Partial RNA Design

Frederic Runge*
Department of Computer Science
University of Freiburg
Freiburg, 79110, Germany
runget@cs.uni-freiburg.de

Jörg K.H. Franke
Department of Computer Science
University of Freiburg
Freiburg, 79110, Germany
frankej@cs.uni-freiburg.de

Daniel Fertmann
Department of Computer Science
University of Freiburg
Freiburg, 79110, Germany
fertmand@cs.uni-freiburg.de

Rolf Backofen
Department of Computer Science
University of Freiburg
Freiburg, 79110, Germany
backofen@informatik.uni-freiburg.de

Frank Hutter
Department of Computer Science
University of Freiburg
Freiburg, 79110, Germany
fh@cs.uni-freiburg.de

Abstract

In this work, we propose *partial RNA design*, a novel RNA design paradigm formulated as a constraint satisfaction problem. Partial RNA design describes the problem of designing RNA sequences from arbitrary RNA sequence and structure motifs. Our formulation enables RNA design of variable-lengths candidates, while still allowing precise control over sequence and structure constraints at individual positions. Based on this formulation, we improve an existing inverse RNA folding algorithm with a masked training objective to derive *libLEARNA*, a new algorithm capable of efficiently solving different tasks in the realm of RNA design. A comprehensive analysis across various problems, including a realisitic riboswitch design task, reveals the outstanding performance of libLEARNA and its robustness.

1 Introduction

Ribonucleic acids (RNAs) are versatile regulators at all levels of cellular machinery that are connected to numerous diseases like Parkinson's [Cao et al., 2018], Alzheimer's [Zhou and Xu, 2015], diabetes [Tian et al., 2018], cancer [Prensner et al., 2011], or cardiovascular diseases [Choong et al., 2017]. The function of an RNA molecule strongly depends on its folding into a complex structure that allows interactions with other cellular compounds. In contrast to protein folding, which is dominated by hydrophobic forces acting globally, RNAs exhibit a hierarchical folding process [Tinoco Jr and Bustamante, 1999]. Intra-molecular interactions via hydrogen bonds pair corresponding nucleotides of the RNA sequence, resulting in restrictive local geometries and a fast formation of the so-called *secondary structure* of RNA. The spatial arrangement and complex *tertiary structures* strongly depend on the local secondary structure features [Tinoco Jr and Bustamante, 1999]. The broad range of different cellular functions, as well as a strong structure-to-function relationship that is already achieved on a secondary structure level [Hammer et al., 2019], allow to design RNA sequences based

*Corresponding author.

Preprint.

on a given structure to accomplish a certain function. The fields of applications are diverse, including vaccine development [Mueller et al., 2010], design of riboswitches [Wachsmuth et al., 2012], RNA nanostructures [Li et al., 2018], catalytic ribozymes [Dotu et al., 2014], CRISPR-based control of gene expression [Ferry et al., 2017], light-dependent control of regulatory RNA molecules [Pilsl et al., 2020], and genetic circuit engineering [Kim et al., 2019].

Typically, RNA design endeavours start from carefully selected RNA fragments that are associated with a certain function and combine these so-called motifs to achieve new functionality [Wachsmuth et al., 2012, Li et al., 2018, Nozawa et al., 2019]. Computational RNA design is employed to narrow down the tremendous search space of RNA candidates to a reasonable number of sequences that could then be experimentally screened for the desired function. However, depending on the selected RNA motifs, the resulting design tasks may impose different requirements on the design algorithms. For example, the design of functional riboswitches by Wachsmuth et al. [2012] was essentially dictated by the existence of a motif with an explicit sequence and a specific structure for binding a ligand (aptamer), a more variable downstream structural motif (expression platform), and a spacer domain of variable length to separate the two. Fernandez-Chamorro et al. [2016] on the other hand designed several candidate sequences for IRES-elements with different fixed secondary structures to study the influence of structural motifs on IRES activity. To be applicable to any setting, a RNA design algorithm thus (1) has to be able to strictly preserve structure and sequence features at desired positions, while at the same time (2) allow to define less restricted domains that can be explored by the machine, optimally (3) enabling designs of variable lengths. We observe that current RNA design formulations and algorithms do not fully meet these requirements: Current formulations assume *a priori* knowledge about the exact pairing scheme of the desired structure and algorithms based on the inverse RNA folding formulation [Hofacker et al., 1994] are restricted to design fixed-length candidates [Hofacker et al., 1994, Andronescu et al., 2004, Busch and Backofen, 2007, Esmaili-Taheri and Ganjtabesh, 2015], while approaches that design RNA sequences based on abstract shapes [Shapiro, 1988, Giegerich et al., 2004] are not limited to a specific output length but require a fully defined secondary structure as template to derive the shape [Avihoo et al., 2011, Drory Retwitzer et al., 2016, Retwitzer et al., 2020]. To produce reasonable amounts of RNA candidates that can be screened in an RNA design pipeline, this limitation of the design spaces is suboptimal. Since a single initial structure template is often not sufficient to achieve the desired design goals, these goals and especially the desired structure often have to be adjusted during a design process which typically requires repetitive runs of the same design algorithm for a given application [Fernandez-Chamorro et al., 2016].

In this work, we propose *partial RNA design*, a novel RNA design formulation to tackle these problems. In essence, the idea is to view RNA design as a search in an open, partially restricted space, rather than a search given a single target structure. This allows us to design RNAs from provided motifs that define positional constraints in the design space, while being capable to design RNAs of variable lengths by leveraging unconstrained regions. Using this formulation, we enhance a previous automated reinforcement learning (AutoRL) approach, *LEARNA* [Runge et al., 2019], to develop *libLEARNA*, a first algorithm for solving the partial RNA design problem. We see our main contributions as follows.

- We propose *partial RNA design*, a novel RNA design formulation based on a constraint satisfaction problem, that allows constructing partially restricted design spaces. Partial RNA design is more general than any previous RNA design formulation and allows the design of RNA sequences with different lengths, while providing precise control over local and global restrictions of the sequence and structure space (Section 2).
- We improve an existing RNA design framework, *LEARNA* [Runge et al., 2019], with a masked training objective to develop *libLEARNA*, a new algorithm that is capable of efficiently navigating large RNA design spaces, ensuring the inclusion of desired motifs while exploring the space within the set parameters (Section 3).
- We evaluate *libLEARNA* on a broad range of applications (Section 4). These include evaluations on different RNA design benchmarks (Section 4.1), the design of theophylline riboswitch constructs following a previously published protocol [Wachsmuth et al., 2012] (Section 4.2.1), the design of RNAs that belong to a certain RNA family (Section 4.2.2), as well as the design of RNAs for a given interaction target (Section 4.2.3).

2 Partial RNA Design

In this section, we cover necessary background information (Section 2.1) before we formulate *partial RNA design* as a constraint satisfaction problem (Section 2.2).

2.1 Background

RNA Folding and Design We consider RNA folding as the problem of predicting the process of a RNA sequence $\phi \in \Phi^* := \{\,\mathsf{A},\,\mathsf{C},\,\mathsf{G},\,\mathsf{U}\,\}^*$ acquiring a spatial structure $\omega \in \Omega^* := \{\,.,\,),\,(\,\}^*$, where * denotes the *Kleene closure* [Fletcher et al., 1991]. RNA folding is often modeled as a function $\mathcal{F}_{\mathcal{O}} : \Phi^* \to \Omega^*$ that maps a primary sequence ϕ to its secondary structure ω (represented in the commonly used dot-bracket notation [Hofacker et al., 1994] throughout this work) with respect to a folding objective $\mathcal{O}$. Examples of this objective are maximizing the structure's stability by minimizing the free energy (MFE) [Zuker and Stiegler, 1981], maximizing base pair probabilities (MEA) [McCaskill, 1990, Do et al., 2006], or learning a RNA folding process from data [Singh et al., 2019, Chen et al., 2020, Franke et al., 2022]. RNA design addresses the inverse problem: given a target secondary structure ω, the goal is to predict an RNA sequence ϕ, such that $\omega = \mathcal{F}_{\mathcal{O}}(\phi)$. However, RNA design applications might not follow a strict inverse folding paradigm. For example, consider a typical application in a synthetic RNA design process with the goal to design a construct for ligand-dependend regulation of gene expression. In this case, one could e.g. combine a riboswitch [Mironov et al., 2002] with a downstream Shine-Dalgarno sequence [Shine and Dalgarno, 1975] for initialization of ribosome binding. This would translate to the following selection of motifs to define the design task: (1) a motif with explicit sequence and secondary structure requirements (the aptamer) followed by (2) an unspecified variable length motif (a spacer region), (3) a motif that considers specific secondary structure features only (the expression platform), and (4) a motif of an explicit sequence composition without further requirements on the secondary structure (the Shine-Dalgarno sequence). While recent RNA design formulations try to relax the originally proposed inverse RNA folding problem to enable the design of candidates for locally unrestricted design spaces (e.g. [Kleinkauf et al., 2015, Drory Retwitzer et al., 2016, Retwitzer et al., 2020, Minuesa et al., 2021]), to the best of our knowledge, none of the existing design formulations could model the described task because it contains unconstrained regions in the secondary structure and the nucleotide sequence that are not fixed to a specific length.

Constraint Satisfaction Problems CSPs are general formulations of combinatorial problems defined as a triple $(X,\,D,\,C)$, where

X is a set of variables,
D is a set of their corresponding domains of values,
C is a set of constraints.

Each variable can take on the values of its corresponding non-empty domain. A constraint $c \in C$ is a tuple (t, R) with $t \subseteq X$ being a subset of k variables from X and R being a k-ary relation over these variables. An *evaluation* of the variables is a function from a subset of variables to a set of values from their corresponding domains that satisfies a constraint (t, R) if the assigned values to the variables in t satisfy the relation R. An evaluation is called *consistent* if it does not violate any of the constraints and *complete* if it includes all variables in X. A solution to the respective CSP is an evaluation that is consistent and complete.

While CSP formulations for modeling RNA design exist [Garcia-Martin et al., 2013, 2015, Minuesa et al., 2021], the variables of these CSPs correspond to individual positions in the RNA sequence. In contrast, we formulate a general CSP where variables are defined over RNA primary- and secondary structure motifs.

2.2 The Partial RNA Design Problem

An RNA molecule can be viewed as a specific instance in an infinite, multi-dimensional RNA space, described by certain characteristics. To model such a molecule, we typically use a set of these characteristics to either locally or globally restrict the vast space in certain dimensions. For example, an RNA is often described by its nucleotide sequence, its secondary structure, e.g. as a string using the dot-bracket notation [Hofacker et al., 1994], its tertiary structure via atom-coordinates in a 3D

space, a certain GC content, its energy, or other molecular characteristics. The task of computationally modeling an RNA molecule can then be described as a search for specific characteristics of one or more dimensions of the space, given specific values of characteristics that restrict the space in certain dimensions, following an objective function that depends on an underlying model of the world. For instance, the inverse RNA folding problem can be described as a search in the primary sequence dimension, while the secondary structure dimension is restricted to a specific structure. The objective is that the primary sequence folds into the secondary structure, using for example an energy minimization approach based on a simplified biophysical world model derived from thermodynamic parameters at a certain temperature. In principle, we are not limited in the definition of dimensions, the definitions of characteristics, nor the objective function or the underlying world model when defining the search space for an RNA modeling task at hand.

This idea led us to develop partial RNA design. In essence, we formulate motifs that constrain positions of the sequence and structure dimensions of an RNA design space, while leaving the other regions of the dimension unconstrained. These unconstrained parts can then be defined as variable length regions which can be filled with arbitrary motifs by an algorithm. The design can then be viewed as a search in a partially restricted design space that allows for exploration by a machine to provide diverse solutions. While we define partial RNA design based on the primary sequence and secondary structure dimensions using the dot-bracket notation [Hofacker et al., 1994], we note that extensions to further dimensions, different notations, an other molecules are rather straightforward.

To generalize RNA design to partial RNA design, we first extend the RNA nucleotide and secondary structure alphabets, $\Phi = \{\,\mathsf{A},\ \mathsf{C},\ \mathsf{G},\ \mathsf{U}\,\}$ and $\Omega = \{\,.,\ ),\ (,\ \}$, with a *wildcard symbol* , "?", indicating a position that is not constrained. Similarly, we add a *variable length wildcard symbol*, "$\hat{?}$", to indicate an unconstrained region of variable length:

$$\begin{aligned} \widehat{\Phi} &:= \Phi \cup \{?\} \cup \{\hat{?}\} \\ \widehat{\Omega} &:= \Omega \cup \{?\} \cup \{\hat{?}\} \qquad . \end{aligned} \tag{1}$$

We define a *motif* $\mathbf{m}$ as a pair $(\hat{\phi}, \hat{\omega})$, consisting of an RNA sequence part $\hat{\phi} \in \widehat{\Phi}^*$, and an RNA structure part $\hat{\omega} \in \widehat{\Omega}^*$.

At each position $i \in \{1, \ldots, |\hat{\phi}|\}$ of the sequence part, $\hat{\phi}$, we can define a set of valid assignments, $\mathbb{V}_{\hat{\phi}_i}$, for the particular position i:

$$\mathbb{V}_{\hat{\phi}_i} = \begin{cases} \{\varphi \mid \varphi \in \{\hat{\phi}_i\}\} & \text{if } \hat{\phi}_i \in \{\mathsf{A},\ \mathsf{C},\ \mathsf{G},\ \mathsf{U}\} \\ \{\varphi \mid \varphi \in \{\mathsf{A},\ \mathsf{C},\ \mathsf{G},\ \mathsf{U}\}\} & \text{if } \hat{\phi}_i = ? \\ \{\varphi \mid \varphi \in \{\mathsf{A},\ \mathsf{C},\ \mathsf{G},\ \mathsf{U}\}^*\} & \text{if } \hat{\phi}_i = \hat{?} \end{cases} \quad , i \in \{1, \ldots, |\hat{\phi}|\}. \tag{2}$$

A valid assignment $\phi \in \Phi^*$ for the entire sequence part of the motif is then the concatenation ($\circ$) of a single element picked from each set of valid assignments at each position,

$$\mathbb{V}_{\hat{\phi}} = \left\{ \phi \;\middle|\; \phi \in \left\{ \varphi_1 \circ \cdots \circ \varphi_{|\hat{\phi}|} \;\middle|\; \varphi_i \in \mathbb{V}_{\hat{\phi}_i}\,,\ i \in \left\{1, \ldots, |\hat{\phi}|\right\} \right\} \right\} \qquad . \tag{3}$$

Similarly, for the structure part, $\hat{\omega}$, of the motif $\mathbf{m}$, we can define the sets of valid assignments $\mathbb{V}_{\hat{\omega}_i}$ and $\mathbb{V}_{\hat{\omega}}$ as

$$\mathbb{V}_{\hat{\omega}_i} = \begin{cases} \{\gamma \mid \gamma \in \{\hat{\omega}_i\}\} & \text{if } \hat{\omega}_i \in \{),\ .,\ (\} \\ \{\gamma \mid \gamma \in \{),\ (,.\}\} & \text{if } \hat{\omega}_i = ? \\ \{\gamma \mid \gamma \in \{),\ (,.\}^*\} & \text{if } \hat{\omega}_i = \hat{?} \end{cases} \quad , i \in \{1, \ldots, |\hat{\omega}|\}, \tag{4}$$

and

$$\mathbb{V}_{\hat{\omega}} = \left\{ \omega \;\middle|\; \omega \in \left\{ \gamma_1 \circ \cdots \circ \gamma_{|\hat{\omega}|} \;\middle|\; \gamma_i \in \mathbb{V}_{\hat{\omega}_i}\,,\ i \in \{1, \ldots, |\hat{\omega}|\} \right\} \right\} \qquad . \tag{5}$$

For the entire motif $\mathbf{m}$, the set of valid assignments, $\mathbb{V}_{\mathbf{m}}$, can then be written as follows:

$$\mathbb{V}_{\mathbf{m}} = \left\{ (\phi, \omega) \;\middle|\; \phi \in \mathbb{V}_{\hat{\phi}}\,,\ \omega \in \mathbb{V}_{\hat{\omega}} \right\} \qquad . \tag{6}$$

We define two variables, x and y, for the sequence and the structure part of a motif, respectively. The corresponding domains of values for x and y are defined as

$$D_x := \{\phi \mid \phi \in \Phi^*\} \qquad \text{and} \qquad D_y := \{\omega \mid \omega \in \Omega^*\} \qquad . \tag{7}$$

To meet the constraints given by the motif $\mathbf{m}$, the assigned values for x and y have to be in the set of valid assignments. We define the relation $\mathcal{R}_{\mathbb{V}_\mathbf{m}} : D_x \times D_y$ as

$$\mathcal{R}_{\mathbb{V}_\mathbf{m}} := \{(x,\, y) \mid (x,\, y) \in \mathbb{V}_\mathbf{m}\} \qquad . \tag{8}$$

For a sequence, $\mathbb{S} = (\mathbf{m}_1, \ldots, \mathbf{m}_k), k \in \mathbb{N}$, of desired motifs, we use two variables x_i and y_i, $i \in \{1, \ldots, k\}$, for each motif. These variables will define the set of variables X of our CSP formulation for partial RNA design.

An objective $O((x_1, \ldots, x_k, y_1, \ldots, x_k))$ defines a relation between the variables x_i and y_i. The solution space $\mathbb{O}$ of a given objective O defines the set of tuples $(x_1, \ldots, x_k, y_1, \ldots, y_k)$ that satisfy the objective. We can write this as the relation $\mathcal{R}_\mathbb{O} : \prod_{i=1}^k D_{x_i} \times \prod_{i=1}^k D_{y_i}$, with

$$\mathcal{R}_\mathbb{O} := \{(x_1, \ldots, x_k,\, y_1, \ldots, y_k) \mid O((x_1, \ldots, x_k,\, y_1, \ldots, y_k)) \text{ is satisfied}\} \qquad . \tag{9}$$

For instance, to define a global folding relation, we can define the relation $\mathcal{R}_{\mathcal{F}_\mathcal{O}}$ over the concatenated sequence parts, $x_1 \circ \cdots \circ x_k$, that have to fold into the concatenation of the structure parts, $\mathcal{F}_\mathcal{O}(x_1 \circ \cdots \circ x_k) = y_1 \circ \cdots \circ y_k$. We then define the relation $\mathcal{R}_{\mathcal{F}_\mathcal{O}} : \prod_{i=1}^k D_{x_i} \times \prod_{i=1}^k D_{y_i}$, with

$$\mathcal{R}_{\mathcal{F}_\mathcal{O}} := \{(x_1, \ldots, x_k,\, y_1, \ldots, y_k) \mid \mathcal{F}_\mathcal{O}(x_1 \circ \cdots \circ x_k) = y_1 \circ \cdots \circ y_k\} \qquad . \tag{10}$$

The *partial RNA design problem* can then be defined as follows.

Definition 1 ***Partial RNA Design Problem*** *Given a sequence of RNA motifs, $\mathbb{S} = (\mathbf{m}_1, \ldots, \mathbf{m}_k)$, of length $k \in \mathbb{N}$ and a set of $n \in \mathbb{N}$ objectives $\mathbb{O} = \{O_1, \ldots, O_n\}$, a solution to the partial RNA design problem associated with the sequence $\mathbb{S}$ corresponds to a solution of the constraint satisfaction problem $(X,\, D,\, C)$, defined as follows:*

$$\begin{aligned} X &= \bigcup_{i=1}^k \{x_i,\, y_i\} \qquad , \\ D &= \bigcup_{i=1}^k \{D_{x_i}, D_{y_i}\} \qquad , \\ C &= \left(\bigcup_{i=1}^k \left\{ \left(\{x_i,\, y_i\},\, \mathcal{R}_{\mathbb{V}_{\mathbf{m}_i}} \right) \right\} \right) \cup \left(\bigcup_{i=1}^n \{(X,\, \mathcal{R}_{\mathbb{O}_i})\} \right) \qquad . \end{aligned} \tag{11}$$

Now recall our example of a synthetic RNA design application combining a riboswitch and a Shine-Dalgarno sequence that could not be modeled with any of the existing RNA design formulations. Using Definition 1 and the modeling alphabets $\widehat{\Phi} = \Phi \cup \{?\} \cup \{\hat{?}\}$ and $\widehat{\Omega} = \Omega \cup \{?\} \cup \{\hat{?}\}$, this problem translates to the following task definition: (1) The aptamer can be expressed as a motif $(\hat{\phi}_1 \in \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{U}\}^{l_1}, \hat{\omega}_1 \in \{), ., (\}^{l_1})$ of an explicit sequence and structure part, $\hat{\phi}_1$ and $\hat{\omega}_1$, with a fixed length l_1. (2) The spacer region translates to a motif $(\hat{\phi}_2 \in \{\hat{?}\}, \hat{\omega}_2 \in \{\hat{?}\})$ of variable length and composition. (3) A motif $(\hat{\phi}_3 \in \{?\}^{l_3}, \hat{\omega}_3 \in \{), ., (\}^{l_3})$ with an undefined sequence part $\hat{\phi}_3$ and explicit secondary structure $\hat{\omega}_3$ of fixed length l_3 can be used for the expression platform. (4) The Shine-Dalgarno sequence corresponds to a motif $(\hat{\phi}_4 \in \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{U}\}^{l_4}, \hat{\omega}_4 \in \{?\}^{l_4})$ of a desired sequence part $\hat{\phi}_4$ and an undefined structure part $\hat{\omega}_4$ of length l_4. We design theophylline riboswitch constructs using a similar search space in Section 4.2.1.

The partial RNA design problem describes a completely new RNA design paradigm: In contrast to previous work that tries to find an optimal solution for a given target structure, our approach seeks to design RNAs that are contained in a certain design space. A narrow space for inverse RNA folding could e.g. be defined with a single motif of a complete secondary structure with unknown sequence, while a large design space might contain multiple regions of variable length wildcard symbols ($\hat{?}$) that allow for exploration. We further note that we do not limit the definition of the motifs. For example structure parts can contain any composition of brackets, balanced or unbalanced. This allows to create structural diversity and further enables us to extend the task at any given point since we are no longer bound to explicit pairing positions. However, one can define an objective to define specific positions that have to be paired if desired. In libLEARNA we allow to index corresponding pairing positions

Table 1: Overview of the training and validation sets.

DATA SET	TASKS	MEAN LENGTH	MEDIAN LENGTH	MIN/MAX LENGTH
TRAINING1 ("LONG")	100000	470.9	300	200–8033
TRAINING2 ("SHORT")	100000	103.9	99	23–200
TRAINING3 ("RANDOM")	100000	142.7	106	23–6361
VALIDATION	100	135.8	94	46–1802

to explicitly indicate base pairs. These explicit pairs are treated as soft constraints to increase the likelihood of a pair by placing Watson-Crick base pairs at the marked positions.

In the following section, we describe libLEARNA, an algorithm that is capable of designing RNAs from provided sequence and structure motifs.

3 RNA Design from Desired Motifs

Before we describe our new algorithm, libLEARNA, we briefly recap its foundation, the automated reinforcement learning framework *LEARNA*. We then detail the changes we made to the state representations, the training procedure, and the AutoRL process.

3.1 The LEARNA Framewrok

LEARNA is an automated deep reinforcement learning (AutoRL) [Parker-Holder et al., 2022] algorithm for the generative design of RNA sequences, following an inverse RNA folding approach [Hofacker et al., 1994]. The algorithm has two stages: In an inner-loop, a deep reinforcement learning (RL) algorithm learns an RNA design policy across thousands of inverse RNA folding tasks. More precisely, an agent periodically interacts with an environment via trial and error. The agent receives a state from the environment and chooses an action based on the current state. The action is communicated to the environment to compute a reward signal and inform the agent about the value of the chosen action. In LEARNA, states are defined as local representations of the input structure in dot-bracket format [Hofacker et al., 1994] using an n-gram centered around the current position. Actions correspond to placing a nucleotide (A, C, G, or U), or directly placing Watson-Crick pairs (A-U, U-A, G-C, or C-G) in case the structure indicates that the given position is paired to another nucleotide. The reward is computed once all sites have been assigned nucleotides; after applying a folding algorithm to the designed sequence, the reward function is based on the Hamming distance between the folded candidate and the desired structure. Over the time, the agent learns a policy to decide which action to choose given a certain state. This policy is approximated with a neural network. The trained agent is evaluated on a validation set and the validation loss is communicated to an efficient Bayesian Optimization method, BOHB [Falkner et al., 2018]. BOHB jointly optimizes the configuration of the RL system in the outer loop, seeking to minimize the observed loss. Finally, the best-performing configuration on the validation set is evaluated at test time.

Runge et al. [2019] propose three versions within this framework: (1) A version that is not trained at all but allows weight updates during evaluation (LEARNA), (2) a version that leverages the pretraining but fixes the weights during evaluation for fast sampling of candidates (Meta-LEARNA), and (3) a pretrained version that allows weight updates at evaluation time (Meta-LEARNA-Adapt). We use the Meta-LEARNA-Adapt (3) approach to develop libLEARNA, described in the following section.

3.2 libLEARNA

We improve Meta-LEARNA-Adapt to design RNA candidates from arbitrary sequence and structure constraints. To do so, we extend the state representation, change the training procedure, and change the objective of the meta-optimization process while also adding further dimensions to the configuration space.

State Representations To enable libLEARNA to process sequence and structure information, we extend the state representations of Meta-LEARNA-Adapt: We numerically encode pairs of a sequence

Table 2: Examples of different tasks in the training data.

TASK DESCRIPTION	TASK SPACE	EXAMPLE	PERCENTAGE OF DATA
INVERSE RNA FOLDING	STRUCTURE	`...........((((....))))......`	11,5
	SEQUENCE	`??????????????????????????????`	
ALTERNATING CONSTRAINTS	STRUCTURE	`?....???.?.((((...??????????`	66,7
	SEQUENCE	`C????UAU?C????????UCAGAUAAAAC`	
RANDOM MASKING	STRUCTURE	`?....???.?.((((...??????????`	21,8
	SEQUENCE	`C????U?U?C????????UCAGA????AC`	

symbol and its corresponding structure symbol for each position of a given design task. This enlarges the state space of libLEARNA but allows to apply it to a broad range of design tasks.

Training We use a masked training objective similar to the masked language model training in BERT [Devlin et al., 2018]. More precisely, we follow the protocol of Runge et al. [2019] to generate three training datasets from sequences of the Rfam [Griffiths-Jones et al., 2003] database version 14.1 with different length distributions ($\leq$ 200 nucleotides (nt), $\geq$ 200 nt, random length) of 100000 samples each, and a non-overlapping validation set of 100 samples using *RNAfold* [Lorenz et al., 2011] to receive the structures. The datasets are described in Table 1. The choice of the training dataset is an additional hyperparameter during the optimization process; the best performing configuration of libLEARNA was trained on the dataset with samples $\leq$ 200 nucleotides.

Instead of specifically training for the task of inverse RNA folding by masking only the entire sequences of the samples, we first mask up to five parts of the structures, each covering up to 20% of the total length, while the positions, the lengths, and the number of parts are sampled uniformly at random. In the second step, we mask corresponding parts of the sequences that remain unmasked in the structures to derive tasks of alternating sequences and structure constraints. Finally, we randomly mask the sequences of $\sim$20% of the samples to derive tasks that correspond to RNA design from arbitrary sequence and structure motifs. Examples of the resulting training tasks are shown in Table 2. The task of *libLEARNA* during training is to fill the masked parts of the sequence such that, after folding the designed sequence, all positions of the resulting folding satisfy the positional constraints of the masked structure. Similar to Runge et al. [2019], the reward is based on the Hamming distance, while masked positions in the structure are ignored. We show examples of tasks of the training data in Table 2.

Meta-Optimization For *libLEARNA*, we mainly adopt the configuration space proposed by Runge et al. [2019] but introduce four new dimensions: (1) While algorithms for the inverse RNA folding problem typically benefit from directly predicting Watson-Crick base pairs at sites that ought to be paired, the pairing partners for many paired sites are not trivially distinguishable as a result of our masking procedure during training. However, we include the choice to make use of the direct prediction of pairs if the pairing partner can be identified, i.e. there is no masking between the pairing positions. (2) We add a choice to the configuration space to dynamically adapt the states based on the design progress to inform the agent about its own decisions. (3) The choice of training data and (4) the schedule of the tasks during training can have a strong impact on the final performance of the learning algorithm. We, therefore, include two dimensions to choose from three different training data distributions and curricula (unsorted tasks or sorted by length). We further allowed searching over an additional LSTM layer. The result is an 18-dimensional search space to jointly optimize over the network architecture, all parts of the RL system, as well as training hyperparameters, task distributions, and schedule, using BOHB [Falkner et al., 2018]. We use the exact same setup during meta-optimization as Runge et al. [2019], with the same training budgets and validation protocols. However, while Runge et al. [2019] optimized for an RL algorithm without any policy updates at test time, we directly optimize for an algorithm with policy updates at evaluation time to increase the adaptation capabilities of our approach. Figure 1 gives an overview over the meta-optimization procedure.

4 Experiments

We assess the performance of libLEARNA across a wide range of RNA design tasks. We evaluate libLEARNA in restricted design spaces using different benchmark datasets (Section 4.1), and demonstrate its capabilities to explore large open design spaces when designing massive amounts

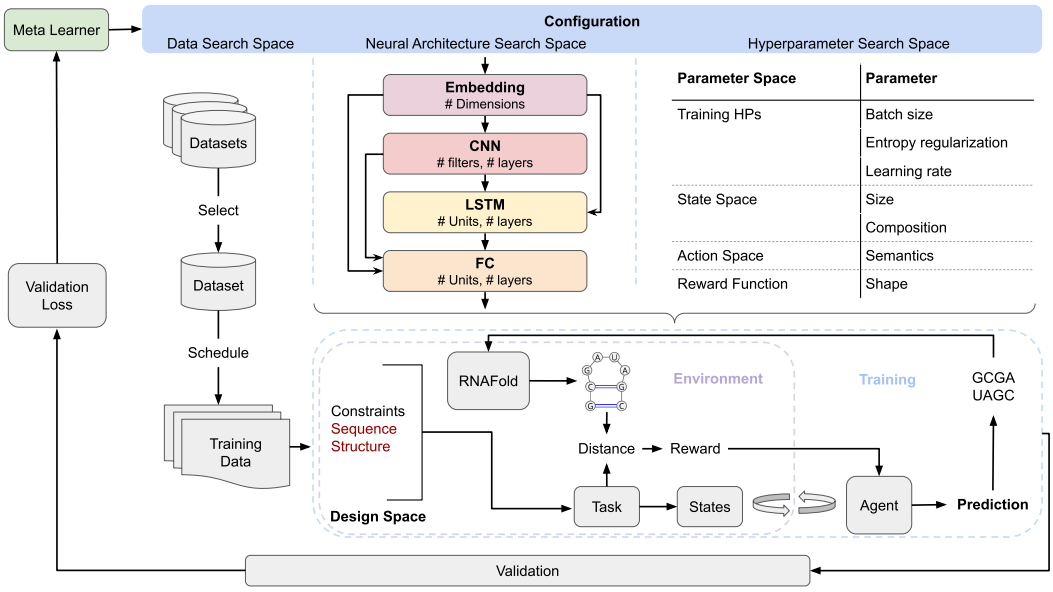

Figure 1: Meta-optimization loop. In each iteration, the meta-learner (BOHB) samples a configuration from a rich configuration space. The sampled configuration defines the learning algorithm's hyperparameters, a specific environment, a particular network architecture of the agent, a training curriculum, and a training data set. These components together formulate the learner; a deep reinforcement learning algorithm, which is trained on the selected training set and evaluated on a validation set. The resulting validation loss is communicated to the meta-learner to update its model, seeking to learn to sample better configurations with each iteration.

of theophylline riboswitch constructs following a previously proposed protocol [Wachsmuth et al., 2012] (Section 4.2.1), with the design of RNAs of a specific family represented as a covariance model (Section 4.2.2), and when designing RNAs that interact with a given target RNA (Section 4.2.3). Finally, we analyze its robustness to changes of the objective function (Section 4.3.1) and the impact of the different components of libLEARNA in an ablation study (Section 4.3.2). We note that we use the same model of libLEARNA for all experiments without retraining or changing any parameters.

4.1 RNA Design in Restricted Search Spaces

We benchmark libLEARNA on three tasks of fixed-length design spaces. We evaluate each algorithm in five independent runs, allowing one core with 5GB ram using Broadwell E5-2630v4 2.2 GHz CPUs. Following the standard protocol in the field, we report the accumulated number of solved tasks across all five runs for the task of inverse RNA folding (Table 3) and inverse RNA folding with desired GC-contents (Table 4). For a new benchmark of randomly masked sequences, we report the average number of solved tasks with the standard deviation around the mean (Figure 2).

4.1.1 Inverse RNA Folding

We first compare libLEARNA against its predecessors LEARNA, Meta-LEARNA, and Meta-LEARNA-Adapt on the challenging Eterna100 benchmark version 2 [Koodli et al., 2021] to assess the influence of the applied changes to the framework. We follow the proposed evaluation protocol on this benchmark, using a timeout of 24 hours per target. Since libLEARNA is specifically based on the Meta-LEARNA-Adapt algorithm, we use the exact same setup as for Meta-LEARNA-Adapt, using a restart timeout of 1800 seconds, i.e. the algorithm resets all weights to the initial values after 1800 seconds,

Table 3: Inverse RNA folding results.

Variation	Solved Tasks [%]
LEARNA	70
Meta-LEARNA	74
Meta-LEARNA-Adapt	70
libLEARNA (ours)	**76**

essentially starting from scratch to avoid getting stucked in local minima as proposed by Runge et al. [2019]. Similarly, LEARNA uses the same restart timeout, while Meta-LEARNA does not apply weight updates at all. Table 3 shows the result of the evaluation. We observe that libLEARNA outperforms its predecessors, solving 76 targets of the benchmark followed by Meta-LEARNA with 74 solved tasks. Both, LEARNA and Meta-LEARNA-Adapt, solve a total of 70 tasks. This is a surprising result, since in contrast to LEARNA, Meta-LEARNA, and Meta-LEARNA-Adapt, libLEARNA was not specifically trained for the inverse RNA folding problem.

4.1.2 Inverse RNA Folding with GC-Content

To evaluate the capabilities of libLEARNA to solve tasks with a desired GC content, we implement an additional GC-loss term in the reward function. In essence, we substract the deviation from the desired GC content with a given tolerance of 0.01 (setting the loss to zero if the GC content is in the epsilon interval around the desired GC content) from the existing structure loss to define the reward. The reward function then is a weighted sum (without any tuning, setting all weights to 1) of the structure- and the GC-loss. However, similar to the structural improvement step described by Runge et al. [2019], we implement a GC-improvement step (GCIS) to guide the agent for this challenging task. The GCIS becomes active if the Hamming distance is less than 5 and replaces nucleotides (or pairs) to improve the GC content. We note that libLEARNA was not trained to include desired GC contents. We evaluate libLEARNA against antaRNA [Kleinkauf et al., 2015] and MCTS-RNA [Yang et al., 2017] on all pseudoknot-free tasks of the ArchiveII dataset [Sloma and Mathews, 2016] with a timeout of one hour and report the accumulated number of solved tasks across five independent runs. Table 4 shows the results. Remarkably, libLEARNA is on par with MCTS-RNA while clearly outperforming antaRNA. Since libLEARNA was not trained with desired GC-contents the results show that it can quickly adapt to changes of the objective. We analyse the robustness of libLEARNA to further changes of the objective function in more detail in Section 4.3.1.

Table 4: Results for inverse RNA folding with desired GC-contents.

Variation	Solved Tasks [%]
antaRNA	52
MCTS-RNA	**73**
libLEARNA (ours)	**73**

4.1.3 RNA Design on Randomly Masked Targets

To evaluate on tasks that are closer related to the partial RNA design problem, we create a third benchmark dataset from tasks derived from the Rfam database version 14.1. We select a total of 100 sequences that do not overlap with the training nor validation data and randomly masked parts of the sequences and the structures after folding with ViennaRNA's RNAFold [Lorenz et al., 2011]. We compare libLEARNA against antaRNA [Kleinkauf et al., 2015] and MoiRNAiFold [Minuesa et al., 2021], two algorithms that are capable of predicting RNAs from masked structures. However, in contrast to libLEARNA, these algorithms require that all unmasked brackets in the structures are balanced and represent desired pairs. We therefore unmasked the respecitve pairing brackets if only one part of a pair was masked during the random masking procedure. A task counts as solved if all constraints in the sequence and the structure are satisfied, i.e. multiple designs are valid for a given input task. We run each algorithm for one hour on the created dataset in five independent runs and report the average number of solved tasks with the standard deviation around the mean. Figure 2 shows the results of the evaluation. We observe that libLEARNA clearly outperforms antaRNA and MoiRNAiFold.

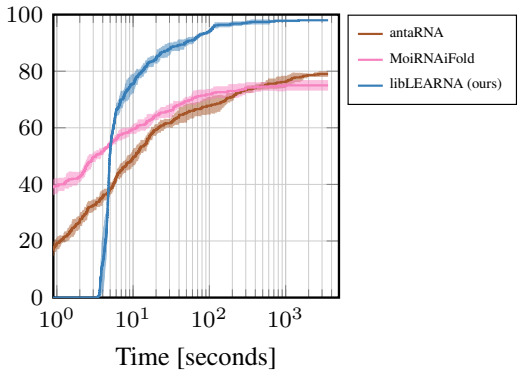

Figure 2: Comparison on randomly masked tasks with balanced brackets.

4.2 RNA Design in Open Search Spaces

In the previous sections, we demonstrate libLEARNA's ability to solve tasks of very restricted design spaces like inverse RNA folding, analyzed its ability to adapt to new objective functions when designing RNAs with desired GC contents, while also showing that it can explore design spaces if the solution spaces are less restricted. However, all previous results were obtained in design spaces with a fixed length. For the exploration of partially restricted RNA spaces of variable lengths as proposed in the partial RNA design problem, libLEARNA employ a one-shot strategy. The input to libLEARNA is a partially restricted design space with certain length restrictions. To generate candidates, libLEARNA then internally samples a fixed length task from the design space at random, while positions marked with a variable length wildcard symbol, i.e. unconstrained regions of arbitrary length, are filled with wildcard symbols for unconstrained positions ('?'), sampled with a random length within the given length constraints. libLEARNA then tries to solve the task with a single shot.

However, before evaluating libLEARNA across tasks of the partial RNA design problem, we first evaluate its ability to solve a given task with a single shot as this is an essential requirement for successfully applying the proposed strategy to tasks of the partial RNA design problem. We, therefore, evaluate libLEARNA's first shot performance against Meta-LEARNA, the best performing approach of the LEARNA algorithms proposed by Runge et al. [2019], on the 100 tasks of the Eterna100 benchmark version 2 [Koodli et al., 2021]. Figure 3 compares the average Hamming distances of the first predictions on each of the tasks between Meta-LEARNA and libLEARNA. We observe that libLEARNA clearly outperforms Meta-LEARNA in its own discipline, showing generally strong 1-shot performance. We take this as an indicator that libLEARNA should be capable to also follow the proposed 1-shot strategy on other tasks, thus, being able to solve partial RNA design tasks. We further note that, in contrast to Meta-LEARNA, libLEARNA further updates its policy to adapt to a given target or across targets in a single run similar to its training strategy. This essentially allows libLEARNA to adapt to an entire design space by updating its weights after each sampled target at each iteration.

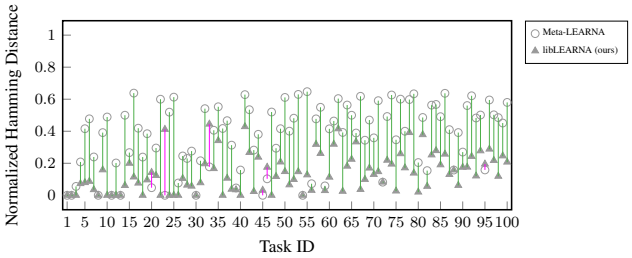

Figure 3: Comparison of the first predictions of Meta-LEARNA and libLEARNA on all tasks of the Eterna100 benchmark version 2. The plot shows the average normalized Hamming distance of the first prediction for all targets of the benchmark. Green bars indicate tasks where libLEARNA achieves lower Hamming distance, purple bars indicate tasks where the Hamming distance is above that of Meta-LEARNA's prediction.

In the following, we demonstrate libLEARNA's ability to solve tasks of the partial RNA design problem with three explicit experiments: the design of theophylline riboswitch constructs following a previously published protocol [Wachsmuth et al., 2012], the design of sequences that match a given RNA family using covariance models, and the design of RNAs that interact with a given target RNA.

4.2.1 Design of Theophylline Riboswitches

Originally, Wachsmuth et al. [2012] constructed theophylline riboswitch candidates for transcriptional activation from (1) the TCT8-4 theophylline aptamer sequence and structure, (2) a spacer sequence of 6 to 20 nucleotides (nt), (3) a sequence of 10nt to 21nt complementary to the $3'$-end of the aptamer, and (4) a U-stretch of 8nt at the $3'$-end of the construct. To generate candidate sequences, Wachsmuth et al. [2012] create a large library of random sequences for the spacer region (6-20nt) and a library of sequences complementary to the $3'$-end of the aptamer (10-21nt). From these sets, randomly sampled sequences were combined with the aptamer and the 8-U-stretch.

Design Space Formulation We start the formulation of the design space from the entire sequence and the structure part of the unbound aptamer, a spacer region of unknown sequence, a region complementary to the $3'$-end of the aptamer of at least 10nt and the unpaired 8-U-stretch, similar to

Table 5: Originally proposed theophylline riboswitch constructs and partial RNA design space formulation. (Top) The sequence parts of the six originally proposed riboswitch constructs by Wachsmuth et al. [2012] and the sequence part of the design space. (Bottom) The corresponding structure parts of the constructs and the structural part of the design space. Red: TCT8-4 theophylline aptamer; green: variable length spacer domain; blue: domain ought to pair with the aptamer (complementary to the 3′-end of the aptamer sequence in the original design); black: 8-U-stretch. Masked positions are indicated with ?, positions for extensions are indicated with $\hat{?}$.

Construct	Aptamer	Spacer	Complementary Region	8-U-Stretch
RS1 Sequence	AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCA	UUACAUC	UGAAGUGCUGCC	UUUUUUUU
RS2 Sequence	AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCA	UGAUCUCGCU	UGAAGUGCUGC	UUUUUUUU
RS3 Sequence	AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCA	UUUACAUACUCGGUAAAC	UGAAGUGCUGCCA	UUUUUUUU
RS4 Sequence	AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCA	AACCGAAAUUUGCGCU	UGAAGUGCUGC	UUUUUUUU
RS8 Sequence	AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCA	CUCCUAGUGGAG	UGAAGUGCUG	UUUUUUUU
RS10 Sequence	AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCA	GAAAUCUC	UGAAGUGCUG	UUUUUUUU
Task Sequence Parts	AAGUGAUACCAGCAUCGUCUUGAUGCCCUUGGCAGCACUUCA	$\hat{?}$??????$\hat{?}$	UGAAGUGCUG$\hat{?}$	UUUUUUUU
RS1 Structure	...........(((((.....)))))....((((((((((((	.....	))))))))))))	.....
RS2 Structure	...........(((((.....)))))....((((((((((((	.....	))))))))))))	).....)
RS3 Structure	...........(((((.....)))))...(((((((((((((	(((((.....))))).	)))))))))))))	.....
RS4 Structure	...........(((((.....)))))....((((((((((((	(..((.....)).)	))))))))))))	).....)
RS8 Structure	.....((((((((.....)))))...)))(((((((((((	((((....))))	))))))))))	.....
RS10 Structure	.....(((((((((.....)))))...)))(((((((((((	((....))	))))))))))	.....
Task Structure Parts	.....???(((((.....)))))...???(((((((((((	$\hat{?}$??....$\hat{?}$	)))))))))))$\hat{?}$	?.....

Wachsmuth et al. [2012]. To ensure a length of 6nt of the spacer region, we use the shared structure constraints of the six final constructs proposed by Wachsmuth et al. [2012], which were also used to restrict the unknown parts of the aptamer. We then introduce three positions where the task can be extended (indicated by $\hat{?}$ in Table 5) to fit the requirements and fix the length of the design space to 66nt-91nt according to Wachsmuth et al. [2012]. The final definition of the design space and the proposed constructs of Wachsmuth et al. [2012] are shown in Table 5.

We assess the performance of libLEARNA against the originally proposed library generation procedure proposed by Wachsmuth et al. [2012] in two experiments: (1) The design of a riboswitch library based on sequence and structure constraints only, and (2) the design of sequences when additionally querying libLEARNA to design candidates with a specific G and C nucleotide ratio (GC-content), given a tolerance of 0.01. Again, the GC-loss term was simply added to the reward function of libLEARNA. For each experiment, we generate 50000 candidates with the approach of Wachsmuth et al. [2012] and libLEARNA with five different random seeds and evaluate them as follows.

Evaluation The evaluation of the designed candidates was performed following Wachsmuth et al. [2012]: after dropping duplicates, the designed sequences were folded using RNAfold and verified for (1) the existence of two hairpin structure elements, the aptamer hairpin for binding the ligand and the terminator hairpin formed between the 3′-end of the aptamer, the spacer and the region complementary to the aptamer sequence, (2) no pairing within the last seven nucleotides of the 8-U-stretch, (3) no pairing between the spacer and the aptamer in a sequence of folding steps, simulating co-transcriptional folding with a fixed elongation speed of five. A candidate that did not pass all criteria was rejected.

Results We observe that libLEARNA generates considerably more candidates that pass the design criteria compared to the original procedure proposed by Wachsmuth et al. [2012], yielding 30% more satisfying candidates on average (Table 6). Further, the candidates are nearly uniformly distributed across the lengths of the design space, especially for the longer sequences (Figure 4 left), and the structure diversity generated by libLEARNA is around 23% higher (Table 6). When designing candidates with desired GC-contents, libLEARNA provides up to 47% more candidates that satisfy the design criteria and contain the specific G and C nucleotide distribution (Figure 4 right). Remarkably, libLEARNA can also design such candidates on the margins of possible GC-contents (0.3 and 0.6) which is between 0.29 and 0.63 for the given riboswitch design space.

Table 6: Overview of designed riboswitch candidates.

Method	Valid Candidates [%]	Unique Structures
Wachsmuth et al. [2012]	41,7	6316.6
libLEARNA	**70,9**	**8269.6**

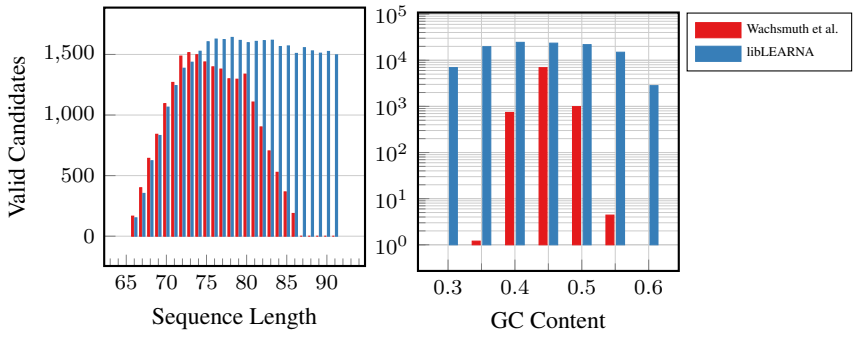

Figure 4: Average number of designed riboswitches across different length and GC-contents.

4.2.2 Design of RNAs that Match Other RNAs

Another interesting application for the exploration of partially restricted design spaces is the design of candidates that match a set of other sequences. We explore this task by designing candidates for a given covariance model. As an example, we use the family of Hammerhead ribozymes (Type III) (Rfam family: RF00008), provided as a covariance model with a CLEN of 54. We use the consensus structure with the conserved nucleotides as the task, while allowing for exploration at the beginning, end, and in the loop regions. We design candidates with a total length between 50 and 60 nucleotides. To make libLEARNA applicable to this objective, we implement a new reward function to optimize the bitscore of the designed sequence, ϕ, directly: $R = bitscore(\phi)$. The reward for a candidate that does not match the covariance model at all was set to -200. We make five independent runs for the design procedure and report the average bitscore every 100 episodes across all runs. Figure 5 shows the results of the design with libLEARNA and with a random agent that uniformly samples actions. libLEARNA quickly optimizes the bitscore to roughly 25 on average after very few steps and further optimizes the candidates to a score of roughly 30 on average, while the average bitscores of the random agent stay below zero all the time. For reference, we also sampled 100 sequences directly from the covariance model using cmemit of the Infernal package [Nawrocki and Eddy, 2013]. The bitscores of these sequences were between 18 and 67.

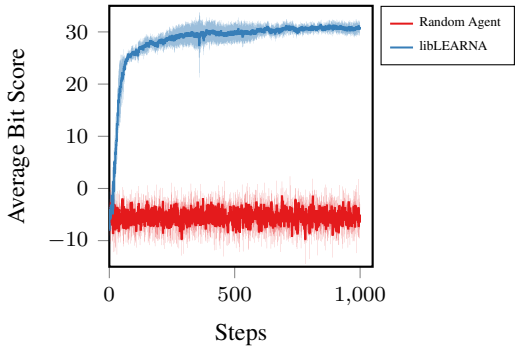

Figure 5: Average Bit Score of designed candidates for the covarinance model of Hammerhead Ribozyme (Type III). A step in the figure corresponds to 100 episodes.

4.2.3 RNA-RNA Interactions

To investigate the influence of the design space formulation on the performance, we use libLEARNA to design candidates for RNA-RNA interactions (RRI). More precisely, we design candidate sequences for a given target RNA and evaluate the energy of the interaction using IntaRNA [Mann et al., 2017]. We use the example 1 inputs from the IntaRNA web server[2] as templates for our formulation of design spaces. In particular, we define three search spaces: (1) A completely unrestricted search space that only contains a variable length wildcard symbol such that libLEARNA can design any sequence without knowledge about the structure nor the sequence. The design is only guided by

[2]The web server is available at `https://rna.informatik.uni-freiburg.de/IntaRNA/Input.jsp?example=IntaRNA_5724482`.

the reward function. (2) A search space that only restricts the structure part. We fold the original sequence provided in the IntaRNA example and introduce a point of extension (a variable length wildcard symbol $\hat{?}$) between the resulting two hairpins. Here, the design is guided by desired structure motifs. (3) A search space with restrictions in the structure and the sequence part, also divided into two motifs. The exact search space definitions look as follows:

Unconstrained	$\hat{?}$ $\hat{?}$
Structure Only	????????????????????????????$\hat{?}$???????????????????????????????? ((((((((((....))))))...))))).....$\hat{?}$.....((((((....)))))).....
Sequence and Structure	AGG?????????????CA????????????$\hat{?}$????????????????????????UUUUUU ((((((((((....))))))...))))).....$\hat{?}$.....((((((....)))))).....

We implement a new reward function that optimizes for the inverse of the energy of the two RNAs. We use the following reward function with a slight shaping factor of 3.

$$R = (Energy \cdot (-1))^3 \qquad (12)$$

However, we did not tune the reward function; we used our first guess. Again we compare libLEARNA with its random variant and run each algorithms five times for 500000 episodes. Figure 6 shows the average results of the RRI design. We observe that libLEARNA can design candidates with a lower energy of the complex compared to the random agent and the initial example sequence (indicated as a red line in Figure 6). Interestingly, libLEARNA quickly adapts to the reward function even without any sequence and structure information (unconstrained search space). However, the structure information seems to be beneficial in the long run since the final energy for the designed candidates using the design space with the structure information only is the lowest of all three search space formulations. The search space that contains sequence and structure restrictions seems to impose more challenges to the design process. While the resulting candidates still clearly show a lower energy with the target RNA compared to the original sequence, the optimization is slower and results in a higher final energy. Overall, we conclude that the formulation of the search space can have substantial impact on the outcome of a given design task.

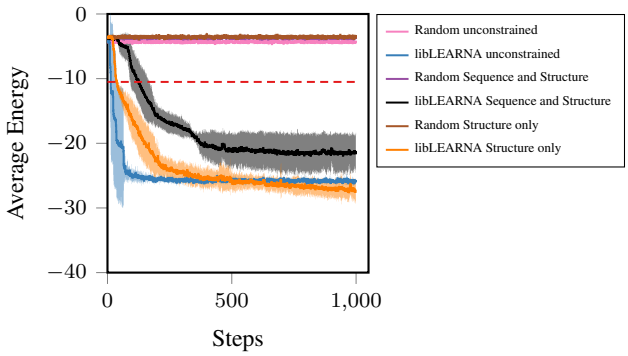

Figure 6: Average energy of RRI complex. A step in the figure corresponds to 500 episodes. The values are averages across five independent runs with standard deviation around the mean displayed as confidence bounds. Dashed red line indicates the energy level of the templae sequence in complex with the target RNA.

4.3 Robustness and Ablation Study

4.3.1 Robustness

We observe that libLEARNA quickly adapts to changes of the objective function. To further investigate this feature, we analyzed the behavior of libLEARNA with an additional experiment. We again use the Eterna100 benchmark dataset for inverse RNA folding to define a rather restricted design space with a fixed length for each task. For our evaluation, we either change the distance measure in the reward function, the folding algorithm, or both and evaluate each variation

Table 7: Analysis of robustness.

Variation	Solved Tasks [%]	Solved Tasks (no restart) [%]
libLEARNA	76	72
libLEARNA-WL	**77**	**73**
libLEARNA-LV	**76**	**75**
libLEARNA-MEA	75	**78**
libLEARNA-LinearFold	73	**72**
libLEARNA-MEA-WL	71	**78**
libLEARNA-MEA-LV	70	**75**
libLEARNA-LinearFold-WL	73	**75**
libLEARNA-LinearFold-LV	72	**74**

with and without restarts of the weights every 1800 seconds (see Section 4.1.1), using five independent runs for 24 hours on each of the 100 tasks. For the distance metric, we replace the Hamming distance either with a Weisfeiler-Lehman graph kernel as recently proposed for secondary structure evaluation [Runge et al., 2023] , or with the Levenshtein distance to measure structural similarity. For the folding engines, we either use the linear approximation of RNAfold implemented in LinearFold-V [Huang et al., 2019], or the RNAFold variant for the maximum expected accuracy (MEA) structure instead of the minimum free energy structure. Table 7 shows the accumulated number of solved tasks across all runs for each of the variants. Surprisingly, the changes to the distance metrics both seem to be beneficial, showing better performance than the original libLEARNA baseline with and without restarts of the algorithm, although libLEARNA was trained with the Hamming distance. For the change of the folding algorithm, however, the restarting procedure seems to be harmful: While each change of the folding engine decreases the performance compared to the libLEARNA baseline in the setting with resets, the change of the folding engine even results in the best performance (RNAFold-MEA, 78 solved tasks; RNAFold-MEA with WL, 78 solved tasks) when the restart is not performed at all. We conclude that switching the underlying folding engine is a more critical change to the algorithm and libLEARNA requires to adapt to this change longer than the given restart timeout allows. However, when libLEARNA is given enough time for adaptation, the change might even be beneficial. Overall, libLEARNA seems to be very robust and is capable to adapt to new, unseen settings.

4.3.2 Ablation Study

Finally, we perform an ablation study where we remove one component of libLEARNA at a time: The weight updates during evaluation (no Adapt), the restart procedure (no Restart), the improvement step proposed by Runge et al. [2019] that we adopted (no IS), the model by using a random agent (no Model), and the prediction of pairs when the pairing partner is known (no Pairs). We again use the Eterna100 benchmark data set version 2 to evaluate the algorithm in a fixed setup. Table 8 shows the results. We observe that all components are important for the performance of libLEARNA. However, the model and the direct prediction of pairs seems to have the most impact on the performance. Regarding the prediction of pairs, however, this impact is potentially largely attributed to the design of tasks of the inverse RNA folding problem where the entire structure is known. As soon as there is an unconstrained position in the structure, libLEARNA cannot fully leverage the prediction of pairs due to ambiguous structures in the design space

Table 8: Ablation Study.

Variation	**Avg. Solved Tasks [%]**
libLEARNA	76
no Restart	72
no Adapt	71
no IS	68
no Model	42
no Pairs	41

5 Conclusion

In this work, we propose *partial RNA design*, a new RNA design paradigm formulated as a constraint satisfaction problem. Partial RNA design considers the problem of designing RNA sequences from provided sequence and structure motifs with given properties. The general idea is to treat RNA design as a search in a partially restricted design space which can be explored by a machine. We then propose libLEARNA, a robust automated reinforcement algorithm capable of efficiently exploring such partially restricted design spaces. We show that libLEARNA shows strong performance across a wide range of different RNA design tasks formulated as partial RNA design problems, including benchmarking tasks in very restricted design settings and tasks that require exploration of open design spaces like the design of riboswitches or the design of RNAs for RNA-RNA interactions. Our work describes a novel approach to RNA design where the algorithm provides large amounts of solutions, given restrictions of its search space rather than designing one or few solutions for a very narrow task definition, e.g. a provided secondary structure. We believe that our new approach adds a new dimension to the RNA design problem and that libLEARNA is a useful tool for future RNA design endeavors.

Acknowledgments

This research was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant number 417962828. The authors further acknowledge support by the state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through grant no INST 39/963-1 FUGG.

References

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

A. Avihoo, A. Churkin, and D. Barash. Rnaexinv: An extended inverse rna folding from shape and physical attributes to sequences. *BMC bioinformatics*, 12(1):319, 2011.

A. Busch and R. Backofen. Info-rna—a server for fast inverse rna folding satisfying sequence constraints. *Nucleic acids research*, 35(suppl_2):W310–W313, 2007.

B. Cao, T. Wang, Q. Qu, T. Kang, and Q. Yang. Long noncoding rna snhg1 promotes neuroinflammation in parkinson's disease via regulating mir-7/nlrp3 pathway. *Neuroscience*, 388:118 – 127, 2018. ISSN 0306-4522.

X. Chen, Y. Li, R. Umarov, X. Gao, and L. Song. {RNA} secondary structure prediction by learning unrolled algorithms. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=S1eALyrYDH`.

O. K. Choong, D. S. Lee, C.-Y. Chen, and P. C. Hsieh. The roles of non-coding rnas in cardiac regenerative medicine. *Non-coding RNA Research*, 2(2):100 – 110, 2017. ISSN 2468-0540.

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

C. B. Do, D. A. Woods, and S. Batzoglou. Contrafold: Rna secondary structure prediction without physics-based models. *Bioinformatics*, 22(14):e90–e98, 2006.

I. Dotu, J. A. Garcia-Martin, B. L. Slinger, V. Mechery, M. M. Meyer, and P. Clote. Complete RNA inverse folding: computational design of functional hammerhead ribozymes. *Nucleic Acids Research*, 42(18):11752–11762, 2014. doi: 10.1093/nar/gku740. URL `http://dx.doi.org/10.1093/nar/gku740`.

M. Drory Retwitzer, V. Reinharz, Y. Ponty, J. Waldispühl, and D. Barash. incaRNAfbinv: a web server for the fragment-based design of RNA sequences. *Nucleic Acids Research*, 44(W1):W308–W314, 05 2016. ISSN 0305-1048. doi: 10.1093/nar/gkw440. URL `https://doi.org/10.1093/nar/gkw440`.

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

S. Falkner, A. Klein, and F. Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1437–1446, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

J. Fernandez-Chamorro, G. Lozano, J. A. Garcia-Martin, J. Ramajo, I. Dotu, P. Clote, and E. Martinez-Salas. Designing synthetic rnas to determine the relevance of structural motifs in picornavirus ires elements. *Scientific reports*, 6:24243, 2016.

Q. R. Ferry, R. Lyutova, and T. A. Fulga. Rational design of inducible crispr guide rnas for de novo assembly of transcriptional programs. *Nature communications*, 8(1):1–10, 2017.

P. Fletcher, H. Hoyle, and C. W. Patty. *Foundations of discrete mathematics*. Boston, MA: PWS-Kent Publishing Company, 1991. ISBN 0-534-92373-9.

J. Franke, F. Runge, and F. Hutter. Probabilistic transformer: Modelling ambiguities and distributions for rna folding and molecule design. *Advances in Neural Information Processing Systems*, 35: 26856–26873, 2022.

J. A. Garcia-Martin, P. Clote, and I. Dotu. Rnaifold: a constraint programming algorithm for rna inverse folding and molecular design. *Journal of Bioinformatics and Computational Biology*, 11 (02):1350001, 2013.

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

R. Giegerich, B. Voß, and M. Rehmsmeier. Abstract shapes of rna. *Nucleic acids research*, 32(16): 4843–4851, 2004.

S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna, and S. R. Eddy. Rfam: an RNA family database. *Nucleic Acids Research*, 31(1):439–441, 01 2003. ISSN 0305-1048. doi: 10.1093/nar/gkg006. URL `https://doi.org/10.1093/nar/gkg006`.

S. Hammer, C. Günzel, M. Mörl, and S. Findeiß. Evolving methods for rational de novo design of functional rna molecules. *Methods*, 161:54 – 63, 2019. ISSN 1046-2023. doi: https://doi.org/10.1016/j.ymeth.2019.04.022. URL `http://www.sciencedirect.com/science/article/pii/S1046202318302895`. Development and engineering of artificial RNAs.

I. Hofacker, W. Fontana, P. Stadler, S. Bonhoeffer, M. Tacker, and P. Schuster. Fast Folding and Comparison of RNA Secondary Structures. *Monatshefte fuer Chemie/Chemical Monthly*, 125: 167–188, 02 1994.

L. Huang, H. Zhang, D. Deng, K. Zhao, K. Liu, D. A. Hendrix, and D. H. Mathews. Linearfold: linear-time approximate rna folding by 5'-to-3'dynamic programming and beam search. *Bioinformatics*, 35(14):i295–i304, 2019.

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

R. Kleinkauf, T. Houwaart, R. Backofen, and M. Mann. antaRNA–Multi-objective inverse folding of pseudoknot RNA using ant-colony optimization. *BMC bioinformatics*, 16(1):389, 2015.

R. V. Koodli, B. Rudolfs, H. K. Wayment-Steele, and R. Das. Redesigning the eterna100 for the vienna 2 folding engine. *bioRxiv*, 2021.

M. Li, M. Zheng, S. Wu, C. Tian, D. Liu, Y. Weizmann, W. Jiang, G. Wang, and C. Mao. In vivo production of rna nanostructures via programmed folding of single-stranded rnas. *Nature communications*, 9(1):1–9, 2018.

R. Lorenz, S. H. Bernhart, C. Höner zu Siederdissen, H. Tafer, C. Flamm, P. F. Stadler, and I. L. Hofacker. Viennarna package 2.0. *Algorithms for Molecular Biology*, 6(1):26, Nov 2011. ISSN 1748-7188. doi: 10.1186/1748-7188-6-26. URL `https://doi.org/10.1186/1748-7188-6-26`.

M. Mann, P. R. Wright, and R. Backofen. Intarna 2.0: enhanced and customizable prediction of rna–rna interactions. *Nucleic acids research*, 45(W1):W435–W439, 2017.

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

G. Minuesa, C. Alsina, J. A. Garcia-Martin, J. C. Oliveros, and I. Dotu. Moirnaifold: a novel tool for complex in silico rna design. *Nucleic acids research*, 49(9):4934–4943, 2021.

A. S. Mironov, I. Gusarov, R. Rafikov, L. E. Lopez, K. Shatalin, R. A. Kreneva, D. A. Perumov, and E. Nudler. Sensing small molecules by nascent rna: a mechanism to control transcription in bacteria. *cell*, 111(5):747–756, 2002.

S. Mueller, J. R. Coleman, D. Papamichail, C. B. Ward, A. Nimnual, B. Futcher, S. Skiena, and E. Wimmer. Live attenuated influenza virus vaccines by computer-aided rational design. *Nature biotechnology*, 28(7):723–726, 2010.

E. P. Nawrocki and S. R. Eddy. Infernal 1.1: 100-fold faster rna homology searches. *Bioinformatics*, 29(22):2933–2935, 2013.

Y. Nozawa, M. Hagihara, M. S. Rahman, S. Matsumura, and Y. Ikawa. Rational design of an orthogonal pair of bimolecular rnase p ribozymes through heterologous assembly of their modular domains. *Biology*, 8(3):65, 2019.

J. Parker-Holder, R. Rajan, X. Song, A. Biedenkapp, Y. Miao, T. Eimer, B. Zhang, V. Nguyen, R. Calandra, A. Faust, F. Hutter, and M. Lindauer. Automated reinforcement learning (autorl): A survey and open problems. *Journal of Artificial Intelligence Research (JAIR)*, 74:517–568, 2022.

S. Pilsl, C. Morgan, M. Choukeife, A. Möglich, and G. Mayer. Optoribogenetic control of regulatory rna molecules. *Nature Communications*, 11(1):1–10, 2020.

J. R. Prensner, M. K. Iyer, O. A. Balbin, S. M. Dhanasekaran, Q. Cao, J. C. Brenner, B. Laxman, I. A. Asangani, C. S. Grasso, H. D. Kominsky, X. Cao, X. Jing, X. Wang, J. Siddiqui, J. T. Wei, D. Robinson, H. K. Iyer, N. Palanisamy, C. A. Maher, and A. M. Chinnaiyan. Transcriptome sequencing across a prostate cancer cohort identifies pcat-1, an unannotated lincrna implicated in disease progression. *Nature Biotechnology*, 29(8):742–749, 2011.

M. D. Retwitzer, V. Reinharz, A. Churkin, Y. Ponty, J. Waldispühl, and D. Barash. incarnafbinv 2.0: a webserver and software with motif control for fragment-based design of rnas. *Bioinformatics*, 36 (9):2920–2922, 2020.

F. Runge, D. Stoll, S. Falkner, and F. Hutter. Learning to design RNA. In *International Conference on Learning Representations*, 2019.

F. Runge, J. K. H. Franke, D. Fertmann, and F. Hutter. Rethinking performance measures of rna secondary structure problems, 2023.

B. A. Shapiro. An algorithm for comparing multiple rna secondary structures. *Bioinformatics*, 4(3): 387–393, 1988.

J. Shine and L. Dalgarno. Determinant of cistron specificity in bacterial ribosomes. *Nature*, 254 (5495):34–38, 1975.

J. Singh, J. Hanson, K. Paliwal, and Y. Zhou. Rna secondary structure prediction using an ensemble of two-dimensional deep neural networks and transfer learning. *Nature communications*, 10(1): 1–13, 2019.

M. F. Sloma and D. H. Mathews. Exact calculation of loop formation probability identifies folding motifs in rna secondary structures. *RNA*, 22(12):1808–1818, 2016.

Y. Tian, J. Xu, X. Du, and X. Fu. The interplay between noncoding rnas and insulin in diabetes. *Cancer Letters*, 419:53 – 63, 2018. ISSN 0304-3835.

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

M. Wachsmuth, S. Findeiß, N. Weissheimer, P. F. Stadler, and M. Mörl. De novo design of a synthetic riboswitch that regulates transcription termination . *Nucleic Acids Research*, 41(4):2541–2551, 12 2012. ISSN 0305-1048.

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

X. Zhou and J. Xu. Identification of alzheimer's disease–associated long noncoding rnas. *Neurobiology of Aging*, 36(11):2925 – 2931, 2015. ISSN 0197-4580.

M. Zuker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, 9(1):133–148, 1981.