Enforcing Constraints in RNA Secondary Structure Predictions: A Post-Processing Framework Based on the Assignment Problem

Geewon Suh [*1,2] **Gyeongjo Hwang** [*1] **Seokjun Kang** [1] **Doojin Baek** [3] **Mingeun Kang** [1,2]

Abstract

RNA properties, such as function and stability, are intricately tied to their two-dimensional conformations. This has spurred the development of computational models for predicting the RNA secondary structures, leveraging dynamic programming or machine learning (ML) techniques. These structures are governed by specific rules; for example, only Watson-Crick and Wobble pairs are allowed, and sequences must not form sharp bends. Recent efforts introduced a systematic approach to post-process the predictions made by ML algorithms, aiming to modify them to respect the constraints. However, we still observe instances violating the requirements, significantly reducing biological relevance. To address this challenge, we present a novel post-processing framework for ML-based predictions on RNA secondary structures, inspired by the assignment problem in integer linear programming. Our algorithm offers a theoretical guarantee, ensuring that the resulting predictions adhere to the fundamental constraints of RNAs. Empirical evidence supports the efficacy of our approach, demonstrating improved predictive performance with no constraint violation, while requiring less running time.

1. Introduction

A nucleotide serves as a building block for deoxyribonucleic acids (DNAs) and ribonucleic acids (RNAs), comprising a nitrogenous base, a pentose sugar, and a phosphate group. Due to the presence of hydrogen donors and acceptors in the nitrogenous base, a nucleotide from one DNA strand can engage in inter-molecular hydrogen bonding with a counterpart from a different strand. This fundamental principle of base pairing gives rise to the double-stranded helical structure of DNAs (Watson & Crick, 1953). In contrast, within a single-stranded RNA molecule, base pairing facilitates the formation of intra-molecular hydrogen bonding, resulting in complex folding patterns with various local structures (Kim et al., 1974; Noller, 1984; Tinoco Jr & Bustamante, 1999). Investigating these structures not only sheds light on the principles governing RNA functions and mechanisms, but also holds promise for advances in biotechnology.

Over the last few decades, numerous achievements have been made in methodologies aimed at observing RNA secondary structures. Technologies such as X-ray crystallography, nuclear magnetic resonance spectroscopy, and cryo-electron microscopy have been prominent strategies for visualizing molecular structures (Neidle & Sanderson, 2021). While these can yield structures of high resolution, it is both cost-prohibitive and labor-intensive to collect samples (Kappel et al., 2020). An alternative is probing-based methods that give us enough number of samples, yet the quality of the data is compromised (Spitale & Incarnato, 2022).

Independently, researchers have also developed computational algorithms to predict RNA secondary structures. In the early 1980s, algorithms finding the most thermodynamically stable structure were proposed (Nussinov & Jacobson, 1980; Zuker & Stiegler, 1981). Since this problem is NP-complete for a general class of RNA structures (Lyngsø & Pedersen, 2000; Bonnet et al., 2017), they usually narrow the optimization space and employ dynamic programming (DP) to efficiently explore the reduced space. Later on, improvements have been made to achieve a better computational complexity (Bringmann et al., 2016; Huang et al., 2019), or to expand the search space including pseudoknotted-structures (Rivas & Eddy, 1998; Akutsu, 2000).

Recently, machine learning (ML) models have been brought into the spotlight with an accumulation of experimentally validated samples (Sussman et al., 1998; Griffiths-Jones et al., 2003; Andronescu et al., 2008). This data-driven strategy makes a model free from handcrafted rules, but their outputs typically do not satisfy the constraints of the RNAs. For example, RNAs usually permit only Watson-Crick and Wobble pairs, and sharp bends are geometrically impossi-

[*]Equal contribution [1]Spidercore Inc., Daejeon, South Korea [2]Department of Electrical Engineering, KAIST, Daejeon, South Korea [3]School of Computing, KAIST, Daejeon, South Korea. Correspondence to: Mingeun Kang <minkang@spidercore.io>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

ble, whereas ML models do not consider such restrictions. Hence, one recent work introduced a post-processing approach to modify the model predictions to comply with the constraints (Chen et al., 2020). This two-stage scheme has now become a default for recent ML-based models (Fu et al., 2022; Chen & Chan, 2023). However, we find that the post-processing algorithm does not guarantee perfect adherence to the requirements, which can severely limit the practicality of current ML algorithms. We provide an explicit toy example where the existing post-processing fails (see Remark 3.1) and show empirical evidence of constraint violations in real-world datasets (see Section 5.2). To address such challenges, we propose a novel post-processing optimization problem and its solver, inspired by the assignment problem in combinatorial optimization.

Contributions. Our contribution lies in the development of a novel post-processing framework for the RNA structure prediction. Intriguingly, we discover that the relevant optimization can be cast as a linear sum assignment problem (LSAP), a type of integer linear programming (ILP) in combinatorial optimization (see Section 4.1). Using a known solver, the Hungarian algorithm, we find that the solutions to the LSAP perfectly adhere to the requirements of RNAs. Lastly, our empirical study shows that the proposed algorithm takes much less time than the existing post-processing method, while maintaining competitive predictive performances in the RNA secondary structure prediction.

2. Related Works

The most widely used computational approach for the RNA secondary structure prediction is based on the principle of minimum free energy (Nussinov & Jacobson, 1980; Zuker & Stiegler, 1981), where the free energy contribution of each local substructure has been updated through experiments (Schroeder & Turner, 2009; Turner & Mathews, 2010). There are several implementations that improved upon the original algorithm, such as ViennaRNA (Hofacker, 2003; 2009), UNAFold (Zuker, 2003; Markham & Zuker, 2008), and RNAstructure (Reuter & Mathews, 2010; Bellaousov et al., 2013). Unfortunately, many of these algorithms suffer from incapability of producing a particular class of output: structures including pseudoknots. To mitigate this, a few methods have been proposed to include a subset of pseudoknotted-structures while compromising time complexity (Rivas & Eddy, 1998; Akutsu, 2000; Bellaousov & Mathews, 2010; Sato et al., 2011).

More recent learning-based models can be categorized into two: (1) a combined algorithm of DP and ML, and (2) a pure ML algorithm. For the combined method, SimFold tries to estimate energy parameters required for the DP algorithm in a data-driven way (Andronescu et al., 2007; 2010). MXfold takes a similar strategy, where the energy estimation model is replaced by a support vector machine (Akiyama et al., 2018). CDPfold and MXfold2 are additional variants of this kind with deep neural networks (Zhang et al., 2019; Sato et al., 2021).

The second category takes a more direct approach, where the ML models output structure predictions without a follow-up DP algorithm. For example, SPOT-RNA (Singh et al., 2019) uses Bidirectional Long Short-Term Memory (Hochreiter & Schmidhuber, 1997; Schuster & Paliwal, 1997) and ResNet (He et al., 2016) to predict the base pairing probabilities between every pair of nucleotide. E2Efold (Chen et al., 2020), UFold (Fu et al., 2022) and REDfold(Chen & Chan, 2023) also predict the base pairing probabilities with different model architectures and input data preprocessing. Specifically, E2Efold employs a transformer (Vaswani et al., 2017) that takes an RNA sequence as input, while UFold and REDfold both employ U-net architectures (Ronneberger et al., 2015) that accept a collection of matrices as input.

It is important to note that making independent predictions for all pairs of nucleotides can result in a secondary structure that violates the RNA constraints. Hence, E2Efold proposes a post-processing framework to enforce these restrictions through constrained optimization. CNNfold (Saman Booy et al., 2022) proposes yet another method that serves a similar purpose, inspired by the classical Blossom algorithm (Galil, 1986). However, we find that their optimizations do not guarantee a solution that fully satisfies the constraints. Indeed, empirical evidence shows that the predictions still violate the constraints despite the post-processing.

Our framework that tackles the aforementioned challenges is based on LSAP. Originating from Monge back in 1780s, LSAP has been studied together with transportation, bipartite matching, and traveling salesman problems in the context of operations research (Schrijver, 1998; Kuhn, 2012). A breakthrough in LSAP came with the simplex algorithm. Despite its exponential time complexity, the simplex algorithm can be employed for LSAP, as it automatically produces integer solutions (Dantzig, 1951). The first polynomial time solver of LSAP is the Hungarian algorithm (Kuhn, 1955; Munkres, 1957), named after pioneering works of two Hungarians (Konig, 1931; Egerváry, 1931). Since then, several variants have been proposed to improve the performance of the original method (Tomizawa, 1971; Edmonds & Karp, 1972), with Jonker–Volgenant algorithm (Jonker & Volgenant, 1988) being the most widely used.

3. Problem Setup

3.1. RNA Secondary Structure Prediction

Let $\boldsymbol{x}$ be an RNA with a nucleotide sequence $\boldsymbol{x} := (x_1, x_2, \ldots, x_L)$, where each $x_i \in \{\texttt{A}, \texttt{G}, \texttt{C}, \texttt{U}\}$ corresponds to one of the four building blocks: adenine, guanine,

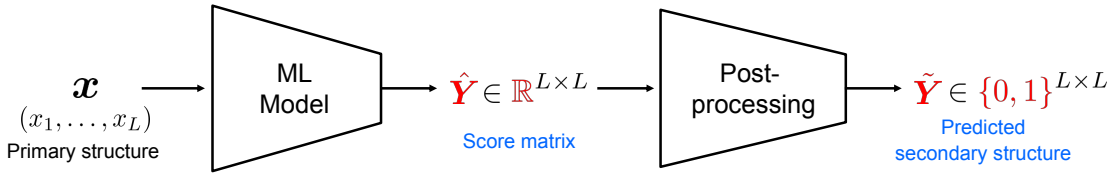

Figure 1. A graphical overview of the two-stage architecture for the ML-based RNA secondary structure prediction. An input RNA $\boldsymbol{x}$ of length L is sequentially processed into $\hat{\boldsymbol{Y}}$ and $\tilde{\boldsymbol{Y}}$, where $\hat{\boldsymbol{Y}}$ is the direct output of the model and $\tilde{\boldsymbol{Y}}$ is the final prediction.

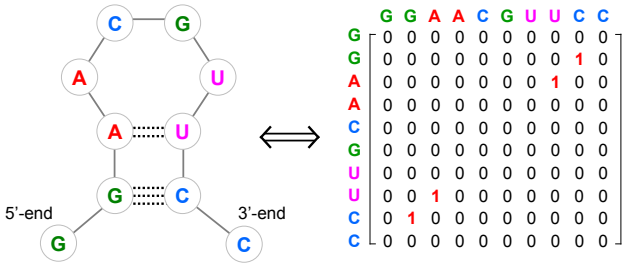

Figure 2. A matrix representation of RNA secondary structure. An RNA `GGAACGUUCC` with a hairpin structure (left) is equivalent to the matrix on the right. The two hydrogen bonds `G-C` and `A-U` are marked with dotted lines (left) and red ones (right).

cytosine, and uracil. Then the corresponding secondary structure of $\boldsymbol{x}$ can be represented as a binary symmetric matrix $\boldsymbol{Y} \in \{0,1\}^{L \times L}$, where each entry $Y_{ij} = 1$ if and only if the bases x_i and x_j are paired (see Figure 2). The goal of RNA secondary structure prediction is to construct such a matrix containing base pairing information from an input RNA.

The structure of RNA molecules is governed by physical laws. Thus, there are certain "hard" constraints that the secondary structure matrix $\boldsymbol{Y}$ should satisfy:

(C1) $\boldsymbol{Y}$ is a binary and symmetric matrix; $Y_{ij} \in \{0,1\}$ and $\boldsymbol{Y} = \boldsymbol{Y}^T$,

(C2) Only the Watson-Crick base pairs (`A-U`, `G-C`) and Wobble pairs (`G-U`) are allowed; $Y_{ij} = 0$ if $x_i x_j \notin \mathcal{B} := \{$`AU`, `UA`, `GC`, `CG`, `GU`, `UG`$\}$,

(C3) No sharp loops are allowed; $Y_{ij} = 0$ if $|i - j| < 4$,

(C4) There is no overlap of pairs, i.e., each row or column contains at most one 1's; $\sum_{j=1}^{L} Y_{ij} \leq 1 \; \forall i$, or $\boldsymbol{Y}\mathbf{1} \leq \mathbf{1}$ where $\mathbf{1}$ is an $L \times 1$ matrix filled with ones.

3.2. Model Architecture

An ML-based approach to RNA secondary structure builds upon a well-known two-stage scheme (Chen et al., 2020; Fu et al., 2022; Chen & Chan, 2023). Stage 1 outputs a predicted score matrix for the input. Stage 2 outputs a finely-tuned matrix that respects the constraints introduced above, given the score matrix computed from Stage 1. A graphical overview of the entire system is illustrated in Figure 1.

Stage 1 (Deep Score Network) The first stage of the architecture is a deep learning model parameterized by θ, whose output $\hat{\boldsymbol{Y}} = \hat{\boldsymbol{Y}}(\boldsymbol{x})$ is an $L \times L$ symmetric matrix for an input RNA $\boldsymbol{x}$ of length L. Each entry of the output, $\hat{Y}_{ij}$, represents the predicted contact score between the nucleotides x_i and x_j in the input. In this paper, we employ the deep neural networks in the prior works, a transformer and a U-net with Dense-net components, from E2Efold (Chen et al., 2020) and REDfold (Chen & Chan, 2023), respectively.

Stage 2 (Post-processing) The second stage bears our key idea. Since the score matrix computed above is not guaranteed to obey the hard constraints, an additional procedure is required. This post-processing can be formulated as a constrained optimization problem, for which a solving algorithm based on a primal-dual method is proposed in E2Efold (Chen et al., 2020). Nonetheless, in certain prediction instances, constraints are violated due to the binary thresholding with an offset term. Also, the proposed gradient-descent solver may not converge to the global optimum, as the objective function is not convex-concave.

To address these challenges, we propose an alternative aimed at achieving a precise solution to the optimization problem while enforcing the constraints in RNA secondary structure prediction. The idea is to transform the optimization into an assignment problem, where it is possible to find the exact optimal solution in polynomial time. See Section 4 for more details.

Remark 3.1 (A toy example)**.** We provide a toy example where the existing post-processing algorithms fail to produce secondary structure predictions that respect the constraints (C1)-(C4). In Figure 3 (top), an intermediate output $\hat{\boldsymbol{Y}}_T$ of the E2Efold post-processing algorithm is shown for an input RNA sequence of `AACCGUU`. Here, T indicates the number of proximal gradient descent steps taken in the algorithm, and "PP" is an abbreviate for post-processing. Since E2Efold then takes the thresholding operation to yield the final output, the algorithm has the potential to violate (C4): the overlapping pair constraint. On the middle left of

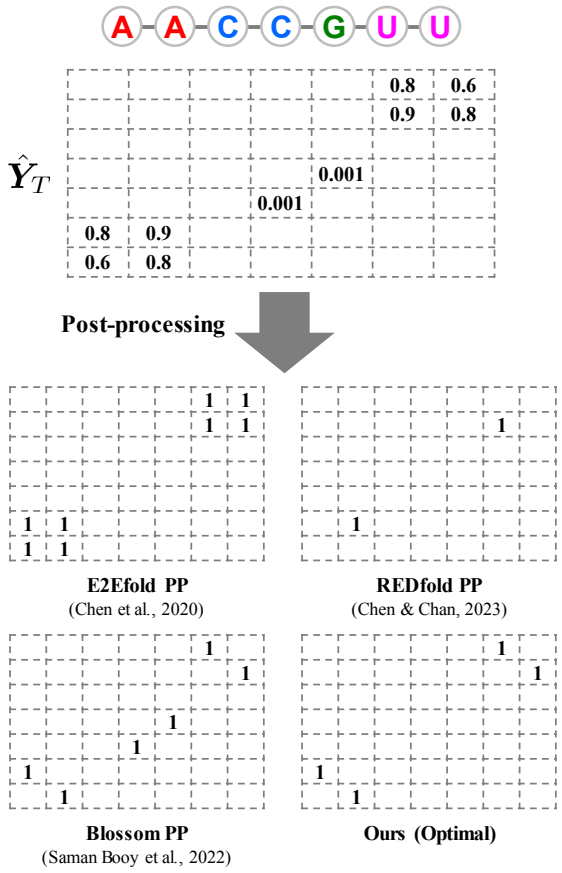

Figure 3. A toy example of RNA sequence and its score matrix. (Top) Output of proximal gradient descent algorithm from E2Efold is shown. (Bottom) Outputs obtained from respective post-processing algorithms are shown.

Figure 3, we can see that there are multiple $\mathbf{1}$'s in a few rows and columns. To avoid such an invalid structure, REDfold post-processing further modifies $\hat{\boldsymbol{Y}}_T$ by taking row-wise and column-wise arg max (middle right). However, this heuristics may erase the base pair that is part of the optimal solution. CNNfold, on the other hand, can find the optimal entries, yet it lacks a filtering operation that can eliminate insignificant values (bottom left). In contrast, the proposed post-processing is designed to identify a maximum entry-sum solution within the provided score matrix while adhering to the constraints of RNAs, thus mitigating the aforementioned issues (bottom right).

4. Post-process Optimization

The goal of post-processing step is to derive a secondary structure matrix $\tilde{\boldsymbol{Y}} := \tilde{\boldsymbol{Y}}(\boldsymbol{x})$ that is slightly different from the initial model prediction $\hat{\boldsymbol{Y}} := \hat{\boldsymbol{Y}}(\boldsymbol{x})$, while respecting the constraints (C1)-(C4). Since a neural network model usually produces a real-valued output $\hat{\boldsymbol{Y}} \in \mathbb{R}^{L\times L}$, we need to come up with a way to convert this into a discrete-valued secondary structure matrix $\tilde{\boldsymbol{Y}}$. Such desiderata can be written as a constrained optimization of the following form:

$$\begin{aligned} \max_{\tilde{\boldsymbol{Y}}} \quad & \langle \hat{\boldsymbol{Y}} - s, \tilde{\boldsymbol{Y}} \rangle \\ \text{s.t.} \quad & \tilde{Y}_{ij} \in \{0,1\}, \quad \tilde{\boldsymbol{Y}} = \tilde{\boldsymbol{Y}}^T, \quad \tilde{\boldsymbol{Y}}\mathbf{1} \le \mathbf{1}, \\ & \tilde{Y}_{ij} = 0 \quad \text{if } x_i x_j \notin \mathcal{B} \text{ or } |i-j| < 4. \end{aligned} \tag{1}$$

where $\mathcal{B}$ and $\mathbf{1}$ are as defined in (C2) and (C4), respectively. $\langle \cdot, \cdot \rangle$ is a matrix inner product and $s \in \mathbb{R}$ is an offset term that weighs the entries of importance. In short, the optimization seeks a matrix satisfying (C1)-(C4) whose similarity against the original prediction $\hat{\boldsymbol{Y}}$ captured by the matrix inner product is the highest.

We see that this is a binary ILP. In general, ILP problem is NP-complete, meaning that the problem cannot be solved in a polynomial running time. To deal with such intractable problems, one natural way is to ignore the binary value constraint: allowing $0 \le \tilde{Y}_{ij} \le 1$. Solving the relaxed Linear Programming (LP) and rounding the solution gives a tractable approximation to the ILP solution, even though it may be sub-optimal or infeasible in the original ILP problem. The algorithm proposed in E2Efold takes such a strategy, where the final prediction matrix is obtained by solving a quadratic programming relaxation by simplifying the search space of the LP relaxation (Chen et al., 2020).

In this work, we find that the optimization problem (1) is in a special case of ILP class such that exact solution is tractable. To see this, we present a technique to transform the original optimization problem into a non-constrained 'assignment problem' (or 'maximum weight bipartite matching'), which is well-known to be solvable in a polynomial time.

4.1. Assignment Problem

The assignment problem is a problem of finding the minimum overall cost on the worker-job assignment. Suppose that there are L workers and L jobs. Any worker i can be assigned to perform job j, incurring cost C_{ij}. The objective is to assign every worker to every other job at the minimum overall cost. i.e., we want to find a bijective assignment whose overall cost is minimal. One can state this in a matrix form as follows:

$$\begin{aligned} \min_{\boldsymbol{Z}} \quad & \langle \boldsymbol{C}, \boldsymbol{Z} \rangle \\ \text{s.t.} \quad & \boldsymbol{Z}_{ij} \in \{0,1\}, \quad \boldsymbol{Z}\mathbf{1} = \mathbf{1}, \quad \boldsymbol{Z}^T\mathbf{1} = \mathbf{1}, \end{aligned} \tag{2}$$

where $\boldsymbol{C} = \{C_{ij}\}_{[L]\times[L]}$ is a cost matrix and $\boldsymbol{Z}$ is an assignment matrix.

We show that the post-process optimization (1) can be transformed into an assignment problem (2) with a cost matrix

carefully designed from the score map $\hat{\boldsymbol{Y}}$.

Theorem 4.1 (Equivalence)**.** *Let $\boldsymbol{x}$ be a sequence of nucleotide with length L, $\hat{\boldsymbol{Y}} \in \mathbb{R}^{L\times L}$ be a symmetric matrix that represents initial model prediction, and s be an offset term that weighs the entries of importance. Given $\boldsymbol{x}, \hat{\boldsymbol{Y}}, s$, let $\boldsymbol{M} := \boldsymbol{M}(\boldsymbol{x}, \hat{\boldsymbol{Y}}, s)$ be a binary matrix whose (i,j)-th entry is defined as $\boldsymbol{M}_{ij} = 1$ if $x_i x_j \in \{\texttt{AU}, \texttt{GU}, \texttt{GC}\}$, $|i-j| \geq 4$, and $\hat{\boldsymbol{Y}}_{ij} > s$, 0 otherwise. Let $\boldsymbol{C}(\hat{\boldsymbol{Y}}) := -2(\hat{\boldsymbol{Y}} - s) \odot \boldsymbol{M}$, where $\odot$ is an element-wise product operation. Then,*

- *For an optimal solution $\tilde{\boldsymbol{Y}}^*$ of problem (1), define an $L \times L$ binary matrix $\boldsymbol{Z}^*$ as $\boldsymbol{Z}^*_{ij} = 1$ if and only if $\tilde{\boldsymbol{Y}}^*_{ij} = 1$, $x_i x_j \in \{\texttt{AU}, \texttt{GU}, \texttt{GC}\}$ and $\hat{\boldsymbol{Y}}_{ij} > s$. Then, $\boldsymbol{Z}^*$ is an optimal solution of problem (2) with $\boldsymbol{C} = \boldsymbol{C}(\hat{\boldsymbol{Y}})$.*
- *For an optimal solution $\boldsymbol{Z}^*$ with $\boldsymbol{C} = \boldsymbol{C}(\hat{\boldsymbol{Y}})$ of problem (2), $\tilde{\boldsymbol{Y}}^* := \boldsymbol{Z}^* \odot \boldsymbol{M} + (\boldsymbol{Z}^* \odot \boldsymbol{M})^T$ is an optimal solution of problem (1).*

The theorem states that any optimal solution of each optimization can be reduced to an optimal solution of the other problem. In this context, we argue that the two problems (1) and (2) are equivalent.

4.2. Proof Outline

According to the base-pairing properties of nucleotides, there are only three types of pairings: $\{\texttt{AU}, \texttt{GU}, \texttt{GC}\}$. In this point of view, the interested problem can be expressed as an assignment problem from bases $\{\texttt{A}, \texttt{G}\}$ to $\{\texttt{U}, \texttt{C}\}$, where each entry of the assignment matrix equals 1 if and only if the two corresponding bases form a pair.

It is reasonable that an entry $\tilde{Y}^*_{ij}$ of optimal solution in problem (1) is equal to 0 when $\hat{Y}_{ij} < s$. For a given $\tilde{\boldsymbol{Y}}$ satisfying the secondary structure constraints, let $\boldsymbol{Z}$ be an $L \times L$ binary matrix satisfying $Z_{ij} = 1$ if $\tilde{Y}_{ij} = 1$ and $x_i x_j \in \{\texttt{AU}, \texttt{GU}, \texttt{GC}\}$, 0 otherwise. Then $\tilde{\boldsymbol{Y}} = \boldsymbol{Z} + \boldsymbol{Z}^T$ and $\langle \hat{\boldsymbol{Y}} - s, \tilde{\boldsymbol{Y}} \rangle = 2\langle \hat{\boldsymbol{Y}} - s, \boldsymbol{Z} \rangle$ due to the symmetry of $\hat{\boldsymbol{Y}}$. We can observe that $Z_{ij} = 1$ only if $x_i \in \{\texttt{A}, \texttt{G}\}$ and $x_j \in \{\texttt{U}, \texttt{C}\}$, while $Z^T_{ij} = 1$ only if $x_i \in \{\texttt{U}, \texttt{C}\}$ and $x_j \in \{\texttt{A}, \texttt{G}\}$. Thus the inequality constraint $\tilde{\boldsymbol{Y}}\mathbf{1} \leq \mathbf{1}$ is equivalent to $\boldsymbol{Z}\mathbf{1} \leq \mathbf{1}$ and $\boldsymbol{Z}^T\mathbf{1} \leq \mathbf{1}$. So we can re-write (1) as follows:

$$\begin{aligned} \max_{\boldsymbol{Z}} \quad & 2\langle \hat{\boldsymbol{Y}} - s, \boldsymbol{Z} \rangle \\ \text{s.t.} \quad & Z_{ij} \in \{0,1\}, \quad \boldsymbol{Z}\mathbf{1} \leq \mathbf{1}, \quad \boldsymbol{Z}^T\mathbf{1} \leq \mathbf{1}, \\ & Z_{ij} = 0 \quad \text{if } x_i x_j \notin \{\texttt{AU}, \texttt{GU}, \texttt{GC}\} \\ & \qquad\qquad \text{or } |i-j| < 4 \text{ or } \hat{Y}_{ij} < s. \end{aligned} \tag{3}$$

Now define a mask matrix $\boldsymbol{M} := \boldsymbol{M}(\boldsymbol{x}, \hat{\boldsymbol{Y}}, s)$ to be $M_{ij} := 1$ if $x_i x_j \in \{\texttt{AU}, \texttt{GU}, \texttt{GC}\}$ and $|i-j| \geq 4$ and $\hat{Y}_{ij} > s$, 0 otherwise. Considering the corresponding constraints as an inner product between the objective and the mask matrix, the problem (3) can be re-written as:

$$\begin{aligned} \max_{\boldsymbol{Z}} \quad & 2\langle (\hat{\boldsymbol{Y}} - s) \odot \boldsymbol{M}, \boldsymbol{Z} \rangle \\ \text{s.t.} \quad & Z_{ij} \in \{0,1\}, \quad \boldsymbol{Z}\mathbf{1} \leq \mathbf{1}, \quad \boldsymbol{Z}^T\mathbf{1} \leq \mathbf{1}. \end{aligned} \tag{4}$$

Note that if the cost function is non-negative, the optimal solution of (4) can be extended to satisfy the boundary condition of inequalities. Therefore, we can formulate the optimization problem as:

$$\begin{aligned} \max_{\boldsymbol{Z}} \quad & 2\langle (\hat{\boldsymbol{Y}} - s) \odot \boldsymbol{M}, \boldsymbol{Z} \rangle \\ \text{s.t.} \quad & Z_{ij} \in \{0,1\}, \quad \boldsymbol{Z}\mathbf{1} = \mathbf{1}, \quad \boldsymbol{Z}^T\mathbf{1} = \mathbf{1}, \end{aligned} \tag{5}$$

which is equivalent to the assignment problem (2) with the cost matrix $-2(\hat{\boldsymbol{Y}} - s) \odot \boldsymbol{M}$ by taking negative of the objective. We leave a full proof in Appendix A.

4.3. Proposed Algorithm

Algorithm 1 Proposed post-processing

Input: RNA sequence $\boldsymbol{x}$ of length L, score matrix $\hat{\boldsymbol{Y}}$, offset parameter s
$\boldsymbol{M} \longleftarrow L \times L$ matrix filled with zeros
for $(i,j) \in [L] \times [L]$ **do**
  if $x_i x_j \in \{\texttt{AU}, \texttt{GU}, \texttt{GC}\}$ **and** $|i-j| \geq 4$ **and** $\hat{\boldsymbol{Y}}_{ij} > s$ **then**
    $M_{ij} \longleftarrow 1$
  end if
end for
$\boldsymbol{C} \longleftarrow -2(\hat{\boldsymbol{Y}} - s) \odot \boldsymbol{M}$
$\boldsymbol{Z}^* \longleftarrow$ Output of Zonker-Volgenant algorithm with an input $\boldsymbol{C}$
$\tilde{\boldsymbol{Y}}^* \longleftarrow \boldsymbol{Z}^* \odot \boldsymbol{M} + (\boldsymbol{Z}^* \odot \boldsymbol{M})^T$
Output: Secondary structure matrix $\tilde{\boldsymbol{Y}}^*$

A naive way to solve the $L \times L$ assignment problem is a brute-force algorithm, which takes $O(L!)$ to check all the permutations. Fortunately, there are some algorithms for solving the problem in polynomial time. The Hungarian algorithm by (Kuhn, 1955; Munkres, 1957), also known as the Kuhn-Munkres algorithm, is a well-known polynomial time solver for the assignment problem. It turns out that the algorithm can be modified to achieve an $O(L^3)$ running time (Tomizawa, 1971; Edmonds & Karp, 1972; Jonker & Volgenant, 1988). The overall post-processing method is presented in Algorithm 1, while keeping the description of Zonker-Volgenant algorithm in Appendix B.

Remark 4.2 (Hyperparameter)**.** The only hyperparameter that is added to the overall model architecture is s. We empirically find that the tuning cost for s is marginal, because we do not need to post-process the model outputs during model training. The only part where increase in complexity comes from is in the evaluation phase, and a few runs of post-processing suffice to find good enough values of s.

5. Experiments

We present experimental results on two real-world datasets. The most heavily used RNA secondary structure prediction algorithms are considered as baselines. The result shows superior performance of ours compared with the baselines in terms of adherence to constraints, predictive accuracy, and execution time. All the experiments were conducted on a system running Ubuntu 18.04.5 LTS, equipped with Intel(R) Xeon(R) Silver 4210R CPU @ 2.40GHz, Nvidia TITAN RTX GPU, and 256 GB of RAM.

5.1. Dataset & Metric

Table 1. Dataset Statistics.

	RNAStrAlign	**ncRNA**
Train	15,639	16,896
Validation	2,390	8,448
Test	2,406	8,449
Total	**20,435**	**33,793**

We consider RNAStrAlign (Tan et al., 2017) and a collection of non-coding RNAs, ncRNA, from the Rfam database (Griffiths-Jones et al., 2003; Kalvari et al., 2021) as benchmark datasets. To ensure data quality, we remove RNA sequences with incorrect base pair information, and eliminate redundant sequences using CD-HIT-EST (Fu et al., 2012). Additionally, we filter out minority examples whose lengths are longer than 600 and 720 in RNAStrAlign and ncRNA, respectively, following the experimental design in the literature. Detailed statistics of the final datasets can be found in Table 1. We split the data into train, validation and test sets at an 8:1:1 ratio for the RNAStrAlign dataset, and a 2:1:1 ratio for the ncRNA dataset.

For the performance metric, we use precision (Prc) = $\frac{TP}{TP+FP}$, recall (Rec) = $\frac{TP}{TP+FN}$, and F1 score (F1) = $2\frac{Prc \cdot Rec}{Prc+Rec}$. Although they all provide meaningful information, we emphasize F1 as a major metric as the label $\boldsymbol{Y}$ is highly imbalanced: there are much more zeros than ones in the secondary structure matrix. We also compare the running time of the post-processing methods per sequence, presented in seconds.

5.2. Results

To fairly demonstrate the performance of the proposed post-processing method, we employ the deep neural networks introduced in E2Efold (Chen et al., 2020) and REDfold (Chen & Chan, 2023) for the first stage of the architecture. That is, we consider two different models: a transformer and a U-Net. The output of these models is then accordingly modified using respective post-processing algorithms (if needed). We compare this to several others, including RNAfold (Hofacker, 2009), RNAstructure (Bellaousov et al., 2013), CONTRAfold (Do et al., 2006), SPOT-RNA (Singh et al., 2019), MXfold2 (Sato et al., 2021), E2Efold and REDfold.

For E2Efold and REDfold, we train the exact same models with three different random seeds in training, and report the mean performance on the held-out test set. Since RNAStrAlign provided by the authors of E2Efold fixes the train/test split, we use the random seeds for different model initialization while following the fixed data split. In case of ncRNA dataset, we repeated the three experiments with different train/test split, while fixing the parameter initialization. For the other baselines, we use the packages available online without any further modifications.

Table 2. Constraint violations on RNAStrAlign.

Method	**(C2)**	**(C3)**	**(C4)**
CONTRAfold	0%	10.2%	0%
SPOT-RNA	91.1%	14.1%	0%
E2Efold + *E2E PP*	0%	10.3%	56.8%
E2Efold + *Blossom*	86.4%	18.4%	0.1%
E2Efold + *Ours*	**0%**	**0%**	**0%**
REDfold + *RED PP*[1]	0%	0.6%	0%
REDfold + *Blossom*	8.4%	1.6%	0%
REDfold + *Ours*	**0%**	**0%**	**0%**

Table 3. Constraint violations on ncRNA.

Method	**(C2)**	**(C3)**	**(C4)**
CONTRAfold	0%	8.1%	0%
SPOT-RNA	57.7%	30.3%	0%
E2Efold + *E2E PP*	0%	12.8%	58.3%
E2Efold + *Blossom*	73.0%	21.0%	0.3%
E2Efold + *Ours*	**0%**	**0%**	**0%**
REDfold + *RED PP*[1]	0%	0.5%	0%
REDfold + *Blossom*	9.4%	0.3%	0%
REDfold + *Ours*	**0%**	**0%**	**0%**

We first check whether the outputs generated by the deep learning-based models respect the structural constraints, described in Section 3.1. Table 2 and Table 3 provide the number of invalid outputs generated by different prediction algorithms in the benchmark datasets. We show which constraints are violated by baselinie algorithms on

[1]Despite the fact that REDfold guarantees adhesion to the constraints, a few invalid predictions occurred although we faithfully implemented using the official source code.

the RNAStrAlign and ncRNA dataset, respectively. In an expression "A + *B*" in the table, we indicate "A" by the backbone architecture that is either a transformer (E2Efold) or a U-Net (REDfold). Meanwhile, "*B*" refers to the type of post-processing algorithm.

ML models without post-processing, namely CONTRAfold and SPOT-RNA, tend to produce predictions that violate the structural requirements as expected. In contrast to what one can expect, however, *E2E PP* cannot address the challenge of constraint violation. The same is true for *Blossom* post-processing, as the algorithm considers only (C4). *RED PP* tends to generate valid structures with an additional $\arg\max$ operation, though it can possibly compromise predictive accuracy. Still, we can observe a stark contrast between models with and without the proposed post-processing; in particular, *Ours* produce absolutely zero invalid predictions.

Table 4. Performances on the RNAStrAlign dataset.

Method	F1	Rec	Prc	Time (s)
RNAfold	0.602	0.634	0.576	-
RNAstructure	0.596	0.621	0.576	-
CONTRAfold	0.664	0.699	0.635	-
SPOT-RNA	0.758	0.826	0.717	-
MXfold2	0.783	0.798	0.772	-
E2Efold + *E2E PP*	0.809	0.797	0.826	0.055
E2Efold + *Blossom*	0.759	0.826	0.706	4.959[2]
E2Efold + *Ours*	0.822	0.822	0.823	1.010[2]
REDfold + *RED PP*	0.904	0.874	**0.975**	0.079
REDfold + *Blossom*	0.917	**0.909**	0.941	0.496
REDfold + *Ours*	**0.918**	0.907	0.944	**0.014**

Table 5. Performances on the ncRNA dataset.

Method	F1	Rec	Prc	Time (s)
RNAfold	0.606	0.679	0.566	-
RNAstructure	0.599	0.668	0.562	-
CONTRAfold	0.626	0.690	0.596	-
SPOT-RNA	0.647	0.683	0.640	-
MXfold2	0.631	0.687	0.608	-
E2Efold + *E2E PP*	0.595	0.575	0.631	0.049
E2Efold + *Blossom*	0.489	0.615	0.415	2.212[2]
E2Efold + *Ours*	0.608	0.602	0.622	0.308[2]
REDfold + *RED PP*	0.844	0.849	**0.877**	0.053
REDfold + *Blossom*	0.840	**0.873**	0.838	0.378
REDfold + *Ours*	**0.847**	0.867	0.858	**0.005**

Table 4 demonstrates the predictive performance of the baseline algorithms on RNAStrAlign dataset, together with mean running time per an instance for the respective post-processing methods. Notably, applying our post-processing method not only helps generating valid RNA structure predictions, but also slightly improves the predictive performance of deep learning-based models, as indicated by the bold numbers in the table. In particular, when combined with the REDfold model, *Ours* outperform all the baselines for F1 metric, with a marginal compromise in recall and precision. Also note that the proposed post-processing exhibits the shortest mean running time, showing more than five folds improvement compared with *Blossom* or *RED PP*.

In Table 5, we observe a similar trend in the ncRNA dataset, except that the overall performance has deteriorated by a respectful margin for all the algorithms.

In Figure 4, we visualize the predicted structures of baseline algorithms for an example sequence 'Ake.c.trnL' in the RNAStrAlign database. The drawings were generated using the VARNA tool (Darty et al., 2009). As depicted in Figure 4, it is evident that the predicted structures of the proposed method best reproduce the ground truth structure.

5.3. Pseudoknot Prediction

Table 6. Performances for the RNA sequences with pseudoknot on the ncRNA dataset.

Method	F1	Rec	Prc	Time (s)
RNAfold	0.473	0.483	0.478	-
RNAstructure	0.471	0.479	0.478	-
CONTRAfold	0.503	0.505	0.522	-
SPOT-RNA	0.621	0.614	0.645	-
MXfold2	0.507	0.502	0.534	-
E2Efold + *E2E PP*	0.603	0.574	0.651	0.048
E2Efold + *Blossom*	0.509	0.612	0.442	2.273[2]
E2Efold + *Ours*	0.616	0.601	0.635	0.873[2]
REDfold + *RED PP*	0.810	0.785	**0.885**	0.057
REDfold + *Blossom*	0.814	**0.814**	0.848	0.620
REDfold + *Ours*	**0.818**	0.807	0.865	**0.008**

Pseudoknots are a special type of local structures frequently observed in RNAs, often excluded in the search space of conventional algorithms. They occur when a base from a loop (hairpin or internal) pairs with another base outside the enclosing loop. Due to their significant roles in our cells (Brierley et al., 2007; Staple & Butcher, 2005; Namy et al., 2006), identifying RNA structures with pseudoknots is a major challenge in the structure prediction task. In light of this, we pick the samples in the benchmark datasets that contain pseudoknots using a python library called Biotite (Kunzmann & Hamacher, 2018).

In Table 6, the predictive performance against pseudoknotted RNAs is evaluated on the ncRNA dataset. Here, REDfold + *Ours* achieves the best predictive performances yet with a slight compromise in precision and recall, similarly as above.

[2]In the case of the E2Efold model, the distribution of the output matrix values is heavily skewed towards larger values. Consequently, there are many entries that remain after thresholding, which causes the ILP-based algorithm to take a long time to run.

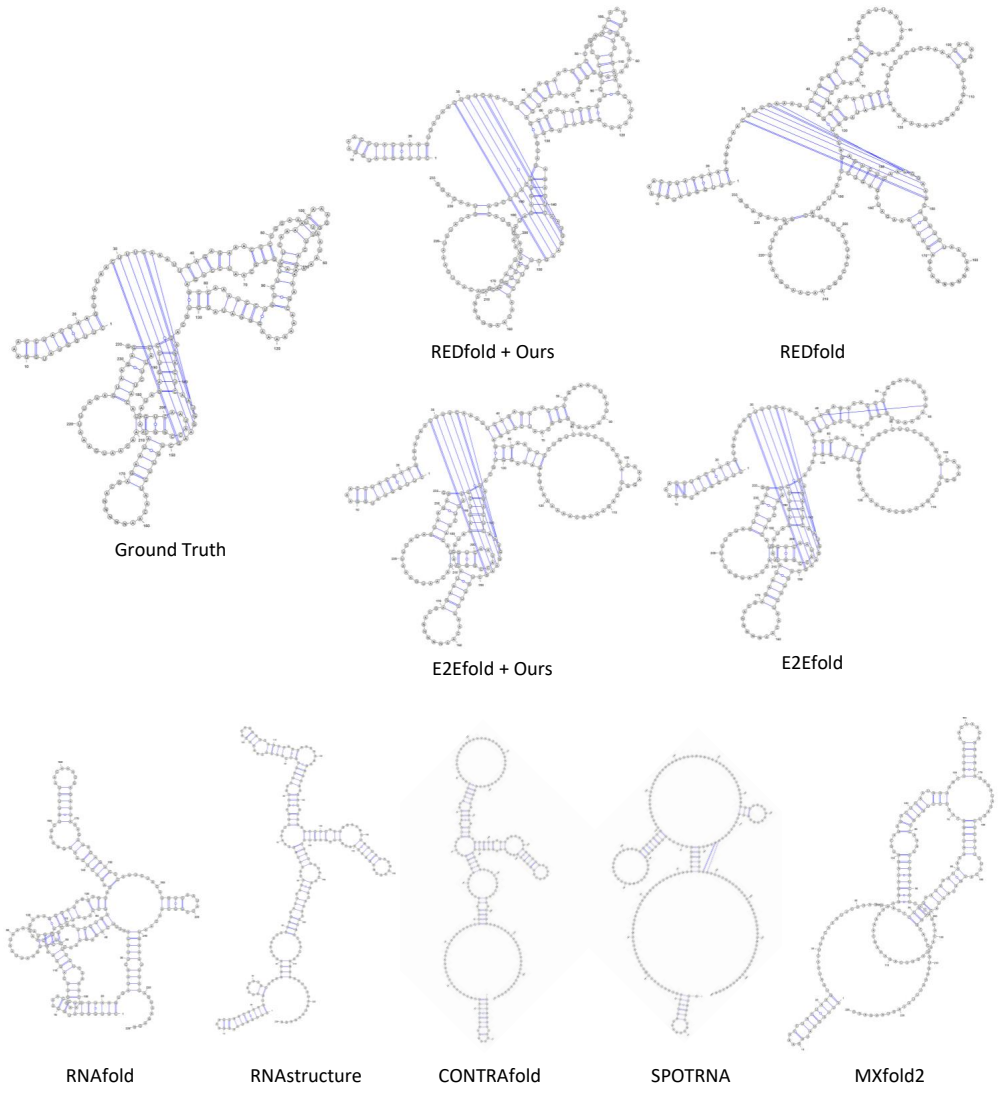

Figure 4. Visualization of the predicted RNA secondary structures of 'Ake.c.trnL' with various methods.

5.4. Empirical Time Complexity

In Figure 5, we plot the average running time of the post-processing methods as a function of RNA length, where y-axis is in a logarithmic scale. We can observe a significantly greater efficiency of *Ours* for all length of RNAs, compared with *E2E PP* which relies on a gradient-based algorithm computed on a GPU environment, and *Blossom* computed on a CPU environment. Considering the fact that our algorithm is not optimized for GPU execution, the empirical running time can be even improved further.

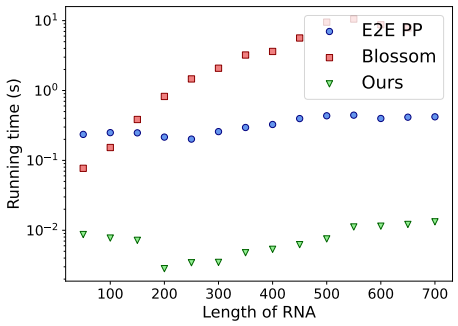

Figure 5. Running time comparison of post-processing algorithms with respect to the length of RNA sequences on the ncRNA dataset, measured in seconds.

6. Discussion

Summary. Our post-processing framework aims to enhance adherence to the fundamental constraints of RNA secondary structures. Leveraging the equivalence theorem in Section 4.1, we employ an efficient algorithm that achieves an exact solution to the ILP problem. Experiments highlight that our post-processing method enables existing ML models to satisfy the structural constraints, with slightly improved performance and a significant reduction in runtime.

Limitation. Our work poses two major limitations: (1) The two stage approach to RNA secondary structure prediction disconnects the computational graph, making it challenging to adopt an end-to-end model. If a backbone model is aware of the subsequent post-processing step, it could make more reasonable predictions in terms of the structural constraints without the need for additional post-processing. Hence, connecting the computational graph of a neural network with discrete optimization is an important yet non-trivial question. (2) The Hungarian algorithm in the second stage has a time complexity of $\mathcal{O}(L^3)$, given an RNA sequence of length L. Given that natural RNAs can be tens of thousands of nucleotides long, cubic time complexity may not be satisfactory for lengthy sequences. One may consider using approximate algorithms faster than $\mathcal{O}(L^3)$ as alternatives, even if they compromise the optimality of the solution.

Future work. One future work of our interest is to push the generalization capability of the state-of-the-art models, especially for various RNA families and longer sequences. Incorporating genetic language models alongside specialized structure prediction algorithms shows promise in this regard. Several attempts have been made to develop universal RNA language models, or equivalently, foundation models for RNAs (Akiyama & Sakakibara, 2022; Chen et al., 2022; Wang et al., 2023). Since our post-processing algorithm is model-agnostic, we expect that it will make a great synergy with such advanced ML models.

Open problems. We argue that two critical open problems in RNA folding remain largely unaddressed: (1) Existing algorithms use the complete sequence information in the input, overlooking the fact that RNAs begin to fold while they are being made (Kramer & Mills, 1981; Pan & Sosnick, 2006; Bushhouse et al., 2022; Szyjka & Strobel, 2023). Put it differently, RNAs can contain structures that are locally optimal, contrary to the underlying assumption in existing algorithms that find structures with globally minimal free energy. Exploring ways to incorporate such dynamics into ML models can be an interesting research direction. (2) RNAs may permit multiple structure solutions. Organisms have evolved to choose genetic sequences that have robust folding patterns, as evidenced by some RNA sequences (Le et al., 2002; Schultes et al., 2005). Alternative structures, however, are also allowed as long as they can serve similar functions in the cells (Russell et al., 2006; Ritz et al., 2013). Integrating this flexibility into computational algorithms and devising methods to evaluate models in light of this possibility can be an important research problem.

Acknowledgement

This work was supported by the Technology Development Program (S3284154, RS-2023-00303099) funded by the Ministry of SMEs and Startups (MSS, Korea).

Impact Statement

The development of advanced computational models for predicting RNA secondary structures has significant potential to impact various scientific and medical fields. These models can accelerate research in molecular biology, leading to a deeper understanding of RNA functions and their roles in various diseases. This knowledge could drive innovations in drug development, enabling more effective treatments for genetic disorders and other RNA-related conditions. Additionally, the integration of machine learning techniques in RNA research can stimulate advancements in computational biology, fostering interdisciplinary collaboration and technological progress. Ultimately, these enhancements in RNA structure prediction will contribute to major breakthroughs in personalized medicine, leading to improved quality of life for patients. In ethical aspects, there are only a few potential consequences of our work, none of which we feel must be specifically highlighted here.

References

Akiyama, M. and Sakakibara, Y. Informative RNA base embedding for RNA structural alignment and clustering by deep representation learning. *NAR genomics and bioinformatics*, 4(1):lqac012, 2022.

Akiyama, M., Sato, K., and Sakakibara, Y. A max-margin training of RNA secondary structure prediction integrated with the thermodynamic model. *Journal of bioinformatics and computational biology*, 16(06):1840025, 2018.

Akutsu, T. Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. *Discrete Applied Mathematics*, 104(1):45–62, 2000.

Andronescu, M., Condon, A., Hoos, H. H., Mathews, D. H., and Murphy, K. P. Efficient parameter estimation for RNA secondary structure prediction. *Bioinformatics*, 23 (13):i19–i28, 2007.

Andronescu, M., Bereg, V., Hoos, H. H., and Condon, A. RNA STRAND: The RNA secondary structure and statistical analysis database. *BMC Bioinformatics*, 9:340 – 340, 2008.

Andronescu, M., Condon, A., Hoos, H. H., Mathews, D. H., and Murphy, K. P. Computational approaches for RNA energy parameter estimation. *RNA*, 16(12):2304–2318, 2010.

Bellaousov, S. and Mathews, D. H. ProbKnot: fast prediction of RNA secondary structure including pseudoknots. *RNA*, 16 10:1870–80, 2010.

Bellaousov, S., Reuter, J. S., Seetin, M. G., and Mathews, D. H. RNAstructure: web servers for RNA secondary structure prediction and analysis. *Nucleic acids research*, 41(W1):W471–W474, 2013.

Bonnet, É., Rzażewski, P., and Sikora, F. Designing RNA secondary structures is hard. *Journal of computational biology : a journal of computational molecular cell biology*, 27(3):302–316, 2017.

Brierley, I., Pennell, S., and Gilbert, R. J. Viral RNA pseudoknots: versatile motifs in gene expression and replication. *Nature Reviews Microbiology*, 5(8):598–610, 2007.

Bringmann, K., Grandoni, F., Saha, B., and Williams, V. V. Truly sub-cubic algorithms for language edit distance and RNA-folding via fast bounded-difference min-plus product. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 375–384, 2016.

Bushhouse, D. Z., Choi, E. K., Hertz, L. M., and Lucks, J. B. How does RNA fold dynamically? *Journal of molecular biology*, 434(18):167665, 2022.

Chen, C.-C. and Chan, Y.-M. REDfold: accurate RNA secondary structure prediction using residual encoder-decoder network. *BMC bioinformatics*, 24(1):1–13, 2023.

Chen, J., Hu, Z., Sun, S., Tan, Q., Wang, Y., Yu, Q., Zong, L., Hong, L., Xiao, J., Shen, T., et al. Interpretable RNA foundation model from unannotated data for highly accurate RNA structure and function predictions. *bioRxiv*, pp. 2022–08, 2022.

Chen, X., Li, Y., Umarov, R., Gao, X., and Song, L. RNA secondary structure prediction by learning unrolled algorithms. In *International Conference on Learning Representations*, 2020.

Dantzig, G. B. Application of the simplex method to a transportation problem. *Activity analysis and production and allocation*, 1951.

Darty, K., Denise, A., and Ponty, Y. VARNA: Interactive drawing and editing of the RNA secondary structure. *Bioinformatics*, 25(15):1974, 2009.

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

Edmonds, J. and Karp, R. M. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264, 1972.

Egerváry, J. Matrixok kombinatorius tulajdonságairól. *Matematikai és Fizikai Lapok*, 38(1931):16–28, 1931.

Fu, L., Niu, B., Zhu, Z., Wu, S., and Li, W. CD-HIT: accelerated for clustering the next-generation sequencing data. *Bioinformatics*, 28(23):3150–3152, 2012.

Fu, L., Cao, Y., Wu, J., Peng, Q., Nie, Q., and Xie, X. UFold: fast and accurate RNA secondary structure prediction with deep learning. *Nucleic acids research*, 50(3):e14–e14, 2022.

Galil, Z. Efficient algorithms for finding maximum matching in graphs. *ACM Computing Surveys (CSUR)*, 18(1): 23–38, 1986.

Griffiths-Jones, S., Bateman, A., Marshall, M., Khanna, A., and Eddy, S. R. Rfam: an RNA family database. *Nucleic Acids Research*, 31(1):439–441, 01 2003.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hofacker, I. L. Vienna RNA secondary structure server. *Nucleic acids research*, 31(13):3429–3431, 2003.

Hofacker, I. L. RNA secondary structure analysis using the Vienna RNA package. *Current protocols in bioinformatics*, 26(1):12–2, 2009.

Huang, L., Zhang, H., Deng, D., Zhao, K., Liu, K., Hendrix, D. A., and Mathews, D. H. LinearFold: linear-time approximate RNA folding by 5'-to-3' dynamic programming and beam search. *Bioinformatics*, 35(14):i295–i304, 07 2019.

Jonker, R. and Volgenant, T. A shortest augmenting path algorithm for dense and sparse linear assignment problems. In *DGOR/NSOR: Papers of the 16th Annual Meeting of DGOR in Cooperation with NSOR/Vorträge der 16. Jahrestagung der DGOR zusammen mit der NSOR*, pp. 622–622, 1988.

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

Kappel, K., Zhang, K., Su, Z., Watkins, A. M., Kladwang, W., Li, S., Pintilie, G., Topkar, V. V., Rangan, R., Zheludev, I. N., et al. Accelerated cryo-EM-guided determination of three-dimensional RNA-only structures. *Nature methods*, 17(7):699–707, 2020.

Kim, S., Suddath, F., Quigley, G., McPherson, A., Sussman, J., Wang, A., Seeman, N., and Rich, A. Three-dimensional tertiary structure of yeast phenylalanine transfer RNA. *Science*, 185(4149):435–440, 1974.

Konig, D. Graphok es matrixok (hungarian)[graphs and matrices]. *Matematikai és Fizikai Lapok*, 38:116–119, 1931.

Kramer, F. R. and Mills, D. R. Secondary structure formation during RNA synthesis. *Nucleic acids research*, 9(19): 5109–5124, 1981.

Kuhn, H. W. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Kuhn, H. W. A tale of three eras: The discovery and rediscovery of the hungarian method. *European Journal of Operational Research*, 219(3):641–651, 2012.

Kunzmann, P. and Hamacher, K. Biotite: a unifying open source computational biology framework in python. *BMC bioinformatics*, 19:1–8, 2018.

Le, S.-Y., Zhang, K., and Maizel Jr, J. V. RNA molecules with structure dependent functions are uniquely folded. *Nucleic acids research*, 30(16):3574–3582, 2002.

Lyngsø, R. B. and Pedersen, C. N. S. RNA pseudoknot prediction in energy-based models. *Journal of computational biology : a journal of computational molecular cell biology*, 7(3-4):409–27, 2000.

Markham, N. R. and Zuker, M. UNAFold: software for nucleic acid folding and hybridization. *Bioinformatics: structure, function and applications*, pp. 3–31, 2008.

Munkres, J. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

Namy, O., Moran, S. J., Stuart, D. I., Gilbert, R. J., and Brierley, I. A mechanical explanation of RNA pseudoknot function in programmed ribosomal frameshifting. *Nature*, 441(7090):244–247, 2006.

Neidle, S. and Sanderson, M. *Principles of nucleic acid structure*. Academic Press, 2021.

Noller, H. F. Structure of ribosomal RNA. *Annual review of biochemistry*, 53(1):119–162, 1984.

Nussinov, R. and Jacobson, A. B. Fast algorithm for predicting the secondary structure of single-stranded RNA. *Proceedings of the National Academy of Sciences*, 77(11): 6309–6313, 1980.

Pan, T. and Sosnick, T. RNA folding during transcription. *Annu. Rev. Biophys. Biomol. Struct.*, 35:161–175, 2006.

Reuter, J. S. and Mathews, D. H. RNAstructure: software for RNA secondary structure prediction and analysis. *BMC bioinformatics*, 11(1):1–9, 2010.

Ritz, J., Martin, J. S., and Laederach, A. Evolutionary evidence for alternative structure in RNA sequence covariation. *PLoS computational biology*, 9(7):e1003152, 2013.

Rivas, E. and Eddy, S. R. A dynamic programming algorithm for RNA structure prediction including pseudoknots. *Journal of molecular biology*, 285(5):2053–68, 1998.

Ronneberger, O., Fischer, P., and Brox, T. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany*, pp. 234–241. Springer, 2015.

Russell, R., Das, R., Suh, H., Travers, K. J., Laederach, A., Engelhardt, M. A., and Herschlag, D. The paradoxical behavior of a highly structured misfolded intermediate in RNA folding. *Journal of molecular biology*, 363(2): 531–544, 2006.

Saman Booy, M., Ilin, A., and Orponen, P. RNA secondary structure prediction with convolutional neural networks. *BMC bioinformatics*, 23(1):58, 2022.

Sato, K., Kato, Y., Hamada, M., Akutsu, T., and Asai, K. IPknot: fast and accurate prediction of RNA secondary structures with pseudoknots using integer programming. *Bioinformatics*, 27(13):i85–i93, 06 2011.

Sato, K., Akiyama, M., and Sakakibara, Y. RNA secondary structure prediction using deep learning with thermodynamic integration. *Nature communications*, 12(1):941, 2021.

Schrijver, A. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

Schroeder, S. J. and Turner, D. H. Optical melting measurements of nucleic acid thermodynamics. *Methods in enzymology*, 468:371–387, 2009.

Schultes, E. A., Spasic, A., Mohanty, U., and Bartel, D. P. Compact and ordered collapse of randomly generated RNA sequences. *Nature structural & molecular biology*, 12(12):1130–1136, 2005.

Schuster, M. and Paliwal, K. K. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

Singh, J., Hanson, J., Paliwal, K., and Zhou, Y. RNA secondary structure prediction using an ensemble of two-dimensional deep neural networks and transfer learning. *Nature communications*, 10(1):5407, 2019.

Spitale, R. C. and Incarnato, D. Probing the dynamic RNA structurome and its functions. *Nature Reviews. Genetics*, 24:178 – 196, 2022.

Staple, D. W. and Butcher, S. E. Pseudoknots: RNA structures with diverse functions. *PLOS Biology*, 3(6), 06 2005.

Sussman, J. L., Lin, D., Jiang, J., Manning, N. O., Prilusky, J., Ritter, O., and Abola, E. E. Protein Data Bank (PDB): database of three-dimensional structural information of biological macromolecules. *Acta Crystallographica Section D: Biological Crystallography*, 54(6):1078–1084, 1998.

Szyjka, C. E. and Strobel, E. J. Observation of coordinated RNA folding events by systematic cotranscriptional RNA structure probing. *Nature Communications*, 14(1):7839, 2023.

Tan, Z., Fu, Y., Sharma, G., and Mathews, D. H. TurboFold II: RNA structural alignment and secondary structure prediction informed by multiple homologs. *Nucleic acids research*, 45(20):11570–11581, 2017.

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

Tomizawa, N. On some techniques useful for solution of transportation network problems. *Networks*, 1(2):173–194, 1971.

Turner, D. H. and Mathews, D. H. NNDB: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure. *Nucleic acids research*, 38(suppl_1):D280–D282, 2010.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, X., Gu, R., Chen, Z., Li, Y., Ji, X., Ke, G., and Wen, H. UNI-RNA: universal pre-trained models revolutionize RNA research. *bioRxiv*, pp. 2023–07, 2023.

Watson, J. D. and Crick, F. H. Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–738, 1953.

Zhang, H., Zhang, C., Li, Z., Li, C., Wei, X., Zhang, B., and Liu, Y. A new method of RNA secondary structure prediction based on convolutional neural network and dynamic programming. *Frontiers in genetics*, 10:467, 2019.

Zuker, M. Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic acids research*, 31(13): 3406–3415, 2003.

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

A. Proof of Theorem 1

Theorem A.1 (Equivalence)**.** *Let $\boldsymbol{x}$ be a sequence of nucleotide with length L, $\hat{\boldsymbol{Y}} \in \mathbb{R}^{L\times L}$ be a symmetric matrix that represents initial model prediction, and s be an offset term that weighs the entries of importance. Given $\boldsymbol{x}, \hat{\boldsymbol{Y}}, s$, let $\boldsymbol{M} := \boldsymbol{M}(\boldsymbol{x}, \hat{\boldsymbol{Y}}, s)$ be a binary matrix whose (i,j)-th entry is defined as $\boldsymbol{M}_{ij} = 1$ if $x_i x_j \in \{\texttt{AU}, \texttt{GU}, \texttt{GC}\}$, $|i-j| \geq 4$, and $\hat{\boldsymbol{Y}}_{ij} > s$, 0 otherwise. Let $\boldsymbol{C}(\hat{\boldsymbol{Y}}) := -2(\hat{\boldsymbol{Y}} - s) \odot \boldsymbol{M}$. Then,*

- *For an optimal solution $\tilde{\boldsymbol{Y}}^*$ of problem (1), define an $L \times L$ binary matrix $\boldsymbol{Z}^*$ as $\boldsymbol{Z}^*_{ij} = 1$ if and only if $\tilde{\boldsymbol{Y}}^*_{ij} = 1$, $x_i x_j \in \{\texttt{AU}, \texttt{GU}, \texttt{GC}\}$ and $\hat{\boldsymbol{Y}}_{ij} > s$. Then, $\boldsymbol{Z}^*$ is an optimal solution of problem (2) with $\boldsymbol{C} = \boldsymbol{C}(\hat{\boldsymbol{Y}})$.*
- *For an optimal solution $\boldsymbol{Z}^*$ with $\boldsymbol{C} = \boldsymbol{C}(\hat{\boldsymbol{Y}})$ of problem (2), $\tilde{\boldsymbol{Y}}^* := \boldsymbol{Z}^* \odot \boldsymbol{M} + (\boldsymbol{Z}^* \odot \boldsymbol{M})^T$ is an optimal solution of problem (1).*

We claim that problems (1) and (2) introduced in Section 4 are equivalent, by showing that any optimal solution of each optimization can be reduced to an optimal solution of the other optimization problem.

Proposition A.2. *Let $\tilde{\boldsymbol{Y}}$ be an $L \times L$ matrix satisfying the constraints in problem (1). Then there exists an $L \times L$ binary matrix $\boldsymbol{Z}$ such that $\tilde{\boldsymbol{Y}} = \boldsymbol{Z} \odot \boldsymbol{M} + (\boldsymbol{Z} \odot \boldsymbol{M})^T$, $\boldsymbol{Z}\mathbf{1} \leq \mathbf{1}$ and $\boldsymbol{Z}^T\mathbf{1} \leq \mathbf{1}$.*

Proof. Set $\boldsymbol{Z} := \boldsymbol{Z}(\tilde{\boldsymbol{Y}})$ to be $\boldsymbol{Z}_{ij} = 1$ if and only if $\tilde{\boldsymbol{Y}}_{ij} = 1$, $x_i x_j \in \{\texttt{AU}, \texttt{GU}, \texttt{GC}\}$ and $\hat{\boldsymbol{Y}}_{ij} > s$. The equality holds since every (i,j) satisfying $\tilde{\boldsymbol{Y}}_{ij} = 1$ are contained in $\mathcal{B} := \{\texttt{AU}, \texttt{GU}, \texttt{GC}, \texttt{UA}, \texttt{UG}, \texttt{CG}\}$. The inequality constraint holds because $\boldsymbol{Z}_{ij} = 1$ only if $\boldsymbol{x}_i \in \{\texttt{A}, \texttt{G}\}$ and $\boldsymbol{x}_j \in \{\texttt{U}, \texttt{C}\}$, while $\boldsymbol{Z}^T_{ij} = 1$ only if $\boldsymbol{x}_i \in \{\texttt{U}, \texttt{C}\}$ and $\boldsymbol{x}_j \in \{\texttt{A}, \texttt{G}\}$. □

Lemma A.3. *Let $\tilde{\boldsymbol{Y}}^*$ be a feasible solution of problem (1). Then, $\boldsymbol{Z}^* = \boldsymbol{Z}(\tilde{\boldsymbol{Y}}^*)$ is a feasible solution of problem (2).*

Proof. For any $\tilde{\boldsymbol{Y}}$, $\tilde{\boldsymbol{Y}} = \boldsymbol{Z} + \boldsymbol{Z}^T$ and $\langle \hat{Y} - s, \tilde{Y} \rangle = 2\langle \hat{Y} - s, \boldsymbol{Z} \rangle$ due to the symmetry of $\hat{\boldsymbol{Y}}$. Therefore,

$$\tilde{\boldsymbol{Y}}^* = \arg\max_{\tilde{\boldsymbol{Y}}} \langle \hat{\boldsymbol{Y}} - s, \tilde{\boldsymbol{Y}} \rangle \quad \text{s.t.} \quad \tilde{\boldsymbol{Y}} = \tilde{\boldsymbol{Y}}^T, \tilde{\boldsymbol{Y}}\mathbf{1} \leq \mathbf{1}, \tilde{\boldsymbol{Y}}_{ij} = 0 \text{ if } \boldsymbol{x}_i\boldsymbol{x}_j \in \mathcal{B} \text{ or } |i-j| < 4 \tag{6}$$

$$= \arg\max_{\tilde{\boldsymbol{Y}}} \langle \hat{\boldsymbol{Y}} - s, \boldsymbol{Z}(\tilde{\boldsymbol{Y}}) + \boldsymbol{Z}(\tilde{\boldsymbol{Y}})^T \rangle \quad \text{s.t.} \quad \tilde{\boldsymbol{Y}} = \tilde{\boldsymbol{Y}}^T, \tilde{\boldsymbol{Y}}\mathbf{1} \leq \mathbf{1}, \tilde{\boldsymbol{Y}}_{ij} = 0 \text{ if } \boldsymbol{x}_i\boldsymbol{x}_j \in \mathcal{B} \text{ or } |i-j| < 4 \tag{7}$$

$$= \arg\max_{\tilde{\boldsymbol{Y}}} 2\langle \hat{\boldsymbol{Y}} - s, \boldsymbol{Z}(\tilde{\boldsymbol{Y}}) \rangle \quad \text{s.t.} \quad \tilde{\boldsymbol{Y}} = \tilde{\boldsymbol{Y}}^T, \tilde{\boldsymbol{Y}}\mathbf{1} \leq \mathbf{1}, \tilde{\boldsymbol{Y}}_{ij} = 0 \text{ if } \boldsymbol{x}_i\boldsymbol{x}_j \in \mathcal{B} \text{ or } |i-j| < 4 \tag{8}$$

$$\boldsymbol{Z}^* = \arg\max_{\boldsymbol{Z}} 2\langle \hat{\boldsymbol{Y}} - s, \boldsymbol{Z} \rangle \quad \text{s.t.} \quad \boldsymbol{Z}_{ij} \in \{0,1\}, \quad \boldsymbol{Z}\mathbf{1} = \mathbf{1}, \quad \boldsymbol{Z}^T\mathbf{1} = \mathbf{1},$$

$$\boldsymbol{Z}_{ij} = \boldsymbol{Z}^T_{ij} = 0 \text{ if } \boldsymbol{x}_i\boldsymbol{x}_j \in \mathcal{B} \text{ or } |i-j| < 4 \tag{9}$$

$$= \arg\max_{\boldsymbol{Z}} 2\langle (\hat{\boldsymbol{Y}} - s) \odot \boldsymbol{M}, \boldsymbol{Z} \rangle \quad \text{s.t.} \quad \boldsymbol{Z}_{ij} \in \{0,1\}, \quad \boldsymbol{Z}\mathbf{1} = \mathbf{1}, \quad \boldsymbol{Z}^T\mathbf{1} = \mathbf{1} \tag{10}$$

$$= \arg\max_{\boldsymbol{Z}} 2\langle (\hat{\boldsymbol{Y}} - s) \odot \boldsymbol{M}, \boldsymbol{Z} \rangle \quad \text{s.t.} \quad \boldsymbol{Z}_{ij} \in \{0,1\}, \quad \boldsymbol{Z}\mathbf{1} \leq \mathbf{1}, \quad \boldsymbol{Z}^T\mathbf{1} \leq \mathbf{1} \tag{11}$$

where equation (8) holds due to the symmetry of $\tilde{\boldsymbol{Y}}$, (9) by Proposition A.2, (11) by extending the boundary conditions. □

Lemma A.4. *Let $\boldsymbol{Z}^*$ be a feasible solution of problem (2). Then $\tilde{\boldsymbol{Y}}^* := \boldsymbol{Z}^* \odot \boldsymbol{M} + (\boldsymbol{Z}^* \odot \boldsymbol{M})^T$ is a feasible solution of problem (1).*

Proof.

$$\boldsymbol{Z}^* = \arg\max_{\boldsymbol{Z}} 2\langle (\hat{\boldsymbol{Y}} - s) \odot \boldsymbol{M}, \boldsymbol{Z} \rangle \quad \text{s.t.} \quad \boldsymbol{Z}_{ij} \in \{0,1\}, \quad \boldsymbol{Z}\mathbf{1} = \mathbf{1}, \quad \boldsymbol{Z}^T\mathbf{1} = \mathbf{1} \tag{12}$$

$$= \arg\max_{\boldsymbol{Z}} 2\langle (\hat{\boldsymbol{Y}} - s) \odot \boldsymbol{M}, \boldsymbol{Z} \rangle \quad \text{s.t.} \quad \boldsymbol{Z}_{ij} \in \{0,1\}, \quad \boldsymbol{Z}\mathbf{1} \leq \mathbf{1}, \quad \boldsymbol{Z}^T\mathbf{1} \leq \mathbf{1} \tag{13}$$

$$= \arg\max_{\boldsymbol{Z}} 2\langle \hat{\boldsymbol{Y}} - s, \boldsymbol{Z} \odot \boldsymbol{M} \rangle \quad \text{s.t.} \quad \boldsymbol{Z}_{ij} \in \{0,1\}, \quad \boldsymbol{Z}\mathbf{1} \leq \mathbf{1}, \quad \boldsymbol{Z}^T\mathbf{1} \leq \mathbf{1} \tag{14}$$

$$= \arg\max_{\boldsymbol{Z}} \langle \hat{\boldsymbol{Y}} - s, \boldsymbol{Z} \odot \boldsymbol{M} \rangle + \langle \hat{\boldsymbol{Y}} - s, (\boldsymbol{Z} \odot \boldsymbol{M})^T \rangle \quad \text{s.t.} \quad \boldsymbol{Z}_{ij} \in \{0,1\}, \quad \boldsymbol{Z}\mathbf{1} \leq \mathbf{1}, \quad \boldsymbol{Z}^T\mathbf{1} \leq \mathbf{1} \tag{15}$$

$$= \arg\max_{\boldsymbol{Z}} \langle \hat{\boldsymbol{Y}} - s, \boldsymbol{Z} \odot \boldsymbol{M} + (\boldsymbol{Z} \odot \boldsymbol{M})^T \rangle \quad \text{s.t.} \quad \boldsymbol{Z}_{ij} \in \{0,1\}, \quad \boldsymbol{Z}\mathbf{1} \leq \mathbf{1}, \quad \boldsymbol{Z}^T\mathbf{1} \leq \mathbf{1} \tag{16}$$

where (13) holds by extending the boundary condition from the inner product with non-negative matrix, and (15) holds due to the symmetry of $\tilde{\boldsymbol{Y}}$. By Proposition A.2, $\boldsymbol{Z}^* \odot \boldsymbol{M} + (\boldsymbol{Z}^* \odot \boldsymbol{M})^T$ is a feasible solution of problem (1). □

Theorem A.1 holds as a corollary of Lemma $A.3$ and $A.4$.

B. Algorithmic Details

Here we provide a detailed explanation of Jonker-Volgenant algorithm (Jonker & Volgenant, 1988), which is an efficient method for solving the assignment problem. The Jonker-Volgenant algorithm systematically adjusts dual variables to explore and update the assignments using a shortest path augmenting algorithm, ensuring the total cost is minimized. The overall pseudocode is presented in Algorithm 2.

At first, the required arrays and variables are initialized in lines 1-4. Line 5 initiates a loop over each row, and lines 6-10 initialize variables specific to the current row. Lines 12-25 iterate through each column to find the minimum value for each un-visited column. It updates the current minimum assignment cost while tracking the column links, and determines the minimum value across all un-visited columns. Then the dual variables are adjusted based on the minimum value in lines 26-33, and row and column with the smallest delta as visited are marked in lines 34-36. After these updates, lines 37-40 update the assignments to ensure optimality using a path augmentation technique, by backtracking through the column links until all the columns are marked. The final output is constructed from lines 42-50.

C. Training Details

We leave two tables containing hyperparameters used in real data experiments. See Table 7 and 8. We conducted a grid search to choose the values of learning rate and s from the respective ranges of 0.01 to 0.00001 and 0 to 5.

Table 7. Hyperparameters used for deep neural networks in experiments.

Hyperparameters	**E2Efold (Chen et al., 2020)**	**REDfold (Chen & Chan, 2023)**
Batch Size	20 (Stage 1), 16 (Stage 3)	1
Number of Epochs	50 (RNAStrAlign), 100 (ncRNA)	200
Learning Rate	0.001	0.001

Table 8. Hyperparameters used for post-processing stage in experiments.

Hyperparameters	***E2E PP***	***RED PP***	***Ours***
Batch Size	1	1	1
s (threshold for score matrix)	3.5 (RNAStrAlign), 4.5 (ncRNA)	1 (RNAStrAlign), 1.5 (ncRNA)	3 (E2E, RNAStrAlign), 4 (E2E, ncRNA), 0 (RED, RNAStrAlign), 0.5 (RED, ncRNA),
Number of Iterations	50	100	Unneeded
ρ (coefficient for L1 penalty)	1	1.6	Unneeded
$\gamma_\alpha, \gamma_\beta$ (decaying coefficient)	0.01, 0.1	0.01, 0.1	Unneeded

In addition, we used a largest-weight edge selection parameter $k = 3$ for the Blossom post-processing (Saman Booy et al., 2022).

Algorithm 2 Jonker-Volgenant algorithm (Jonker & Volgenant, 1988)

Require: $\boldsymbol{C}$ (an $L \times L$ matrix)
$L \leftarrow$ number of rows/columns in $\boldsymbol{C}$
$\boldsymbol{u} \leftarrow$ array of size L (dual variables for rows, initialized to $\mathbf{0}$)
$\boldsymbol{v} \leftarrow$ array of size L (dual variables for columns, initialized to $\mathbf{0}$)
$\boldsymbol{p} \leftarrow$ array of size $L + 1$ (row assignments, initialized to $\mathbf{0}$)
for $i = 1$ to L **do**
  $links \leftarrow$ array of size $L + 1$ (to track column links, initialized to $\mathbf{0}$)
  $mins \leftarrow$ array of size $L + 1$ (to store minimal values, initialized to ∞)
  $visited \leftarrow$ array of size $L + 1$ (boolean array to track visited columns, initialized to $\mathbf{False}$)
  $(marked_i, marked_j) \leftarrow (i, 0)$ (to track the current row and column during the iteration)
  repeat
    $delta \leftarrow \infty$
    for $j = 1$ to L **do**
      if $\neg visited[j]$ **then**
        $cur \leftarrow \boldsymbol{C}[marked_i][j] - \boldsymbol{u}[marked_i] - \boldsymbol{v}[j]$
        if $cur < mins[j]$ **then**
          $mins[j] \leftarrow cur$
          $links[j] \leftarrow marked_j$
        end if
        if $mins[j] < delta$ **then**
          $delta \leftarrow mins[j]$
          $marked_j \leftarrow j$
        end if
      end if
    end for
    for $j = 1$ to L **do**
      if $visited[j]$ **then**
        $\boldsymbol{u}[\boldsymbol{p}[j]] \leftarrow \boldsymbol{u}[\boldsymbol{p}[j]] + delta$
        $\boldsymbol{v}[j] \leftarrow \boldsymbol{v}[j] - delta$
      else
        $mins[j] \leftarrow mins[j] - delta$
      end if
    end for
    $visited[marked_j] \leftarrow$ true
    $marked_i \leftarrow \boldsymbol{p}[marked_j]$
  until $marked_i = 0$
  repeat
    $\boldsymbol{p}[marked_j] \leftarrow \boldsymbol{p}[links[marked_j]]$
    $marked_j \leftarrow links[marked_j]$
  until $marked_j = 0$
end for
$result \leftarrow$ array of size L
for $j = 1$ to L **do**
  $result[\boldsymbol{p}[j]] \leftarrow j$
end for
$\boldsymbol{Z} \leftarrow \mathbf{0}_{L \times L}$ (initialize an $L \times L$ zero matrix for output)
for $i = 1$ to L **do**
  $\boldsymbol{Z}[i][result[i]] \leftarrow 1$
end for
Output: $\boldsymbol{Z}$

D. Source Code and Dataset for Reproduction

To facilitate the replication of our experiments and to encourage further research in this area, we have made the source code for our algorithm available for download. The source code can be accessed via the following link:

`https://github.com/KangSeokjun/RNASecondaryStructure`

The source code at the provided link was created by employing the deep neural network introduced in E2Efold (Chen et al., 2020) and REDfold (Chen & Chan, 2023) for the deep learning stage. The file also include detailed experimental data used in our research.

We encourage readers and researchers to explore, modify, and build upon our codebase to expand the scope of this research.

E. Additional Experiments

E.1. PR Curve

In Figure 6, We plot a PR curve to show the superior performance achieved by REDfold + *Ours* on ncRNA test set compared to REDfold + *RED PP*. We tuned the threshold parameter s to obtain multiple points.

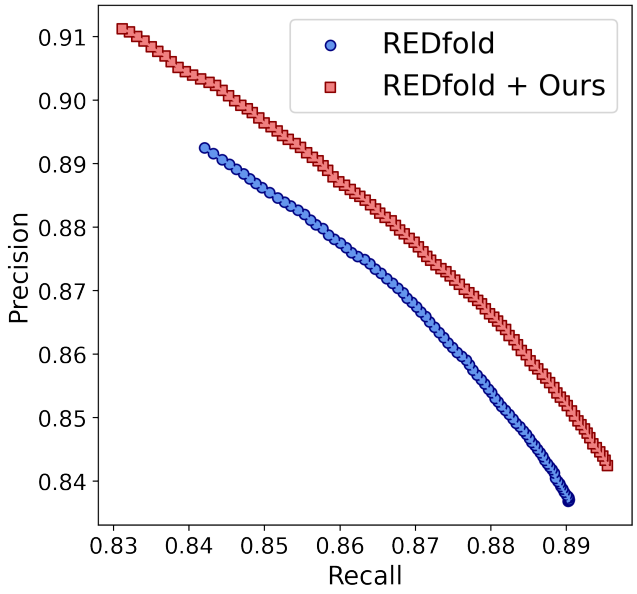

Figure 6. PR curve on the ncRNA dataset.

E.2. Relation of Benchmark Models and the Proposed Post-processing

The proposed PP method is model-agnostic in a sense that it can work with any computational model that makes a binary classification decision (or a regression) for every pair of nucleotide. In Table 9, we conducted an ablation study to find out whether ours can improve the performance of CONTRAfold on the RNAStrAlign dataset.

Table 9. Constraint violations and performances on the RNAStrAlign dataset.

Method	**(C2)**	**(C3)**	**(C4)**	**F1**	**Rec**	**Prc**
CONTRAfold	0%	10.2%	0%	0.664	0.635	0.699
CONTRAfold + *Ours*	0%	0%	0%	0.693	0.705	0.690