RNAInvBench: Benchmark for the RNA Inverse Design Problem

Jack Cole [1] **Fan Li** [1] **Liwen Wu** [2] **Krasmira Tsaneva-Atanasova** [3] **Ke Li** [1]

Abstract: The RNA inverse folding problem involves a challenging one-to-many problem space, where countless RNA sequences can fold into a central secondary structure. The central methodology for generating a varied set of solutions involves an algorithmic approach to navigate the search space. The broad set of solutions thus allows for the optimisation of sequences to improve function and efficacy. Despite a range of impressive algorithms, the lack of a unified platform for comparing models and techniques hinders comprehensive analysis. To address this, we propose RNAInvBench, an open-source framework for Inverse RNA Design, which comprises of two focused tasks, pseudoknot-free and pseudoknot-inclusive design, two essential tasks for understanding and benchmarking model efficacy and accuracy. We further collate and organise key datasets for future model training and testing, and provide several detailed metrics for evaluating the success, function and efficiency of the generated sequences. We provide package environments through dockerfiles, to facilitate easy access for researchers. All source code is publicly available for use and download at: https://github.com/COLA-Laboratory/RNAInvBench

Keywords: Benchmark, bioinformatics, evolutionary strategy, reinforcement learning, adaptive walk, model-based, RNA design, RNA inverse folding

1. Introduction

Ribonucleic Acid (RNA), a significant biological molecule fundamental to the function and regulation of living organisms, has recently garnered interest due to its applicability in therapeutics and vaccine design. (Wang and Farhana, 2024). Composed of a sequence of nucleotides, RNA molecules fold into secondary or tertiary structures to facilitate interactions with other cellular components and perform its biological roles effectively (Ganser et al., 2019). The folding process is achieved through base-pairing interactions, with the length of the sequence and the specific types of interactions being crucial for the folded RNA's stability and function. The secondary structure begins the creation of local structures and motifs within the RNA (Tinoco and Bustamante, 1999). By using a secondary structure with useful interaction regions, we can guide the formation of the final tertiary structure, thereby producing efficient, useful, stable RNA structures.

Inverse RNA folding describes the reverse of the traditional RNA problem; where an RNA sequence is folded into a secondary structure (Hofacker et al., 1994). By reversing this process and taking a secondary structure with a known useful function, we can find RNA sequences that will fold into this structure. This allows biologists to create artificial RNA molecules to complete a specific function, such as artificial ribozymes (Bauer and Suess, 2006), miRNAs (Schwab et al., 2006), or riboswitches (Findeiß et al., 2017). This problem has been proved NP-hard (Bonnet et al., 2020), and thus simple brute-force search is implausible, which has spurred interest and development of a variety of algorithmic and deep learning approaches (Runge et al., 2019; Shi et al., 2018; Minuesa et al., 2021).

The problem complexity is further exacerbated with the inclusion of pseudoknots, a key part of an RNA structure proven to have a profound impact on an RNA's functionality and efficiency (Giedroc et al., 2000). There are four types of pseudoknots discovered thus far, which are classified in terms of complexity by (Kucharík et al., 2016). These being the commonly found H-type (Hairpin) and K-type (Kissing Hairpin), and rarer L-type and M-type pseudoknots (Staple and Butcher, 2005). Despite the importance of pseudoknots within the RNA secondary structure, the majority of inverse RNA folding tools do not consider them when constructing sequences (Runge et al., 2019; Koodli et al., 2019; Lyngsø et al., 2012; Minuesa et al., 2021; Reinharz et al., 2013).

Previous Inverse Folding RNA comparisons have been established, such as (Churkin et al., 2017), however, several new approaches and algorithms have been introduced since, such as Reinforcement Learning (Runge et al., 2019) and Model-based optimisation (Shi et al., 2018) approaches. Further to this, the previously popular benchmark dataset, Eterna100v1, was proven to have 19 unsolvable sequences

[1]Department of Computer Science, University of Exeter, Exeter, UK [2]University of Electronic Science and Technology of China, Sichuan, China [3]Department of Mathematics and Statistics, University of Exeter, Exeter, UK. Correspondence to: Ke Li <k.li@exeter.ac.uk>.

Proceedings of the 5^{th} Workshop on AI for Science at the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

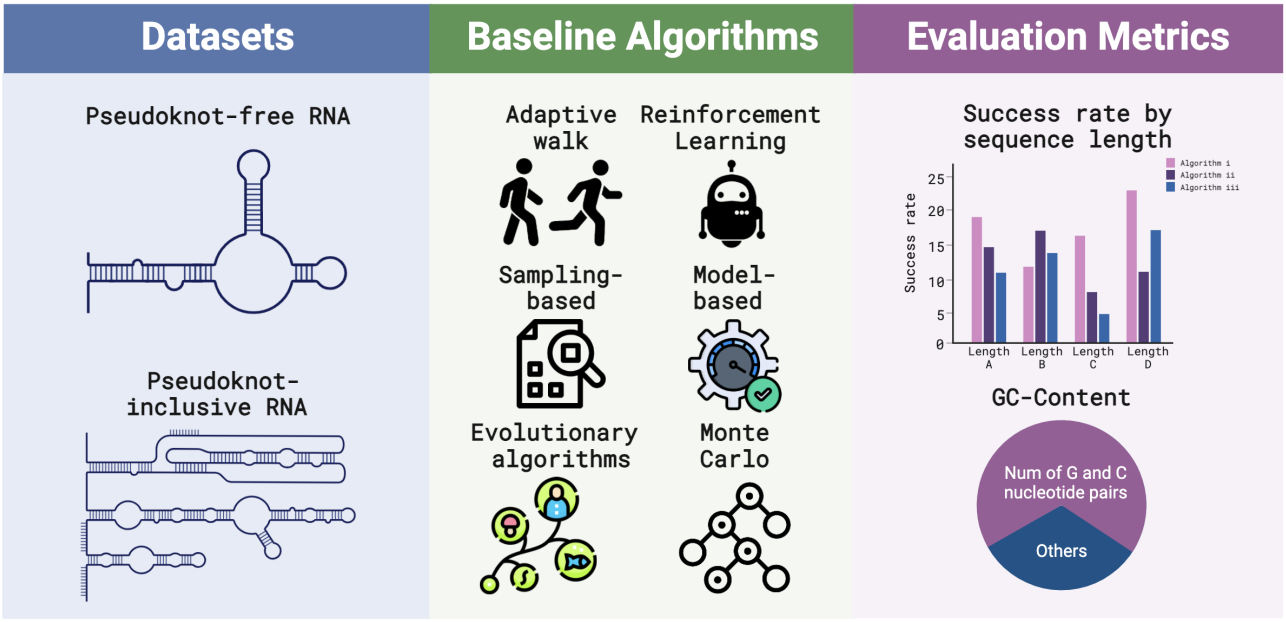

Figure 1. **RNAInvBench benchmarks.** RNAInvBench is composed of three layers. The data layer encompasses the gathering of publicly available datasets for the two key tasks of inverse RNA design, the pseudoknot-free and pseudoknot-inclusive tasks. The baseline algorithms layer is comprised of a diverse set of algorithms, idealised for the pseudoknot-free task, although two algorithms are provided for the pseudoknot-inclusive task. Lastly, the evaluation metrics layer demonstrates the two key metrics we use on-top of success rate, to allow for further analysis into the effectiveness and efficacy of the algorithms.

when using the Vienna 2.4 folding engine (Koodli et al., 2021), resulting in a lowered standard of results for the algorithms used. Whilst a brief update to some algorithms (RNAinverse, EternaBrain, LEARNA, SentRNA, NEMO) was conducted, this was not comprehensive for the current landscape, and as such we aim to update these results with the newly redesigned Eterna100v2, where all 100 sequences are solvable with Vienna 2. We aim to provide results for both newly developed and historically strong approaches and algorithms.

A central contribution of this work is the open-source release of a framework that will allow researchers to examine and evaluate key algorithmic approaches within this field. Previous work such as (Churkin et al., 2017) have provided a comparison, but have not provided a framework that facilitates further use from other researchers. A more recent work, RnaBench (Runge et al., 2024) provides an open-source framework for inverse-folding algorithms to be run and utilised, however these algorithms are limited to evolutionary (antaRNA), model-based (SentRNA, EternaBrain), adaptive walk methods (RNAInverse), and Monte-Carlo Tree Search (MCTS). Further, a comparison of the results of these algorithms using this benchmark has yet to be included within this paper.

Further to this is the benchmark of algorithms that can consider pseudoknots within their designed RNA sequences. Although it has been proven that pseudoknots can have a profound impact on an RNA's functionality and efficiency (Giedroc et al., 2000; Brierley et al., 2007), a comprehensive comparison of pseudoknot algorithms has yet to be undertaken in academic literature, resulting in a critical gap in our understanding of their efficacy and performance. In this work, we provide a novel comparison between the most popular pseudoknot solving algorithms, and include this within our open-source framework to facilitate future research.

This work aims to provide a standardised open-source platform for using, comparing, and bench-marking secondary structure RNA inverse folding algorithms, with datasets that can be easily re-implemented. We include both pseudoknot-free and pseudoknot-inclusive algorithms, as outlined in our experimental design.

2. Benchmarks and Evaluation framework

Within RNAInvBench we propose two key tasks, inverse RNA design, and inverse RNA design with pseudoknots. Both tasks follow a general principle: initialise the algorithms with a target secondary structure stored using dot-bracket notation; choose the algorithm to utilise; narrow down the search space with the algorithm; fold the se-

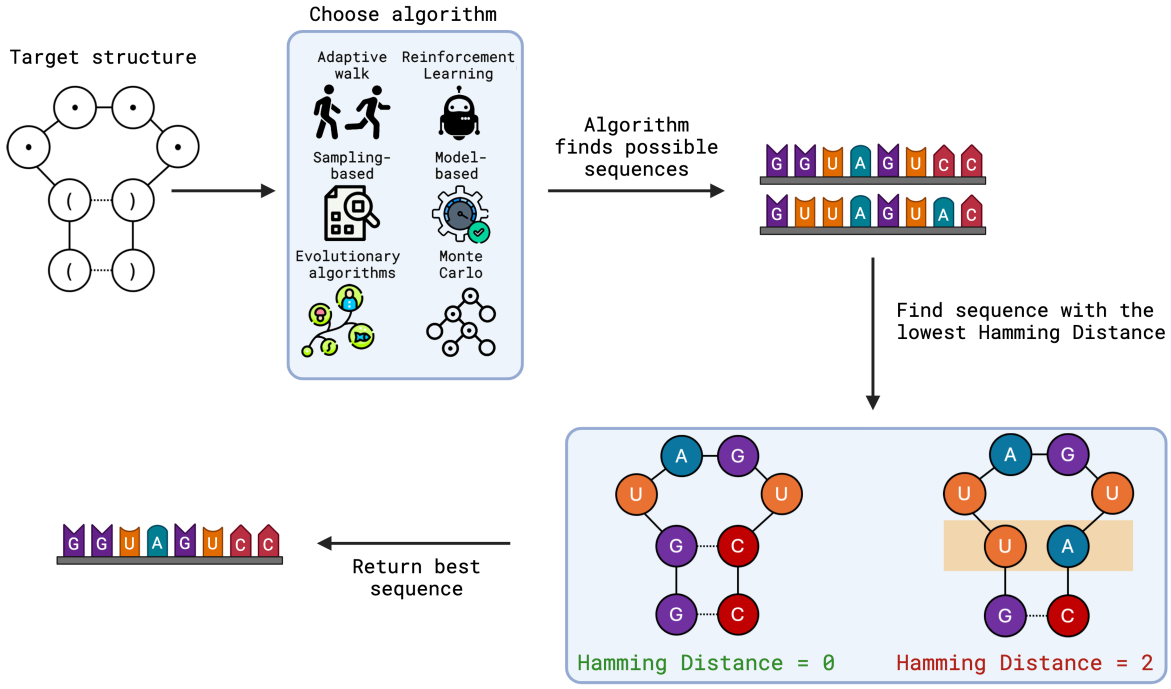

Figure 2. **RNA inverse design methodology.** We provide an outline of the RNA inverse design problem formulation. An initial target structure is identified, and an algorithmic approach is selected. The algorithm is used to narrow down the search space of the problem, and it returns several possible sequences that fit the criteria (e.g., matches the target structure and GC-Content). To select the best sequence out of the ones returned, we first fold the sequence using a folding algorithm, and then compare the hamming distance with the target structure. Finally, the best sequence is returned to the user.

quences using a folding algorithm; evaluate the sequences using hamming distance; return the best sequence to the user. This methodology is outlined in fig. 2.

The folding algorithm that we use to fold our generated sequences is vital to the accuracy of our algorithms. Current work mostly uses ViennaRNA 2 (Lorenz et al., 2011), as it is easy to integrate, fast (as it has a time complexity of $O(n^3)$), and efficient. ViennaRNA 2 converts the sequence to the dot-bracket notation, unable to incorporate pseudoknots within the structures it outputs. Alternatively, we use the pKiss package (Janssen and Giegerich, 2014) to predict the pseudoknots, which increases the time complexity from $O(n^3)$ for pseudoknot-free structures to $O(n^4)$ for predicting pseudoknot inclusive structures with heuristics. The algorithms used in for the pseudoknot problem within this benchmark include antaRNA (Kleinkauf et al., 2015) and MCTS (Yang et al., 2017).

2.1. Dataset types

In this section we describe the variety of datasets that we have collated for the purposes of training and testing both algorithms for both the pseudoknot-free and pseudoknot-inclusive benchmarks. We further describe the two datasets used for benchmarking these algorithms. The Eterna100v2 dataset (Koodli et al., 2021) was used for the non-pseudoknot problem, and PseudoBase++ (Kleinkauf et al., 2015) was used for the pseudoknot problem. Whilst we include the Eterna100v1 (Anderson-Lee et al., 2016) dataset in the benchmark framework, it was notably excluded from this benchmark, due to the usage of the Vienna 2 Folding Engine for the algorithms, as (Koodli et al., 2021) found that 19 sequences within the Eterna100v1 dataset are unsolvable with the Vienna 2 Folding Engine. A complete table of the dataset information is included within table 4.

Pseudoknot-free Training. We include the Rfam-learn set, which consists of $65,000$ sequences, established by (Runge et al., 2019). This dataset was created through folding Rfam sequences with the ViennaRNA package, and as such it is not reliable to use for testing, but can be used to generalise algorithm training.

Pseudoknot-free Testing. We include several test sets gathered from current literature. The Eterna100v1 (Anderson-Lee et al., 2016) and Eterna100v2 (Koodli et al., 2021) test sets are included, each consisting of 100 structures each (although 81 of these sequences are the same). We provide Eterna100v1 to allow for future benchmarks that use a different folding engine to be compared and con-

trasted with the unsolvable sequences. Eterna100v2 was provided as it consists of a varied sequence length and difficulty, allowing for a robust test set. We further include RNA-Strand provided by (Kleinkauf et al., 2015), which contains 50 short-length structures, where the longest structures are only 98 nucleotides. This is provided to be used as a benchmark for short-length structures. We include the Rfam-Test set provided by (Kleinkauf et al., 2015), and the Rfam-Taneda set provided by (Taneda, 2011), with the Rfam-Test set consisting of 63 structures, with a max-length for 273 nucleotides, which is ideal for benchmarking the middle-length structures. Rfam-Taneda is provided as an additional test set alongside Eterna100-v2, consisting of 29 structures with a max-length of 451 nucleotides. Lastly, we include the Rfam-Learn-Test set, containing 100 structures constructed in the same way as Rfam-Learn-Train (Runge et al., 2019). This set is aimed at providing initial testing for algorithms that plan to use the Rfam-Learn-Train dataset.

PseudoBase++. PseudoBase++ (van Batenburg et al., 2000) contains a much smaller set of sequences than Eterna100, with the largest being only 137 nucleotides long, but 252 sequences rather than only 100. Each sequence within this dataset contains a pseudoknot, thereby restricting it's usage to only algorithms that can handle pseudoknots.

2.2. Baselines

We have selected nine RNA inverse design baselines for our benchmark. They are ▶ RNAInverse (Lorenz et al., 2011; Hofacker et al., 1994) as the adaptive walk method; ▶ antaRNA (Kleinkauf et al., 2015) and SAMFEO (Zhou et al., 2023a) as evolutionary algorithms; ▶ SentRNA (Shi et al., 2018) as a model-based method; ▶ LEARNA, Meta-LEARNA, and Meta-LEARNA-Adapt (Runge et al., 2019) as reinforcement learning-based methods; ▶ MCTS (Yang et al., 2017) as monte carlo based methods; and ▶ IncaRNAtion (Reinharz et al., 2013) as the sampling-based method. The details of each algorithm are described in appendix A.2. Below, we provide an explanation of each type of method.

2.2.1. Adaptive walk methods

The adaptive walk method involves two primary steps. In the first step, the algorithm establishes a starting point by generating a random seed sequence. The second step involves progressively modifying or mutating this seed sequence to align it with the target structure through a stochastic search process (Lorenz et al., 2011; Hofacker et al., 1994; Andronescu et al., 2004).

2.2.2. Evolutionary Algorithms

Evolutionary algorithms solve the RNA inverse design problem primarily through iterative optimisation of candidate sequences against predefined structural and functional objectives (Taneda, 2011; Lyngsø et al., 2012; Kleinkauf et al., 2015; Zhou et al., 2023a; Merleau and Smerlak, 2022).

2.2.3. Model-based method

Model-based methods approach the RNA inverse design problem by employing surrogate models that leverage existing data to predict and optimise RNA sequences. These methods train models such as neural networks (Shi et al., 2018) or convolutional neural networks (Koodli et al., 2019) on datasets comprised of RNA sequences and structures, enabling them to learn the underlying patterns critical for effective RNA design. Once trained, these models can generate new RNA sequences that are then iteratively refined through techniques like adaptive walks, which involve making small, random changes to the sequences to enhance their alignment with the target structures. This process allows for the systematic improvement of RNA designs towards optimal configurations that meet predefined criteria.

2.2.4. Reinforcement Learning

Reinforcement Learning (RL) methods rely on an RL algorithm (in this case, Proximal Policy Optimisation (Schulman et al., 2017)) to create an optimal policy through online learning, which is followed by a deep neural network. Before RL can be used, we must first model the problem as a Markov Decision Process (MDP), achieved through the following equation, where T represents the current target structure T. $MDP = (S, A, R_T, P_T)$ (Sutton and Barto, 2018). The state-of-the-art RL-based algorithms were developed by (Runge et al., 2019), although these algorithms require a large amount of computation time to be accurate with long sequences (>=1 hour per sequence).

2.2.5. Monte Carlo

Monte Carlo methods can leverage random sampling and probabilistic modeling to narrow down the search space to only the most promising reasons. Monte Carlo sampling methods are used in-tandem with a search-based method, e.g., tree search (Yang et al., 2017) or heuristics to guide the search and mutations of the sequence (Portela, 2018). These algorithms allow for a more efficient form of solving the search space, but still require significant computational time to fully explore the possible transitions.

2.2.6. Sampling-based

Sampling-based approaches generally use a type of sampling to gather different options for improving the sequence, and select the best sample through a cost or objective function. These techniques fall into two categories: random-based and sample-based. Random-based methods, such as RNAInverse (Lorenz et al., 2011; Andronescu et al.,

2004) generate mutations randomly within the initial sequence and then evaluate these changes against a cost or objective function to assess their goodness. Conversely, sample-based methods like IncaRNAtion (Reinharz et al., 2013) use a weighted distribution for more efficient sampling, or a heuristic to guide sampling to more efficient and effective.

2.3. Evaluation frameworks

A significant problem with the collation of differing algorithmic approaches is the amount of time required for algorithms to find the RNA sequence (Aguirre-Hernández et al., 2007). The length of the sequence will have a massive effect on the overall required computation time (Schnall-Levin et al., 2008), as for each nucleotide in the sequence, there are 4 possible values (G, C, A, U), and 6 possible pairings (GC, CG, AU, UA, GU, UG) (Höner zu Siederdissen et al., 2011). This is represented as $4^u \cdot 6^{p/2}$, where u represents the number of unpaired nucleotides, and p represents the number of paired nucleotides within the sequence. The exponential growth in computational complexity have resulted in algorithms that require a lengthy run period to produce accurate sequences.

Despite this, we challenge the assumption that the extreme amount of computation time used by previous studies (Runge et al., 2019; Churkin et al., 2017; Kleinkauf et al., 2015; Hofacker et al., 1994; Garcia-Martin et al., 2013) is required to properly benchmark these algorithms. By utilising the novel metric, number of solved sequences per length, we conduct further analysis into the efficacy of the algorithms on longer and shorter sequences. We further separate our analysis into key periods of time to demonstrate the effect that computation time has on the efficacy of the algorithms on various sequence lengths. This allows us to identify key areas where performance on sequence types begins to fall off. Furthermore, by restricting the algorithms to the same run-time, we can ensure that the comparison is fair between techniques.

Here we describe the metrics used to evaluate the performance of the algorithms. Computation time was included as part of the ablation study, to show how the difference in computation time affects model performance.

Number of successfully solved sequences (within a time limit): Here we report the percentage of sequences that were successfully solved, where a successful solve is defined as when the Hamming Distance between the target and predicted structure is 0, or the normalised hamming distance is 1. We report this for several time limits, using 1-second as a k-shot benchmark, and 1-hour as our longest benchmark.

Number of solved sequences of a particular length: We report the number of solved sequences within the length categories identified in Table 2. This provides further insight into the type of sequences that the algorithms are struggling to predict.

GC-Content GC-content represents the amount of G and C nucleotide pairs within a sequence. Previous studies (Isaacs et al., 2006; Chan et al., 2009) have shown that a higher concentration of GC pairs could have a strong impact on the function of an RNA, and therefore it is important to include when considering the application of the methods evaluated.

Table 1. Eterna100v2 Sequence Length Distribution

Sequence Length	Number of Sequences
350-400	19
150-349	15
100-149	26
50-99	23
0-49	17

3. Experimental Design

In this section, we demonstrate the usage of our benchmarks to showcase it's applicability in solving research and biological questions. We use our benchmark suite to run a large-scale empirical study on two central tasks, pseudoknot-free RNA design, and pseudoknot-inclusive RNA design.

As the experimental design, and thereby analysis of the results, will differ between the pseudoknot-free and pseudoknot-inclusive tasks, we divide this section according to these two problems, as to present our findings in a clear and concise manner.

Pseudoknot-free. For the pseudoknot-free problem, we aim to evaluate the performance of our baseline algorithms, and quantify the relationship between sequence length and algorithm efficacy. The folding algorithm utilised was kept as ViennaRNA 2 throughout, in order to keep consistent results. We utilise an average of 3 runs with randomised seeds for these results, as to ensure reproducibility and mitigate potential outliers.

Pseudoknot-inclusive. For the pseudoknot-inclusive problem, we aim to evaluate the effectiveness of our algorithms at producing useful sequences through the measure of GC Content, and assess the performance of the algorithm on data that includes varying types of pseudoknots. We do not utilise the short computation times shown within the first benchmark, due to the additional computation required for the analysis of pseudoknot RNA structures. It should be noted that the folding algorithm utilised was kept as pKiss throughout, to ensure for a fair comparison between algorithms.

3.1. Data for experiments

Pseudobase++ We provide the Pseudobase++ training set first released by (Kleinkauf et al., 2015), which consists of 304 structures, of which 37 were excluded for having no base-pairings. We have split the data into training and testing using the same conditions as (Kleinkauf et al., 2015). The training data consists of 13 structures containing H-type pseudoknots, and 3 structures containing bulge hair-pin pseudoknots (B-type). The testing data consists of 251 sequences, of which there are 219 H-type, 29 B-type, and 3 K-type structures.

Eterna100v2 Eterna100v1 was previously used as a benchmark (Runge et al., 2019; Eastman et al., 2018; Koodli et al., 2019; Shi et al., 2018; Yang et al., 2017), however with the introduction of Eterna100v2 and several new algorithms claiming state-of-the-art (Merleau and Smerlak, 2022; Minuesa et al., 2021; Zhou et al., 2023a), this benchmark must be updated to provide a complete comparison. This dataset consists of 100 synthetic RNA sequences, and does not consist of any training data. The sequences used were split into five categories to represent the difference in length for each sequence. The histogram below showcases the distribution of sequences, and table 2 shows the number of sequences per split.

3.2. Baselines

For the pseudoknot-free problem, we implement a diverse set of algorithms according to the types set out previously. As model-based approaches such as Reinforcement Learning (Eastman et al., 2018; Runge et al., 2019) or Model-based Optimisation (Zhou et al., 2023b; Zhang et al., 2023) provide pre-trained models, we utilise the reported model in our experiments. The other approaches, Sampling-based (Reinharz et al., 2013), Evolutionary (Zhou et al., 2023a; Kleinkauf et al., 2015), Monte Carlo (Yang et al., 2017) and Adaptive Walk (Lorenz et al., 2011), do not require a model, and as such, we utilise the reported layout for model use, as to ensure our results are as close to reported as possible. The run-time for each algorithm was kept consistent, and each algorithm was run using 3 seeds for each computational time given. We take the rounded mean of the 3 seeds, and match this with the sequence length results. It should be noted that the computational hardware utilised in this paper is likely poorer than those used to report previous results of the algorithms, and there may be a slight variation of results over reported results due to this. We detail hardware specification in appendix A.3.

For the pseudoknot-inclusive problem, we implement two popular baseline algorithms that are widely regarded by current literature, MCTS and antaRNA (Yang et al., 2017; Kleinkauf et al., 2015). We use the same run-time for both algorithms, and each run-time reported was averaged across 3 seeds.

3.3. Metrics

The central metric for measuring the success of the pseudoknot-free tasks is the success rate, i.e., the amount of sequences that successfully fold into the same target sequence (Esmaili-Taheri et al., 2014). We report this as a percentage, however since the result is out of 100, it can also be interpreted as a raw number. To gain further insight into algorithm performance, we further utilise the number of solved sequences by length, according to the layout presented in table 1.

The pseudoknot-inclusive problem is slightly different, as we use Pseudobase++, the length of the sequences is not nearly as diverse, only between 20-136 nucleotides, thereby making length analysis redundant. We instead utilise the GC-Content, as both algorithms benchmarked can incorporate a target GC-Content, where they aim to use as high a percentage of G and C nucleotides as possible, which will increase the efficiency and stability of the overall RNA structure. By using this combination of success rate and GC-Content, we can explore both the performance of the algorithms and the quality of the sequences provided (Merleau and Smerlak, 2022).

4. Results

In this section, we present the results of the algorithms used within the benchmark. We split this section in accordance with our research questions as presented earlier, to clearly provide an answer for each one proposed. The complete set of results for the Eterna100v2 benchmark is shown in fig. 3, and table 3, and the set of results for the Pseudobase++ benchmark is shown in table 2.

4.1. Pseudoknot-Free

We find that SAMFEO performs exceptionally well for the low computation time benchmarks, even solving 47% of puzzles with just a 10 second computation time. We find that each approach clearly beats our original baseline, RNAInverse, as expected, although some older-style algorithms have improved, with MCTS outperforming the reinforcement learning (LEARNA, Meta-LEARNA, Meta-LEARNA-Adapt) approaches. We find that the evolutionary algorithm approaches are not as heavily affected by the computation time, as the other algorithms, perhaps due to other parameters such as population count having a larger affect on the results. IncaRNAtion, our sampling-based approach, was unable to be run on a lower computation time (due to the way in which it is laid out), however the performance demonstrated was not competitive with the current methods,

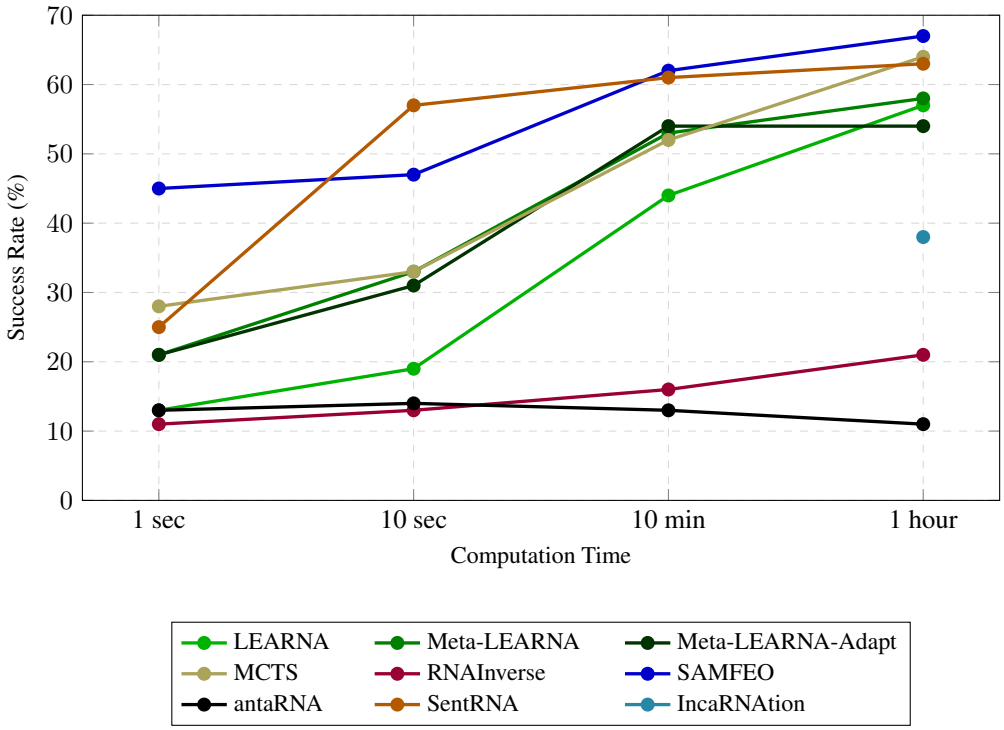

Figure 3. Graphical representation of the baseline RNA inverse design algorithms and their success rates over different computation times.

with only 38% of puzzles completed.

We find that almost every algorithm sees a drop-off in performance for all algorithms on the longest sequence category (350-400), and although the general trend is a decrease in performance as the sequence gets longer, it is not necessarily the case that the algorithms perform better on the shorter sequences. Most top performing algorithms performed very similarly across the 0-149 sequence length benchmark, only beginning to see a drop-off in performance at the 150+ mark. The collapse in performance for the longest sequence length is likely due to the increased computational time and resources required for solving them.

The drop in performance for the reinforcement learning algorithms is due to being unable to generalise to the longest sequences (350-400). This is likely due to lack of computational time, although this is contrary to results initially reported by (Runge et al., 2019).

4.2. Pseudoknot-inclusive

We find that MCTS is consistently outperformed by antaRNA across all 3 analyses run, with antaRNA consistently having a higher number of solved sequences, and a much closer target to the actual GC Content. This likely comes down to the function of the methods, as MCTS uses hamming distance as the central termination condition, it doesn't continue to optimise the sequence (i.e., increase/decrease GC content) after the hamming distance is confirmed to be zero. This is the opposite case for antaRNA, where a termination potential is used, and the maximum number of solutions, structural distance, and if GC-Content is met. Thereby antaRNA is more likely to prioritise GC-Content over matching the structure if the structure is difficult to match. This suggests that, if we were to re-run the tests with a much higher maximum computation time, MCTS may end up with a higher success rate, but the GC Content is always likely to end up lower than antaRNA.

Table 2. Results showcasing the performance of different algorithms on Pseudobase sequences and their G-C content.

Algorithm	Target GC Content	Num Solved	Actual GC Content
antaRNA	25%	170	25.54%
antaRNA	50%	174	49.93%
antaRNA	75%	173	74.70%
MCTS	25%	134	38.96%
MCTS	50%	129	39.45%
MCTS	75%	141	39.29%

5. Resources

We outline our source code and provide a docker environment to allow users to reproduce the results of

Table 3. Pseudoknot-free, Eterna100v2 1-Hour Sequence Length Results

Algorithm	Method	Sequence Length					
		0-49	50-99	100-149	150-349	350-400	Total
Reinforcement Learning	LEARNA	13	14	16	14	0	57
	Meta-LEARNA	13	15	18	11	1	58
	Meta-LEARNA-Adapt	13	15	18	8	0	54
Monte Carlo	MCTS	13	15	18	12	6	64
Adaptive Walk	RNAInverse	8	5	4	3	1	21
Evolutionary Algorithm	SAMFEO	12	17	18	14	6	67
	antaRNA	8	4	1	0	0	13
Model-based	SentRNA	13	14	17	10	9	63
Sampling-based	IncaRNAtion	11	8	10	6	3	38

our analysis. We include an example jupyter notebook or main.py file for users to gather results in the same way as we have presented the data. The code can be found in our public GitHub: https://github.com/COLA-Laboratory/RNAInvBench

6. Discussion and Future Work

We proposed RNAInvBench, an open-source framework for solving the inverse design problem with RNA secondary structures. It serves three key purposes: (a) to provide open-access to data and baseline algorithms, (b) to simplify installation and use by containerising them in a docker-container, and (c) to provide standardised metrics for measuring the performance of both pseudoknot-free and pseudoknot-inclusive algorithms.

Although we provide a framework for both pseudoknot-free and pseudoknot-inclusive algorithms, we do not discuss the constrained problem, where part of the RNA sequence is provided to limit the search space for the algorithms. Future work can extend the baseline algorithms to this problem and include both training and testing datasets.

We have conducted a large-scale study of our framework by benchmarking and evaluating the nine baseline algorithms, thereby demonstrating their performance and robustness across both pseudoknot-free and pseudoknot-inclusive challenges. This demonstrates the suitability of our framework for future research and use. Lastly, our framework is open-source, thus future contributions of new algorithms and data sources can ensure it remains up-to-date.

Acknowledgments

KL is collectively supported by the UKRI Future Leaders Fellowship (MR/X011135/1, MR/S017062/1); NSFC (62376056, 62076056); the Kan Tong Po Fellowship (KTP/R1/231017); Alan Turing Fellowship; and Amazon Research Award.

We gratefully acknowledge Mingyu Huang and Shasha Zhou for their useful comments and suggestions on earlier drafts of this manuscript.

References

Raquel Aguirre-Hernández, Holger H. Hoos, and Anne Condon. Computational RNA secondary structure design: empirical complexity and improved methods. *BMC Bioinformatics*, 8(1):34, 2007.

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, Part A):748–757, 2016. ISSN 0022-2836.

M. Andronescu, A. P. Fejes, F. Hutter, et al. A new algorithm for RNA secondary structure design. *Journal of Molecular Biology*, 336:607–624, 2004.

Mirela Andronescu, Victor Bereg, Holger H. Hoos, and Anne Condon. Rna strand: The rna secondary structure and statistical analysis database. *BMC Bioinformatics*, 9: 340, 2008.

G. Bauer and B. Suess. Engineered riboswitches as novel tools in molecular biology. *Journal of Biotechnology*, 124:4–11, 2006.

É. Bonnet, P. Rzążewski, F. Sikora, et al. Designing RNA secondary structures is hard. *Journal of Computational Biology*, 27:302–316, 2020.

Ian Brierley, Simon Pennell, and Robert J. C. Gilbert. Viral rna pseudoknots: versatile motifs in gene expression and replication. *Nature Reviews Microbiology*, 5(8):598–610, 2007. ISSN 1740-1534.

Christina Y Chan, Collin S Carmack, Dustin D Long, Anil Maliyekkel, Yong Shao, Igor B Roninson, and Yousong Ding. A structural interpretation of the effect of GC-content on efficiency of RNA interference. *BMC Bioinformatics*, 10 Suppl 1(Suppl 1):S33, 2009.

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, 01 2017. ISSN 1477-4054.

Peter Eastman, Jiayi Shi, Bharath Ramsundar, and Vijay S Pande. Solving the RNA design problem with reinforcement learning. *PLOS Computational Biology*, 14(6), 2018.

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

S. Findeiß, M. Etzel, S. Will, et al. Design of artificial riboswitches as biosensors. *Sensors*, 17:1990, 2017.

L. R. Ganser, M. L. Kelly, D. Herschlag, and H. M. Al-Hashimi. The roles of structural dynamics in the cellular functions of rnas. *Nature Reviews Molecular Cell Biology*, 20(8):474–489, August 2019.

Juan Antonio Garcia-Martin, Peter Clote, and Ivan Dotu. RNAiFOLD: A constraint programming algorithm for RNA inverse folding and molecular design. *Journal of Bioinformatics and Computational Biology*, 11:1350001, 2013.

D. P. Giedroc, C. A. Theimer, and P. L. Nixon. Structure, stability and function of RNA pseudoknots involved in stimulating ribosomal frameshifting. *Journal of Molecular Biology*, 298(2):167–185, 2000.

Ivo Hofacker, Walter Fontana, Peter Stadler, Sebastian Bonhoeffer, Manfred Tacker, and Peter Schuster. Fast folding and comparison of rna secondary structures. *Monatshefte für Chemie/Chemical Monthly*, 125:167–188, February 1994.

Christian Höner zu Siederdissen, Stephan H. Bernhart, Peter F. Stadler, and Ivo L. Hofacker. A folding algorithm for extended RNA secondary structures. *Bioinformatics*, 27(13):i129–i136, 2011.

Farren J Isaacs, Daniel J Dwyer, and James J Collins. RNA synthetic biology. *Nature Biotechnology*, 24(5):545–554, 2006.

Stefan Janssen and Robert Giegerich. The RNA shapes studio. *Bioinformatics*, 31(3):423–425, 10 2014. ISSN 1367-4803.

Robert Kleinkauf, Martin Mann, and Rolf Backofen. antaRNA: ant colony-based RNA sequence design. *Bioinformatics*, 31(19):3114–3121, 05 2015. ISSN 1367-4803.

Rohan V. Koodli, Benjamin Keep, Katherine R. Coppess, Fernando Portela, and Rhiju Das. Eternabrain: Automated RNA design through move sets and strategies from an internet-scale RNA videogame. *PLoS Comput. Biol.*, 15(6), 2019.

Rohan V. Koodli, Boris Rudolfs, Hannah K. Wayment-Steele, Eterna Structure Designers, and Rhiju Das. Redesigning the eterna100 for the vienna 2 folding engine. *bioRxiv*, August 2021.

M. Kucharík, I. L. Hofacker, P. F. Stadler, and J. Qin. Pseudoknots in RNA folding landscapes. *Bioinformatics*, 32: 187–194, 2016.

Ronny Lorenz, Stephan H. Bernhart, Christian Höner 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. ISSN 1748-7188.

Rune B. Lyngsø, James W. Anderson, Elena Sizikova, et al. Frnakenstein: Multiple target inverse RNA folding. *BMC Bioinformatics*, 13:260, 2012.

N. S. C. Merleau and M. Smerlak. arnaque: An evolutionary algorithm for inverse pseudoknotted RNA folding inspired by lévy flights. *BMC Bioinformatics*, 23:335, 2022.

Gerard Minuesa, Cristina Alsina, Juan Antonio Garcia-Martin, Juan Carlos Oliveros, and Ivan Dotu. MoiRNAiFold: a novel tool for complex in silico RNA design. *Nucleic Acids Research*, 49(9):4934–4943, 05 2021. ISSN 0305-1048.

Eric P. Nawrocki, Sarah W. Burge, Alex Bateman, Jennifer Daub, Ryann Y. Eberhardt, Sean R. Eddy, Emily W. Floden, Paul P. Gardner, Todd A. Jones, John Tate, and Robert D. Finn. Rfam 12.0: updates to the rna families database. *Nucleic Acids Research*, 43:D130–D137, 2015.

Fernando Portela. An unexpectedly effective monte carlo technique for the rna inverse folding problem. *bioRxiv*, 2018.

Vladimir Reinharz, Yann Ponty, and Jérôme Waldispühl. A weighted sampling algorithm for the design of RNA sequences with targeted secondary structure and nucleotide distribution. *Bioinform.*, 29(13):308–315, 2013.

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

Frederic Runge, Karim Farid, Jörg K. H. Franke, and Frank Hutter. RnaBench: A comprehensive library for In Silico RNA modelling. *bioRxiv*, 2024.

Michael Schnall-Levin, Leonid Chindelevitch, and Bonnie Berger. Inverting the Viterbi algorithm: An abstract framework for structure design. In *International Conference on Machine Learning*, pages 904–911, 2008.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

R. Schwab, S. Ossowski, M. Riester, et al. Highly specific gene silencing by artificial micrornas in arabidopsis. *Plant Cell*, 18:1121–1133, 2006.

J. Shi, R. Das, V. S. Pande, et al. Sentrna: Improving computational RNA design by incorporating a prior of human design strategies. *arXiv preprint*, 2018.

David W Staple and Samuel E Butcher. Pseudoknots: Rna structures with diverse functions. *PLoS Biol*, 3(6):e213, Jun 2005.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, 2018. ISBN 0262039249.

Akito Taneda. MODENA: A multi-objective RNA inverse folding. *Advances and Applications in Bioinformatics and Chemistry : AABC*, 4:1–12, 2011.

Michela Taufer, Antonio Licon, Ricardo Araiza, Darío Mireles, Frank H. van Batenburg, Alexander P. Gultyaev, and M. Y. Leung. PseudoBase++: an extension of PseudoBase for easy searching, formatting and visualization of pseudoknots. *Nucleic Acids Research*, 37:D127–D135, 2009.

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

Frans H. van Batenburg, Alexander P. Gultyaev, Cornelis W. A. Pleij, Joey Ng, and Jeroen Oliehoek. PseudoBase: a database with RNA pseudoknots. *Nucleic Acids Research*, 28(1):201–204, 2000.

D. Wang and A. Farhana. *Biochemistry, RNA Structure*. StatPearls Publishing, Treasure Island (FL), 2024.

Xiaodong Yang, Kazuki Yoshizoe, Akito Taneda, et al. Rna inverse folding using monte carlo tree search. *BMC Bioinformatics*, 18(1):468, 2017.

Hao Zhang, Ling Zhang, Anran Lin, and et al. Algorithm for optimized mRNA design improves stability and immunogenicity. *Nature*, 621:396–403, 2023.

Tianshuo Zhou, Ning Dai, Sizhen Li, Max Ward, David H. Mathews, and Liang Huang. RNA design via structure-aware multifrontier ensemble optimization. *Bioinformatics*, 39(Supplement-1):563–571, 2023a.

Tianyin Zhou, Nana Dai, Shuang Li, Michelle Ward, David H Mathews, and Leshang Huang. RNA design via structure-aware multifrontier ensemble optimization. *Bioinformatics*, 39(39 Suppl 1):i563–i571, 2023b.

A. Appendix

A.1. Technical Details and Code Availability

Data For the pseudoknot-free analysis, we utilised Eterna100v2 as our benchmark data (Koodli et al., 2021), which can be found at the EternaGame github: https://github.com/eternagame/eterna100-benchmarking

For the pseudoknot-inclusive analysis, we utilised Pseudobase++ as our benchmark data (Taufer et al., 2009), which can be found at the Pseudobase++2.0 website: https://rnavlab.utep.edu/database

Code Availability All of our code is open-source and available for download and use through the github page: `https://github.com/COLA-Laboratory/RNAInvBench`

A.2. Baselines for Inverse RNA Design

- **RNAInverse** (Lorenz et al., 2011; Hofacker et al., 1994; Andronescu et al., 2004), one of the first inverse folding algorithms, initializes the sequence randomly and then uses a simple adaptive walk to randomly sample better sequences.
- **antaRNA** (Kleinkauf et al., 2015) employs ant-colony optimization where initial sequences are generated through a weighted random search, and the quality of these sequences guides the adjustment of search weights, iteratively refining the sequences towards the target structure.
- **MCTS** (Yang et al., 2017) employs monte-carlo tree search, where each node in the tree corresponds to an assigned nucleotide in the sequence. The initial sequence is randomly generated, and MCTS guides iterative mutation towards the target structure.
- **SentRNA** (Shi et al., 2018) is a design agent consisting of a fully-connected neural network trained end-to-end using human-designed RNA sequences.
- **LEARNA** (Runge et al., 2019) employs a deep neural network with an embedding layer for the input state for its policy function, and uses Proximal Policy Optimisation (PPO) to train the policy through online interaction.
- **Meta-LEARNA** (Runge et al., 2019) employs a meta-learning approach that views each RNA structure as a task, and transfers knowledge across each task. As each structure has a different MDP (Markov Decision Process), PPO is used asynchronously to train a single policy network across all the tasks. After training, the parameters are fixed and applied to a new sequence (modelled by a decision process).
- **Meta-LEARNA-Adapt** (Runge et al., 2019) combines Meta-LEARNA and LEARNA. It uses Meta-LEARNA to obtain the initial policy parameters, and LEARNA is used to further adapt the policy parameters.
- **IncaRNAtion** (Reinharz et al., 2013) is a successor of RNA-ensign that can specifically design sequences with a specified GC content using a GC-weighted Boltzmann ensemble and stochastic backtracking.
- **SAMFEO** (Zhou et al., 2023a) leverages both structure-level and ensemble-level information to guide iterative searches, optimizing objectives through cycles of sampling, mutation, and updating to produce a large number of successfully designed RNA sequences.

A.3. Hardware Details

All computations were completed using an 24-Core Intel(R) Xeon(R) Gold 6248R CPU (3.00GHz), with 252GB of RAM.

A.4. Benchmark data specification

Table 4. Benchmark data details, including the task, the range of sequence lengths, and the number of sequences within the dataset.

Dataset	Pseudoknot Type	Task	Count	Range
Eterna100-v1	Pseudoknot-free	Test	100	11-399
Eterna100-v2	Pseudoknot-free	Test	100	11-399
Pseudobase++-Test	Pseudoknot-inclusive	Test	251	20-136
Pseudobase++-Train	Pseudoknot-inclusive	Train	16	24-134
Rfam-Learn-Test	Pseudoknot-free	Test	100	50-446
Rfam-Learn-Train	Pseudoknot-free	Train	65000	50-450
Rfam-Test	Pseudoknot-free	Test	63	35-273
Rfam-Taneda	Pseudoknot-free	Test	29	54-451
RNA-Strand	Pseudoknot-free	Test	50	20-98

A.5. Dataset Pre-Processing

The RNA families used (Rfam, Strand, and Pseudobase) are highly redundant. Rfam, which compiles data from Wikipedia, taxonomy, and ontology resources and is manually curated (Nawrocki et al., 2015). Strand is compiled of a wide collection of secondary structures collated from public databases (Andronescu et al., 2008). Pseudobase consists of a database of pseudoknot secondary structures developed from Leiden University through crystallography, NMR, mutational experiments and sequence comparisons (Taufer et al., 2009). Therefore, pre-processing is required to ensure structural and sequential diversity. Below, we provide the pre-processing steps to maintain this.

- **RNA-Strand and Rfam:** For each family, (Kleinkauf et al., 2015) obtained the seed alignment, and removed all alignments with less than 21 entries. From here, the sequence and structure were aligned to ensure that the structural elements were correctly mapped to the nucleotide sequence. This keeps the sequence and structure length consistent, and ensures they are accurately aligned.
- **Pseudobase++:** Cases where non-canonical base pairs were identified and removed by (Kleinkauf et al., 2015). The remaining pseudoknots fall into four categories, simple hairpin, complex hairpin, kissing hairpin and bulge hairpin pseudoknots. Pseudoknots of other categories were excluded.
- **Eterna100v1/v2:** This dataset consists of 100 synthetic RNA sequences. For Eterna100v1, all 100 synthetic RNA sequences were created by (Anderson-Lee et al., 2016), considering a diverse range of sequence and mean stem length, symmetry and specific difficult-to-design motifs. For Eterna100v2, the 19 sequences that were unsolvable were replaced with synthetically created sequences by the Eterna community, which were then validated and selected by (Koodli et al., 2021).