LEARNING TO DESIGN RNA

Anonymous authors
Paper under double-blind review

ABSTRACT

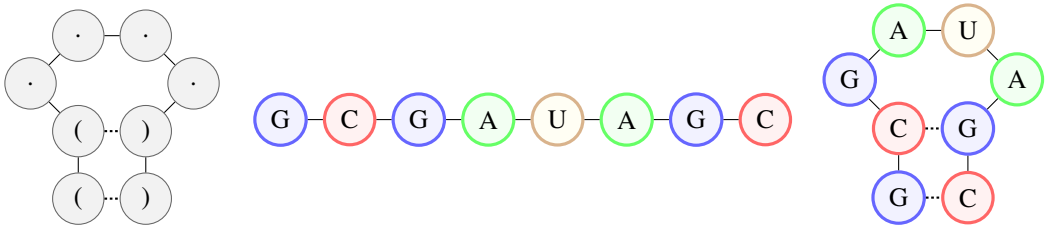
Designing RNA molecules has garnered recent interest in medicine, synthetic biology, biotechnology and bioinformatics since many functional RNA molecules were shown to be involved in regulatory processes for transcription, epigenetics and translation. Since an RNA's function depends on its structural properties, the *RNA Design* problem is to find an RNA molecule that folds into a specified secondary structure. Here, we propose a new algorithm for the RNA Design problem, dubbed *LEARNA*. *LEARNA* uses deep reinforcement learning to train a policy network to sequentially design an entire RNA sequence given a specified secondary target structure. By meta-learning across thousands of different RNA target structures, our extension *Meta-LEARNA* constructs an RNA design policy that can be applied out of the box to solve novel RNA target structures. Comprehensive empirical results on two widely-used RNA secondary structure design benchmarks, as well as a third one that we introduce, show that our approach achieves new state-of-the-art performance on all benchmarks while also being up to $450\times$ faster than any previous approach. We achieve this by a joint optimization of the policy network's architecture, the training hyperparameters, and the state space representation. In an ablation study, we analyze our method's different components.

1 INTRODUCTION

RNA is one of the major classes of information-carrying biopolymers in the cells of living organisms. Recent studies revealed a key role of functional non-protein-coding RNAs (ncRNAs) in regulatory processes and transcription control, which have also been connected to certain diseases like Parkinson's disease and Alzheimer's disease (ENCODE Project Consortium and others, 2004; Gstir et al., 2014; Kaushik et al., 2018). Functional ncRNAs are involved in the modulation of epigenetic marks, altering of messenger RNA (mRNA) stability, mRNA translation, alternative splicing, signal transduction and scaffolding of large macromolecular complexes (Vandivier et al., 2016). Therefore, engineering of ncRNA molecules is of growing importance with applications ranging from biotechnology and medicine to synthetic biology (Delebecque et al., 2011; 2012; Guo et al., 2010; Terns & Terns, 2014; Meyer et al., 2015). In fact, successful attempts to create functional RNA sequences *in vitro* and *in vivo* have been reported (Dotu et al., 2014; Wachsmuth et al., 2013).

At its most basic form, RNA is a sequence of the four nucleotides *Adenine (A)*, *Guanine (G)*, *Cytosine (C)* and *Uracile (U)*. This nucleotide sequence is called the RNA sequence, or primary structure. While the RNA sequence serves as the blueprint, the functional structure of the RNA molecule is determined by the folding translating the RNA sequence into its 3D tertiary structure. The intrinsic thermodynamic properties of the sequence dictate the resulting fold. The hydrogen bonds formed between two corresponding nucleotides constitute one of the driving forces in the thermodynamic model and influence the tertiary structure heavily. The structure that encompasses these hydrogen bonds is commonly referred to as the secondary structure of RNA. Many algorithms for RNA tertiary structure design directly work on RNA secondary structures (Kerpedjiev et al., 2015; Zhao et al., 2012; Reinharz et al., 2012). Therefore, fast and accurate algorithms for RNA secondary structure design could advance the current state of the art in RNA engineering.

The problem of finding an RNA sequence that folds into a desired secondary structure is known as the *RNA Design problem* or *RNA inverse folding* (Hofacker et al., 1994). Most algorithms for RNA Design focus on search strategies that start with an initial nucleotide sequence and modify it to find a solution for the given secondary structure (Hofacker et al., 1994; Andronescu et al., 2004; Taneda,



(a) Target structure ω (b) RNA sequence $\phi \in N^*$ (c) Folded structure $\mathcal{F}(\phi)$

Figure 1: Illustration of the *RNA Design problem* using folding algorithm $\mathcal{F}$. Given the desired RNA secondary structure and its dot-bracket notation (a), the task is to design an RNA sequence (b) that folds into the desired secondary structure (c).

2011; Esmaili-Taheri et al., 2014; Eastman et al., 2018). In contrast, in this paper we describe a novel deep reinforcement learning approach to this problem. Our contributions are as follows:

- We describe *LEARNA*, a deep reinforcement learning (RL) algorithm for RNA design. LEARNA trains a policy network that, given a target secondary structure, can be rolled out to sequentially predict the entire RNA sequence. After generating an RNA sequence, our approach folds this sequence, locally adapts it, and uses the distance of the resulting structure to the target structure as an error signal for the RL agent.
- We describe *Meta-LEARNA*, a version of LEARNA that learns a single policy across many RNA design problems directly applicable to new RNA design problems. Specifically, it learns a conditional generative model from which we can sample candidate RNA sequences for a given RNA target structure and solves many problems with the first sample.
- We introduce a new benchmark data set, which we used to develop and tune our algorithm.
- We jointly optimize the architecture of the policy network together with training hyperparameters, and the state representation on our development benchmark set.
- A comprehensive empirical analysis shows that our approach achieves new state-of-the-art performance on the two most commonly used RNA design benchmark datasets: Rfam (following Yang et al. (2017)) and Eterna100 (following Anderson-Lee et al. (2016)), as well as on the test split of our new benchmark. Furthermore, Meta-LEARNA achieves the same results as the other algorithms up to $26\times$, $450\times$ and $4\times$ faster than any previous approach on the respective datasets.

2 Background and Related Work

In this section, we give a brief overview of the RNA Design problem and current approaches, followed by a short summary of RL approaches for similar problems.

2.1 RNA Design

The RNA Design problem is the inverse of the RNA folding problem. Formally, an RNA folding algorithm $\mathcal{F}$ maps from an RNA sequence $\phi \in N^* = \{A, G, C, U\}^*$ to a representation of the RNA's secondary structure. The RNA design problem aims to find an inverse mapping of this folding process:

Definition 1 (RNA Design)**.** *Given a folding algorithm, $\mathcal{F}$ and a target RNA secondary structure ω, the* RNA design problem *is to find a RNA sequence $\phi \in N^* = \{A, G, C, U\}^*$ that satisfies $\omega = \mathcal{F}(\phi)$.*

In this paper, we employ the most common folding algorithm: the *Zuker algorithm* (Zuker & Stiegler, 1981; Zuker & Sankoff, 1984), which uses a thermodynamic model to minimize the free energy to find the most stable conformation of the RNA structure. We note, however, that our approach is not limited to it and would directly apply for any other RNA folding algorithm.

Figure 1 illustrates this problem: In the dot bracket notation of the target structure (a), nucleotides are replaced by a dot for unbound sites, and opening and closing brackets for nucleotides connected by a hydrogen bond. The primary structure (b) folds into the desired target according to $\mathcal{F}$ (c).

Most algorithms for RNA design employ a structural loss function $L(\omega, \mathcal{F}(\phi))$ to quantify the difference between the target structure ω and the structure resulting from folding an RNA sequence ϕ. The minimizer of this loss corresponds to a solution to the RNA design problem:

$$\phi^* \in \arg\min_{\phi \in N^*} L(\omega, \mathcal{F}(\phi)). \tag{1}$$

A common loss function, which we also employ in this work, is the Hamming distance between two structures.

We note that multiple RNA sequences may fold to the same secondary structure, such that the RNA design problem does not generally have a unique solution; one could distinguish between solution by preferring more stable folds, targeting a specific GC content, or satisfying other constraints; all of these could be incorporated into the loss function being optimized.

Most algorithms targeting the RNA Design problem are either local or global algorithms. Local approaches commonly operate on a single sequence and try to find a solution by changing a small number of nucleotides at a time with the loss function guiding the search. Global methods, on the other hand, either have a large number of candidates being manipulated, or model a global distribution from which samples are sampled. They differ mostly by the specif loss used, which can be tailored for specific domains.

Local Methods Many early approaches follow the seminal work by Hofacker et al. (1994), which introduced *RNAInverse*, the first algorithm for the RNA design problem. Among others RNA-SSD (Andronescu et al., 2004), INFO-RNA (Busch & Backofen, 2006), NUPACK (Dirks & Pierce, 2004; Zadeh et al., 2010), *ERD* (Esmaili-Taheri et al., 2014), and the approach by Eastman et al. (2018) follow the same principle on manipulating a single sequence.

Global Methods The algorithms in this category employ different methods to globally search the space of primary structures. The genetic approach *MODENA* (Taneda, 2011) operates on an ensemble of sequences by applying local mutations and global cross-over events. *AntaRNA* (Kleinkauf et al., 2015) uses ant-colony optimization to find the most promising RNA sequence, while *MCTS-RNA* (Yang et al., 2017) employs Monte-Carlo Tree Search to model a global distribution from which it draws samples.

We refer the interested reader to Churkin et al. (2017) and the references within for a more detailed review.

2.2 Reinforcement Learning

In recent years Reinforcement Learning, which considers the setting of an agent interacting with an environment, achieved state-of-the-art results in a variety of different tasks (Mnih et al., 2013; Silver et al., 2016). In the field of Neural Architecture Search, Reinforcement Learning recently has been successfully used to design architectures for Neural Networks by sequentially sampling from sets of building blocks (Zoph & Le, 2017; Zoph et al., 2018; Pham et al., 2018).

The work by Bello et al. (2016) heavily influenced our work. In it, the authors apply RL to combinatorial problems, namely the Traveling Salesman Problem. The agent proposes complete solutions rather than manipulating an existing on, and is trained using an episodic reward, in this case the tour length. Inspired by this work, we propose to frame the *RNA-Design problem* as a Reinforcement Learning problem where each candidate solution is designed from scratch. In our approach, the agent predicts which nucleotide to place next into the sequence, learning to design RNA in end-to-end.

This is stark contrast to the recent work by Eastman et al. (2018) carried out in parallel to and independently from ours. The authors used reinforcement learning (RL) to perform a local search starting from a randomly initialized sequence. The RL agent applies local modifications to design a solution that folds into the desired target structure. The current sequence constitutes the state and each action represents changing an unpaired nucleotide or a pair of nucleotides. After each action the current sequence is evaluated utilizing Zuker's algorithm. The agent only receives a nonzero

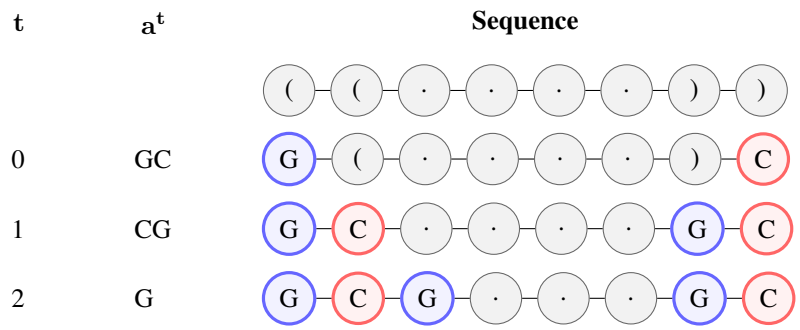


Figure 2: Illustration of the design of a candidate solution. Starting at time point 0 the candidate solution is sequentially built by placing nucleotides from the action sequence $(a^t)_{t \in \{0, \ldots, T-1\}}$.

reward signal once it finds a correct sequence. The agent's policy is a convolutional neural network pre-trained on fixed length, randomly generated sequences. In the remainder of the paper, we refer to this approach as RL-LS, as the RL agent performs a local search.

3 Learning to Design RNA

In this section we describe our novel generative approach for the RNA Design problem based on reinforcement learning. We first formulate RNA Design as a decision process and then propose several strategies to yield agents that learn to design RNA in an end-to-end fashion.

3.1 A Decision Process Modelling RNA Design

We propose to model the RNA Design problem with respect to a given target structure ω as the decision process $D_\omega := (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$. In our experiments, we used the popular *Zuker algorithm* (Zuker & Stiegler, 1981; Zuker & Sankoff, 1984) for folding, but our approach would directly be applicable to any other folding algorithm. In this section we denote the mapping, given by the *Zuker algorithm*, from the space of candidate solutions $N^* = \{A, G, C, U\}^*$ to the set of dot bracket encoded RNA secondary structures Ω, with $\mathrm{fold}(\cdot)$.

Action space In each episode an agent has the task to design an RNA sequence that folds into a given $\omega \in \Omega$. To design a candidate solution $\phi \in N^*$, at each time step t the agent places nucleotides by choosing an action a^t from the action space $\mathcal{A}$. For unpaired sites a^t corresponds to one of the four RNA nucleotides (G, C, A or U); for paired sites a^t corresponds to one of the Watson-Crick base pairs (GC, CG, AU, or UA). The action space at time step t then is

$$\mathcal{A} := \{0,\, 1,\, 2,\, 3\} \equiv \begin{cases} \{A,\, G,\, C,\, U\} & \text{for } \omega[t] = . \\ \{GC,\, CG,\, AU,\, UA\} & \text{for } \omega[t] \in \{(,\,)\} \end{cases} \quad , \tag{2}$$

where $\omega[t]$ is the t-th character of the target structure ω. See Figure 2 for an illustration.

State space The agent chooses an action a^t based on the state s^t provided by the environment. We formulated states to provide local information to the agent. For this we set s^t to the $(2\kappa + 1)$-gram centered around the t-th site of the target structure ω, where κ is a hyperparameter we dub the state radius. To be able to construct this centered n-gram at all sites we introduced κ padding characters at the start and the end of the target structure. Formally, the state space can then be written as

$$\mathcal{S} := \{0,\, 1,\, 2,\, 3\}^{2\kappa+1} \equiv (\mathcal{B} \cup \{\text{padding}\})^{2\kappa+1} \quad , \tag{3}$$

where $\mathcal{B}$ is the set of symbols in the dot-bracket notation: a dot, an opening and a closing bracket.

Transition Function Since at each time step t the state s^t is set to a fixed $(2\kappa + 1)$-gram, the transition function $\mathcal{P}$ is deterministic and defined accordingly.

Reward At the terminal time step T the agent has assigned nucleotides to all sites and the environment generates the (only non-zero) reward r^T:

$$r^T = -\left(\frac{\mathrm{d_H}(\mathrm{fold}(\phi),\, \omega)}{|\omega|}\right)^{\alpha} \quad , \tag{4}$$

where $\mathrm{d_H}(\cdot,\cdot)$ is the *Hamming distance* and $\alpha > 1$ is a hyperparameter to shape the reward. If the distance between the fold of the proposed sequence and the target structure becomes small enough, i.e., $\mathrm{d_H}(\mathrm{fold}(\phi),\, \omega) < \xi$ (where ξ is a hyperparameter), we employ a local improvement step before computing the reward. The local strategy tries all nucleotide combinations for the sites $\mathrm{fold}(\phi)$ and ω differ. Since the described decision process has only one non-zero reward per episode, we set the discount γ to 1.

3.2 Obtaining Policies for RNA Design

We use deep reinforcement learning to optimize policy networks π^θ with parameters θ, that define a posterior distribution on the action space $\mathcal{A}$ conditional on the state space $\mathcal{S}$. Our policy networks consist of an embedding layer for the input state and a deep neural network; this neural network optionally contains convolutional, recurrent and fully-connected layers, and its precise architecture is jointly optimized together with the hyperparameters as described in Section 4. We propose several strategies to learn the parameters θ of a given policy network as detailed below.

LEARNA The LEARNA strategy learns to design a sequence for the target structure ω in an online fashion, from scratch. The parameters θ are randomly initialized before the agent episodically interacts with the decision process $\mathcal{D}_\omega$. For updating the parameters we use the policy gradient method PPO (Schulman et al., 2017), which was recently successfully applied to several other applications of machine learning (Heess et al., 2017; Bansal et al., 2018; Zoph et al., 2018).

Meta-LEARNA Meta-LEARNA uses a meta-learning approach (Lemke et al., 2015) that views the RNA design of the target structures in the training set Ω_{train} as tasks and learns to transfer knowledge across them. Each of the target structures $\omega_i \in \Omega_{train}$ defines a different decision process $\mathcal{D}_{\omega_i}$, and we train a single policy network on all of them also using PPO. Once the training is finished, the parameters θ are fixed and π^θ can be applied to the decision process $\mathcal{D}_\omega$ by sampling from the learned generative model.

Meta-LEARNA-Adapt Meta-LEARNA-Adapt combines the previous two strategies: First, we run Meta-LEARNA to train parameters θ in an offline training phase on Ω_{train}. However, to work on target structure ω, the policy is not fixed but is only used to initialize LEARNA running on the decision process $\mathcal{D}_\omega$.

4 Joint Architecture Search and Hyperparameter Optimization

It is not obvious a priori which architecture would work best for our policy network. To decide this automatically, we perform a search over various possible structures, including elements of convolutional neural networks (CNNs) and recurrent neural networks (RNNs). We construct our architecture as follows: The input state is first processed by an optional embedding layer that converts the symbol-based representation into a numerical one for each site. Then, an optional CNN with at most two layers can be applied to the state, followed by an optional LSTM with at most two layers. As the final stage, we always add a shallow fully-connected network with one or two layers, which outputs the distribution over actions. This parameterization covers a broad range of possible architectures while keeping the dimensionality of the search space relatively modest.

Optimizing the architecture alone does not suffice to find very well-performing agents, since the performance of neural networks is also very sensitive to their training hyperparameters governing optimization and regularization. As we employ PPO for training the policy, we optimize some of its hyperparameters, namely the learning rate, batch size, and strength of the entropy regularization. Two additional hyperparameter are the exponent α used for reward shaping and the dimensionality

of the embedding. Finally, we also optimize the state space radius κ (see Section 3.1), since our ultimate goal is not to solve a specific MDP, but to use the best MDP for solving our problem.

Overall, these design choices yield a 14-dimensional search space comprising mostly integer variables. The complete list of parameters, their types, ranges, and the priors we used over them can be found in Appendix A. We used almost identical search spaces for LEARNA and Meta-LEARNA, but adapted the ranges for the learning rate and the entropy regularization slightly based on preliminary experiments.

We used the recently-proposed optimizer BOHB (Falkner et al., 2018) to find good configurations, since it supports optimization in mixed integer/continuous search spaces and can utilize parallel resources. Additionally, BOHB allows to exploit cheap approximations of expensive-to-evaluate objective functions to speed up the optimization; these so-called low-fidelity approximations can be achieved in numerous ways, e.g., limiting the training time, data, or number independent repetitions of the evaluations (Falkner et al., 2018). In our setting, we decided to limit the wall-clock time for training (Meta-LEARNA) or the evaluations (LEARNA); this means that instantiations of our 14 design dimensions are first evaluated for a rather short time, and only the most promising candidates advance to higher budgets (here: computational time).

To properly optimize the listed design choices without overfitting on our target RNA benchmarks, we needed a designated training and validation dataset of RNA target structures. The training set constitutes the corpus of sequences Ω_{train} on which we train Meta-LEARNA, and the validation set is used to estimate the performance of a configuration on unseen RNA target structures. Note that LEARNA requires no training, so we evaluated configurations for it directly on the validation set. To ensure a large enough and interesting dataset, we downloaded all families of the Rfam database version 13.0 (Kalvari et al., 2017) and removed all secondary structure duplicates. We filtered the remaining dataset for the 5% and 95% quantiles regarding the structure length to remove outliers. We only kept the sequences that a single run of *MCTS-RNA* for 30 seconds could not solve. (We chose *MCTS-RNA* for filtering as it was the fastest and most accurate algorithm at the time.) We split the remaining secondary structures into a training set of 65 000, a validation set of 100, and a test set of 100 secondary structures. We refer to these datasets as Rfam-Learn-Train, Rfam-Learn-Validation and Rfam-Learn-Test, respectively, for the remainder of the paper.

Despite the fact that RL is known to often yield noisy or unreliable outcomes in single runs (Henderson et al., 2017), we actively decided to only use a single training run and a single validation set for each configuration to keep the optimization manageable. To counteract the problems associated with single noisy observations, we studied three different loss functions:

- *The Number of unsolved sequences*. This is the quantity we ultimately try to optimize, but it is rather noisy.
- *The sum of minimal distances*. This is also minimal when all sequences are solved, but it prefers almost solving a structure over complete failure and does not punish exploration.
- *The sum of mean distances*. By using the mean, rather than the minimum distance of a run, this choice encourages fast convergence of the agent while penalizing exploration.

In preliminary experiments, we found that the sum of minimum distances was the best choice: the optimizer found good configurations consistently, and repeated evaluations yielded consistent losses. This is in contrast to optimizing the number of unsolved sequences directly, which found less stable configurations (we attribute this due to the intrinsic noise of this quantity; with more evaluations on the validation set, this could likely be overcome at the expense of additional computational costs). Finally, the sum of mean rewards was not a good proxy loss, yielding only a weak correlation to the number of solved sequences; we attribute this to the fact that exploration is helpful and should not be discouraged.

5 EXPERIMENTS

We evaluate our approaches against state-of-the-art methods and perform an ablation study to assess the importance of our method's components. We report results on two established benchmarks from the literature and on our own benchmark. Following standard evaluation protocols for these benchmarks, we performed multiple runs with a fixed time limit on each target structure, reporting

Table 1: Summary of the final results for *RNAInverse*, *MCTS-RNA*, *RL-LS*, *antaRNA*, *LEARNA*, and *Meta-LEARNA* regarding the total number of solved target structures on the three benchmarks in percent. A target structure counts as solved if any run on the specific dataset solved it.

METHOD	ETERNA100	RFAM-TANEDA	RFAM-LEARN-TEST
MCTS-RNA	57	79	97
ANTARNA	58	66	**100**
RL-LS	59	62	62
RNAINVERSE	60	59	95
LEARNA-10MIN	-	79	95
LEARNA-30MIN	63	-	97
META-LEARNA-ADAPT	64	**83**	98
META-LEARNA	**65**	79	**100**

the accumulated number of solved targets across all runs. Additionally, we provide means and confidence bounds for all experiments. The three benchmarks we used are as follows:

- The Rfam benchmark as described at Taneda (2011) encompasses 29 target structures from living organisms as listed in the Rfam database version 9.0 (Gardner et al., 2009) Evaluation is based on 50 runs with a timeout of 10 minutes. We denote the dataset described at Taneda (2011) with Rfam-Taneda.
- The Eterna100 benchmark (Anderson-Lee et al., 2016) is comprised of 100 secondary structures that span a wide range of lengths and difficulties. Measurements comprise of 5 runs with a timeout of 24 hours.
- The Rfam-Learn benchmark we propose here consists of 100 target structures with a minimum length of 50- and a maximum length of 450 nucleotides. Performance is measured over 5 runs with a time limit of 1 hour.

We used the implementation of the *Zuker algorithm* provided by *ViennaRNA* (Lorenz et al., 2011) versions 2.4.8 (*MCTS-RNA*, *RL-LS* and *LEARNA*), 2.1.9 (*AntaRNA*) and 2.4.9 (*RNAInverse*). Our implementation uses the Reinforcement Learning library *tensorforce*, version 0.3.3 (Schaarschmidt et al., 2017) working with *TensorFlow* version 1.4.0 (Abadi et al., 2015). All computations were done on Broadwell E5-2630v4 2.2 GHz CPUs with 5 GByte RAM. For the training phase of Meta-LEARNA, we used all 20 cores of these machines, but at evaluation time, all methods were only allowed a single core (using core binding).

5.1 COMPARATIVE STUDY

On the Eterna100 benchmark all variants of *LEARNA* achieve new state-of-the-art results. While *LEARNA* succeeds on 63% of the target structures, *Meta-LEARNA* solves upto 65%. Additionally, *Meta-LEARNA* only needs about 90 seconds to be on par with the final performance of any other method and achieves new state-of-the-art results in less than 3 minutes. This performance is stable through all of the 5 runs performed. Furthermore, *Meta-LEARNA* also achieves new state-of-the-art performance in each single run (solving at least 64% of the target structures). Our two pre-trained approaches, *Meta-LEARNA* and *Meta-LEARNA-Adapt*, outperform all other methods in terms of speed and the number of solved target structures. However, it seems counter-intuitive that further adaptation decreases performance of *Meta-LEARNA-Adapt* compared to *Meta-LEARNA*. We suspect that the loss in performance of *Meta-LEARNA-Adapt* is caused by the overhead associated with the weight updates which results in fewer evaluations in the same time. For more details, we refer to Appendix E. The results on the Eterna100 benchmark are visualized in Figure 3.

Concerning the Rfam-Taneda benchmark, *Meta-LEARNA* and *LEARNA* are on par with the current state-of-the-art results of *MCTS-RNA*, see Figure 3. Remarkably, *Meta-LEARNA* needs less than 10 seconds to achieve this performance. *Meta-LEARNA-adapt* achieves new state-of-the-art results after 1 minute (see Appendix E) solving 83% of the target structures.

The results on our new Rfam-Learn benchmark are shown in Figure 8 in Appendix B. Only *Meta-LEARNA* and *antaRNA* were able to solve all of the target structures in 1 hour; 5 minutes and 20

minutes respectively. Except for *RL-LS*, all algorithms could solve at least 95% of the target structures within the one hour timelimit.

In general, our approach achieved the best performance on all of the three benchmarks while being much faster than all other algorithms we compared against. The final performance of all algorithm of our comparison for the three benchmarks is summarized in Table 1.

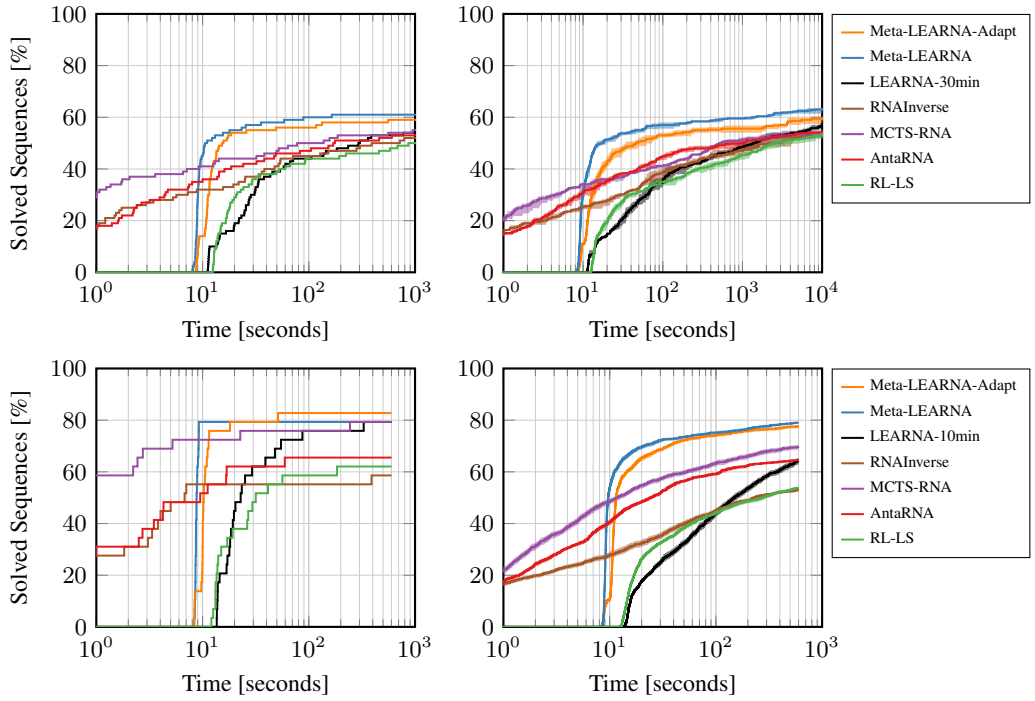


Figure 3: Comparison of all methods on the Eterna 100 (top) and the RFAM-Taneda benchmark (bottom). On the left we show the minumum over 5 runs, while the right panels show the mean and the confidence interval.

5.2 Ablation Study

To study the influence of the different components on the performance of our approach we performed an ablation study. Therefore we either excluded the adaptation option, the local improvement step or the restart option. For all experiments we observed that the local improvement step boosts performance for all variants of our approach. The restart option stabilizes the single runs (see Figure 12 in the Appendix C) because it is effectively averaging over more shorter runs. We also observed a performance boost directly after the restart. Figure 9 exemplarily shows the results of the ablation study for *Meta-LEARNA-adapt*. All results of our ablation study are illustrated in the Appendix C.

6 Conclusion

We proposed the deep reinforcement learning algorithm *LEARNA* for the *RNA-Design problem* to sequentially construct candidate solutions in an end-to-end fashion. By pre-training on a large corpus of biological sequences, a local improvement step to aid the agent, and extensive architecture and hyperparameter optimization, we arrived at *Meta-LEARNA*, a ready-to-use agent that achieves state-of-the-art results on the Eterna100 benchmark (Anderson-Lee et al., 2016), and our own new dataset. By continuing training, dubbed *Meta-LEARNA-Adapt*, we can also improve over all previous results on the Rfam benchmark proposed by Taneda (2011). Our ablation study shows the importance of all components, suggesting that RL with an additional local improvement step can solve the RNA Design problem efficiently. Code and data to reproduce our results will be made publicly available after acceptance of this paper.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Jeff Anderson-Lee, Eli Fisker, Vineet Kosaraju, Michelle Wu, Justin Kong, Jeehyung Lee, Minjae Lee, Mathew Zada, Adrien Treuille, and Rhiju Das. Principles for predicting RNA secondary structure design difficulty. *Journal of molecular biology*, 428(5):748–757, 2016.

Mirela Andronescu, Anthony P. Fejes, Frank Hutter, Holger H. Hoos, and Anne Condon. A New Algorithm for RNA Secondary Structure Design. *Journal of Molecular Biology*, 336(3): 607–624, 2004. ISSN 0022-2836. doi: https://doi.org/10.1016/j.jmb.2003.12.041. URL `http://www.sciencedirect.com/science/article/pii/S0022283603015596`.

Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. In *Proceedings of the International Conference on Learning Representations (ICLR'18)*, 2018. Published online: `iclr.cc`.

Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

Anke Busch and Rolf Backofen. INFO-RNA—a fast approach to inverse RNA folding. *Bioinformatics*, 22(15):1823–1831, 2006. doi: 10.1093/bioinformatics/btl194. URL `http://dx.doi.org/10.1093/bioinformatics/btl194`.

Alexander Churkin, Matan Drory Retwitzer, Vladimir Reinharz, Yann Ponty, Jérôme Waldispühl, and Danny Barash. Design of rnas: comparing programs for inverse rna folding. *Briefings in bioinformatics*, 19(2):350–358, 2017.

Camille J. Delebecque, Ariel B. Lindner, Pamela A. Silver, and Faisal A. Aldaye. Organization of Intracellular Reactions with Rationally Designed RNA Assemblies. *Science*, 333(6041):470–474, 2011. ISSN 0036-8075. doi: 10.1126/science.1206938. URL `http://science.sciencemag.org/content/333/6041/470`.

CJ Delebecque, PA Silver, and AB Lindner. Designing and using RNA scaffolds to assemble proteins in vivo. *Nature Protocols*, 7(10):1797–807, 2012. doi: 10.1038/nprot.2012.102.

Robert M. Dirks and Niles A. Pierce. An Algorithm for Computing Nucleic Acid Base-Pairing Probabilities Including Pseudoknots. *Journal of Computational Chemistry*, 25(10):295—-1304, 2004. doi: 10.1002/jcc.20057.

Ivan Dotu, Juan Antonio Garcia-Martin, Betty L. Slinger, Vinodh Mechery, Michelle M. Meyer, and Peter 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`.

Peter Eastman, Jade Shi, Bharath Ramsundar, and Vijay S Pande. Solving the RNA design problem with reinforcement learning. *PLoS computational biology*, 14(6):e1006176, 2018.

ENCODE Project Consortium and others. The ENCODE (encyclopedia of DNA elements) project. *Science*, 306(5696):636–640, 2004.

Ali Esmaili-Taheri, Mohammad Ganjtabesh, and Morteza Mohammad-Noori. Evolutionary solution for the RNA design problem. *Bioinformatics*, 30(9):1250–1258, 2014. doi: 10.1093/bioinformatics/btu001. URL `http://dx.doi.org/10.1093/bioinformatics/btu001`.

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, pp. 1436–1445, July 2018.

Paul P. Gardner, Jennifer Daub, John G. Tate, Eric P. Nawrocki, Diana L. Kolbe, Stinus Lindgreen, Adam C. Wilkinson, Robert D. Finn, Sam Griffiths-Jones, Sean R. Eddy, and Alex Bateman. Rfam: updates to the RNA families database. *Nucleic Acids Research*, 37(suppl-1):D136–D140, 2009. doi: 10.1093/nar/gkn766. URL http://dx.doi.org/10.1093/nar/gkn766.

Ronald Gstir, Simon Schafferer, Marcel Scheideler, Matthias Misslinger, Matthias Griehl, Nina Daschil, Christian Humpel, Gerald J. Obermair, Claudia Schmuckermair, Joerg Striessnig, Bernhard E. Flucher, and Alexander Hüttenhofer. Generation of a neuro-specific microarray reveals novel differentially expressed noncoding RNAs in mouse models for neurodegenerative diseases. *RNA*, 20(12):1929–1943, 2014.

Peixuan Guo, Oana Coban, Nicholas M. Snead, Joe Trebley, Steve Hoeprich, Songchuan Guo, and Yi Shu. Engineering RNA for Targeted siRNA Delivery and Medical Application. *Advanced Drug Delivery Reviews*, 62(6):650 – 666, 2010. ISSN 0169-409X. doi: https://doi.org/10.1016/j.addr.2010.03.008. URL http://www.sciencedirect.com/science/article/pii/S0169409X10000773. From Biology to Materials: Engineering DNA and RNA for Drug Delivery and Nanomedicine.

Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin A. Riedmiller, and David Silver. Emergence of locomotion behaviours in rich environments. *CoRR*, abs/1707.02286, 2017. URL http://arxiv.org/abs/1707.02286.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560*, 2017.

Ivo Hofacker, Walter Fontana, Peter Stadler, Sebastian Bonhoeffer, Manfred Tacker, and Peter Schuster. Fast Folding and Comparison of RNA Secondary Structures. *Monatshefte fuer Chemie/Chemical Monthly*, 125:167–188, 02 1994.

Ioanna Kalvari, Joanna Argasinska, Natalia Quinones-Olvera, Eric P Nawrocki, Elena Rivas, Sean R Eddy, Alex Bateman, Robert D Finn, and Anton I Petrov. Rfam 13.0: shifting to a genome-centric resource for non-coding RNA families. *Nucleic acids research*, 46(D1):D335–D342, 2017.

Kriti Kaushik, Ambily Sivadas, Shamsudheen Karuthedath Vellarikkal, Ankit Verma, Rijith Jayarajan, Satyaprakash Pandey, Tavprithesh Sethi, Souvik Maiti, Vinod Scaria, and Sridhar Sivasubbu. RNA secondary structure profiling in zebrafish reveals unique regulatory features. *BMC Genomics*, 20:147, 2018.

P Kerpedjiev, C Höner zu Siederdissen, and IL Hofacker. Predicting RNA 3D structure using a coarse-grain helix-centered model. *RNA*, 21(6):1110–1121, 2015. doi: 10.1261/rna.047522.114.

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

Christiane Lemke, Marcin Budka, and Bogdan Gabrys. Metalearning: a survey of trends and technologies. 44(1):117–130, Jun 2015.

Ronny Lorenz, Stephan H Bernhart, Christian Hoener Zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F Stadler, and Ivo L Hofacker. Viennarna package 2.0. *Algorithms for Molecular Biology*, 6(1):26, 2011.

Sarai Meyer, James Chappell, Sitara Sankar, Rebecca Chew, and Julius B. Lucks. Improving fold activation of small transcription activating RNAs (STARs) with rational RNA engineering strategies. *Biotechnology and Bioengineering*, 113(1):216–225, 2015. doi: 10.1002/bit.25693. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/bit.25693.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *CoRR*, abs/1802.03268, 2018. URL http://arxiv.org/abs/1802.03268.

Vladimir Reinharz, François Major, and Jérôme Waldispühl. Towards 3d structure prediction of large rna molecules: an integer programming framework to insert local 3d motifs in rna secondary structure. *Bioinformatics*, 28(12):i207–i214, 2012. doi: 10.1093/bioinformatics/bts226. URL http://dx.doi.org/10.1093/bioinformatics/bts226.

Michael Schaarschmidt, Alexander Kuhnle, and Kai Fricke. Tensorforce: A tensorflow library for applied reinforcement learning. Web page, 2017. URL https://github.com/reinforceio/tensorforce.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Akito Taneda. MODENA: a multi-objective RNA inverse folding. *Advances and applications in bioinformatics and chemistry: AABC*, 4:1, 2011.

Rebecca M. Terns and Michael P. Terns. CRISPR-based technologies: prokaryotic defense weapons repurposed. *Trends in Genetics*, 30(3):111–118, 2014. doi: https://doi.org/10.1016/j.tig.2014.01.003.

Lee E. Vandivier, Stephen J. Anderson, Shawn W. Foley, and Brian D. Gregory. The conservation and function of RNA secondary structure in plants. *Annual review of plant biology*, 67:463–488, 2016.

Manja Wachsmuth, Sven Findeiß, Nadine Weissheimer, Peter F. Stadler, and Mario Mörl. De novo design of a synthetic riboswitch that regulates transcription termination. *Nucleic Acids Research*, 41(4):2541–2551, 2013. doi: 10.1093/nar/gks1330. URL http://dx.doi.org/10.1093/nar/gks1330.

Xiufeng Yang, Kazuki Yoshizoe, Akito Taneda, and Koji Tsuda. RNA inverse folding using Monte Carlo tree search. *BMC bioinformatics*, 18(1):468, 2017.

Joseph N. Zadeh, Conrad D. Steenberg, Justin S. Bois, Brian R. Wolfe, Marshall B. Pierce, Asif R. Khan, Robert M. Dirks, and Niles A. Pierce. Nupack: Analysis and design of nucleic acid systems. *Journal of Computational Chemistry*, 32(1):170–173, 2010. doi: 10.1002/jcc.21596. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.21596.

Y Zhao, Y Huang, Z Gong, Y Wang, J Man, and Y Xiao. Automated and fast building of three-dimensional RNA structures. *Scientific Reports*, 2(734), 2012. doi: 10.1038/srep00734.

B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR'17)*, 2017. Published online: iclr.cc.

B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR'18)*, 2018.

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.

Michael Zuker and David Sankoff. RNA secondary structures and their prediction. *Bulletin of Mathematical Biology*, 46(4):591 – 621, 1984. ISSN 0092-8240. doi: https://doi.org/10.1016/S0092-8240(84)80062-2. URL http://www.sciencedirect.com/science/article/pii/S0092824084800622.

Table 2: Search space for the agent's architecture and the trainings hyperparameter used for training Meta-Learna

Parameter Name	Type	Range	Prior
filter size in 1st conv layer	integer	$[0, 8]$	uniform
filter size in 2nd conv layer	integer	$[0, 4]$	uniform
# filters in 1st conv layer	integer	$[1, 32]$	uniform
# filters in 2nd conv layer	integer	$[1, 32]$	uniform
# fully connected layers	integer	$[1, 2]$	uniform
# units in fully connected layer(s)	integer	$[8, 64]$	log-uniform
# LSTM layers	integer	$[0, 2]$	uniform
# units in every LSTM layer	integer	$[1, 64]$	log-uniform
# state space radius	integer	$[0, 32]$	uniform
embedding dimensionality	integer	$[0, 4]$	uniform
batch size	integer	$[32, 128]$	log-uniform
entropy regularization	float	$[5 \cdot 10^{-5}, 5 \cdot 10^{-3}]$	log-uniform
learning rate for PPO	float	$[1 \cdot 10^{-6}, 1 \cdot 10^{-4}]$	log-uniform
reward exponent	float	$[1, 10]$	uniform

Table 3: Search space for the agent's architecture and the trainings hyperparameter used for training Learna for both the 10 and 30 minutes budget

Parameter Name	Type	Range	Prior
filter size in 1st conv layer	integer	$[0, 8]$	uniform
filter size in 2nd conv layer	integer	$[0, 4]$	uniform
# filter in 1st conv layer	integer	$[1, 32]$	log-uniform
# filter in 2nd conv layer	integer	$[1, 32]$	log-uniform
# fully connected layers	integer	$[1, 2]$	uniform
# units in fully connected layer(s)	integer	$[8, 64]$	log-uniform
# LSTM layers	integer	$[0, 2]$	uniform
# units in every LSTM layer	integer	$[1, 64]$	log-uniform
# state space radius	integer	$[1, 32]$	uniform
embedding dimensionality	integer	$[0, 4]$	uniform
batch size	integer	$[32, 128]$	log-uniform
entropy regularization	float	$[1 \cdot 10^{-5}, 1 \cdot 10^{-2}]$	log-uniform
learning rate for PPO	float	$[1 \cdot 10^{-5}, 1 \cdot 10^{-3}]$	log-uniform
reward exponent	float	$[1, 10]$	uniform

A ARCHITECTURE AND HYPERPARAMETER SEARCH SPACE

Table 4: The selected configurations for each scenario and budget.

Parameter Name	LEARNA-10min	LEARNA-30min	Meta-LEARNA
filter size in 1st conv layer	5	0	5
filter size in 2nd conv layer	3	3	7
# filters in 1st conv layer	8	10	32
# filters in 2nd conv layer	1	3	14
# fully connected layers	1	1	1
# units in fully connected layer(s)	52	32	9
# LSTM layers	2	2	0
# units in every LSTM layer	4	7	53
# state space radius	16	2	26
embedding dimensionality	0	0	1
batch size	32	79	80
entropy regularization	$4.44 \cdot 10^{-4}$	$1.63 \cdot 10^{-4}$	$1.98 \cdot 10^{-4}$
learning rate for PPO	$5.49 \cdot 10^{-4}$	$3.38 \cdot 10^{-4}$	$6.37 \cdot 10^{-5}$
reward exponent	5.72	9.43	9.22

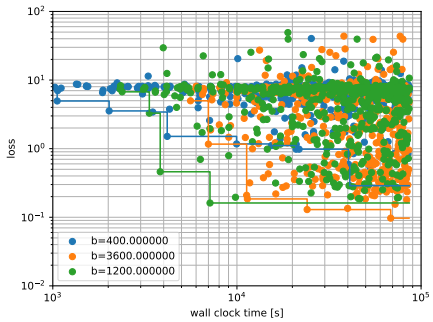


Figure 4: Observed validation loss during the BOHB run for Meta-LEARNA. The different budgets b correnspond to the training time on 20 CPUs in parallel. The results seem to suggest that one can achieve a very similar performance with only 20 minutes of training, which could imply that much longer training of the agent might be required for substantially better performance.

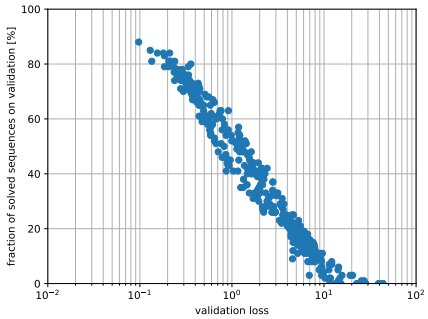


Figure 5: Relationship between the obseverd loss and the number of solved sequences. The plot suggests that our loss metric correlates strongly with the number of successfully found primary sequences.

B SOLVED VS TIME

B.1 ETERNA

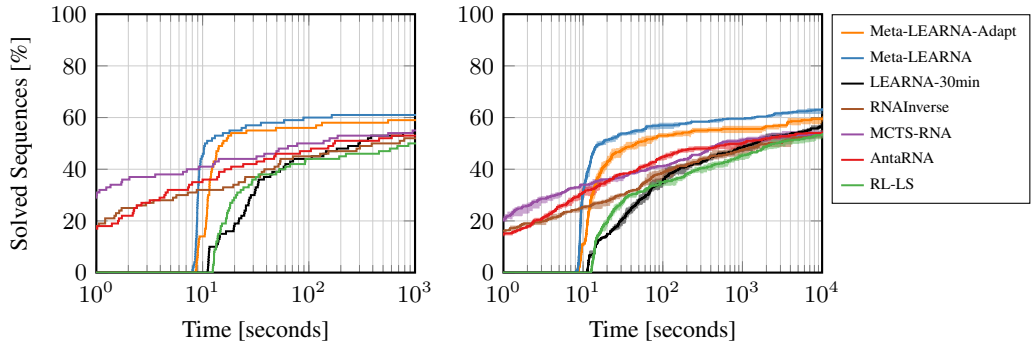


Figure 6: Comparison of all methods on Eterna.

B.2 RFAM TANEDA

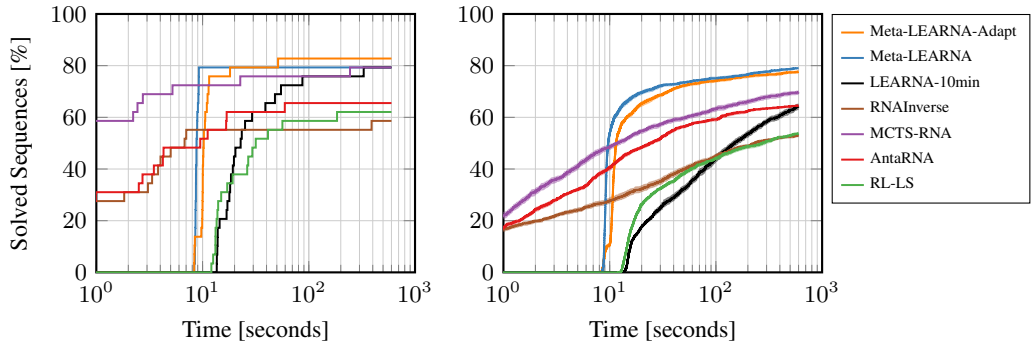


Figure 7: Comparison of all methods on RFAM Taneda.

B.3 RFAM LEARN

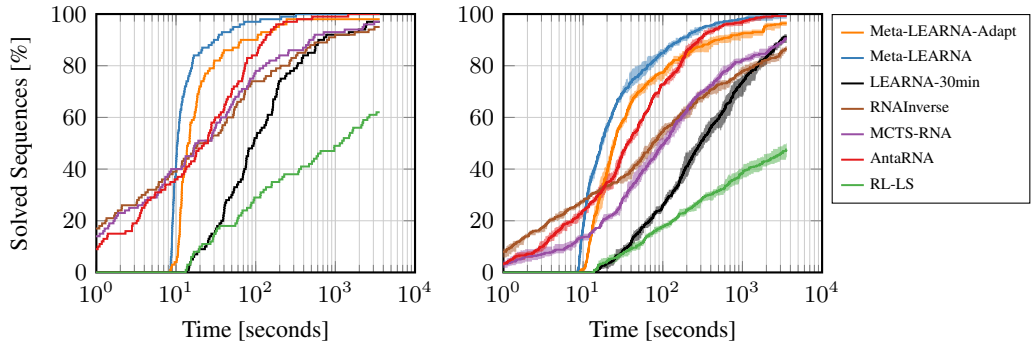


Figure 8: Comparison of all methods on RFAM LEARNA.

C ABLATION

C.1 META-LEARNA-ADAPT

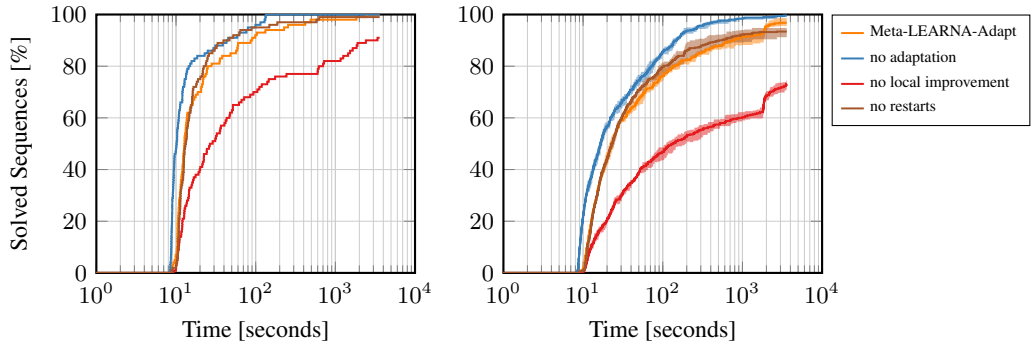


Figure 9: Ablation study of Meta-LEARNA-Adapt on RFAM LEARNA.

C.2 META-LEARNA

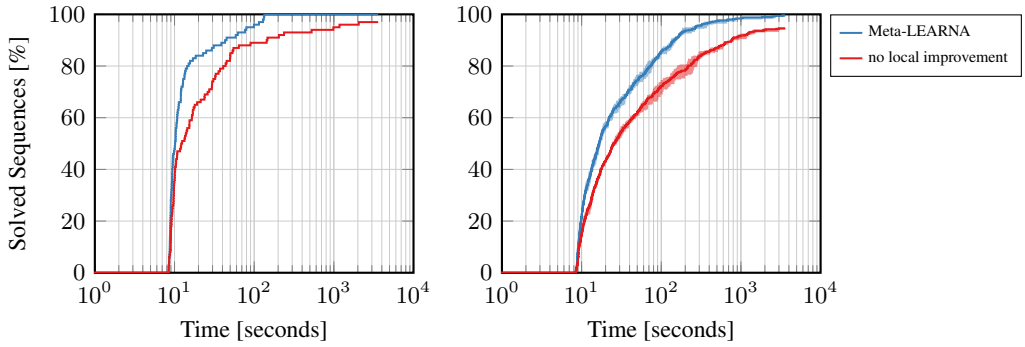


Figure 10: Aplation study of Meta-LEARNA on RFAM LEARNA.

C.3 LEARNA-30MIN

C.4 LEARNA-10MIN

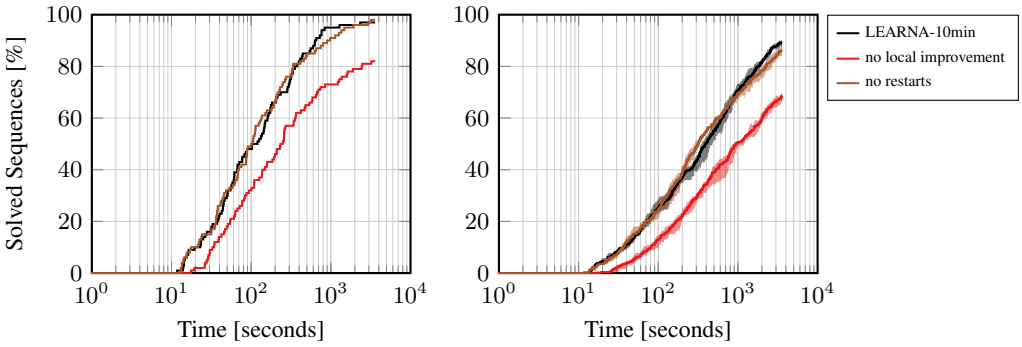


Figure 11: Aplation study of LEARNA-30min on RFAM LEARNA.

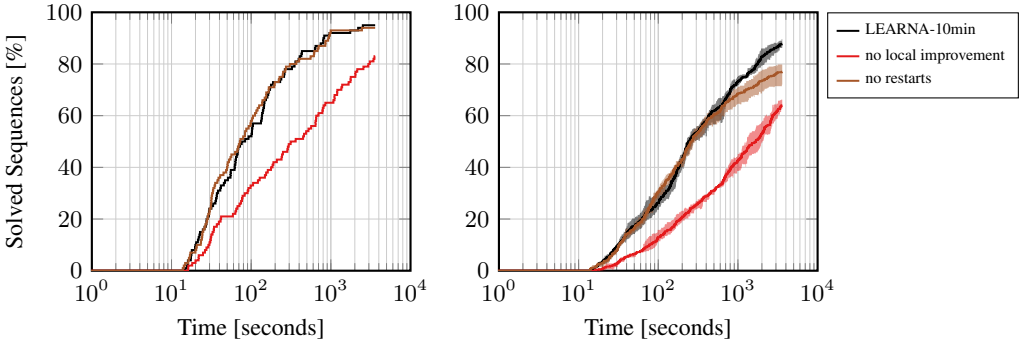


Figure 12: Aplation study of LEARNA-10min on RFAM LEARNA.

D VALIDATION

D.1 META-LEARNA

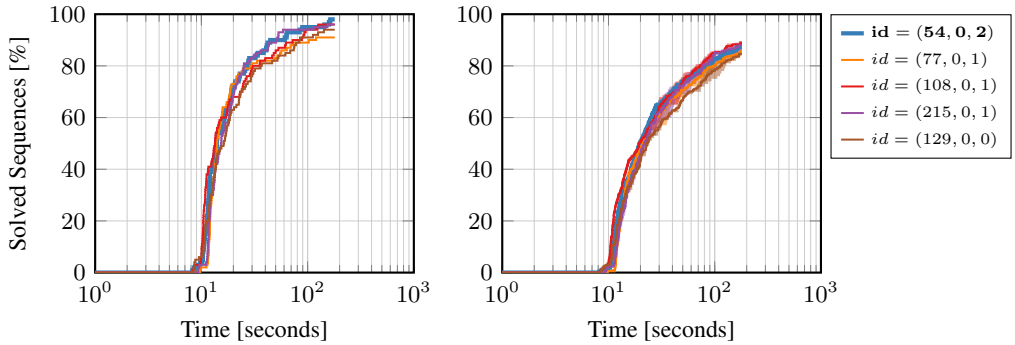


Figure 13: 5 independent runs of the best 5 configurations for Meta-LEARNA on the validation set to asses the robustness. Chosen configuration highlighted in bold.

D.2 LEARNA-30MIN

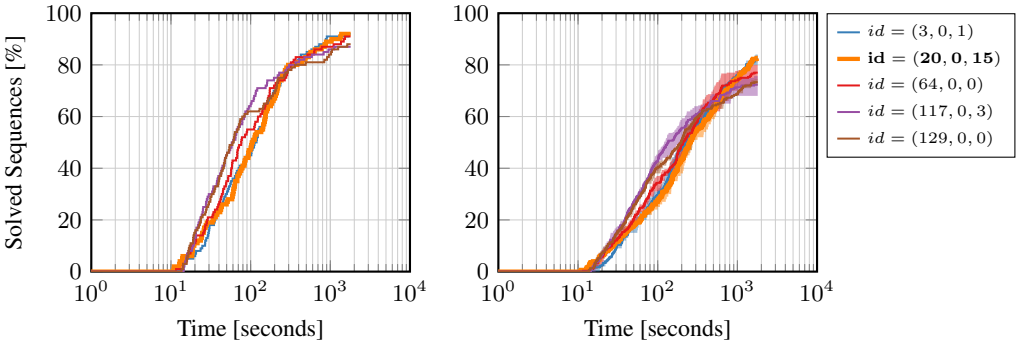


Figure 14: 5 independent runs of the best 5 configurations for LEARNA with a 30 minute budget on the validation set to asses the robustness. Chosen configuration highlighted in bold.

D.3 LEARNA-10MIN

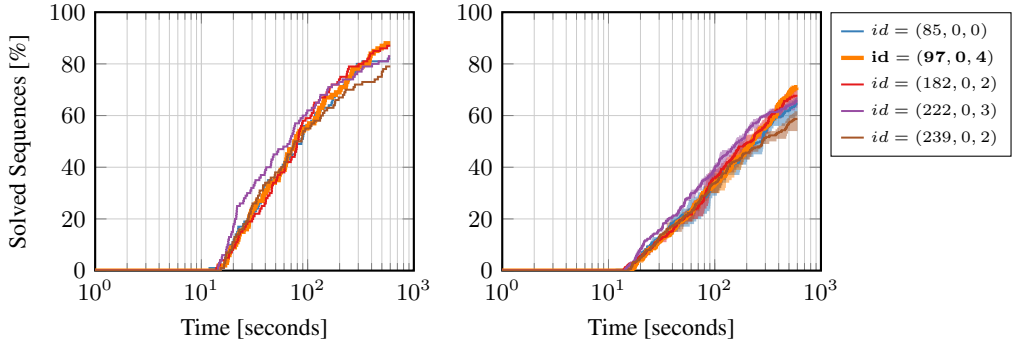


Figure 15: 5 independent runs of the best 5 configurations for LEARNA with a 10 minute budget on the validation set to asses the robustness. Chosen configuration highlighted in bold.

Table 5: Comparison between *RNAInverse*, *MCTS-RNA*, *RL-LS*, *antaRNA*, *LEARNA*, and our random baseline regarding the total number of solved target structures of the Eterna benchmark in accordance to Anderson-Lee et al. (2016) in percent. 5 runs with a timelimit of 24 hours were performed for each of the 100 target structures. A target structure was counted as solved if any of the 5 runs solved it.

METHOD	TOTAL (%)	2 RUNS	3 RUNS	4 RUNS	5 RUNS
RANDOM-NO-MUTATION	30	27	26	26	26
RANDOM	39	36	35	34	32
MCTS-RNA	57	57	56	54	51
ANTARNA	58	58	58	56	56
RL-LS	59	59	58	57	55
RNAINVERSE	60	60	59	59	58
LEARNA-30MIN	63	61	60	60	58
META-LEARNA-ADAPT	64	63	62	60	60
META-LEARNA	**65**	**65**	**64**	**64**	**64**

Table 6: Comparison between *MCTS-RNA*, *antaRNA*, *RL-LS*, *RNAInverse*, *LEARNA* and our random baseline regarding the total number of solved target structures of the Rfam derived benchmark in accordance to Taneda (2011) in percent. 50 runs with a timelimit of 10 minutes were performed for each of the 29 target structures. A target structure was counted as solved if any of the 50 runs solved it.

METHOD	TOTAL (%)	5 RUNS	10 RUNS	25 RUNS	50 RUNS
RANDOM-NO-MUTATION	28	21	21	21	17
RANDOM	35	28	28	24	21
RNAINVERSE	59	55	55	52	48
RL-LS	62	62	55	52	48
ANTARNA	66	66	66	66	62
MCTS-RNA	79	76	72	72	59
LEARNA-10MIN	79	76	76	66	28
META-LEARNA-ADAPT	**83**	**83**	**79**	**79**	69
META-LEARNA	79	79	79	**79**	**76**

E SUMMARY TABLES

Table 7: Comparison between *MCTS-RNA*, *antaRNA*, *RL-LS*, *RNAInverse*, *LEARNA* and our random baseline regarding the total number of solved target structures of the Rfam derived benchmark in percent. 3 runs with a timelimit of 4 hours were performed for each of the 100 target structures. A target structure was counted as solved if any of the 3 runs solved it.

METHOD	TOTAL (%)	2 RUNS	3 RUNS	4 RUNS	5 RUNS
RANDOM-NO-MUTATION	17	14	13	12	10
RANDOM	31	26	25	22	21
RL-LS	62	53	45	41	37
RNAINVERSE	95	90	87	83	78
MCTS-RNA	97	94	91	89	82
ANTARNA	**100**	99	99	**99**	**99**
LEARNA-10MIN	95	94	92	84	74
LEARNA-30MIN	97	94	93	90	82
META-LEARNA-ADAPT	98	98	98	96	92
META-LEARNA	**100**	**100**	**100**	**99**	98

Table 8: Comparison between *RNAInverse*, *MCTS-RNA*, *RL-LS*, *antaRNA*, *LEARNA*, and our random baseline regarding the total number of solved target structures of the Eterna benchmark in accordance to Anderson-Lee et al. (2016) in percent. 5 runs with a timelimit of 24 hours were performed for each of the 100 target structures. A target structure was counted as solved if any of the 5 runs solved it.

METHOD	TOTAL (%)	10S	1MIN	30MIN	1H	4H	12H
RANDOM-NO-MUTATION	30	15	21	25	26	29	30
RANDOM	39	19	22	29	33	37	39
MCTS-RNA	57	41	48	55	56	57	57
ANTARNA	58	36	46	54	55	55	58
RL-LS	59	0	40	53	55	58	59
RNAINVERSE	60	32	44	55	57	59	60
LEARNA-30MIN	63	0	41	58	58	60	63
META-LEARNA-ADAPT	64	14	56	59	59	63	64
META-LEARNA	**65**	**45**	**59**	**61**	**64**	**64**	**65**

Table 9: Comparison between *MCTS-RNA*, *antaRNA*, *RL-LS*, *RNAInverse*, *LEARNA* and our random baseline regarding the total number of solved target structures of the Rfam derived benchmark in accordance to Taneda (2011) in percent. 50 runs with a timelimit of 10 minutes were performed for each of the 29 target structures. A target structure was counted as solved if any of the 50 runs solved it.

METHOD	TOTAL (%)	10S	30S	1MIN	5MIN
RANDOM-NO-MUTATION	28	17	21	24	28
RANDOM	35	28	28	31	31
ANTARNA	66	52	62	66	66
RNAINVERSE	59	55	55	55	55
RL-LS	62	0	48	52	55
MCTS-RNA	79	72	76	76	79
LEARNA-10MIN	79	0	62	72	76
META-LEARNA-ADAPT	**83**	28	**79**	**83**	**83**
META-LEARNA	79	**79**	**79**	79	79

Table 10: Comparison between *MCTS-RNA*, *antaRNA*, *RL-LS*, *RNAInverse*, *LEARNA* and our random baseline regarding the total number of solved target structures of the Rfam derived benchmark in percent. 3 runs with a timelimit of 4 hours were performed for each of the 100 target structures. A target structure was counted as solved if any of the 3 runs solved it.

METHOD	TOTAL (%)	10S	30S	1MIN	5MIN	10MIN	20MIN	30MIN
RANDOM-NO-MUTATION	17	3	4	6	10	12	13	14
RANDOM	31	11	14	16	23	25	30	30
RL-LS	62	0	14	21	38	45	50	56
RNAINVERSE	95	39	53	66	83	89	92	93
MCTS-RNA	97	40	55	68	86	92	93	94
ANTARNA	**100**	36	58	73	97	99	99	**100**
LEARNA-10MIN	95	0	24	39	78	85	92	93
LEARNA-30MIN	97	0	15	34	78	88	92	93
META-LEARNA-ADAPT	98	4	81	87	98	98	98	98
META-LEARNA	**100**	**41**	**90**	**95**	**100**	**100**	**100**	**100**

F DETAIL TABLES

Table 11: Results for 5 independent attempts on the first half of the 100 targets of the Rfam benchmark. The timeout for each attempt was set to 4 hours.

ID	LEARNA-30MIN	META-LEARNA	MCTS-RNA	RL-SL	RNAINVERSE	ANTARNA
1	5/5	5/5	5/5	5/5	5/5	5/5
2	5/5	5/5	5/5	5/5	5/5	5/5
3	5/5	5/5	5/5	5/5	5/5	5/5
4	5/5	5/5	5/5	5/5	5/5	5/5
5	5/5	5/5	5/5	5/5	5/5	5/5
6	5/5	5/5	5/5	5/5	5/5	5/5
7	5/5	5/5	5/5	5/5	5/5	5/5
8	5/5	5/5	5/5	5/5	5/5	5/5
9	5/5	5/5	5/5	5/5	5/5	5/5
10	5/5	5/5	4/5	-	5/5	5/5
11	5/5	5/5	5/5	5/5	5/5	5/5
12	5/5	5/5	5/5	5/5	5/5	5/5
13	5/5	5/5	5/5	5/5	5/5	5/5
14	5/5	5/5	5/5	5/5	5/5	5/5
15	5/5	5/5	5/5	5/5	5/5	5/5
16	5/5	5/5	5/5	5/5	5/5	5/5
17	5/5	5/5	5/5	5/5	5/5	5/5
18	5/5	5/5	5/5	5/5	5/5	5/5
19	5/5	5/5	5/5	5/5	5/5	5/5
20	5/5	5/5	5/5	5/5	5/5	5/5
21	5/5	5/5	5/5	5/5	5/5	5/5
22	5/5	5/5	5/5	-	5/5	5/5
23	5/5	5/5	5/5	5/5	5/5	5/5
24	5/5	5/5	5/5	5/5	5/5	5/5
25	5/5	5/5	5/5	4/5	5/5	5/5
26	5/5	5/5	4/5	2/5	5/5	5/5
27	5/5	5/5	5/5	5/5	5/5	5/5
28	2/5	5/5	5/5	1/5	5/5	5/5
29	5/5	5/5	3/5	3/5	5/5	5/5
30	5/5	5/5	5/5	5/5	5/5	5/5
31	5/5	5/5	5/5	5/5	5/5	5/5
32	5/5	5/5	5/5	5/5	5/5	5/5
33	5/5	5/5	5/5	5/5	5/5	5/5
34	5/5	5/5	5/5	5/5	5/5	5/5
35	5/5	5/5	5/5	4/5	5/5	5/5
36	5/5	5/5	4/5	3/5	5/5	5/5
37	5/5	5/5	5/5	5/5	5/5	5/5
38	5/5	5/5	5/5	1/5	5/5	5/5
39	5/5	5/5	5/5	5/5	5/5	5/5
40	5/5	5/5	5/5	5/5	5/5	5/5
41	5/5	5/5	5/5	5/5	5/5	5/5
42	5/5	5/5	5/5	5/5	5/5	5/5
43	5/5	5/5	5/5	5/5	5/5	5/5
44	5/5	5/5	1/5	-	4/5	3/5
45	1/5	5/5	-	-	-	1/5
46	5/5	5/5	1/5	-	5/5	5/5
47	5/5	5/5	5/5	3/5	5/5	5/5
48	5/5	5/5	5/5	5/5	5/5	5/5
49	5/5	5/5	5/5	5/5	5/5	5/5
50	5/5	5/5	5/5	4/5	5/5	5/5
TOTAL	243/250	250/250	232/250	205/250	244/250	244/250
SOLVED	50/50	50/50	49/50	45/50	49/50	50/50

Table 12: Results for 3 independent attempts on the second half of the 100 targets of the Rfam benchmark. The timeout for each attempt was set to 4 hours.

ID	LEARNA	META-LEARNA	MCTS-RNA	RL-SL	RNAINVERSE	ANTARNA
51	5/5	5/5	5/5	1/5	5/5	5/5
52	5/5	5/5	5/5	5/5	5/5	5/5
53	5/5	5/5	5/5	1/5	5/5	5/5
54	5/5	5/5	5/5	2/5	5/5	5/5
55	5/5	5/5	5/5	3/5	5/5	5/5
56	5/5	5/5	5/5	1/5	5/5	5/5
57	5/5	5/5	5/5	-	4/5	5/5
58	5/5	5/5	5/5	-	3/5	5/5
59	5/5	5/5	5/5	1/5	5/5	5/5
60	5/5	5/5	5/5	1/5	3/5	5/5
61	5/5	5/5	4/5	1/5	1/5	5/5
62	5/5	5/5	5/5	1/5	5/5	5/5
63	5/5	5/5	5/5	1/5	5/5	5/5
64	3/5	5/5	5/5	-	5/5	5/5
65	5/5	5/5	5/5	-	5/5	5/5
66	5/5	5/5	5/5	-	5/5	5/5
67	5/5	5/5	5/5	-	5/5	5/5
68	5/5	5/5	5/5	2/5	5/5	5/5
69	4/5	5/5	5/5	-	3/5	5/5
70	5/5	5/5	5/5	1/5	5/5	5/5
71	5/5	5/5	5/5	-	5/5	5/5
72	5/5	5/5	5/5	-	5/5	5/5
73	4/5	5/5	5/5	-	5/5	5/5
74	5/5	5/5	5/5	-	2/5	5/5
75	5/5	5/5	5/5	-	5/5	5/5
76	5/5	5/5	5/5	1/5	2/5	5/5
77	5/5	5/5	5/5	4/5	5/5	5/5
78	5/5	5/5	5/5	-	4/5	5/5
79	5/5	5/5	5/5	-	5/5	5/5
80	-	4/5	-	-	1/5	5/5
81	4/5	5/5	5/5	-	2/5	5/5
82	4/5	5/5	4/5	-	4/5	5/5
83	3/5	5/5	5/5	-	3/5	5/5
84	5/5	5/5	4/5	-	5/5	5/5
85	-	5/5	1/5	-	5/5	5/5
86	5/5	5/5	5/5	2/5	5/5	5/5
87	4/5	5/5	5/5	-	5/5	5/5
88	3/5	5/5	5/5	-	1/5	5/5
89	1/5	5/5	2/5	-	-	5/5
90	4/5	5/5	2/5	-	5/5	5/5
91	5/5	5/5	5/5	-	5/5	5/5
92	5/5	5/5	5/5	-	5/5	5/5
93	4/5	5/5	4/5	-	4/5	5/5
94	5/5	5/5	5/5	-	5/5	5/5
95	5/5	5/5	5/5	-	1/5	5/5
96	5/5	5/5	5/5	2/5	5/5	5/5
97	1/5	3/5	2/5	-	1/5	5/5
98	5/5	5/5	5/5	-	-	5/5
99	4/5	5/5	5/5	-	5/5	5/5
100	-	5/5	3/5	-	-	5/5
TOTAL	218/250	247/250	226/250	30/250	198/250	250/250
SOLVED	47/50	50/50	49/50	17/50	47/50	50/50

Table 13: Results for 5 independent attempts on each of the 100 targets of the Eterna100 benchmark in accordance to (Anderson-Lee et al., 2016). The timeout for each attempt was set to 24 hours.

ID	LEARNA-30MIN	META-LEARNA	MCTS-RNA	RL-SL	RNAINVERSE	ANTARNA
1	5/5	5/5	5/5	5/5	5/5	5/5
2	5/5	5/5	5/5	5/5	5/5	5/5
3	5/5	5/5	5/5	5/5	5/5	5/5
4	5/5	5/5	5/5	5/5	5/5	5/5
5	5/5	5/5	5/5	5/5	5/5	5/5
6	5/5	5/5	5/5	-	5/5	5/5
7	5/5	5/5	5/5	5/5	5/5	5/5
8	5/5	5/5	5/5	5/5	5/5	5/5
9	-	5/5	-	4/5	-	5/5
10	5/5	5/5	-	-	5/5	5/5
11	5/5	5/5	5/5	5/5	5/5	5/5
12	5/5	5/5	5/5	5/5	5/5	5/5
13	5/5	5/5	5/5	5/5	5/5	5/5
14	5/5	5/5	5/5	5/5	5/5	5/5
15	4/5	5/5	5/5	5/5	5/5	5/5
16	5/5	5/5	-	3/5	-	-
17	5/5	5/5	4/5	5/5	2/5	-
18	5/5	5/5	5/5	5/5	5/5	5/5
19	5/5	5/5	5/5	5/5	5/5	5/5
20	5/5	5/5	4/5	5/5	5/5	5/5
21	5/5	5/5	5/5	5/5	5/5	5/5
22	-	5/5	5/5	5/5	-	5/5
23	5/5	5/5	-	5/5	5/5	5/5
24	5/5	5/5	5/5	5/5	5/5	5/5
25	5/5	5/5	5/5	5/5	5/5	5/5
26	5/5	5/5	5/5	5/5	5/5	5/5
27	5/5	5/5	5/5	5/5	5/5	5/5
28	5/5	5/5	5/5	5/5	5/5	5/5
29	5/5	5/5	5/5	5/5	5/5	5/5
30	5/5	5/5	-	5/5	5/5	5/5
31	5/5	5/5	5/5	5/5	5/5	5/5
32	5/5	5/5	5/5	5/5	5/5	5/5
33	5/5	5/5	-	-	5/5	5/5
34	5/5	5/5	5/5	5/5	5/5	5/5
35	-	5/5	-	-	-	-
36	5/5	5/5	5/5	5/5	5/5	5/5
37	5/5	5/5	5/5	2/5	4/5	-
38	5/5	5/5	5/5	-	-	-
39	5/5	5/5	5/5	5/5	5/5	5/5
40	5/5	5/5	5/5	5/5	5/5	5/5
41	5/5	5/5	-	5/5	5/5	5/5
42	5/5	5/5	4/5	5/5	5/5	5/5
43	5/5	5/5	5/5	5/5	5/5	5/5
44	5/5	5/5	5/5	5/5	5/5	5/5
45	5/5	5/5	5/5	5/5	5/5	5/5
46	5/5	5/5	5/5	5/5	5/5	5/5
47	5/5	5/5	5/5	5/5	5/5	5/5
48	5/5	5/5	5/5	5/5	5/5	5/5
49	5/5	5/5	5/5	5/5	5/5	5/5
50	-	-	-	-	-	-
TOTAL	228/250	245/250	202/250	219/250	216/250	218/250
SOLVED	46/50	49/50	41/50	44/50	44/50	44/44

Table 14: Results for 5 independent attempts on each of the 100 targets of the Eterna100 benchmark in accordance to (Anderson-Lee et al., 2016). The timeout for each attempt was set to 24 hours.

ID	LEARNA-30MIN	META-LEARNA	MCTS-RNA	RL-SL	RNAINVERSE	ANTARNA
51	5/5	5/5	5/5	5/5	5/5	5/5
52	-	-	-	-	-	-
53	-	5/5	-	-	-	-
54	5/5	5/5	5/5	5/5	5/5	5/5
55	5/5	5/5	5/5	5/5	5/5	5/5
56	5/5	5/5	5/5	5/5	5/5	5/5
57	-	-	-	-	-	-
58	5/5	5/5	5/5	5/5	5/5	5/5
59	5/5	5/5	5/5	5/5	5/5	-
60	-	-	-	-	-	-
61	-	-	-	-	-	-
62	-	-	-	-	5/5	3/5
63	5/5	5/5	5/5	5/5	5/5	5/5
64	-	-	-	-	-	-
65	-	-	2/5	-	-	-
66	1/5	-	-	-	5/5	5/5
67	-	-	-	-	-	-
68	-	-	-	-	-	-
69	4/5	5/5	5/5	-	-	-
70	5/5	5/5	3/5	-	-	4/5
71	-	-	-	-	-	-
72	-	-	-	5/5	5/5	-
73	-	-	-	-	-	-
74	-	2/5	-	-	-	-
75	5/5	5/5	5/5	5/5	5/5	5/5
76	-	-	-	-	-	-
77	2/5	-	3/5	4/5	5/5	-
78	-	-	-	-	-	-
79	-	-	-	-	-	-
80	-	-	-	-	-	-
81	-	-	-	-	-	-
82	5/5	5/5	5/5	5/5	5/5	5/5
83	-		-	-	-	-
84	5/5	5/5	5/5	5/5	5/5	5/5
85	-	-	-	-	-	-
86	-	-	-	-	-	-
87	-	-	-	-	-	-
88	-	-	-	-	-	-
89	-	-	-	-	-	-
90	-	-	-	-	-	-
91	-	-	-	-	-	-
92	-	-	-	-	-	-
93	5/5	5/5	5/5	5/5	5/5	5/5
94	-	-	-	-	-	-
95	5/5	5/5	5/5	5/5	5/5	5/5
96	-	-	-	-	-	-
97	-	-	-	-	-	-
98	1/5	-	-	-	-	-
99	-	-	-	-	-	-
100	-	-	-	-	-	-
TOTAL	73/250	77/250	73/250	69/250	80/250	67/250
SOLVED	17/50	16/50	16/50	14/50	16/50	14/50

Table 15: Results for 50 independent attempts on each of the 29 targets of the Rfam benchmark in accordance to (Taneda, 2011). The timeout for each attempt was set to 10 minutes.

ID	LEARNA-10MIN	META-LEARNA	MCTS-RNA	RL-SL	RNAINVERSE	ANTARNA
1	24/50	50/50	32/50	7/50	20/50	50/50
2	40/50	50/50	28/50	5/50	-	50/50
3	1/50	46/50	4/50	-	-	50/50
4	48/50	50/50	50/50	50/50	50/50	50/50
5	50/50	50/50	50/50	50/50	50/50	50/50
6	50/50	50/50	50/50	50/50	50/50	50/50
7	46/50	50/50	50/50	48/50	50/50	50/50
8	50/50	50/50	50/50	50/50	50/50	50/50
9	21/50	50/50	44/50	-	-	-
10	-	-	-	-	-	-
11	-	-	-	-	-	-
12	41/50	50/50	50/50	50/50	3/50	50/50
13	47/50	50/50	50/50	50/50	50/50	50/50
14	50/50	50/50	50/50	50/50	50/50	50/50
15	50/50	50/50	50/50	50/50	50/50	50/50
16	-	-	-	-	-	-
17	29/50	50/50	47/50	-	50/50	50/50
18	-	-	-	-	-	-
19	50/50	50/50	50/50	50/50	50/50	50/50
20	-	-	-	-	-	-
21	50/50	50/50	50/50	50/50	50/50	50/50
22	48/50	50/50	50/50	50/50	50/50	50/50
23	-	-	-	-	-	-
24	43/50	50/50	50/50	19/50	-	50/50
25	49/50	50/50	50/50	50/50	46/50	50/50
26	50/50	50/50	50/50	50/50	50/50	50/50
27	10/50	50/50	6/50	-	-	-
28	48/50	50/50	50/50	50/50	50/50	50/50
29	35/50	50/50	50/50	-	-	36/50
TOTAL	930/1450	1146/1450	1011/1450	779/1450	769/1450	1036/1450
SOLVED	23/29	23/29	23/29	18/29	17/29	21/29