sequence design leveraging protein language models. *CoRR*, abs/2407.03154, 2024.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *NIPS'99: Proc. of the 12th Annual Conference on Neural Information Processing Systems*, pp. 1057–1063, 1999.

Taft, R. J., Pang, K. C., Mercer, T. R., Dinger, M., and Mattick, J. S. Non-coding RNAs: regulators of disease. *J. Pathol.*, 220(2):126–139, 2010.

Taneda, A. Multi-objective genetic algorithm for pseudoknotted RNA sequence design. *Front. Genet.*, 3:36, 2012.

Trabucco, B., Geng, X., Kumar, A., and Levine, S. Design-bench: Benchmarks for data-driven offline model-based optimization. In *ICML'22: Proc. of the 39th International Conference on Machine Learning*, pp. 21658–21676, 2022.

Vandivier, L. E., Anderson, S. J., Foley, S. W., and Gregory, B. D. The conservation and function of RNA secondary structure in plants. *Annual review of plant biology*, 67(1): 463–488, 2016.

Wang, Y., Tang, H., Huang, L., Pan, L., Yang, L., Yang, H., Mu, F., and Yang, M. Self-play reinforcement learning guides protein engineering. *Nature Machine Intelligence*, 5(8):845–860, 2023.

Wang, Y., Ge, Y., Li, Z., Li, L., and Chen, R. M^3Rec: A Context-Aware Offline Meta-Level Model-Based Reinforcement Learning Approach for Cold-Start Recommendation. *ACM Transactions on Information Systems*, 42 (6):1–27, 2024.

Wolpert, D. H. and Macready, W. G. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

Yang, H., Chen, R., and Li, K. Bridging sequence-structure alignment in RNA foundation models. In *AAAI'25: Proc. of the 39th AAAI Conference on Artificial Intelligence*, pp. 21929–21937, 2025.

Yang, X., Yoshizoe, K., Taneda, A., and Tsuda, K. RNA inverse folding using Monte Carlo tree search. *BMC bioinformatics*, 18(1):468, 2017.

Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., and Wu, Y. The surprising effectiveness of PPO in cooperative multi-agent games. In *NeurIPS'22: Proc. of the 36th Annual Conference on Neural Information Processing Systems*, pp. 24611–24624, 2022.

Zhou, T., Dai, N., Li, S., Ward, M., Mathews, D. H., and Huang, L. RNA design via structure-aware multifrontier ensemble optimization. *Bioinformatics*, 39 (Supplement_1):i563–i571, 2023.

A. Pseudocode for CoCoRNA

Algorithm 1 presents the pseudocode for the proposed CoCoRNA method. CoCoRNA leverages the MAPPO algorithm, an extension of the PPO method tailored for multi-agent environments. In our implementation of CoCoRNA, each agent is equipped with an Actor network responsible for selecting actions based on local observations, while the centralized Critic network evaluates the global state to provide value estimates for policy updates.

Algorithm 1 CoCoRNA: Cooperative Multi-Agent RNA Design

Input: Target RNA secondary structure dataset $\mathbb{X}$, maximum episodes E, maximum steps per episode T, horizon H, number of agents n
Initialize Actor networks π_{θ^i} and Critic network V_ϕ for each agent i
Initialize experience replay buffer $\mathcal{D}$
for episode $= 1$ to E **do**
  Reset environment, randomly select a target RNA secondary structure x_t from $\mathbb{X}$, and initialize RNA sequence s_{seq}
  Decompose the RNA design task among n agents according to the chosen decomposition scheme
  for step $t = 1$ to T **do**
    for each agent $i = 1$ to n **in parallel do**
      Observe local observation o_t^i
      Obtain validity mask M_t^i from the environment based on assigned paired partners
      Select masked action $a_t^i \sim \pi_{\theta^i}(\cdot \mid o_t^i, M_t^i)$
    end for
    Execute joint action $\boldsymbol{a}_t = (a_t^1, a_t^2, \ldots, a_t^n)$ in the environment's fixed update order
    Update RNA sequence s_{seq} with masked valid actions $\boldsymbol{a}_t$
    Predict folded structure x_f using MFE folding algorithm
    Calculate Hamming distance $H_t = H(x_f, x_t)$
    Calculate reward r_t based on H_t and H_{t-1}
    Store transition $(\boldsymbol{o}_t, \boldsymbol{a}_t, r_t, \boldsymbol{o}_{t+1})$ in buffer $\mathcal{D}$
    if using SAE **then**
      Perform local search to find better actions $\boldsymbol{a}'_t$
      Update reward r'_t based on improved Hamming distance H'_t
      Replace transition with $(\boldsymbol{o}_t, \boldsymbol{a}'_t, r'_t, \boldsymbol{o}_{t+1})$ in buffer $\mathcal{D}$
    end if
    if $H_t = 0$ **then**
      Assign final reward C to all agents
      Break
    end if
    if buffer size reaches horizon H **then**
      Update Actor networks π_{θ^i} and Critic network V_ϕ using mini-batch updates from buffer $\mathcal{D}$ with MAPPO
      Clear experience buffer $\mathcal{D}$
    end if
  end for
end for

B. Theoretical Analysis

B.1. Convergence Analysis

In this section, we provide a convergence analysis of the proposed algorithm. We demonstrate that, under certain conditions, the multi-agent actor-critic method used in CoCoRNA converges to a local maximum of the expected joint return.

In CoCoRNA, we employ a centralized Critic network to evaluate the joint policy. Specifically, we use the joint advantage function $A(s, \boldsymbol{a})$, defined as:

$$A(s, \boldsymbol{a}) = Q(s, \boldsymbol{a}) - V(s), \tag{8}$$

where $Q(s, \boldsymbol{a})$ denotes the joint action-value function and $V(s)$ is the state-value function. The joint policy gradient is then

given by:

$$g = \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_i \nabla_\theta \log \pi^i(a^i \mid o^i)\, A(s, \boldsymbol{a})\right] \tag{9}$$

$$= \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_i \nabla_\theta \log \pi^i(a^i \mid o^i)\, Q(s, \boldsymbol{a})\right] - \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_i \nabla_\theta \log \pi^i(a^i \mid o^i)\, V(s)\right], \tag{10}$$

where $\boldsymbol{\pi}$ denotes the joint policy, and θ represents the parameters of the Actor networks. The second term of (10) can be expanded as:

$$g_V = -\mathbb{E}_{\boldsymbol{\pi}}\left[\sum_i \nabla_\theta \log \pi^i(a^i \mid o^i)\, V(s)\right] \tag{11}$$

$$= -\sum_s d^{\boldsymbol{\pi}}(s) \sum_i \sum_{\boldsymbol{a}^{-i}} \boldsymbol{\pi}(\boldsymbol{a}^{-i} \mid o^{-i}) \sum_{a^i} \nabla_\theta \pi^i(a^i \mid o^i)\, V(s), \tag{12}$$

where $d^{\boldsymbol{\pi}}(s)$ denotes the discounted ergodic state distribution (Sutton et al., 1999), $\boldsymbol{a}^{-i}$ and o^{-i} represent the joint actions and observations of all agents except agent i.

Due to the normalization property of the policy:

$$\sum_{a^i} \pi^i(a^i \mid o^i) = 1, \tag{13}$$

we have:

$$\sum_{a^i} \nabla_\theta \pi^i(a^i \mid o^i) = \nabla_\theta \sum_{a^i} \pi^i(a^i \mid o^i) = \nabla_\theta 1 = 0. \tag{14}$$

Therefore,

$$g_V = -\sum_s d^{\boldsymbol{\pi}}(s) \sum_i \sum_{\boldsymbol{a}^{-i}} \boldsymbol{\pi}(\boldsymbol{a}^{-i} \mid o^{-i})\, V(s)\, \nabla_\theta 1 \tag{15}$$

$$= 0. \tag{16}$$

Thus, the policy gradient in (10) reduces to only the first term:

$$g = \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_i \nabla_\theta \log \pi^i(a^i \mid o^i)\, Q(s, \boldsymbol{a})\right]. \tag{17}$$

Since the joint policy can be expressed as the product of individual agent policies:

$$\boldsymbol{\pi}(\boldsymbol{a} \mid s) = \prod_i \pi^i(a^i \mid o^i), \tag{18}$$

and using the logarithmic identity $\log \prod_i x_i = \sum_i \log x_i$, we can rewrite the policy gradient as:

$$g = \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_i \nabla_\theta \log \pi^i(a^i \mid o^i)\, Q(s, \boldsymbol{a})\right] \tag{19}$$

$$= \mathbb{E}_{\boldsymbol{\pi}}\left[\nabla_\theta \log \prod_i \pi^i(a^i \mid o^i)\, Q(s, \boldsymbol{a})\right] \tag{20}$$

$$= \mathbb{E}_{\boldsymbol{\pi}}\left[\nabla_\theta \log \boldsymbol{\pi}(\boldsymbol{a} \mid s)\, Q(s, \boldsymbol{a})\right]. \tag{21}$$

This result shows that (21) is equivalent to the single-agent actor-critic policy gradient, where the multiple agents are considered as a single joint agent (or super-agent). Konda & Tsitsiklis (1999) proved the convergence of policy gradient algorithms under the condition that the policy $\boldsymbol{\pi}$ is differentiable and the algorithm parameters satisfy certain conditions.

Therefore, by following the gradient in (9), the multi-agent actor-critic algorithm converges to a local maximum of the expected joint return $J^{\boldsymbol{\pi}}$.

B.2. Limitation and Discussion

In theory, gradient-based deep learning and reinforcement learning methods generally guarantee convergence to local optima rather than global optima due to the non-convex nature of the optimization landscape. However, in practice, the ability of reinforcement learning algorithms to go beyond local optima and approach global optima depends significantly on how well they balance exploration and exploitation.

Specifically, CoCoRNA is built upon PPO, which includes an entropy bonus in the objective function, which explicitly encourages the policy to maintain high entropy. This prevents the policy from becoming too deterministic too quickly, avoiding premature convergence to suboptimal policies and promoting continued exploration.

In the context of the RNA design task, the multi-agent framework of CoCoRNA further aids in overcoming local optima. Unlike single-agent methods that mutate one nucleotide at one step, our method allows multiple agents to act simultaneously on different parts of the RNA sequence. This introduces greater diversity and increases the exploration of the state space, thereby increasing the opportunities to discover better policies.

C. Additional Results

In this section, we present a series of experiments under various settings to comprehensively demonstrate the effectiveness of CoCoRNA.

C.1. Dataset Splitting

In the main experiments, we cluster the dataset based on structural similarity and randomly select one cluster out of 100 as the test set. To more rigorously rule out potential data leakage, we further construct six different train-test splits by randomly selecting six distinct clusters as test sets. Independent training and evaluation are then conducted on each split. Additionally, we evaluate two of the strongest baselines on these test sets. The results are reported in Table 3.

The results show that CoCoRNA achieves an average success rate of 97.44% across these new train-test splits, which is highly consistent with the results reported in the main paper. These findings further demonstrate the strong generalization ability of CoCoRNA.

Table 3. Performance of CoCoRNA, Meta-LEARNA-Adapt-600s, and SAMFEO-600s across six different train-test splits. The mean and standard deviation are reported.

Method	Number of solved samples	Mean success rate
CoCoRNA	633.33 ± 5.47	97.44%
Meta-LEARNA-Adapt-600s	578.17 ± 9.60	88.95%
SAMFEO-600s	568.16 ± 6.09	87.41%

Table 4. Performance of CoCoRNA with different problem decomposition schemes on Rfam dataset across six different train-test splits. The mean and standard deviation over are reported.

Method	Number of solved samples	Mean success rate
CoCoRNA-PBD	633.33 ± 5.47	97.44%
CoCoRNA-SBD	628.50 ± 4.99	96.69%
CoCoRNA-random	596.33 ± 6.70	91.74%

C.2. Using Different Decomposition Schemes

Table 4 provides the testing results for different problem decomposition schemes used in CoCoRNA. We employ three decomposition methods: Position-based decomposition (CoCoRNA-PBD), Structure-type-based decomposition (CoCoRNA-SBD) and random decomposition(CoCoRNA-random). For Position-based decomposition, the complete RNA structure

Table 5. Performance of CoCoRNA and ablated versions on Rfam dataset. The mean and standard deviation over five independent runs are reported.

Method	Number of solved samples	Mean success rate
CoCoRNA	635.40 ± 2.58	97.75%
CoCoRNA-ablated	599.60 ± 7.64	92.25%
Single-agent version	340.80 ± 5.63	52.43%

is divided into approximately equal segments, with each segment assigned to a different agent. For Structure-type-based decomposition, agents are assigned specific structural types based on the dot-bracket notation; for example, if Agent 1 is responsible for non-paired structures, it sequentially designs nucleotides at all non-paired positions within the structure.

In our experiments, CoCoRNA-PBD and CoCoRNA-SBD only show minor performance differences, demonstrating the flexibility of our method concerning decomposition choices. For CoCoRNA-SBD, structure-type-based decomposition allows agents to focus more on specific structural elements, but the relative positional relationships between different agents may become completely disrupted as the design process progresses, potentially causing slight adverse effects on the learning process. Although CoCoRNA-SBD performed slightly worse than CoCoRNA-PBD in testing, both methods significantly outperformed the single-agent approach. CoCoRNA-random is worse than the other two decomposition methods. We hypothesize that this is because randomly assigning design positions is entirely unstructured. This increases the difficulty for agents to cooperate and interferes with the learning process. However, the performance of random decomposition does not completely collapse, suggesting that the MARL mechanism itself is robust.

C.3. Ablation Study on SAE

Table 5 presents the testing results of two ablated versions of CoCoRNA. Removing the SAE method results in a slight performance decrease in CoCoRNA-ablated, demonstrating the effectiveness of the SAE approach. Additionally, as shown in the right plot of Figure 5, CoCoRNA-ablated continues to improve towards the end of the training process, suggesting that further training may yield better performance. The single-agent version shows significantly poorer performance. In fact, the single-agent version is similar to the LEARNA method but lacks architectural and hyperparameter optimizations and does not undergo additional training during the design phase.

Figure 6. Learning curves under different reward settings on the Rfam dataset. Each experiment is performed over 6 independent runs with different random seeds. The shaded areas represent the 95% confidence intervals. **Left**: Learning curves using delayed reward (22). **Right**: Learning curves using terminal reward (23).

Table 6. Performance of CoCoRNA and CoCoRNA-ablated under different reward settings on the Rfam dataset.

Reward Setting / Method	Number of solved samples	Success rate
Delayed Reward / CoCoRNA	634/650	97.54%
Delayed Reward / CoCoRNA-ablated	621/650	95.54%
Terminal Reward / CoCoRNA	604/650	92.92%
Terminal Reward / CoCoRNA-ablated	570/650	87.69%

C.4. Ablation Study on Reward Signals

In addition to the standard reward signal, we test two different delayed/sparse reward settings on the Rfam dataset.

- **Delayed Reward:** The reward signal is calculated every 10 steps instead of every step. The original reward function (7) is modified as follows:

$$R_t = \begin{cases} \frac{H_{t-10} - H_t}{l}, & \text{if } H_t > 0 \text{ and } t \bmod 10 = 0, \\ C, & \text{if } H_t = 0 \text{ and } t \bmod 10 = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{22}$$

- **Terminal Reward Only:** The reward is given only at the end of an episode or upon successful design. The reward function is defined as:

$$R_t = \begin{cases} 1 - \frac{H_t}{l}, & \text{if } H_t \neq 0 \text{ and } t \text{ reaches maximum steps (episode ends)}, \\ C, & \text{if } H_t = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{23}$$

We also test the impact of removing the SAE method under these two reward settings. The results are shown in Figure 6 and Table 6. We denote the ablated version without SAE as CoCoRNA-ablated. It can be observed that under the delayed reward setting, CoCoRNA experiences almost no performance loss. When only terminal rewards are available, the algorithm's performance decreases slightly due to the excessively sparse reward signal significantly increasing the difficulty of policy learning. Under both reward settings, CoCoRNA-ablated performs worse than CoCoRNA, demonstrating the effectiveness of the SAE method. Overall, this experiment shows that CoCoRNA is robust to different reward signals and performs well even under harsh reward settings. Notably, the delayed reward setting reduces the number of calls to the RNA folding algorithm to one-tenth of the original, significantly speeding up training.

Table 7. Performance of CoCoRNA under different agent size settings on the Rfam dataset.

Number of Agents (n)	Number of solved samples	Success rate
$n = 2$	601/650	92.46%
$n = 4$	636/650	97.85%
$n = 6$	629/650	96.77%
$n = 8$	615/650	94.62%

C.5. Effect of Varying the Number of Agents

In the main results presented earlier, we set the number of agents to $n = 4$. In this section, we adjust the parameter n to examine the impact of different numbers of agents on the algorithm's performance. We experiment with four different settings: using 2, 4, 6, and 8 agents. Table 7 shows the performance of CoCoRNA under these different agent sizes on the Rfam dataset. The corresponding learning curves are presented in Figure 7.

Figure 7. Learning curves of CoCoRNA under different agent size settings on the Rfam dataset. Each experiment is performed over 6 independent runs with different random seeds. The shaded areas represent the 95% confidence intervals.

From the results, we observe that using 4 or 6 agents achieves the best performance. When the number of agents is too small or too large, the performance tends to decrease. With too few agents, the benefits of multi-agent problem decomposition cannot be fully exploited, as each agent handles a larger portion of the task, leading to higher complexity per agent and potentially less efficient learning. On the other hand, with too many agents, the environment becomes more non-stationary from the perspective of each agent due to the increased interactions and dependencies among agents, making cooperation more challenging.

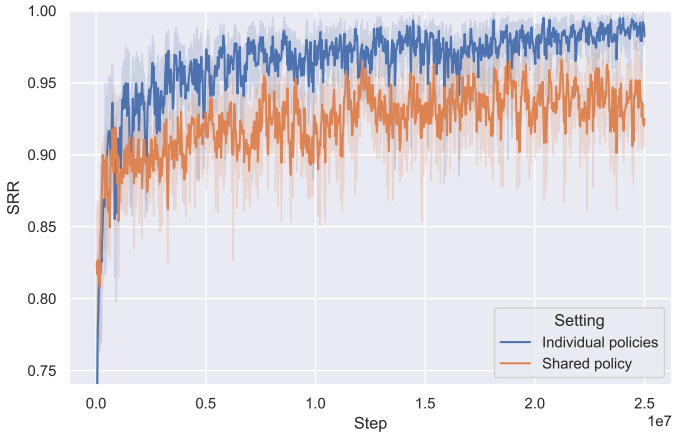

Figure 8. Learning curves of CoCoRNA-PBD and CoCoRNA-PBD-shared on the Rfam dataset. Each experiment is performed over 6 independent runs with different random seeds. The shaded areas represent the 95% confidence intervals.

C.6. Effect of Sharing Policy Parameters

We investigate the performance of CoCoRNA when agents share policy parameters under both decomposition methods. In these experiments, although different agents receive different observations, all agents share the same set of policy parameters. This approach is referred to as CoCoRNA-PBD-shared and CoCoRNA-SBD-shared for position-based decomposition and structure-type-based decomposition, respectively.

The learning curves and test results are shown in Figure 8 and Table 8, respectively. From the results, we observe that

Table 8. Performance of CoCoRNA with independent and shared policies on the Rfam dataset.

Method	Number of solved samples	Success rate
CoCoRNA-PBD	636/650	97.85%
CoCoRNA-PBD-shared	622/650	95.69%
CoCoRNA-SBD	629/650	96.77%
CoCoRNA-SBD-shared	621/650	95.54%

enforcing shared policy parameters among multiple agents leads to a slight decrease in algorithm performance compared to using independent policies, although the gap is relatively small.

These results potentially suggest that different agents learn distinct local policies that are specialized for their assigned sub-tasks. When agents share policy parameters, the policy learning process is constrained because all agents must use the same policy function, despite facing different sub-tasks or local environments. This can limit the agents' ability to adapt their policies to their specific roles in the RNA design task. In contrast, when using independent policies, each agent can tailor its policy parameters to its particular sub-task without interference from the data of other agents, providing more flexibility in policy learning.

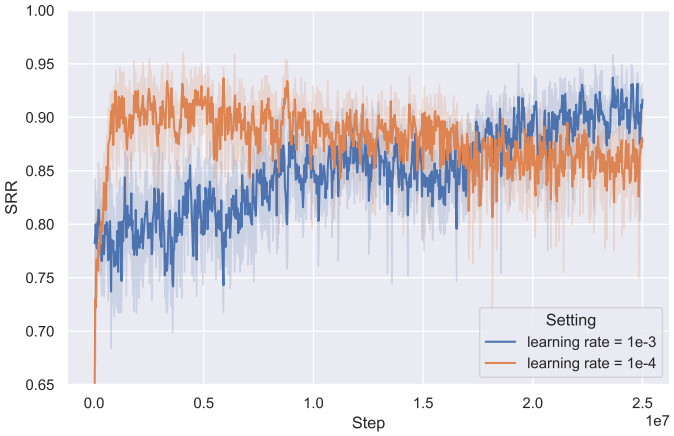

Figure 9. Learning curves of CoCoRNA using different learning rates. Each experiment is performed over 6 independent runs with different random seeds. The shaded areas represent the 95% confidence intervals.

C.7. Training Parameters

We observe that the algorithm is relatively sensitive to the learning rate compared to other hyperparameters. In our main experiments, we use a learning rate of 1×10^{-5}, which is lower than the settings commonly used in RL algorithms. Figure 9 shows the learning curves when using higher learning rates. It can be seen that excessively high learning rates lead to unstable training and poor convergence.

In multi-agent environments, a high learning rate can exacerbate the non-stationarity of the environment. This issue arises because each agent's policy update affects the environment dynamics perceived by other agents. With higher learning rates, these changes become more abrupt, making it difficult for agents to adapt and learn stable policies. Consequently, the training process becomes highly unstable.

For other important hyperparameters, such as the discount factor γ, the GAE parameter λ, and the PPO clip range ϵ, we use standard values commonly adopted in the literature. These settings are detailed in Appendix F.

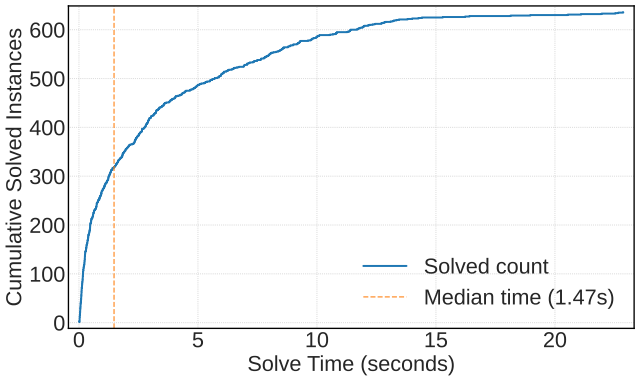

Figure 10. Number of solved samples as a function of the allotted runtime.

Table 9. Evaluation of CoCoRNA and the single-agent ablated version on biological sequence design tasks. Results represent the average of 8 independent runs, with standard deviations also reported.

	TF Bind 8	**TF Bind 10**	**ChEMBL**	**UTR**
CoCoRNA	1.000 ± 0.000	1.000 ± 0.000	0.811 ± 0.001	0.810 ± 0.001
Single-agent version	0.901 ± 0.093	0.725 ± 0.140	0.595 ± 0.166	0.690 ± 0.010

C.8. Other Results

Figure 10 shows how the number of solved samples increases as a function of the allotted runtime. The median solving time of CocoRNA is only 1.47 s.

Table 10 summarizes the number of redesign attempts (iterations) n_{i} required to design different RNA structures on the Rfam dataset using CoCoRNA, LEARNA-600s, and Meta-LEARNA-Adapt-600s. For the majority of sequences (95.54%), our method succeeds within a single episode. In contrast, the two baseline methods require numerous iterative optimizations. Although Meta-LEARNA-Adapt performs meta-learning on the entire training set, only 3.38% of sequences can be solved in a single attempt.

D. Other biological sequence design tasks

Beyond RNA inverse design, we further apply CoCoRNA to four biological sequence design tasks. These tasks are formulated as RL environments based on DesignBench (Trabucco et al., 2022). Specifically, they include the DNA functional optimization tasks TF Bind 8, TF Bind 10 (Barrera et al., 2016), and UTR (Sample et al., 2019), as well as the drug molecule design task ChEMBL (Gaulton et al., 2012). Apart from minor adjustments to the state space, action space, and hyperparameters such as the learning rate, CoCoRNA requires no additional modifications.

Results Given that the algorithms in Section 4.1 are specifically tailored for RNA design, we can hardly adapt them to other tasks without nearly redesigning a new algorithm from scratch. Therefore, we compare CoCoRNA with its single-agent ablated variant to further validate the effectiveness of the multi-agent framework. The results, presented in Table 9, clearly show that the MARL framework of CoCoRNA consistently outperforms its single-agent counterpart across all tasks. Moreover, CoCoRNA successfully reaches the global optimum in TF Bind 8 and TF Bind 10. Considering that CoCoRNA was not originally designed for these tasks, these results further demonstrate its adaptability and effectiveness across diverse problem settings.

Table 10. Distribution of RNA sequences by redesign attempts n_{i} intervals for different methods on the Rfam dataset.

Method	$n_{\mathrm{i}} = 1$	$1 < n_{\mathrm{i}} \leq 15$	$n_{\mathrm{i}} > 15$
CoCoRNA	621	29	0
LEARNA	1	16	633
Meta-LEARNA	22	137	491

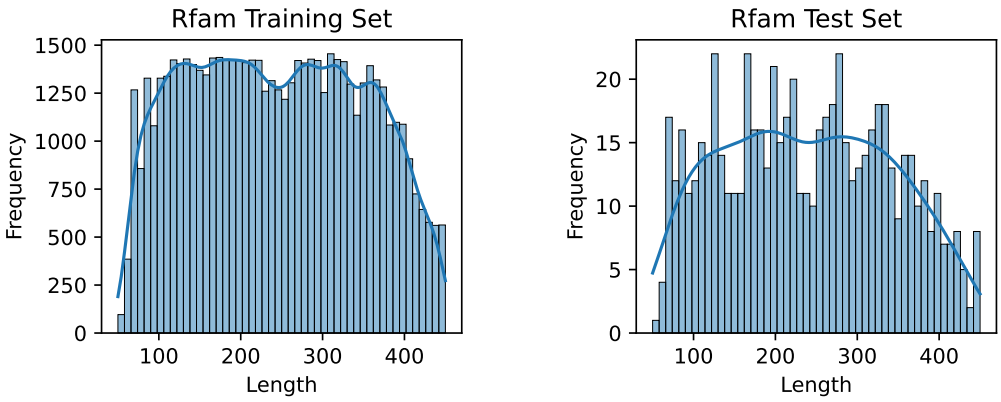

Figure 11. Length distributions of RNA secondary structures in different datasets.

E. Datasets Details

We use the Rfam dataset (Runge et al., 2024a) to train and evaluate our algorithm. The dataset contains only RNA secondary structure information without explicit sequence labels, as the task is to design sequences that fold into the provided structures.

The dataset is constructed by applying RNA folding algorithms to RNA sequences from the Rfam database (Kalvari et al., 2021). The structures in this dataset are computationally predicted using tools like the ViennaRNA package (Lorenz et al., 2011), which produces secondary structures based on minimum free energy (MFE) folding.

Figure 11 shows the length distributions of RNA secondary structures in the Rfam training set and test set. Table 11 provides statistical information about the RNA structures in these datasets, including the sample size, average length, maximum length, and minimum length of the RNA secondary structures.

F. Algorithm Details and Hyperparameters

This section provides a detailed description of the CoCoRNA algorithm, including network architectures, training details, and hyperparameter settings.

F.1. Network Architectures

Each agent in the CoCoRNA framework is equipped with its own Actor and Critic networks. Although the Critic network leverages shared global information, the network parameters are not shared across agents. The Actor network takes local observations as input and outputs a discrete action (the nucleotide to be placed in the RNA sequence). The Critic network, on the other hand, processes the global state, which includes the full RNA sequence and secondary structure, and outputs a scalar value representing the estimated cumulative future reward for the entire system.

- **Actor Network:** The Actor network consists of several fully connected layers followed by a softmax output layer. Each agent's observation is passed through the network to compute a probability distribution over the action space $\{A, U, G, C\}$.

Table 11. Statistics of the datasets.

Dataset	Sample Size	Average Length	Max Length	Min Length
Rfam (Training Set)	60000	243.73	450	50
Rfam (Test Set)	650	238.78	450	50

Table 12. Training hyperparameters.

Hyperparameter	Value
Learning Rate	0.00001
Discount Factor (γ)	0.99
GAE Parameter (λ)	0.95
PPO Clip Range (ϵ)	0.2
Entropy Coefficient	0.02
Horizon Length	256
Batch Size	512
Number of Epochs	10
Gradient Clipping	0.5

- **Critic Network:** The Critic network takes the entire RNA sequence and secondary structure as input. This input is processed through two separate CNN modules: one for the sequence and one for the structure. The outputs of these modules are concatenated and passed through a fully connected network to produce a value estimate for the entire system.

F.2. Training and Optimization

The training procedure follows on-policy reinforcement learning paradigm: experiences are collected over a fixed horizon, stored in a replay buffer, and subsequently used to update the Actor and Critic networks. We adopt PPO to optimize the policy by maximizing the clipped surrogate objective, which stabilizes training, and employ Generalized Advantage Estimation (GAE) to compute advantages with reduced variance.

Training on a single NVIDIA 4090 GPU requires approximately 8 to 10 hours. As shown in Appendix C.4, this can be reduced to 3 to 4 hours under a delayed reward formulation with negligible performance degradation. Inference is highly efficient, averaging only 3.24 seconds per successful design.

F.3. Hyperparameter Settings

Table 12 and Table 13 summarize the key hyperparameters used in training the CoCoRNA model, including training parameters and network parameters.

F.4. Other Settings

Table 14 presents additional parameter settings used in the CoCoRNA framework. During training, the maximum number of steps per episode is set to 350, while for testing, it is increased to 400. The training is conducted for a total of 25 million steps. Each agent has an observation radius of 30, resulting in a local observation window size of 61. Since the maximum RNA structure length in the dataset is 450, we standardize the global state dimension to 900 (450 for the sequence information and 450 for the structure information). We use the ViennaRNA package (Lorenz et al., 2011) version 2.6.4 to compute the folded RNA structures during the design process.

Table 13. Network hyperparameters.

Hyperparameter	Value
Actor Network Layers	$[128, 64]$
Critic Network Layers	$[256, 64]$
CNN Filter Size	$[8, 4, 3]$
CNN Number of Filters	$[32, 64, 64]$
Activation Function	ReLU
Optimizer	Adam

Table 14. Additional hyperparameters.

Parameter	Value
Maximum Steps per Episode (Training)	350
Maximum Steps per Episode (Testing)	400
Observation Radius	30
Local Observation Window Size	61
Global State Dimension	900
Total Training Steps	2.5×10^7

G. Discussion

G.1. Graph Neural Networks

Many biological molecules are well-suited to be modeled as graphs due to their inherent structural properties. Transforming RNA structures into graph representations and utilizing Graph Neural Networks (GNNs) (Gilmer et al., 2017) could potentially better capture the relationships between different nucleotides, enabling more effective policy learning. Graph-based representations naturally model the interactions and dependencies within RNA structures, which could enhance the agent's decision-making process.

However, employing more complex network architectures like GNNs also introduces higher computational and memory overheads, potentially reducing training efficiency. In our work, although we did not use GNNs, we addressed the issue of partial observations by employing CNNs within the centralized Critic to extract global structural information. This approach helps mitigate the limitations of local observations by providing a holistic view of the RNA structure, thereby supporting coordinated policy optimization without incurring excessive computational costs. Nevertheless, integrating GNNs with reinforcement learning-based methodologies represents a promising direction for future research.

G.2. Limitations and Future Work

While CoCoRNA demonstrates promising results in RNA secondary structure design, there are several limitations.

First, it is non-trivial to design an accurate and reliable reward model given the complexity of biological systems. We may resort to large-scale pretrained models (Abramson et al., 2024). However, these models are often too large and thus are computationally expensive. One potential solution is to go with a model-based RL, where an explicit model of the environment is learned and used to predict future states and rewards more efficiently. Another future direction is to extend the framework to multi-objective settings (Hu et al., 2023), jointly considering structural fidelity and additional biological objectives such as free energy (Lorenz et al., 2011) and GC content.

Another potential limitation is the decomposition method considered in the current CocoRNA. Given the high-dimensional

nucleotide design space, it will be more promising to study an adaptive decomposition mechanism. This might be achieved by designing hierarchical policies (Kulkarni et al., 2016), where high-level agents make decisions about task decomposition and low-level agents focus on specific sub-tasks.

In the current framework, all agents share a global reward defined by the structural quality of the full RNA sequence, and credit assignment is handled implicitly through centralized value estimation rather than explicit per-agent rewards. As the number of interacting agents increases, the environment becomes increasingly non-stationary from the perspective of each individual agent, making coordination and credit assignment more challenging. This effect is reflected in our ablation studies, where performance degrades moderately when the number of agents becomes too large. Developing more scalable coordination and credit-assignment mechanisms for large multi-agent biological design problems remains an important direction for future work.