Sampling-based Continuous Optimization with Coupled Variables for RNA Design

Wei Yu Tang[a,b], Ning Dai[a], Tianshuo Zhou[a], David H. Mathews[d,e,f], and Liang Huang[a,c,1]

This manuscript was compiled on December 13, 2024

The task of RNA design given a target structure aims to find a sequence that can fold into that structure. It is a computationally hard problem where some version(s) have been proven to be NP-hard. As a result, heuristic methods such as local search have been popular for this task, but by only exploring a fixed number of candidates. They can not keep up with the exponential growth of the design space, and often perform poorly on longer and harder-to-design structures. We instead formulate these discrete problems as continuous optimization, which starts with a distribution over all possible candidate sequences, and uses gradient descent to improve the expectation of an objective function. We define novel distributions based on coupled variables to rule out invalid sequences given the target structure and to model the correlation between nucleotides. To make it universally applicable to any objective function, we use sampling to approximate the expected objective function, to estimate the gradient, and to select the final candidate. Compared to the state-of-the-art methods, our work consistently outperforms them in key metrics such as Boltzmann probability, ensemble defect, and energy gap, especially on long and hard-to-design puzzles in the Eterna100 benchmark. Our code is available at: http://github.com/weiyutang1010/ncrna_design.

RNA design | inverse folding | continuous optimization | sampling

Ribonucleic acid (RNA) is vital for fundamental cellular processes such as transcription and translation, catalyzing reactions, and controlling gene expression (1–3). Its importance is also evidenced by COVID-19, an RNA virus, as well as the last two Nobel Prizes in Physiology and Medicine for messenger RNA vaccines (2023) and microRNAs (2024). The problem of *RNA design* aims to find sequences that is capable of folding into a target structure (4–8). This process enables the creation of artificial RNA molecules with specific function, such as artificial ribozymes (9, 10), artificial miRNAs (11), artificial RNA aptamers (12), and artificial riboswitches (13, 14).

Computationally, the RNA design problem is extremely challenging due to its exponentially large search space, $\{A, C, G, U\}^n$, which has the size of $O(4^n)$. Indeed, it has been proved NP-hard at least for a simplified energy model (15). Therefore probably the most popular approach for this problem uses heuristic methods such as local search (4, 6), which starts with a single sequence and tries to optimize an objective function by revising one or a few nucleotides in each step. However, such methods are only capable of exploring a fixed number of candidate sequences, thus not able to keep up with the exponential growth of the design space. As a result, they tend to perform poorly on longer and harder-to-design structures.

We instead cast the RNA design problem as *continuous optimization* (16, 17). The basic idea is to start with a distribution over *all possible* candidate sequences and use gradient descent to gradually sharpen the distribution, with each step changing *all* positions simultaneously in contrast to local search methods. In this work, we first define novel sequence distributions for any given RNA structure using coupled variables for paired and mismatch positions, which not only rules out invalid sequences but also models the positional correlations explicitly. We then aim to optimize the *expectation* of an *arbitrary* objective function over these distributions. However, given the diversity of various objective functions in RNA design (such as Boltzmann probability or ensemble defect), it is often computationally prohibitive to compute the exact values of the expected objective function or its gradient over the whole distribution of exponentially many sequences. Therefore, in contrast to previous work, we use sampling to approximate the expected objective function and its gradient. At the end, we return the best sample in terms of the objective function among all samples collected, which yields high-quality designs.

When tested on the Eterna100 benchmark, our work consistently outperforms the state-of-the-art RNA design methods (4, 6) in (almost) all metrics such as

Significance Statement

Since structure determines function in biology, we often want to design (non-coding) RNA sequences that fold into a particular structure. This is an important problem with wide applications in science and medicine, yet is notoriously difficult due to the vast design space and numerous competing structures. Previous work mostly uses local search methods such as random walk, which can not keep up with the exponential growth of the design space. We instead cast this discrete search problem as continuous optimization, and develop a general-purpose sampling-based framework which can be used for any objective function. Our work substantially outperforms existing methods in almost all metrics, especially on long and hard-to-design structures.

Author affiliations: [a]School of EECS; [b]Dept. of Quantitative and Computational Biology, University of Southern California; [c]Dept. of Biochemistry & Biophysics, Oregon State University, Corvallis, 97330, OR, USA; [d]Dept. of Biochemistry & Biophysics; [e]Center for RNA Biology; [f]Dept. of Biostatistics & Computational Biology, University of Rochester Medical Center, Rochester, 14642, NY, USA

[1]To whom correspondence should be addressed. E-mail: liang.huang.sh@gmail.com

Boltzmann probability, ensemble defect, and free energy gap. The advantage of our work is especially salient on longer and harder-to-design structures, demonstrating the advantage of this distributional approach that models the whole design space over local search methods that performs local changes on a single sequence.

1. RNA Design as Discrete Optimization

An RNA sequence $\boldsymbol{x}$ of length n is specified as a string of base nucleotides $x_1 x_2 \ldots x_n$, where $x_i \in \mathcal{N}$ ($\mathcal{N} \triangleq \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{U}\}$ is the set of nucleotides) for $i = 1, 2, ..., n$. A pseudoknot-free secondary structure for sequence $\boldsymbol{x}$ is a well-balanced dot-bracket string $\boldsymbol{y} = y_1 y_2 \ldots y_n$ where $y_i = $ "." indicates that x_i is unpaired, and $y_i = $ "(" indicates that x_i is paired with some downstream x_j and $y_i = $ ")" indicates that x_i is paired with some upstream x_j. The set of unpaired indices is denoted $unpaired(\boldsymbol{y})$ and the set of paired indices $pairs(\boldsymbol{y})$. For example, if $\boldsymbol{x} = \texttt{CCCAAAGGG}$ and $\boldsymbol{y} = (((...)))$, we have $unpaired(\boldsymbol{y}) = \{4, 5, 6\}$ and $pairs(\boldsymbol{y}) = \{(1, 9), (2, 8), (3, 7)\}$. We assume each base-pair is a Watson-Crick-Franklin or wobble pair, i.e., $\forall (i, j) \in pairs(\boldsymbol{y}), x_i x_j \in \mathcal{P}$ where $\mathcal{P} \triangleq \{\texttt{CG}, \texttt{GC}, \texttt{AU}, \texttt{UA}, \texttt{GU}, \texttt{UG}\}$.

1.A. RNA Folding. The *ensemble* of an RNA sequence $\boldsymbol{x}$ is the set of all possible secondary structures of $\boldsymbol{x}$, denoted as $\mathcal{Y}(x)$. In thermodynamic RNA folding models, *Gibbs free energy change* $\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y})$ is used to characterize the stability of $\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})$. The lower the free energy $\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y})$, the more stable the secondary structure $\boldsymbol{y}$ for $\boldsymbol{x}$. The structure with the *minimum free energy* is the most stable structure in the ensemble, i.e., *MFE structure*,

$$\mathrm{MFE}(\boldsymbol{x}) \triangleq \operatorname*{argmin}_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} \Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}). \qquad [1]$$

Note that for most methods for secondary structure prediction, ties for argmin are broken arbitrarily when there are multiple lowest free energy structures. This issue was often neglected in the literature, but it deserves clarification here. To be precise, we define

$$\mathrm{MFEs}(\boldsymbol{x}) \triangleq \{\boldsymbol{y} \mid \Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}) = \min_{y' \in \mathcal{Y}(\boldsymbol{x})} \Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}')\} \qquad [2]$$

to be the *set of MFE structures* for $\boldsymbol{x}$. When it is a singleton set, we say $\boldsymbol{x}$ has a *unique MFE* (uMFE) structure.

The *partition function* sums the contribution of all structures in an ensemble:

$$Q(\boldsymbol{x}) \triangleq \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} e^{-\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y})/RT}, \qquad [3]$$

where R is the molar gas constant and T is the absolute temperature. Accordingly, the equilibrium probability of a sequence $\boldsymbol{x}$ folding into a structure $\boldsymbol{y}$ is defined as

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{e^{-\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y})/RT}}{Q(\boldsymbol{x})}. \qquad [4]$$
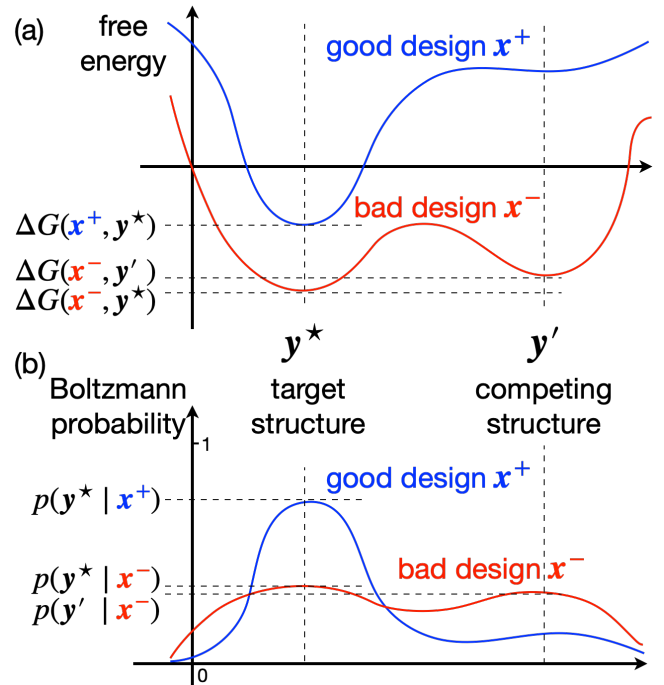

Fig. 1. RNA design criteria: (a) MFE vs. (b) Boltzmann probability. In (a), both designs $\boldsymbol{x}^+$ and $\boldsymbol{x}^-$ are MFE solutions for the target structure $\boldsymbol{y}^\star$, but $\boldsymbol{x}^-$ is not a good design due to the competing structure $\boldsymbol{y}'$ having similar free energy, which results in low probability of $\boldsymbol{y}^\star$ in the ensemble (b). By contrast, $\boldsymbol{x}^+$ is a better design with sharper energy landscape (less competition), thus higher probability of $\boldsymbol{y}^\star$ (i.e., it is more likely to fold into $\boldsymbol{y}^\star$).

1.B. RNA Design as Inverse Folding. Given a target structure $\boldsymbol{y}^\star$, RNA design aims to find a suitable RNA sequence $\boldsymbol{x}$ that can naturally and easily fold into $\boldsymbol{y}^\star$, within the design space $\mathcal{X}(\boldsymbol{y}^\star)$ of all valid sequences for $\boldsymbol{y}^\star$:

$$\mathcal{X}(\boldsymbol{y}^\star) \triangleq \{\boldsymbol{x} \in \mathcal{N}^{|\boldsymbol{y}^\star|} \mid \forall (i, j) \in pairs(\boldsymbol{y}^\star), x_i x_j \in \mathcal{P}\} \qquad [5]$$

But there are different ways to quantify how "naturally" or "easily" $\boldsymbol{x}$ folds into $\boldsymbol{y}^\star$, which we categorize into two broad groups: (a) MFE-based and (b) ensemble-based criteria.

MFE criteria and uMFE criteria Sequence $\boldsymbol{x}$ is said to be an *MFE solution* of $\boldsymbol{y}^\star$ if $\boldsymbol{y}^\star$ is one of the MFE structures of $\boldsymbol{x}$:

$$\boldsymbol{y}^\star \in \mathrm{MFEs}(\boldsymbol{x}) \qquad [6]$$

Or equivalently, $\forall \boldsymbol{y} \neq \boldsymbol{y}^\star, \Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}) \leq \Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star)$. As a stricter criteria, sequence $\boldsymbol{x}$ is said to be a *uMFE solution* of $\boldsymbol{y}^\star$ if $\boldsymbol{y}^\star$ is the unique MFE structure of $\boldsymbol{x}$, or equivalently:

$$\forall \boldsymbol{y} \neq \boldsymbol{y}^\star, \Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}) < \Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star) \qquad [7]$$

To search for an MFE or uMFE solution, we can start from a random sequence $\boldsymbol{x}$ and gradually update it to minimize one of the following metrics:

- *structural distance* $d(\boldsymbol{y}^\star, \mathrm{MFE}(\boldsymbol{x}))$, where $d(\cdot, \cdot)$ is a standard distance metric between two secondary structures, returning the number of differently folded nucleotides:

$$d(\boldsymbol{y}, \boldsymbol{y}') \triangleq |\boldsymbol{y}| - 2 \cdot |pairs(\boldsymbol{y}) \cap pairs(\boldsymbol{y}')| \\ - |unpaired(\boldsymbol{y}) \cap unpaired(\boldsymbol{y}')|. \qquad [8]$$

- *free energy gap* $\Delta\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star)$, which is the difference between the free energies of $\boldsymbol{y}^\star$ and $\mathrm{MFE}(\boldsymbol{x})$:

$$\begin{aligned}
\Delta\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star) &\overset{\triangle}{=} \Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star) - \Delta G^\circ(\boldsymbol{x}, \mathrm{MFE}(\boldsymbol{x})) \\
&= \Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star) - \min_{\boldsymbol{y}} \Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}) \geq 0
\end{aligned} \quad [9]$$

Clearly, when $d(\boldsymbol{y}^\star, \mathrm{MFE}(\boldsymbol{x}))$ or $\Delta\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star)$ reaches 0, we have an MFE solution.

Ensemble-based criteria: Boltzmann probability and ensemble defect However, the above two criteria only consider MFE structures, and neglect the other competing structures. Even if $\boldsymbol{y}^\star$ is the unique MFE structure, there could still be many highly competitive structures that are very close in energy to $\boldsymbol{y}^\star$; see design $\boldsymbol{x}^-$ in Fig. 1(a) for an example. As a result, the Boltzmann probability $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ could still be arbitrarily small due to competition, which means $\boldsymbol{x}$ is highly unlikely to fold into $\boldsymbol{y}^\star$ in equilibrium (see Fig. 1(b)). So a better criteria is to look at the whole Boltzmann ensemble to minimize the competition from alternative structures. We consider two such metrics:

- *conditional (i.e., Boltzmann) probability* $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$. Since each $\boldsymbol{x}$ has exponentially many possible structures in the ensemble, this probability can be arbitrarily small. So for numerical stability, we *minimize* the negative log probability $-\log p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ instead.

- *ensemble defect* ED $(\boldsymbol{x}, \boldsymbol{y}^\star)$, which is the expected structural distance between $\boldsymbol{y}^\star$ and all structures in the ensemble (18). This metric not only considers competition, but also how (dis)similar the competing structures are from $\boldsymbol{y}^\star$; we want to penalize highly competitive structures that are very different from $\boldsymbol{y}^\star$. The value of ensemble defect can be normalized to between 0 and 1, known as *normalized ensemble defect* (NED):

$$\begin{aligned}
\mathrm{NED}(\boldsymbol{x}, \boldsymbol{y}^\star) &\overset{\triangle}{=} \frac{1}{|\boldsymbol{x}|} \, \mathbb{E}_{\boldsymbol{y} \sim p(\cdot \mid \boldsymbol{x})} \, d(\boldsymbol{y}^\star, \boldsymbol{y}) \\
&= \frac{1}{|\boldsymbol{x}|} \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} p(\boldsymbol{y} \mid \boldsymbol{x}) \cdot d(\boldsymbol{y}^\star, \boldsymbol{y}).
\end{aligned} \quad [10]$$

By plugging in Eq. 8 and some simplifications (18, 19), we get

$$\mathrm{NED}(\boldsymbol{x}, \boldsymbol{y}^\star) = 1 - \frac{2}{|\boldsymbol{x}|} \sum_{(i,j) \in pairs(\boldsymbol{y}^\star)} p_{ij} - \frac{1}{|\boldsymbol{x}|} \sum_{j \in unpaired(\boldsymbol{y}^\star)} q_j, \quad [11]$$

where $p_i j$ is the base-pairing probability of nucleotides i and j, while $q_j = 1 - \sum_i p_{ij}$ is the probability of j being unpaired. $\mathrm{NED}(\boldsymbol{x}, \boldsymbol{y}^\star)$ can also be further decomposed into the sum of *positional defect* (ϵ_i):

$$\mathrm{NED}(\boldsymbol{x}, \boldsymbol{y}^\star) = \frac{1}{|\boldsymbol{x}|} \sum_{1 \leq i \leq |\boldsymbol{x}|} \epsilon_i(\boldsymbol{x}, \boldsymbol{y}^\star) \quad [12]$$

where

$$\epsilon_i = \begin{cases} 1 - q_i & \text{if } i \in unpaired(\boldsymbol{y}^\star); \\ 1 - p_{ij} & \text{if } (i,j) \in pairs(\boldsymbol{y}^\star) \text{ for some } j > i; \\ 1 - p_{ji} & \text{if } (j,i) \in pairs(\boldsymbol{y}^\star) \text{ for some } j < i. \end{cases} \quad [13]$$

Now we can formulate the RNA design problem as optimizing some objective function $f(\boldsymbol{x}, \boldsymbol{y}^\star)$ over the design space $\mathcal{X}(\boldsymbol{y}^\star)$:

$$\boldsymbol{x}^\star = \underset{\boldsymbol{x} \in \mathcal{X}(\boldsymbol{y}^\star)}{\operatorname{argmin}} f(\boldsymbol{x}, \boldsymbol{y}^\star) \quad [14]$$

where the objective function can be one of these four:

$$f(\boldsymbol{x}, \boldsymbol{y}^\star) = \begin{cases} d(\mathrm{MFE}(\boldsymbol{x}), \boldsymbol{y}^\star) & \text{structural distance} \\ \Delta\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star) & \text{free energy gap} \\ -\log p(\boldsymbol{y}^\star \mid \boldsymbol{x}) & \text{conditional probability} \\ \mathrm{NED}(\boldsymbol{x}, \boldsymbol{y}^\star) & \text{ensemble defect} \end{cases} \quad [15]$$

2. RNA Design as Continuous Optimization

However, it is well known that the above discrete optimization formulation is hard to optimize. For any target structure, the RNA design space is exponentially large:

$$|\mathcal{X}(\boldsymbol{y}^\star)| = 4^{|unpaired(\boldsymbol{y}^\star)|} \cdot 6^{|pairs(\boldsymbol{y}^\star)|} \quad [16]$$

But most commonly used local search methods (20–23) considers only one (or a few) candidate sequence in each step and only modifies one (or a few) nucleotides, which seems highly inefficient in exploring the exponentially large design space.

Can we instead modify *all* positions of the candidate sequence in each step, or consider *all* candidate sequences simultaneously and promote the better ones? Here we replace the discrete representation of a single candidate sequence by a probability distribution $p_{\boldsymbol{y}^\star}(\boldsymbol{x})$ over all possible sequences $\boldsymbol{x}$ in $\mathcal{X}(\boldsymbol{y}^\star)$. Essentially, we propose the following continuous relaxation of the optimization problem minimizing the following objective function:

$$\mathcal{J} \overset{\triangle}{=} \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{y}^\star}(\cdot)} [f(\boldsymbol{x}, \boldsymbol{y}^\star)] = \sum_{\boldsymbol{x} \in \mathcal{X}(\boldsymbol{y}^\star)} p_{\boldsymbol{y}^\star}(\boldsymbol{x}) f(\boldsymbol{x}, \boldsymbol{y}^\star) \quad [17]$$

This new objective function is to find a *distribution* $p_{\boldsymbol{y}^\star}(\cdot)$ *of RNA candidates* whose expectation of the objective function $f(\boldsymbol{x}, \boldsymbol{y}^\star)$ is minimized. If the probability mass concentrates on only one sequence, then this new relaxed objective degenerates to the original discrete objective.

The way of modeling the probability distribution $p_{\boldsymbol{y}^\star}(\cdot)$ over the design space $\mathcal{X}(\boldsymbol{y}^\star)$ could potentially affect the complexity of the optimization and the convergence of the final solution. We aim to find a method that can represent the design space efficiently while being easy to manage.

2.A. Independent distributions (v0). The most obvious modeling of $p_{\boldsymbol{y}}(\boldsymbol{x})$ is to use an independent distribution over $\mathcal{N} = \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{U}\}$ for each position, so that the distribution over sequences is simply the product of individual distributions:

$$p_{\boldsymbol{y}}^0(\boldsymbol{x}) \overset{\triangle}{=} \prod_i p_i(x_i) \quad [18]$$

This is the same distribution in previous work (16). However, this distribution is simplistic and overlooks the fact that each base-pair $(i,j) \in \boldsymbol{y}$ requires $x_i x_j$ to be one of the 6 possible pairs in $\mathcal{P}$, which is impossible with independent variables ($4 \times 4 = 16$ choices for $x_i x_j$). As a result, the domain of this $p_{\boldsymbol{y}}^0(\cdot)$ distribution is *all possible sequences* $\mathcal{N}^{|\boldsymbol{y}|}$ (of size $4^{|\boldsymbol{y}|}$) rather than the set $\mathcal{X}(\boldsymbol{y})$ of *valid sequences* (of
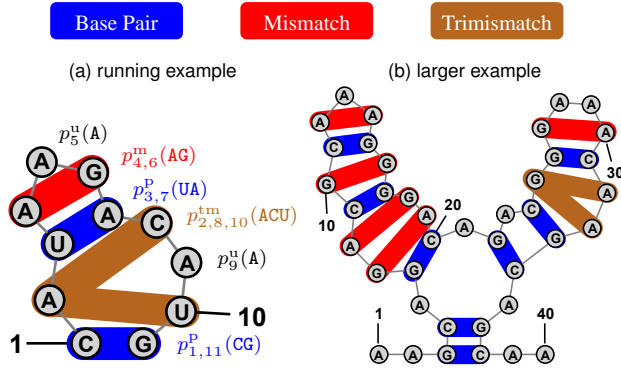
Fig. 2. Positions of mismatches and trimismatches in different types of loops.

size $4^{|unpaired(\boldsymbol{y})|} \cdot 6^{|pairs(\boldsymbol{y})|}$). In other words, for any input structure $\boldsymbol{y}^\star$ (except for the trivial case of fully-unpaired), this naive distribution includes (exponentially many) invalid sequences.

2.B. Coupled variables for pairs (v1). In order to model the dependencies between paired positions, we separate the positions into two groups: the set of unpaired indices, denoted $unpaired(\boldsymbol{y})$, and the set of paired indices, $pairs(\boldsymbol{y})$. We now factorize the joint distribution of the entire sequence as

$$p_{\boldsymbol{y}}^1(\boldsymbol{x}) \triangleq \prod_{i \in unpaired(\boldsymbol{y})} p_i^{\mathrm{u}}(x_i) \cdot \prod_{(i,j) \in pairs(\boldsymbol{y})} p_{i,j}^{\mathrm{P}}(x_i x_j)$$

where $p_i^{\mathrm{u}}(\cdot)$ is the local distribution (over $\mathcal{N}$) for unpaired position i, and $p_{i,j}^{\mathrm{P}}(\cdot)$ is the local distribution (over 6 choices in $\mathcal{P}$) for paired positions (i,j). This is the first distribution over the set of valid sequences $\mathcal{X}(\boldsymbol{y})$ for a given $\boldsymbol{y}$.

As an example, consider Fig. 2(a). Here $\boldsymbol{y}^\star = (.(...)...)$, so $unpaired(\boldsymbol{y}^\star) = \{2,4,5,6,8,9,10\}$ and $pairs(\boldsymbol{y}^\star) = \{(1,11),(3,7)\}$. The probability distribution of the design space is factorized as:

$$p_{\boldsymbol{y}^\star}^1(\boldsymbol{x}) = p_2^{\mathrm{u}}(x_2) \cdot p_4^{\mathrm{u}}(x_4) \cdot p_5^{\mathrm{u}}(x_5) \cdot p_6^{\mathrm{u}}(x_6) \cdot p_8^{\mathrm{u}}(x_8)$$
$$\cdot p_9^{\mathrm{u}}(x_9) \cdot p_{10}^{\mathrm{u}}(x_{10}) \cdot p_{1,11}^{\mathrm{P}}(x_1 x_{11}) \cdot p_{3,7}^{\mathrm{P}}(x_3 x_7)$$

Therefore, for the particular design in Fig. 2(a),

$$p_{\boldsymbol{y}^\star}^1(\mathtt{CAUAAGACAUG}) = p_2^{\mathrm{u}}(\mathtt{A}) \cdot p_4^{\mathrm{u}}(\mathtt{A}) \cdot p_5^{\mathrm{u}}(\mathtt{A}) \cdot p_6^{\mathrm{u}}(\mathtt{G}) \cdot p_8^{\mathrm{u}}(\mathtt{C})$$
$$p_9^{\mathrm{u}}(\mathtt{A}) \cdot p_{10}^{\mathrm{u}}(\mathtt{U}) \cdot p_{1,11}^{\mathrm{P}}(\mathtt{CG}) \cdot p_{3,7}^{\mathrm{P}}(\mathtt{UA})$$

2.C. Coupled variables for terminal mismatches (v2 and v3). The next two versions (v2 and v3) are refinements of the above v1. First, we note that in the standard energy models (Turner rules (24)), there are terminal mismatches lookup tables. For example, for a hairpin loop defined by the pair (i,j), the first and last nucleotide of the loop, x_{i+1} and x_{j-1}, are the terminal mismatch, and will be looked up together in the energy table (such as x_4 and x_6 in Fig. 2(a)). Therefore, it is better to make a coupled variable over $\mathcal{N}^2$ ($4 \times 4 = 16$ choices) for each terminal mismatch position-pair:

$$p_{\boldsymbol{y}}^2(\boldsymbol{x}) \triangleq \prod_{i \in unpaired(\boldsymbol{y})} p_i^{\mathrm{u}}(x_i) \cdot \prod_{(i,j) \in pairs(\boldsymbol{y})} p_{i,j}^{\mathrm{P}}(x_i x_j) \cdot \prod_{(i,j) \in mismatches(\boldsymbol{y})} p_{i,j}^{\mathrm{m}}(x_i x_j)$$

Moreover, there is a special case that deserves our attention. Let us consider the 1-by-3 internal loop in Fig. 2.

For such 1-by-x ($x > 1$) internal loops, there is exactly one unpaired nucleotide on one of the two branches, and that single unpaired nucleotide (x_2 in our example) is included in two mismatches (x_2 and x_8 on one side and x_2 and x_{10} on the other). Therefore, it is better to model all these three nucleotides together in a coupled variable over $\mathcal{N}^3$ ($4^3 = 64$ choices), which we call a "trimismatch":

$$p_{\boldsymbol{y}}^3(\boldsymbol{x}) \triangleq \prod_{i \in unpaired(\boldsymbol{y})} p_i^{\mathrm{u}}(x_i) \cdot \prod_{(i,j) \in pairs(\boldsymbol{y})} p_{i,j}^{\mathrm{P}}(x_i x_j)$$
$$\cdot \prod_{(i,j) \in mismatches(\boldsymbol{y})} p_{i,j}^{\mathrm{m}}(x_i x_j) \cdot \prod_{(i,j,k) \in trimismatches(\boldsymbol{y})} p_{i,j,k}^{\mathrm{tm}}(x_i x_j x_k)$$

Note that, a 1-by-1 internal loop is a special case of mismatch (see Fig. 2(b), nucleotides 10 and 16).

Now for the example structure $\boldsymbol{y}^\star$ in Fig. 2(a), our final joint distribution is:

$$p_{\boldsymbol{y}^\star}^3(\boldsymbol{x}) = p_5^{\mathrm{u}}(x_5) \cdot p_9^{\mathrm{u}}(x_9) \cdot p_{1,11}^{\mathrm{P}}(x_1 x_{11}) \cdot p_{3,7}^{\mathrm{P}}(x_3 x_7)$$
$$\cdot p_{4,6}^{\mathrm{m}}(x_4 x_6) \cdot p_{2,8,10}^{\mathrm{tm}}(x_2 x_8 x_{10})$$

And for the particular design in Fig. 2(a),

$$p_{\boldsymbol{y}^\star}^3(\mathtt{CAUAAGACAUG}) = p_5^{\mathrm{u}}(\mathtt{A}) \cdot p_9^{\mathrm{u}}(\mathtt{A}) \cdot p_{1,11}^{\mathrm{P}}(\mathtt{CG}) \cdot p_{3,7}^{\mathrm{P}}(\mathtt{UA})$$
$$\cdot p_{4,6}^{\mathrm{m}}(\mathtt{AG}) \cdot p_{2,8,10}^{\mathrm{tm}}(\mathtt{ACU})$$

3. Sampling for Objective Evaluation, Gradient Estimation, and Design Space Exploration

Given the complexity and variety of RNA design problem settings, a method that can seamlessly switch between various objective functions $f(\boldsymbol{x}, \boldsymbol{y}^\star)$ is desirable. Even though we factorize $p_{\boldsymbol{y}^\star}(\cdot)$ into many tractable local distributions, without making any assumptions or requirements about the structure of $f(\boldsymbol{x}, \boldsymbol{y}^\star)$, the exact calculation of the expectation $\mathcal{J} = \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{y}^\star}(\cdot)}[f(\boldsymbol{x}, \boldsymbol{y}^\star)]$ is generally intractable. Therefore, we employ gradient descent for optimization and adopt random sampling for estimating the objective and its gradients.

3.A. Sampling for Objective Evaluation. We approximate the expectation by averaging over a set of samples $\mathcal{S}$ from the distribution:

$$\mathcal{J} = \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{y}^\star}(\cdot)}[f(\boldsymbol{x}, \boldsymbol{y}^\star)] \approx \frac{1}{|\mathcal{S}|} \sum_{\boldsymbol{x} \in \mathcal{S}} f(\boldsymbol{x}, \boldsymbol{y}^\star)$$

where $\mathcal{S} = \{\boldsymbol{x}^{(l)} \sim p_{\boldsymbol{y}^\star}(\cdot)\}_{l=1}^{|\mathcal{S}|}$

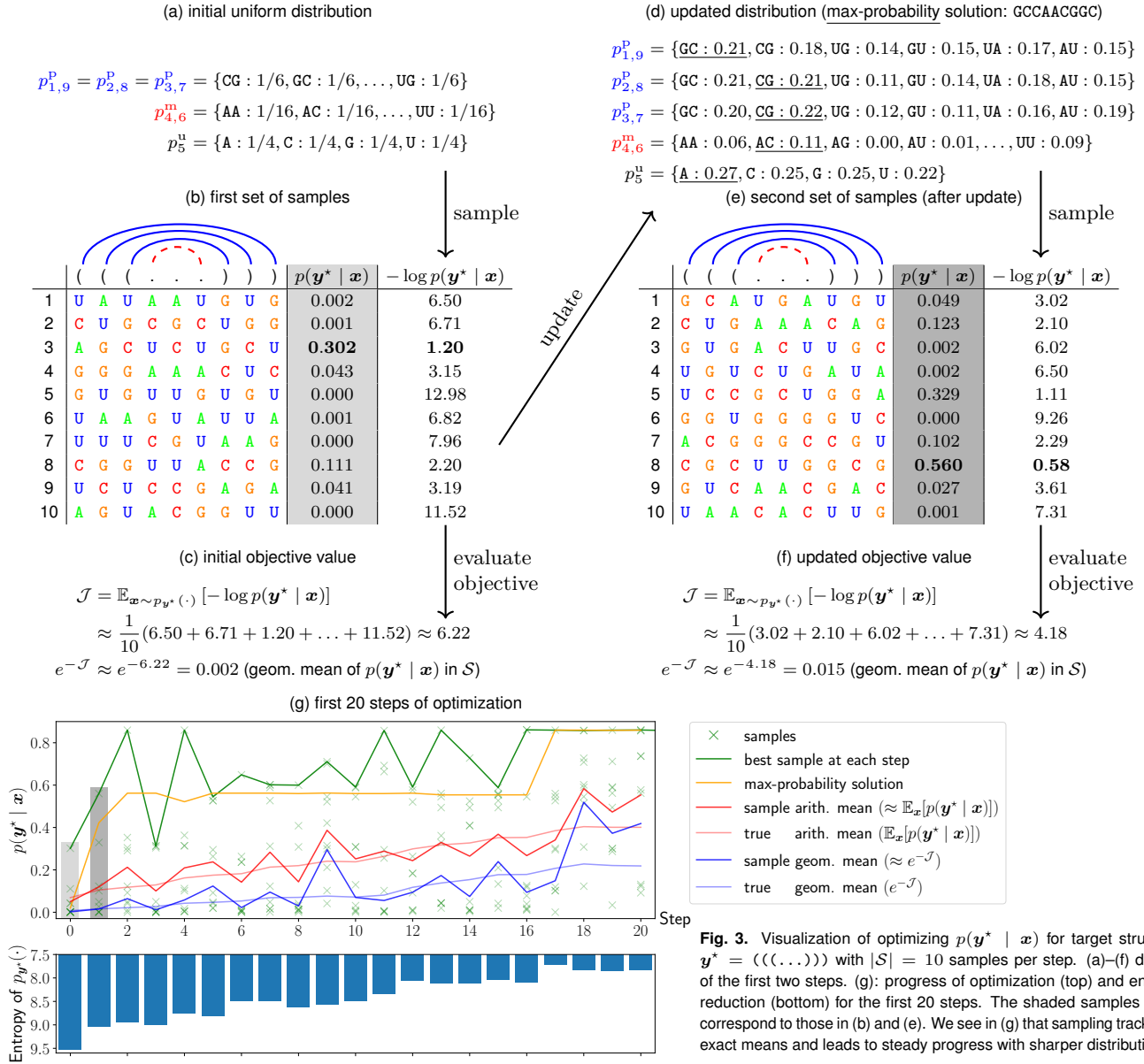Theoretically, as $|\mathcal{S}| \to \infty$, this approximation will converge to the true expectation.

(a) initial uniform distribution

$$p^{\mathrm{P}}_{1,9} = p^{\mathrm{P}}_{2,8} = p^{\mathrm{P}}_{3,7} = \{\texttt{CG}:1/6, \texttt{GC}:1/6, \ldots, \texttt{UG}:1/6\}$$
$$p^{\mathrm{m}}_{4,6} = \{\texttt{AA}:1/16, \texttt{AC}:1/16, \ldots, \texttt{UU}:1/16\}$$
$$p^{\mathrm{u}}_5 = \{\texttt{A}:1/4, \texttt{C}:1/4, \texttt{G}:1/4, \texttt{U}:1/4\}$$

(d) updated distribution (max-probability solution: GCCAACGGC)

$$p^{\mathrm{P}}_{1,9} = \{\texttt{GC}:0.21, \texttt{CG}:0.18, \texttt{UG}:0.14, \texttt{GU}:0.15, \texttt{UA}:0.17, \texttt{AU}:0.15\}$$
$$p^{\mathrm{P}}_{2,8} = \{\texttt{GC}:0.21, \texttt{CG}:0.21, \texttt{UG}:0.11, \texttt{GU}:0.14, \texttt{UA}:0.18, \texttt{AU}:0.15\}$$
$$p^{\mathrm{P}}_{3,7} = \{\texttt{GC}:0.20, \texttt{CG}:0.22, \texttt{UG}:0.12, \texttt{GU}:0.11, \texttt{UA}:0.16, \texttt{AU}:0.19\}$$
$$p^{\mathrm{m}}_{4,6} = \{\texttt{AA}:0.06, \texttt{AC}:0.11, \texttt{AG}:0.00, \texttt{AU}:0.01, \ldots, \texttt{UU}:0.09\}$$
$$p^{\mathrm{u}}_5 = \{\texttt{A}:0.27, \texttt{C}:0.25, \texttt{G}:0.25, \texttt{U}:0.22\}$$

(b) first set of samples

	Sequence	$p(\boldsymbol{y}^\star \mid \boldsymbol{x})$	$-\log p(\boldsymbol{y}^\star \mid \boldsymbol{x})$
1	U A U A A U G U G	0.002	6.50
2	C U G C G C U G G	0.001	6.71
3	A G C U C U G C U	**0.302**	**1.20**
4	G G G A A A C U C	0.043	3.15
5	G U G U U G U G U	0.000	12.98
6	U A A G U A U U A	0.001	6.82
7	U U U C G U A A G	0.000	7.96
8	C G G U U A C C G	0.111	2.20
9	U C U C C G A G A	0.041	3.19
10	A G U A C G G U U	0.000	11.52

(e) second set of samples (after update)

	Sequence	$p(\boldsymbol{y}^\star \mid \boldsymbol{x})$	$-\log p(\boldsymbol{y}^\star \mid \boldsymbol{x})$
1	G C A U G A U G U	0.049	3.02
2	C U G A A A C A G	0.123	2.10
3	G U G A C U U G C	0.002	6.02
4	U G U C U G A U A	0.002	6.50
5	U C C G C U G G A	0.329	1.11
6	G G U G G G G U C	0.000	9.26
7	A C G G G C C G U	0.102	2.29
8	C G C U U G G C G	**0.560**	**0.58**
9	G U C A A C G A C	0.027	3.61
10	U A A C A C U U G	0.001	7.31

(c) initial objective value

$$\mathcal{J} = \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{y}^\star}(\cdot)} \left[-\log p(\boldsymbol{y}^\star \mid \boldsymbol{x}) \right]$$
$$\approx \frac{1}{10}(6.50 + 6.71 + 1.20 + \ldots + 11.52) \approx 6.22$$
$$e^{-\mathcal{J}} \approx e^{-6.22} = 0.002 \ \text{(geom. mean of } p(\boldsymbol{y}^\star \mid \boldsymbol{x}) \text{ in } \mathcal{S})$$

(f) updated objective value

$$\mathcal{J} = \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{y}^\star}(\cdot)} \left[-\log p(\boldsymbol{y}^\star \mid \boldsymbol{x}) \right]$$
$$\approx \frac{1}{10}(3.02 + 2.10 + 6.02 + \ldots + 7.31) \approx 4.18$$
$$e^{-\mathcal{J}} \approx e^{-4.18} = 0.015 \ \text{(geom. mean of } p(\boldsymbol{y}^\star \mid \boldsymbol{x}) \text{ in } \mathcal{S})$$

(g) first 20 steps of optimization

- × samples
- best sample at each step
- max-probability solution
- sample arith. mean ($\approx \mathbb{E}_{\boldsymbol{x}}[p(\boldsymbol{y}^\star \mid \boldsymbol{x})]$)
- true arith. mean ($\mathbb{E}_{\boldsymbol{x}}[p(\boldsymbol{y}^\star \mid \boldsymbol{x})]$)
- sample geom. mean ($\approx e^{-\mathcal{J}}$)
- true geom. mean ($e^{-\mathcal{J}}$)

Fig. 3. Visualization of optimizing $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ for target structure $\boldsymbol{y}^\star = (((\ldots)))$ with $|\mathcal{S}| = 10$ samples per step. (a)–(f) details of the first two steps. (g): progress of optimization (top) and entropy reduction (bottom) for the first 20 steps. The shaded samples in (g) correspond to those in (b) and (e). We see in (g) that sampling tracks the exact means and leads to steady progress with sharper distributions.

3.B. Sampling for Gradient Estimation.

Next, we derive the true gradient of $\mathcal{J}$ with respect to $p_{\boldsymbol{y}^\star}(\cdot)$ as:

$$\nabla_{p_{\boldsymbol{y}^\star}} \mathcal{J}$$
$$= \nabla_{p_{\boldsymbol{y}^\star}} \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{y}^\star}(\cdot)} \left[f(\boldsymbol{x}, \boldsymbol{y}^\star) \right]$$
$$= \nabla_{p_{\boldsymbol{y}^\star}} \sum_{\boldsymbol{x} \in \mathcal{X}(\boldsymbol{y}^\star)} p_{\boldsymbol{y}^\star}(\boldsymbol{x}) f(\boldsymbol{x}, \boldsymbol{y}^\star) \quad \text{(def. of expectation)}$$
$$= \sum_{\boldsymbol{x} \in \mathcal{X}(\boldsymbol{y}^\star)} \nabla_{p_{\boldsymbol{y}^\star}} p_{\boldsymbol{y}^\star}(\boldsymbol{x}) f(\boldsymbol{x}, \boldsymbol{y}^\star) \quad \text{(linearity of } \nabla)$$
$$= \sum_{\boldsymbol{x} \in \mathcal{X}(\boldsymbol{y}^\star)} p_{\boldsymbol{y}^\star}(\boldsymbol{x}) \frac{\nabla_{p_{\boldsymbol{y}^\star}} p_{\boldsymbol{y}^\star}(\boldsymbol{x})}{p_{\boldsymbol{y}^\star}(\boldsymbol{x})} f(\boldsymbol{x}, \boldsymbol{y}^\star)$$
$$= \sum_{\boldsymbol{x} \in \mathcal{X}(\boldsymbol{y}^\star)} p_{\boldsymbol{y}^\star}(\boldsymbol{x}) \nabla_{p_{\boldsymbol{y}^\star}} \log p_{\boldsymbol{y}^\star}(\boldsymbol{x}) f(\boldsymbol{x}, \boldsymbol{y}^\star) \quad \left((\log f(x))' = \frac{f'(x)}{f(x)} \right)$$
$$= \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{y}^\star}(\cdot)} \left[\nabla_{p_{\boldsymbol{y}^\star}} \log p_{\boldsymbol{y}^\star}(\boldsymbol{x}) f(\boldsymbol{x}, \boldsymbol{y}^\star) \right] \quad \text{(def. of expectation)}$$

We again use sampling to estimate the above expectation and derive the approximate gradient for step t:

$$\nabla_{p_{\boldsymbol{y}^\star}} \mathcal{J}^{(t)} \approx \frac{1}{|\mathcal{S}|} \sum_{\boldsymbol{x} \in \mathcal{S}} \nabla_{p_{\boldsymbol{y}^\star}} \log p_{\boldsymbol{y}^\star}(\boldsymbol{x}) f(\boldsymbol{x}, \boldsymbol{y}^\star) \quad [19]$$

$$\mathcal{S}^{(t)} \leftarrow \{\boldsymbol{x}^{(l)} \sim p_{\boldsymbol{y}^\star}^{(t)}(\cdot)\}_{l=1}^N \quad [20]$$

We then use the approximate gradient to update the distribution (see Sec. 4 for details):

$$p_{\boldsymbol{y}^\star}^{(t+1)} \leftarrow \text{update} \left(p_{\boldsymbol{y}^\star}^{(t)}, \nabla_{p_{\boldsymbol{y}^\star}} \mathcal{J}^{(t)} \right) \quad [21]$$

3.C. Sampling-based Design Space Exploration.

At the end of this continuous optimization, we still need to return a single sequence from the distribution, i.e., an "integral solution". This can be done by "rounding" if the distribution is close to one-hot, or more generally by taking the sequence with the highest probability in the final distribution

Tang *et al.*

PNAS | **December 13, 2024** | vol. XXX | no. XX | **5**

$$\boldsymbol{x}^{\star} = \operatorname*{argmax}_{\boldsymbol{x}} p_{\boldsymbol{y}^{\star}}(\boldsymbol{x}) \qquad [22]$$

For example, for independent distributions (v0), since each position is isolated, we simply take the best nucleotide for each position: $x_i^{\star} = \operatorname{argmax}_{a \in \mathcal{N}} p_i(a)$. But for the coupled variable distribution (v1), for each unpaired position $i \in \textit{unpaired}(y^*)$, we take $x_i^{\star} = \operatorname{argmax}_{a \in \mathcal{N}} p_i^{\mathrm{u}}(a)$ same as in v0, and for each paired position-pair $(i, j) \in \textit{pairs}(y^*)$, we take the best pair out of the six pair types: $x_i^{\star} x_j^{\star} = \operatorname{argmax}_{ab \in \mathcal{P}} p_{i,j}^{\mathrm{p}}(ab)$.

However, this max-probability sequence is not necessarily the best sequence in terms of the objective function, since the distribution is often not perfectly aligned with objective function. Here we use an alternative approach that simply takes the best sample in terms of the objective function out of all samples collected in the optimization process:

$$\boldsymbol{x}^{(t)} = \operatorname*{argmin}_{\boldsymbol{x} \in \mathcal{S}^{(t)}} f(\boldsymbol{x}, \boldsymbol{y}^{\star}) \qquad [23]$$

$$\boldsymbol{x}^{\star} = \operatorname*{argmin}_{t} \boldsymbol{x}^{(t)} \qquad [24]$$

This method, which we call "sampling-based candidate exploration", outperforms the max-probability solution, because the samples offer much more diversity in the exploration of the distribution than a single sequence.

4. Parameterization and Optimization

Now we turn to the question of how to parameterize the factorized distribution $p_{\boldsymbol{y}^{\star}}(\cdot)$ as $p_{\boldsymbol{y}^{\star}}(\cdot; \boldsymbol{\Theta})$. The first method (Sec. 4.A) simply uses $\boldsymbol{\Theta}$ as raw probabilities, but the update of $\boldsymbol{\Theta}$ needs to result in probabilities, leading to a harder *constrained* optimization problem. The second method (Sec. 4.B) models the distribution implicitly by applying softmax on $\boldsymbol{\Theta}$, resulting in a simpler *unconstrained* optimization problem.

4.A. Method 1: Direct Parameterization and Constrained Optimization. The obvious way of parameterization is to use *explicit* probabilities. For each unpaired position i, we use a non-negative parameter vector $\boldsymbol{\theta}_i^{\mathrm{u}} \triangleq (\theta_{i,\mathtt{A}}^{\mathrm{u}}, \theta_{i,\mathtt{C}}^{\mathrm{u}}, \theta_{i,\mathtt{G}}^{\mathrm{u}}, \theta_{i,\mathtt{U}}^{\mathrm{u}})$ which sums to 1 as the probability distribution over nucleotides:

$$\forall a \in \mathcal{N}, p_i^{\mathrm{u}}(a; \boldsymbol{\theta}_i^{\mathrm{u}}) \triangleq \theta_{i,a}^{\mathrm{u}}.$$

Similarly for each paired position (i, j), we use a non-negative parameter vector $\boldsymbol{\theta}_{i,j}^{\mathrm{p}} \triangleq (\theta_{i,j,\mathtt{CG}}^{\mathrm{p}}, \theta_{i,j,\mathtt{GC}}^{\mathrm{p}}, \dots, \theta_{i,j,\mathtt{UG}}^{\mathrm{p}})$ which sums to 1, and we have

$$\forall ab \in \mathcal{P}, p_{ij}^{\mathrm{p}}(ab; \boldsymbol{\theta}_{i,j}^{\mathrm{p}}) \triangleq \theta_{i,j,ab}^{\mathrm{p}}.$$

The cases for mismatches and trimismatches are also similar. The whole parameter set $\boldsymbol{\Theta}$ includes all parameter vectors:

$$\begin{aligned}
\boldsymbol{\Theta} = & \{\boldsymbol{\theta}_i^{\mathrm{u}} \mid i \in \textit{unpaired}(\boldsymbol{y}^{\star})\} \\
& \cup \{\boldsymbol{\theta}_{i,j}^{\mathrm{p}} \mid (i, j) \in \textit{pairs}(\boldsymbol{y}^{\star})\} \\
& \cup \{\boldsymbol{\theta}_{i,j}^{\mathrm{m}} \mid (i, j) \in \textit{mismatches}(\boldsymbol{y}^{\star})\} \\
& \cup \{\boldsymbol{\theta}_{i,j,k}^{\mathrm{tm}} \mid (i, j, k) \in \textit{trimismatches}(\boldsymbol{y}^{\star})\}
\end{aligned} \qquad [25]$$

where each $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ is a distribution, i.e.,

$$\forall \theta_a \in \boldsymbol{\theta}, \theta_a \in [0, 1], \text{ and } \sum\nolimits_{\theta_a \in \boldsymbol{\theta}} \theta_a = 1.$$

For example, for the structure y^* in Fig. 2(a), we have its parameters as $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_5^{\mathrm{u}}, \boldsymbol{\theta}_9^{\mathrm{u}}, \boldsymbol{\theta}_{1,11}^{\mathrm{u}}, \boldsymbol{\theta}_{3,7}^{\mathrm{p}}, \boldsymbol{\theta}_{4,6}^{\mathrm{p}}, \boldsymbol{\theta}_{2,8,10}^{\mathrm{tm}}\}$.

Now we can parameterize the whole distribution as

$$\begin{aligned}
p_{\boldsymbol{y}}^3(\boldsymbol{x}; \boldsymbol{\Theta}) = & \prod_{i \in \textit{unpaired}(\boldsymbol{y})} p_i^{\mathrm{u}}(x_i; \boldsymbol{\theta}_i^{\mathrm{u}}) \cdot \prod_{(i,j) \in \textit{pairs}(\boldsymbol{y})} p_{i,j}^{\mathrm{p}}(x_i x_j; \boldsymbol{\theta}_{i,j}^{\mathrm{p}}) \\
& \cdot \prod_{(i,j) \in \textit{mismatches}(\boldsymbol{y})} p_{i,j}^{\mathrm{m}}(x_i x_j; \boldsymbol{\theta}_{i,j}^{\mathrm{m}}) \\
& \cdot \prod_{(i,j,k) \in \textit{trimismatches}(\boldsymbol{y})} p_{i,j,k}^{\mathrm{tm}}(x_i x_j x_k; \boldsymbol{\theta}_{i,j,k}^{\mathrm{tm}}) \\
\triangleq & \prod_{i \in \textit{unpaired}(\boldsymbol{y})} \theta_{i,x_i}^{\mathrm{u}} \cdot \prod_{(i,j) \in \textit{pairs}(\boldsymbol{y})} \theta_{i,j,x_i x_j}^{\mathrm{p}} \cdot \\
& \prod_{(i,j) \in \textit{mismatches}(\boldsymbol{y})} \theta_{i,j,x_i x_j}^{\mathrm{m}} \cdot \prod_{(i,j,k) \in \textit{trimismatches}(\boldsymbol{y})} \theta_{i,j,k,x_i x_j x_k}^{\mathrm{tm}}
\end{aligned}$$

Now we adopt a parameterized version for our objective function:

$$\mathcal{J}(\boldsymbol{\Theta}) = \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{y}^{\star}}(\cdot; \boldsymbol{\Theta})} f(\boldsymbol{x}, \boldsymbol{y}^{\star})$$

The optimization problem can then be formulated as a constrained optimization

$$\begin{aligned}
\min_{\boldsymbol{\Theta}} \quad & \mathcal{J}(\boldsymbol{\Theta}) \\
\text{s.t.} \quad & \text{each } \boldsymbol{\theta} \in \boldsymbol{\Theta} \text{ is a distribution.}
\end{aligned} \qquad [26]$$

To solve this constrained optimization problem, we use the Projected Gradient Descent (PGD) method (25). At each step t, we first perform a gradient descent (with learning rate α):

$$\widehat{\boldsymbol{\Theta}} \leftarrow \boldsymbol{\Theta} - \alpha \nabla_{\boldsymbol{\Theta}} \mathcal{J}(\boldsymbol{\Theta}) \qquad [27]$$

where the gradient components are computed individually for each parameter vector:

$$\nabla_{\boldsymbol{\Theta}} \mathcal{J}(\boldsymbol{\Theta}) = \{\frac{\partial \mathcal{J}(\boldsymbol{\Theta})}{\partial \boldsymbol{\theta}} \mid \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$$

For example, for an unpaired position i, we have:

$$\frac{\partial \mathcal{J}(\boldsymbol{\Theta})}{\partial \boldsymbol{\theta}_i^{\mathrm{u}}} = (\frac{\partial \mathcal{J}(\boldsymbol{\Theta})}{\partial \theta_{i,\mathtt{A}}^{\mathrm{u}}}, \frac{\partial \mathcal{J}(\boldsymbol{\Theta})}{\partial \theta_{i,\mathtt{C}}^{\mathrm{u}}}, \frac{\partial \mathcal{J}(\boldsymbol{\Theta})}{\partial \theta_{i,\mathtt{G}}^{\mathrm{u}}}, \frac{\partial \mathcal{J}(\boldsymbol{\Theta})}{\partial \theta_{i,\mathtt{U}}^{\mathrm{u}}})$$

The first component can be estimated using Eq. 19 as follows:

$$\frac{\partial \mathcal{J}(\boldsymbol{\Theta})}{\partial \theta_{i,\mathtt{A}}^{\mathrm{u}}} \approx \frac{1}{|\mathcal{S}|} \sum_{\boldsymbol{x} \in \mathcal{S}} \frac{\partial \log p_{\boldsymbol{y}^{\star}}(\boldsymbol{x}; \boldsymbol{\Theta})}{\partial \theta_{i,\mathtt{A}}^{\mathrm{u}}} f(\boldsymbol{x}, \boldsymbol{y}^{\star}) \qquad [28]$$

Expanding the term $\log p_{\boldsymbol{y}^{\star}}(\boldsymbol{x}; \boldsymbol{\Theta})$, the gradient can be simplified (details provided in Sec. S1.A) as:

$$\frac{\partial \mathcal{J}(\boldsymbol{\Theta})}{\partial \theta_{i,\mathtt{A}}^{\mathrm{u}}} \approx \frac{1}{|\mathcal{S}|} \sum_{\boldsymbol{x} \in \mathcal{S}} \mathbb{1}[x_i = A] \frac{f(\boldsymbol{x}, \boldsymbol{y}^{\star})}{\theta_{i,\mathtt{A}}^{\mathrm{u}}} = \frac{1}{|\mathcal{S}|} \sum_{\substack{\boldsymbol{x} \in \mathcal{S} \\ x_i = A}} \frac{f(\boldsymbol{x}, \boldsymbol{y}^{\star})}{\theta_{i,\mathtt{A}}^{\mathrm{u}}}$$

$$[29]$$

After the gradient update (Eq. 27), we then project $\widehat{\boldsymbol{\Theta}}$ back onto the set of valid distributions. For each $\widehat{\boldsymbol{\theta}}$ in $\widehat{\boldsymbol{\Theta}}$, we project it back to the probability simplex by finding the vector in the simplex that is closest (in ℓ_2 norm) to $\widehat{\boldsymbol{\theta}}$:

$$\boldsymbol{\Theta}' \leftarrow \{proj(\widehat{\boldsymbol{\theta}}) \mid \widehat{\boldsymbol{\theta}} \in \widehat{\boldsymbol{\Theta}}\}$$

$$proj(\widehat{\boldsymbol{\theta}}) \triangleq \operatorname*{argmin}_{\boldsymbol{\theta}} \quad \|\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta}\|_2^2$$

$$\text{s.t.} \quad \boldsymbol{\theta} \text{ is a distribution.}$$

4.B. Method 2: Softmax Parameterization and Unconstrained Optimization.

An alternative approach to the optimization problem is to introduce a parametrization that naturally enforces the required normalization for a valid distribution, thus converting the problem into an unconstrained optimization problem. This approach eliminates the need for performing gradient projection at each step. A common choice for achieving this normalization is the *softmax* function, which inherently converts a set of real numbers into a valid probability distribution.

Instead of using a parameter vector as a distribution explicitly, now we model a distribution implicitly using softmax and our new parameter vector $\boldsymbol{\theta}_i^{\mathrm{u}} \triangleq (\theta_{i,\mathtt{A}}^{\mathrm{u}}, \theta_{i,\mathtt{C}}^{\mathrm{u}}, \theta_{i,\mathtt{G}}^{\mathrm{u}}, \theta_{i,\mathtt{U}}^{\mathrm{u}})$ no longer sums to 1; instead we have:

$$\forall a \in \mathcal{N}, p_i^{\mathrm{u}}(a; \boldsymbol{\theta}_i^{\mathrm{u}}) \triangleq \frac{\exp(\theta_{i,a}^{\mathrm{u}})}{\sum_{a'} \exp(\theta_{i,a'}^{\mathrm{u}})} \qquad [30]$$

where $\theta_{i,a}^{\mathrm{u}}$ can be any real number. The softmax function ensures that each $p_i^{\mathrm{u}}(\cdot; \boldsymbol{\theta}_i^{\mathrm{u}})$ forms a valid distribution without explicitly imposing this as a constraint. This definition can be extended for other parameter vectors $\theta_{i,j}^{\mathrm{p}}$, $\theta_{i,j}^{\mathrm{m}}$, and $\theta_{i,j,k}^{\mathrm{tm}}$ as follows:

$$\forall ab \in \mathcal{P}, p_{i,j}^{\mathrm{p}}(ab; \boldsymbol{\theta}_{i,j}^{\mathrm{p}}) \triangleq \frac{\exp(\theta_{i,j,ab}^{\mathrm{p}})}{\sum_{a'b' \in \mathcal{P}} \exp(\theta_{i,j,a'b'}^{\mathrm{p}})}$$

$$\forall ab \in \mathcal{N}^2, p_{i,j}^{\mathrm{m}}(ab; \boldsymbol{\theta}_{i,j}^{\mathrm{m}}) \triangleq \frac{\exp(\theta_{i,j,ab}^{\mathrm{m}})}{\sum_{a'b' \in \mathcal{N}^2} \exp(\theta_{i,j,a'b'}^{\mathrm{m}})}$$

$$\forall abc \in \mathcal{N}^3, p_{i,j,k}^{\mathrm{tm}}(abc; \boldsymbol{\theta}_{i,j,k}^{\mathrm{tm}}) \triangleq \frac{\exp(\theta_{i,j,k,abc}^{\mathrm{tm}})}{\sum_{a'b'c' \in \mathcal{N}^3} \exp(\theta_{i,j,k,a'b'c'}^{\mathrm{tm}})}$$

With this new parametrization, the optimization problem becomes an unconstrained problem:

$$\min_{\boldsymbol{\Theta}} \quad \mathcal{J}(\boldsymbol{\Theta}) \qquad [31]$$

Since the constraints have been naturally embedded into the problem formulation through the softmax function, we can directly apply the vanilla gradient descent algorithm to solve this optimization problem. The gradient can be updated by:

$$\boldsymbol{\Theta}' \leftarrow \boldsymbol{\Theta} - \alpha \nabla_{\boldsymbol{\Theta}} \mathcal{J}(\boldsymbol{\Theta})$$

where α is the learning rate.

Due to the softmax parametrization, the specific form of the gradient $\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\Theta})$ differs from that in the constrained optimization problem. Using the chain rule, we express it as:

$$\frac{\partial \log p_{\boldsymbol{y}^\star}(\boldsymbol{x}; \boldsymbol{\Theta})}{\partial \theta_{i,\mathtt{A}}^{\mathrm{u}}} = \sum_{a \in \mathcal{N}} \underbrace{\frac{\partial \log p_{\boldsymbol{y}^\star}(\boldsymbol{x}; \boldsymbol{\Theta})}{\partial p_i^{\mathrm{u}}(a; \boldsymbol{\theta}_i^{\mathrm{u}})}}_{\text{same as Eq. 29}} \cdot \underbrace{\frac{\partial p_i^{\mathrm{u}}(a; \boldsymbol{\theta}_i^{\mathrm{u}})}{\partial \theta_{i,\mathtt{A}}^{\mathrm{u}}}}_{\text{softmax; Eq. 33}} \qquad [32]$$

where the first partial derivative on the right hand side is identical to the case of direct parameterization above (Sec. 4.A; Eq. 29), but the second partial derivative, which used to be 1 in direct parameterization, is now the gradient of the softmax function (see Fig. 4 and Sec. S1.B for details):

$$\frac{\partial p_i^{\mathrm{u}}(a; \boldsymbol{\theta}_i^{\mathrm{u}})}{\partial \theta_{i,\mathtt{A}}^{\mathrm{u}}} = p_i^{\mathrm{u}}(a; \boldsymbol{\theta}_i^{\mathrm{u}}) \cdot (\mathbb{1}[a = \mathtt{A}] - p_i^{\mathrm{u}}(\mathtt{A}; \boldsymbol{\theta}_i^{\mathrm{u}})) \qquad [33]$$
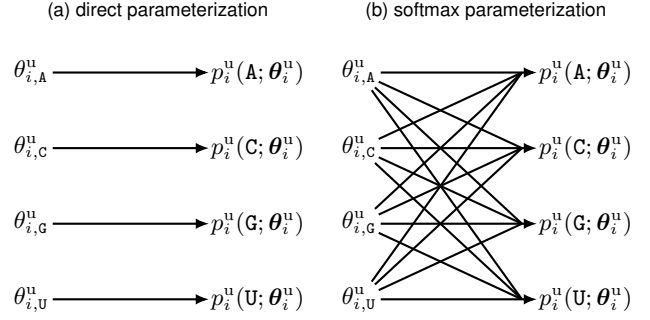

Fig. 4. Computational graph for direct vs. softmax parameterization.

So the gradient for the softmax parameterization is:

$$\frac{\partial \log p_{\boldsymbol{y}^\star}(\boldsymbol{x}; \boldsymbol{\Theta})}{\partial \theta_{i,\mathtt{A}}^{\mathrm{u}}}$$

$$\approx \sum_{a \in \mathcal{N}} \left(\frac{1}{|\mathcal{S}|} \sum_{\substack{\boldsymbol{x} \in \mathcal{S} \\ x_i = a}} \frac{f(\boldsymbol{x}, \boldsymbol{y}^\star)}{\theta_{i,a}^{\mathrm{u}}} \right) \cdot [p_i^{\mathrm{u}}(a; \boldsymbol{\theta}_i^{\mathrm{u}}) \cdot (\mathbb{1}[a = \mathtt{A}] - p_i^{\mathrm{u}}(\mathtt{A}; \boldsymbol{\theta}_i^{\mathrm{u}}))]$$

We run the gradient decent step until the changes in the value of the objective function $\mathcal{J}(\boldsymbol{\Theta})$ become sufficiently small (see Sec. 6), indicating that the solution has converged. Algorithm 1 outlines the procedure of both constraint and unconstraint optimization approach.

Algorithm 1 Sampling-based RNA Design

---

function DESIGN($\boldsymbol{y}^\star$, f, projection=False)   ▷ f: objective
  $\boldsymbol{x}^\star \leftarrow$ random_init($\boldsymbol{y}^\star$)   ▷ current best design
  $\boldsymbol{\Theta} \leftarrow$ init_params($\boldsymbol{y}^\star$)
  while not converged **do**
    sample sequences, $\mathcal{S}$, from distribution $p_{\boldsymbol{y}^\star}(\cdot; \boldsymbol{\Theta})$
    $\boldsymbol{x} \leftarrow \operatorname{argmin}_{\boldsymbol{x}' \in \mathcal{S}} f(\boldsymbol{x}', \boldsymbol{y}^\star)$ ▷ best sample in $\mathcal{S}$
    if $f(\boldsymbol{x}, \boldsymbol{y}^\star) < f(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ **then**   ▷ smaller means better
      $\boldsymbol{x}^\star \leftarrow \boldsymbol{x}$
    estimate objective $\mathcal{J}(\boldsymbol{\Theta})$ using $\mathcal{S}$
    estimate gradient $\nabla_{\boldsymbol{\Theta}} \mathcal{J}(\boldsymbol{\Theta})$ using $\mathcal{S}$
    $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \alpha \nabla_{\boldsymbol{\Theta}} \mathcal{J}(\boldsymbol{\Theta})$
    if projection **then**   ▷ projected gradient descent
      $\boldsymbol{\Theta} \leftarrow \{proj(\boldsymbol{\theta}) \mid \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$ ▷ project onto simplex
  return $\boldsymbol{x}^\star$

---

5. Related Work

Although Matthies et al. (16) also used continuous optimization for RNA design, our approach is vastly different from and substantially outperforms theirs in both scalability and design quality (by all metrics).

- First, their sequence distribution is a simple product of independent distributions for each position (same as our distribution v0 in Sec. 2.A) which is ill-suited for the RNA design problem for two reasons: (a) that distribution includes exponentially many illegal sequences for any input structure due to pair violations and (b) that distribution does not explicitly model the

Tang *et al.*

PNAS | **December 13, 2024** | vol. XXX | no. XX | **7**

covariance between paired positions. Instead, we use coupled variables for paired and mismatch positions (our distributions v1, v2, and v3 in Secs. 2.B–2.C), which rules out invalid sequences and explicitly models the dependencies between correlated positions.

- Second, our sampling framework can work with arbitrary objective functions while their work is specifically designed for one such function, the Boltzmann probability.

- Third, our unbiased sampling yields an *unbiased* approximation to the expectation of an arbitrary objective function over the distribution of sequences. For example, for the case of Boltzmann probability, our sampling results in an unbiased approximation of the expected Boltzmann probability, which converges to the true expectation as the sample size increases:

$$\frac{1}{|\mathcal{S}|}\sum_{\boldsymbol{x}\in\mathcal{S}} p(\boldsymbol{y}^\star \mid \boldsymbol{x}) \approx \mathbb{E}_{\boldsymbol{x}}[p(\boldsymbol{y}^\star \mid \boldsymbol{x})] \qquad [34]$$

By contrast, they optimize a different objective (in red below) that deviates from the true expectation of Boltzmann probability with a bias ($\mathbb{E}[X/Y] \neq \mathbb{E}[X]/\mathbb{E}[Y]$):

$$\color{red}{\frac{\mathbb{E}_{\boldsymbol{x}}\left[e^{-\Delta G^\circ(\boldsymbol{x},\boldsymbol{y}^\star)/RT}\right]}{\mathbb{E}_{\boldsymbol{x}}[Q(\boldsymbol{x})]}} \neq \mathbb{E}_{\boldsymbol{x}}\left[\frac{e^{-\Delta G^\circ(\boldsymbol{x},\boldsymbol{y}^\star)/RT}}{Q(\boldsymbol{x})}\right] \triangleq \mathbb{E}_{\boldsymbol{x}}[p(\boldsymbol{y}^\star \mid \boldsymbol{x})]$$
$$[35]$$

- Fourth, our sampling-based approach is much more efficient: it scales to the longest structures in the Eterna100 benchmark (400 *nt*) while their work only scaled to structures up to 50 *nt* long.

- Last, our results substantially outperform theirs in all metrics (see Table S1).

6. Evaluation Results on Eterna100 Dataset

The Eterna100 dataset (26) is a widely used benchmark for evaluating RNA design programs. It contains 100 secondary structures (i.e., "puzzles") of up to 400 nucleotides, varying in design difficulty from simple hairpins to intricate multiloop structures. We evaluated this work against three baselines using this dataset: SAMFEO (4), NEMO (6), and Matthies et al. (16). To compare their performance, we used the following metrics:

1. Average $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ across all puzzles;

2. Geometric mean of $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ across all puzzles except those 18 that are proven to be undesignable (in the sense that there is no uMFE solution) by our previous work (27, 28); these puzzles have extremely low $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ which bias the geometric mean towards 0;

3. Average $\text{NED}(\boldsymbol{x}, \boldsymbol{y}^\star)$ across all puzzles;

4. Average $d(\text{MFE}(\boldsymbol{x}), \boldsymbol{y}^\star)$ across all puzzles;

5. Average $\Delta\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star)$ across all puzzles;

6. Number of puzzles in which an MFE solution is found;

7. Number of puzzles in which a uMFE solution is found.

Sampling-based Continuous Optimization (This Work) By default, our method samples 2500 sequences at each step. The number of steps is adaptive to each puzzle, specifically the run stops after 50 steps in which the objective function does not improved and the total number of steps is limited to 2000. Our main program is implemented in C++ and utilizes OpenMP for parallelization.

The default initial learning rate is set to 0.01, which works well for all puzzles under the softmax parameterization. However, we observe that the direct parameterization (*projection*) requires smaller learning rates as puzzle lengths increases; otherwise, the objective value does not improve. Therefore, we apply adaptive learning rate decay for the projection method. Additionally, we implement momentum-based optimizers: Adam (29) for softmax and Nesterov accelerate gradient (30) for projection.

We adopt three types of initialization:

- **Uniform**: Each parameter is set to uniform distribution.

- **Targeted**: Assigns 100% A for unpaired, 50% CG and GC for base pairs, and uniform distribution for mismatches and trimismatches.

- ϵ-**Targeted**: A combination of targeted and uniform distribution defined by $\epsilon \cdot \text{targeted} + (1 - \epsilon) \cdot \text{uniform}$.

For the projection method, we use both uniform and targeted initializations. For the softmax method, we use uniform and ϵ-targeted initializations with $\epsilon = 0.75$. The final solutions are selected from the best out of both initializations.

We use our previous work, LinearPartition (19), to compute the partition function and base-pairing probability in linear time with beam pruning. We use beam size $b = 250$ for optimizing $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$, and $b = 100$ for optimizing $\text{NED}(\boldsymbol{x}, \boldsymbol{y}^\star)$. It is only when optimizing for $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$, we use a larger beam size because LinearPartition tends to underapproximate the partition function, resulting in $p(\boldsymbol{y}^\star \mid \boldsymbol{x}) > 1$. For optimizing $d(\text{MFE}(\boldsymbol{x}), \boldsymbol{y}^\star)$ and $\Delta\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star)$, we fold the sequences using LinearFold (31) with $b = 100$.

At each step, we record the best sample among the 2,500 samples. After completing all steps, we reevaluate the recorded samples using ViennaRNA 2.0 (32), selecting the best sequence for each metric. The values reported in this paper are thus, unaffected by the approximation error from beam search.

Baseline 1: SAMFEO SAMFEO is an iterative approach that selects a few nucleotides to mutate by (a) sampling positions based on positional defects and (b) utilizing structural information (4). Similar to our work, SAMFEO is a general approach that can work with any objective function $f(\boldsymbol{x}, \boldsymbol{y}^\star)$. In their paper, they optimize for two ensemble objectives: $1 - p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ and $\text{NED}(\boldsymbol{x}, \boldsymbol{y}^\star)$. SAMFEO is run five times on each puzzle under their default settings with 5000 steps, and we report the best solution obtained from these five runs.

Baseline 2: NEMO NEMO uses Nested Monte Carlo Search with domain-specific knowledge to solve puzzles (6). It maximizes a scoring function defined by:

$$\text{score}(\boldsymbol{x}, \boldsymbol{y}^\star) = K(1 + \Delta\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star))^{-\text{sign}(K)} \qquad [36]$$

where

$$K = 1 - \frac{\text{BPD}(\text{MFE}(\boldsymbol{x}), \boldsymbol{y}^\star)}{2|pairs(\boldsymbol{y}^\star)|}$$

(a) Main results

Methods	Objective	$p(\boldsymbol{y}^\star \mid \boldsymbol{x})(\uparrow)$ arith.	geom.†	NED$(\boldsymbol{x}, \boldsymbol{y}^\star)$ $(\downarrow)$	$d(\text{MFE}(\boldsymbol{x}), \boldsymbol{y}^\star)$ $(\downarrow)$	$\Delta\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star)$ $(\downarrow)$	# of MFE $(\uparrow)$	# of uMFE $(\uparrow)$
NEMO	composite (Eq. (36))	*0.271*	*0.083*	*0.098*	**1.93**	*1.31*	**79**	**77**
SAMFEO	$1 - p(\boldsymbol{y}^\star \mid \boldsymbol{x})$	0.581	0.167	0.043	4.57	2.48	77	74
	NED$(\boldsymbol{x}, \boldsymbol{y}^\star)$	0.512	0.033	<u>0.037</u>	3.92	2.95	72	66
This Work (*projection*)	$\mathbb{E}_{\boldsymbol{x}}[-\log p(\boldsymbol{y}^\star \mid \boldsymbol{x})]$	<u>0.589</u>	0.104	0.048	6.98	2.74	75	71
This Work (*softmax*)	$\mathbb{E}_{\boldsymbol{x}}[-\log p(\boldsymbol{y}^\star \mid \boldsymbol{x})]$	**0.593**	**0.502**	**0.036**	<u>2.10</u>	**0.78**	**79**	<u>76</u>
	$\mathbb{E}_{\boldsymbol{x}}[\text{NED}(\boldsymbol{x}, \boldsymbol{y}^\star)]$	0.541	0.015	0.041	5.48	5.07	63	62
	$\mathbb{E}_{\boldsymbol{x}}[d(\text{MFE}(\boldsymbol{x}), \boldsymbol{y}^\star)]$	0.504	0.022	0.045	4.88	4.50	67	67
	$\mathbb{E}_{\boldsymbol{x}}[\Delta\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star)]$	0.538	<u>0.377</u>	0.047	2.88	<u>1.14</u>	<u>78</u>	72

Fig. 5. (a) Results of various RNA Design methods on the Eterna100 dataset. **Bold**: best value. <u>Underline</u>: second best value. *Italic*: the byproduct is obtained by evaluating the final solution. This work and SAMFEO take the best solution across the entire optimization trajectory. †: geometric mean without 18 undesignable puzzles. (b) – (c) Comparison between this work and SAMFEO for $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$, grouped by puzzle lengths. Each group contains 10 puzzles, except for the geometric means, which exclude undesignable puzzles. (d) Puzzles solved by this work, SAMFEO, and NEMO under the uMFE criterion. (e) – (f) $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ of solutions designed by this work vs. SAMFEO in both original and log scale. Figure S2 provides similar grouped-by-length and individual plots for other metrics. Starred puzzles: #71⋆, #73⋆, #76⋆, #78⋆, #79⋆, #91⋆, #97⋆, and #99⋆ are hyperlinked to their visualizations.

and BPD$(\boldsymbol{y}, \boldsymbol{y}')$ is the base pairing distances,

$$\text{BPD}(\boldsymbol{y}, \boldsymbol{y}') = |pairs(\boldsymbol{y}) \cup pairs(\boldsymbol{y}')| - |pairs(\boldsymbol{y}) \cap pairs(\boldsymbol{y}')|.$$

We ran NEMO five times with the parameters of ViennaRNA 2.5.1 and take the best solutions out of the five runs.

Baseline 3: Matthies et al. (16) Matthies et al. (16) reported the $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ of puzzles up to a length of 50 (18 puzzles) using an 80 GB NVIDIA A100 GPU in their paper. We were able to run their code up to a length of 104 nucleotides (the shortest 51 puzzles) using 80GB NVIDIA H100 GPU under their default settings. However, extending to longer puzzles is not feasible due to GPU memory limit. We compare

with their system in Table S1 and Fig. S1, where our results substantially outperform theirs in all metrics.

Main Results The main results of RNA Design methods on the Eterna100 dataset are shown in Figure 5(a). Our best method uses the softmax parametererization while optimizing for $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$, which performed the best on the whole dataset in categories including the arithmetic mean of $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$, geometric mean of $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ without undesignable puzzles (27, 28), NED$(\boldsymbol{x}, \boldsymbol{y}^\star)$, $\Delta\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star)$, and the number of MFE solved. It also performs the second best on the other two metrics, $d(\text{MFE}(\boldsymbol{x}), \boldsymbol{y}^\star)$ and the number of uMFE solved.

(a)

#73 (370 nt)	$p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ ($\uparrow$)	NED($\boldsymbol{x}, \boldsymbol{y}^\star$) ($\downarrow$)	d(MFE($\boldsymbol{x}$), $\boldsymbol{y}^\star$) ($\downarrow$)	$\Delta\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star)$ ($\downarrow$)	is MFE	is uMFE
This Work	**0.005**	**0.055**	**0**	**0.0 kcal/mol**	**Yes**	**Yes**
SAMFEO	3×10^{-27}	0.417	138	33.2 kcal/mol	No	No

correct base-pair probability
0.0 0.2 0.4 0.6 0.8 1.0

incorrect base-pair probability
0.01 0.2 0.4 0.6 0.8 1.0

nucleotide positional defect
0.0 0.05 0.1 0.2 0.5 0.8 1.0

(b) Target Structure $\boldsymbol{y}^\star$ and MFE Structure (This Work)

(c) MFE Structure (SAMFEO)

(d) Base-Pairing Probabilities (This Work)

(e) Base-Pairing Probabilities (SAMFEO)

Fig. 6. Comparison of the best $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ solutions designed by this work vs. SAMFEO for Puzzle 73 ("Snowflake 4"). (b) – (c) MFE structures of the solutions from this work and SAMFEO. Base-pairs are colored as follows: blue for correct pairs, red for incorrect pairs, with the intensity indicating pairing probability. Nucleotide colors range from blue to red, indicating positional defect. (d) – (e) Base-pairing probabilities of this work and SAMFEO. Orange represents missing correct pairs (i.e. correct pairs with a pairing probability below 0.1).

This work (*softmax*) achieves arithmetic mean $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ of 0.594, outperforming SAMFEO by 0.013. In terms of the geometric mean of $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ without undesignable puzzles, our method performs better than other baselines by a wider margin. We obtained geometric mean of 0.512, surpassing the second-place method, SAMFEO, by 0.345. This indicates that our method is much more effective at designing solutions for longer and harder-to-design puzzles with lower $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ values.

To our surprise, optimizing for $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ yields excellent solutions for other metrics indirectly. Many of these metric are better optimized by optimizing p(y* | x) than when optimizing the metric directly. For example, this work (*softmax*) optimizing for $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ achieves average NED($\boldsymbol{x}, \boldsymbol{y}^\star$) of 0.035, beating both of the approaches (this work and SAMFEO) optimizing for NED($\boldsymbol{x}, \boldsymbol{y}^\star$) by 0.005 and 0.001 respectively.

Additionally, our method solved 79 puzzles under the MFE criterion and 76 under the uMFE criterion, matching NEMO's

performance under the MFE criterion and solving just one fewer puzzle under the uMFE criterion. However, some of NEMO's advantage may stem from heuristic rules specifically tailored for the ViennaRNA energy model. Previous work (4) showed that the vanilla version of NEMO solves only 76 puzzles under the MFE criterion and 75 under the uMFE criterion.

Our method uses both uniform and ϵ-targeted initializations with $\epsilon = 0.75$, and take the best solution from both runs. For $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$, there are 14 puzzles for which the best solution comes from using the uniform initialization. For examples, the $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ for puzzle #74 (380 nt) improved from 0.202 to 0.457 and puzzle #77 (105 nt) improved from 0.334 to 0.400.

Although this work (*projection*) achieves a higher arithmetic mean of $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ than SAMFEO, its geometric mean of $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ without undesignable puzzles is lower than SAMFEO (0.029 vs. .167). This suggests that the projection method performs poorly on hard to design puzzles, which is due to the difficulty in setting the appropriate learning rate for each puzzle.

This work vs. SAMFEO (grouped by puzzle lengths)   Figure 5(b) and (c) compare the performance of this work (*softmax*) against SAMFEO across different puzzle length groups, using the geometric mean of $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ (without undesignable puzzles) and the average $\mathrm{NED}(\boldsymbol{x}, \boldsymbol{y}^\star)$. While both methods perform similarly for shorter puzzles across both metrics, our method demonstrates advantages in the four longest length groups (116–192, 200–316, 337–387 and 389–400). Our geometric mean against SAMFEO in the longest four groups are 0.70 vs. 0.60, 0.34 vs. 0.27, 0.15 vs. 0.00 and 0.22 vs. 0.02, differing from 0.07 to 0.20. Similarly, our method outperform SAMFEO in terms of $\mathrm{NED}(\boldsymbol{x}, \boldsymbol{y}^\star)$, with margins between 0.006 to 0.033 (0.019 vs. 0.025, 0.02 vs. 0.05, 0.05 vs. 0.08, and 0.02 vs. 0.05).

Figure S2(a), (b), and (c) further examine the arithmetic mean of $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$, the average $\Delta\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star)$, and the average $d(\mathrm{MFE}(\boldsymbol{x}), \boldsymbol{y}^\star)$. Our method outperforms SAMFEO in all four longest groups for the arithmetic mean of $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ (0.76 vs. 0.74, 0.53 vs. 0.5, 0.39 vs. 0.38, 0.35 vs. 0.32). In terms of $\Delta\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star)$ and the average of $d(\mathrm{MFE}(\boldsymbol{x}), \boldsymbol{y}^\star)$, the major improvements occur in the three or four longest groups. For $\Delta\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star)$, our method surpasses SAMFEO in the longest three groups (2.23 vs. 0.47, 13.07 vs. 2.92, and 7.40 vs. 2.46). For $d(\mathrm{MFE}(\boldsymbol{x}), \boldsymbol{y}^\star)$, our method also outperforms SAMFEO in the longest four groups (0.00 vs. 1.20, 2.20 vs. 7.50, 6.70 vs. 22.70, and 4.30 vs. 11.40)

uMFE analysis   The Venn digram in Figure 5(d) illustrates puzzles that are solved under the uMFE criterion by this work, SAMFEO, and NEMO. Out of the 100 puzzles, there are 79 solved puzzles and 18 puzzles proven to be undesignable (27, 28). The remaining 3 puzzles (#68, #97, #100) have yet to be solved or proven undesignable.

Our method solved 76 puzzles under the uMFE criterion, one fewer than NEMO. However, we solved an additional puzzle (#89) that was not solved by either SAMFEO or NEMO. Although NEMO solved the most puzzles in the uMFE sense, their solution quality is often substantially worse than ours in terms of $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ and $\mathrm{NED}(\boldsymbol{x}, \boldsymbol{y}^\star)$. See Figure S3 for detailed visualizations of puzzles #71 and #79. The examples show that NEMO's uMFE solutions have

worse positional defects overall and a lower $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ due to competition from alternative structures.

This work vs. SAMFEO (individual puzzle)   The scatterplot in Figure 5(e) compares this work with SAMFEO across individual puzzles in terms of $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$. The two methods perform similarly on most puzzles, with our method showing substantial improvements in a few cases. For example, our method improved puzzle #74 from 0.325 to 0.457, and puzzle #38 from 0.569 to 0.736.

The scatterplot in Figure 5 (f) uses a log scale for $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$, focusing on puzzles longer than $280nt$. In this plot, there are nine puzzles (annotated in the figure) which our solution outperforms SAMFEO's by a factor larger than 10 fold. Most of these puzzles are undesignable and have much lower values of $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$. Figure S2(d) compares the solutions by $\mathrm{NED}(\boldsymbol{x}, \boldsymbol{y}^\star)$, showing similar improvements for puzzles longer than $280nt$.

Visualized Examples   Figures 6–7 and S4–S6 provide detailed comparisons between this work and SAMFEO for puzzles #73, #78, #76, #91, and #99, respectively. In all these examples, our method consistently outperforms SAMFEO across all metrics and is able to find the uMFE (or very close to uMFE) solutions.

For puzzles #73 and #76 (Figs. 6 and S4), our method achieves better $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ than SAMFEO by a substantial factor (0.005 vs. 3×10^{-27} and 0.046 vs. 7×10^{-8}). Furthermore, our solutions meet the uMFE criterion, whereas theirs do not satisfy either the MFE or uMFE criteria. In the base-pairing probabilities plots, we observe many incorrect and missing pairs from SAMFEO's MFE structure, along with many positions with high positional defects.
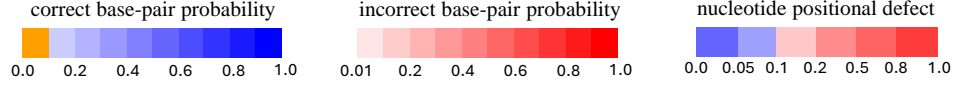
Puzzles #78 (Fig. 7) and #91 (Fig. S5) are undesignable. Our solutions have structural distance of 4 for puzzle #78 and 8 for puzzle #91. In both cases, the structural distances are due to missing pairs in our MFE structures, all of which belong to undesignable motifs. Therefore, we believe our MFE structures are the closest possible solutions to the target structures. In contrast, SAMFEO's MFE structures are very different from the target structure, and the base-pairing probabilities plots show that their solution has very weak base-pairing probabilities becayse of competition from alternative structures.

Puzzle #99 (Fig. S6) is also undesignable. Our MFE solution has a structural distance of 20, compared to SAMFEO's MFE solution, which has a distance of 80. In our solution, one pair is missing from the undesignable motif, along with a few pairs from other bulge loops. Most of the missing pairs in the bulges form a 2×2 internal loop instead. Although these bulges are not undesignable motifs, they may still be undesignable in the context of the entire structure.
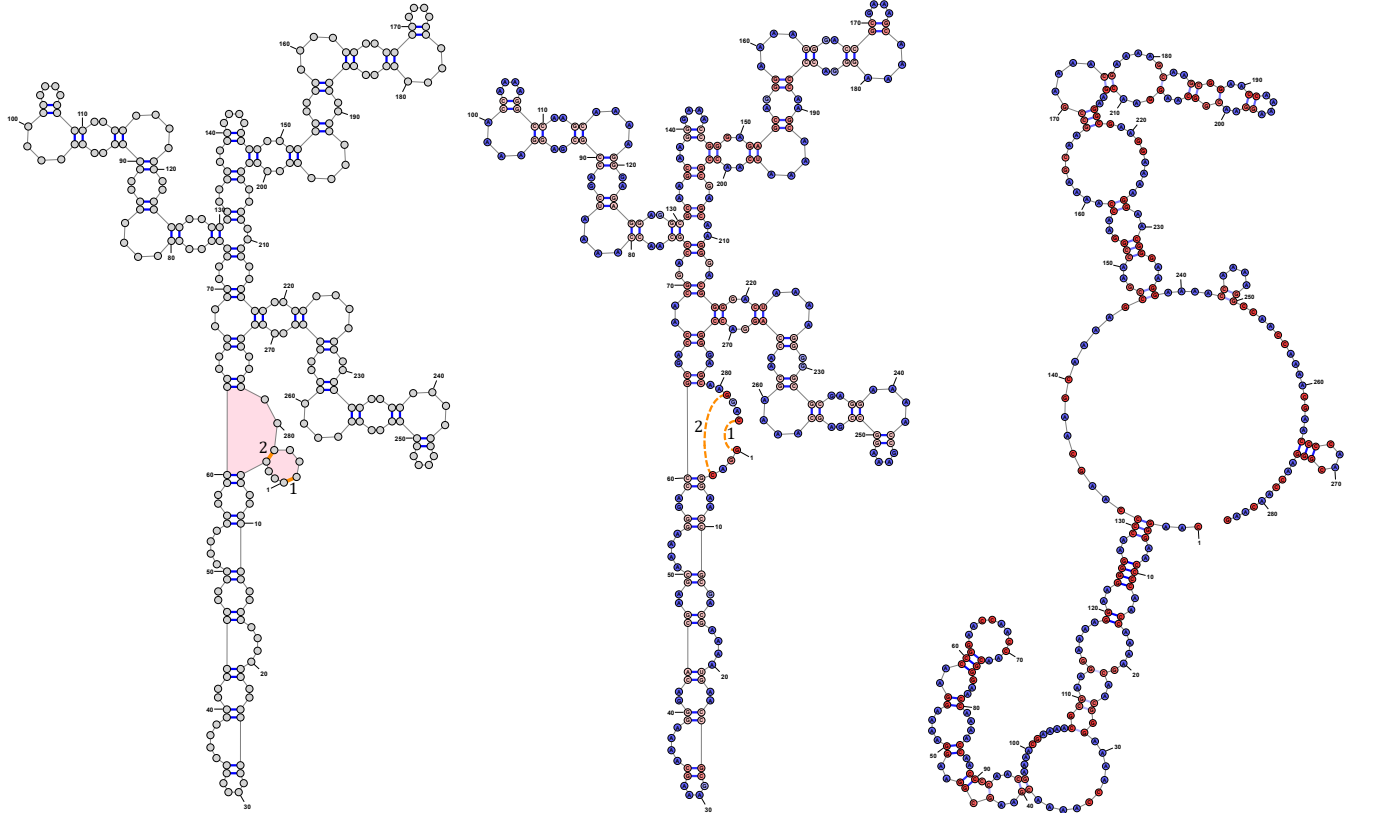
Time Analysis   Figure S7 (b) shows the total number of steps to solve each puzzle based on the stopping criteria: 50 steps since the the objective function improved and a maximum of 2000 steps. Generally, as the puzzle length increases, it takes more steps to find the best solution. However, for puzzles between $100nt$ to $192nt$, most puzzles terminated earlier than others, suggesting that these puzzles are easier to design.

Figure S7 (c) shows the time taken to solve each puzzle when ran on a server with 28 physical cores. The whole

#78 (284 nt)	$p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ ($\uparrow$)	NED($\boldsymbol{x}, \boldsymbol{y}^\star$) ($\downarrow$)	d(MFE($\boldsymbol{x}$), $\boldsymbol{y}^\star$) ($\downarrow$)	$\Delta\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star)$ ($\downarrow$)	is MFE	is uMFE	$p(\tilde{\boldsymbol{y}} \mid \boldsymbol{x})$ ($\uparrow$)
This Work	**0.001**	**0.123**	**4**	**2.6 kcal/mol**	No	No	**0.058**
SAMFEO	9×10^{-25}	0.452	140	29.2 kcal/mol	No	No	2×10^{-20}

Fig. 7. Comparison of the best $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ solution designed by this work vs. SAMFEO for Puzzle 78 ("Mat - Lot 2-2 B"). (b) Target structure: pink-filled regions highlight loops that belong to an undesignable motif, while orange base pairs represent the missing pairs in Sampling's MFE structure. (c) – (d) MFE structures of the best $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ solutions from this work and SAMFEO. (e) – (f) Base-pairing probabilities plots. Base-pairs are colored as follows: blue for correct pairs, red for incorrect pairs, with the intensity indicating pairing probability. Orange represents missing correct pairs (i.e. correct pairs with a pairing probability below 0.1). Nucleotide colors range from blue to red, indicating positional defect. $\tilde{\boldsymbol{y}}$ refers to the target structure with the (orange) base pairs from undesignable motifs removed (i.e. pairs 1 and 2 are removed).

evaluation on the Eterna100 dataset takes about 10 days, with the longest puzzle taking up to 20 hours.

The primary time bottleneck in our method is computing the objective function $f(\boldsymbol{x}, \boldsymbol{y}^\star)$ for all 2500 samples. For example, computing $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ takes up to more than 99% of total time due to cubic time complexity. However, this issue is mitigated to some extent by using beam search from LinearPartition. Additionally, computing the objective

function for each sample is parallelizable. With 28 physical cores, we observed a 27.4× speedup for the longest puzzles with 400 nucleotides, reducing the time from 1233 seconds to 45 seconds per step.

Other speedup options include using a smaller sample size, but it tends to result in longer convergence times and greater difficulty in finding a better solution. Speed improvements can also be achieved by caching the $f(\boldsymbol{x}, \boldsymbol{y}^\star)$ of each sample, which avoids redundant computations for repeated samples. This approach is effective when the puzzle is short and targeted initialization is used, which tends to produce more repeated samples. For example, puzzle #36 with 151 nucleotides obtained a 1.95× speedup from caching, reducing the time from 4.2 seconds to 2.15 seconds per step.

Learning Curves Figure S8 presents the learning curves for puzzles #97 and #98. At each step, the plots illustrate $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ for the best sample and the integral solution, along with the arithmetic and geometric means of $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ computed from 2500 samples. Overall, the learning curves show that the values for the best sample, as well as the arithmetic and geometric means of $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$, consistently increase and converge toward the end of the process. The entropy decreases over time, while the boxplot narrows, indicating that the distribution becomes more concentrated as the optimization progresses. Notably, the best sample converges more quickly and reaches a higher $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ than the integral solution. In our findings, the highest quality solutions are consistently derived from the best sample rather than the integral solution.

Ablation Studies We run two additional experiments with targeted initializations to assess the effectiveness of coupling variables on mismatches and trimismatches. First, we ablated the coupling variables for trimismatches. Then, we ablated the coupling variables for both mismatches and trimismatches. The results of these ablation studies are presented in Table S3.

As more coupling variables are removed, the results deteriorate across all metrics. The arithmetic mean of $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ decreased from 0.589 to 0.578 after ablating trimismatches and further declines to 0.566 when mismatches are also ablated. Similarly, the average $\text{NED}(\boldsymbol{x}, \boldsymbol{y}^\star)$ increases from 0.035 to 0.039 and then to 0.042. This pattern continues across other metrics, suggesting that the coupled variables approach is useful.

7. Conclusions and Future Work

We described a general framework for sampling-based continuous optimization with coupled-variable distributions that is applicable to optimizing any objective function for RNA design. Our method consistently outperformed other state-of-the-art methods across nearly all metrics, such as Boltzmann probability and ensemble defect, particularly on the long and hard-to-design puzzles in the Eterna100 benchmark.

In the future, our work can be improved in ways such as:

1. Our method evaluates the objective for each sample, which takes up the majority of runtime. Currently we cache full sequences (unique samples), but to speed up even further, we can cache subsequences shared among the samples. We can also use importance sampling (33) to reduce the number of samples in each step.

2. Our sequence distributions are products of independent distributions of coupled variables, which are both expressive and easy to sample from. But we can also consider more complex distributions using discriminative models (34) and neural models (35), although sampling would be more difficult.

3. We can extend this framework to protein design which should in principle work for arbitrary objective functions. The challenge is scalability due to much slower objective evaluation using protein folding engines (36, 37).

1. SR Eddy., Non-coding RNA genes and the modern RNA world. *Nat. Rev. Genet.* **2**, 919–929 (2001).
2. JA Doudna, TR Cech, The chemical repertoire of natural ribozymes. *Nature* **418**, 222–228 (2002).
3. JP Bachellerie, J Cavaillé, A Hüttenhofer, The expanding snoRNA world. *Biochimie* **84**, 775–790 (2002).
4. T Zhou, et al., RNA design via structure-aware multifrontier ensemble optimization. *Bioinformatics* **39**, i563–i571 (2023).
5. S Bellaousov, M Kayedkhordeh, RJ Peterson, DH Mathews, Accelerated RNA secondary structure design using preselected sequences for helices and loops. *RNA* **24**, 1555–1567 (2018).
6. F Portela, An unexpectedly effective Monte Carlo technique for the RNA inverse folding problem. *BioRxiv* p. 345587 (2018).
7. JA Garcia-Martin, P Clote, I Dotu, RNAiFOLD: a constraint programming algorithm for RNA inverse folding and molecular design. *J. bioinformatics computational biology* **11**, 1350001 (2013).
8. JN Zadeh, BR Wolfe, NA Pierce, Nucleic Acid Sequence Design via Efficient Ensemble Defect Optimization. *J. Comput. Chem.* **32**, 439–452 (2010).
9. I Dotu, et al., Complete RNA inverse folding: computational design of functional hammerhead ribozymes. *Nucleic Acids Res.* **42**, 11752–11762 (2014).
10. R Yamagami, M Kayedkhordeh, DH Mathews, PC Bevilacqua, Design of highly active double-pseudoknotted ribozymes: a combined computational and experimental study. *Nucleic Acids Res.* **47**, 29–42 (2018).
11. R Schwab, S Ossowski, M Riester, N Warthmann, D Weigel, Highly specific gene silencing by artificial microRNAs in Arabidopsis. *The Plant Cell* **18**, 1121–1133 (2006).
12. M Hamada, In silico approaches to RNA aptamer design. *Biochimie* **145**, 8–14 (2018).
13. G Bauer, B Suess, Engineered riboswitches as novel tools in molecular biology. *J. biotechnology* **124**, 4–11 (2006).
14. S Findeiß, M Etzel, S Will, M Mörl, PF Stadler, Design of artificial riboswitches as biosensors. *Sensors* **17**, 1990 (2017).
15. É Bonnet, P Rzazewski, F Sikora, Designing RNA secondary structures is hard. *J. Comput. Biol.* **27**, 302–316 (2020).
16. M Matthies, R Krueger, A Torda, M Ward, Differentiable Partition Function Calculation for RNA. *Nucleic Acids Res.* (2023).
17. N Dai, WY Tang, T Zhou, DH Mathews, L Huang, Messenger RNA Design via Expected Partition Function and Continuous Optimization. *arXiv preprint arXiv:2401.00037* (2024).
18. JN Zadeh, BR Wolfe, NA Pierce, Nucleic acid sequence design via efficient ensemble defect optimization. *J. computational chemistry* **32**, 439–452 (2011).
19. H Zhang, L Zhang, DH Mathews, L Huang, LinearPartition: linear-time approximation of RNA folding partition function and base-pairing probabilities. *Bioinformatics* **36**, i258–i267 (2020).
20. IL Hofacker, et al., Fast folding and comparison of RNA secondary structures. *Monatshefte fur chemie* **125**, 167–167 (1994).
21. M Andronescu, AP Fejes, F Hutter, HH Hoos, A Condon, A New Algorithm for RNA Secondary Structure Design. *J. Mol. Biol.* **336**, 607–624 (2004).
22. A Busch, R Backofen, INFO-RNA – a fast approach to inverse RNA folding. *Bioinformatics* **22**, 1823–1831 (2006).
23. JN Zadeh, et al., NUPACK: Analysis and design of nucleic acid systems. *J. computational chemistry* **32**, 170–173 (2011).
24. A Mittal, DH Turner, DH Mathews, Nndb: An expanded database of nearest neighbor parameters for predicting stability of nucleic acid secondary structures. *J. Mol. Biol.* **436**, 168549 (2024) Computation Resources for Molecular Biology.
25. A Madry, A Makelov, L Schmidt, D Tsipras, A Vladu, Towards Deep Learning Models Resistant to Adversarial Attacks (2019).
26. J Anderson-Lee, et al., Principles for predicting RNA secondary structure design difficulty. *J. molecular biology* **428**, 748–757 (2016).
27. T Zhou, WY Tang, DH Mathews, L Huang, Undesignable RNA Structure Identification via Rival Structure Generation and Structure Decomposition in *International Conference on Research in Computational Molecular Biology.* (Springer), pp. 270–287 (2024).
28. T Zhou, WY Tang, DH Mathews, L Huang, Scalable Identification of Minimum Undesignable RNA Motifs on Loop-Pair Graphs. *arXiv preprint arXiv:2402.17206* (2024).
29. DP Kingma, J Ba, Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

30. Y Nesterov, A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. *Dokl. Akad. Nauk. SSSR* **269**, 543 (1983).
31. L Huang, et al., LinearFold: linear-time approximate RNA folding by 5'-to-3' dynamic programming and beam search. *Bioinformatics* **35**, i295–i304 (2019).
32. R Lorenz, et al., Viennarna package 2.0. *Algorithms Mol. Biol.* **6**, 26 (2011).
33. H Kahn, TE Harris, Estimation of particle transmission by random sampling. *Natl. Bureau Standards applied mathematics series* **12**, 27–30 (1951).
34. J Lafferty, A McCallum, F Pereira, , et al., Conditional random fields: Probabilistic models for segmenting and labeling sequence data in *Proceedings of ICML*. (Williamstown, MA), Vol. 1, p. 3 (2001).
35. A Vaswani, et al., Attention is all you need, 2017 in *Proceedings of NIPS*. p. S0140525X16001837 (2017).
36. J Jumper, et al., Highly accurate protein structure prediction with alphafold. *nature* **596**, 583–589 (2021).
37. Z Lin, et al., Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **379**, 1123–1130 (2023).

<div align="center">

Supporting Information

Sampling-based Continuous Optimization with Coupled Variables for RNA Design

Wei Yu Tang, Ning Dai, Tianshuo Zhou, David H. Mathews, and Liang Huang

</div>

S1. Derivation of Gradient

S1.A. Direct Parameterization. In the example of direct parameterization (Sec. 4.A), our objective is to compute the gradient of the objective function with respect to the parameter associated with an unpaired position i and nucleotide $\texttt{A}$. As noted in Eq. 28, this gradient can be approximated as follows:

$$\frac{\partial \mathcal{J}(\boldsymbol{\Theta})}{\partial \theta^{\mathrm{u}}_{i,\texttt{A}}} \approx \frac{1}{|\mathcal{S}|} \sum_{\boldsymbol{x} \in \mathcal{S}} \frac{\partial \log p_{\boldsymbol{y}^\star}(\boldsymbol{x}; \boldsymbol{\Theta})}{\partial \theta^{\mathrm{u}}_{i,\texttt{A}}} f(\boldsymbol{x}, \boldsymbol{y}^\star)$$

Here, $\log p_{\boldsymbol{y}^\star}(\boldsymbol{x}; \boldsymbol{\Theta})$ can be expressed as:

$$\begin{aligned}
\log p_{\boldsymbol{y}^\star}(\boldsymbol{x}; \boldsymbol{\Theta}) &= \log \left(\prod_{i \in \mathit{unpaired}(\boldsymbol{y})} \theta^{\mathrm{u}}_{i,x_i} \cdot \prod_{(i,j) \in \mathit{pairs}(\boldsymbol{y})} \theta^{\mathrm{p}}_{i,j,x_i x_j} \cdot \prod_{(i,j) \in \mathit{mismatches}(\boldsymbol{y})} \theta^{\mathrm{m}}_{i,j,x_i x_j} \cdot \prod_{(i,j,k) \in \mathit{trimismatches}(\boldsymbol{y})} \theta^{\mathrm{tm}}_{i,j,k,x_i x_j x_k} \right) \\
&= \sum_{i \in \mathit{unpaired}(\boldsymbol{y})} \log(\theta^{\mathrm{u}}_{i,x_i}) + \sum_{(i,j) \in \mathit{pairs}(\boldsymbol{y})} \log(\theta^{\mathrm{p}}_{i,j,x_i x_j}) + \sum_{(i,j) \in \mathit{mismatches}(\boldsymbol{y})} \log(\theta^{\mathrm{m}}_{i,j,x_i x_j}) + \sum_{(i,j,k) \in \mathit{trimismatches}(\boldsymbol{y})} \log(\theta^{\mathrm{tm}}_{i,j,k,x_i x_j x_k})
\end{aligned}$$

Since we are interested in the gradient with respect to a particular position i and nucleotide $\texttt{A}$, the other terms are constants. This simplifies the equation to:

$$\frac{\partial \log p_{\boldsymbol{y}^\star}(\boldsymbol{x}; \boldsymbol{\Theta})}{\partial \theta^{\mathrm{u}}_{i,\texttt{A}}} = \frac{\partial}{\partial \theta^{\mathrm{u}}_{i,\texttt{A}}} (\log \theta^{\mathrm{u}}_{i,x_i}) = \frac{1}{\theta^{\mathrm{u}}_{i,x_i}} \mathbb{1}\left[x_i = \texttt{A} \right]$$

Finally, the gradient approximation can be expressed as:

$$\frac{\partial \mathcal{J}(\boldsymbol{\Theta})}{\partial \theta^{\mathrm{u}}_{i,\texttt{A}}} \approx \frac{1}{|\mathcal{S}|} \sum_{\boldsymbol{x} \in \mathcal{S}} \mathbb{1}\left[x_i = A \right] \frac{f(\boldsymbol{x}, \boldsymbol{y}^\star)}{\theta^{\mathrm{u}}_{i,\texttt{A}}} = \frac{1}{|\mathcal{S}|} \sum_{\substack{\boldsymbol{x} \in \mathcal{S} \\ x_i = A}} \frac{f(\boldsymbol{x}, \boldsymbol{y}^\star)}{\theta^{\mathrm{u}}_{i,\texttt{A}}}$$

We can also extend the gradient to other cases, including pairs, mismatches, and trimismatches. The gradients are listed below:

$$\frac{\partial \mathcal{J}(\boldsymbol{\Theta})}{\partial \theta^{\mathrm{p}}_{i,j,ab}} \approx \frac{1}{|\mathcal{S}|} \sum_{\substack{\boldsymbol{x} \in \mathcal{S} \\ x_i = a \\ x_j = b}} \frac{f(\boldsymbol{x}, \boldsymbol{y}^\star)}{\theta^{\mathrm{p}}_{i,j,ab}}, \qquad \frac{\partial \mathcal{J}(\boldsymbol{\Theta})}{\partial \theta^{\mathrm{m}}_{i,j,ab}} \approx \frac{1}{|\mathcal{S}|} \sum_{\substack{\boldsymbol{x} \in \mathcal{S} \\ x_i = a \\ x_j = b}} \frac{f(\boldsymbol{x}, \boldsymbol{y}^\star)}{\theta^{\mathrm{m}}_{i,j,ab}}, \qquad \frac{\partial \mathcal{J}(\boldsymbol{\Theta})}{\partial \theta^{\mathrm{tm}}_{i,j,k,abc}} \approx \frac{1}{|\mathcal{S}|} \sum_{\substack{\boldsymbol{x} \in \mathcal{S} \\ x_i = a \\ x_j = b \\ x_k = c}} \frac{f(\boldsymbol{x}, \boldsymbol{y}^\star)}{\theta^{\mathrm{tm}}_{i,j,k,abc}}$$

S1.B. Softmax Parameterization. In Section 4.B, we introduce the softmax parameterization as an alternative approach to solving the optimization problem. The probability of a nucleotide $a \in \mathcal{N}$ at an unpaired position i is defined in Eq. 30 as:

$$p^{\mathrm{u}}_i(a; \boldsymbol{\theta}^{\mathrm{u}}_i) \triangleq \frac{\exp(\theta^{\mathrm{u}}_{i,a})}{Z}, \quad \text{where } Z = \sum_{a' \in \mathcal{N}} \exp(\theta^{\mathrm{u}}_{i,a'})$$

To compute the gradient of the objective function under the softmax parameterization (Eq. 32), we need to derive the derivative of the softmax function. Suppose that we are interested in the derivative of $p^{\mathrm{u}}_i(a; \boldsymbol{\theta}^{\mathrm{u}}_i)$ with respect to the parameter $\theta^{\mathrm{u}}_{i,\texttt{A}}$. For the case where $a = \texttt{A}$, applying the quotient rule yields:

$$\frac{\partial p^{\mathrm{u}}_i(\texttt{A}; \boldsymbol{\theta}^{\mathrm{u}}_i)}{\partial \theta^{\mathrm{u}}_{i,\texttt{A}}} = \frac{\exp(\theta^{\mathrm{u}}_{i,\texttt{A}}) S - \exp(\theta^{\mathrm{u}}_{i,\texttt{A}}) \exp(\theta^{\mathrm{u}}_{i,\texttt{A}})}{Z^2} = \frac{\exp(\theta^{\mathrm{u}}_{i,\texttt{A}})}{Z} \cdot \frac{Z - \exp(\theta^{\mathrm{u}}_{i,\texttt{A}})}{Z} = p^{\mathrm{u}}_i(\texttt{A}; \boldsymbol{\theta}^{\mathrm{u}}_i) \cdot (1 - p^{\mathrm{u}}_i(\texttt{A}; \boldsymbol{\theta}^{\mathrm{u}}_i))$$

Similarly, for the case where $a \neq \texttt{A}$, the derivative is given by:

$$\frac{\partial p^{\mathrm{u}}_i(a; \boldsymbol{\theta}^{\mathrm{u}}_i)}{\partial \theta^{\mathrm{u}}_{i,\texttt{A}}} = \frac{0 - \exp(\theta^{\mathrm{u}}_{i,\texttt{A}}) \exp(\theta^{\mathrm{u}}_{i,a})}{Z^2} = -\frac{\exp(\theta^{\mathrm{u}}_{i,\texttt{A}})}{Z} \cdot \frac{\exp(\theta^{\mathrm{u}}_{i,a})}{Z} = -p^{\mathrm{u}}_i(\texttt{A}; \boldsymbol{\theta}^{\mathrm{u}}_i) p^{\mathrm{u}}_i(a; \boldsymbol{\theta}^{\mathrm{u}}_i) = p^{\mathrm{u}}_i(a; \boldsymbol{\theta}^{\mathrm{u}}_i)(0 - p^{\mathrm{u}}_i(\texttt{A}; \boldsymbol{\theta}^{\mathrm{u}}_i))$$

Combining these results, we can express the derivative of the softmax function with respect to $\theta^{\mathrm{u}}_{i,\texttt{A}}$ as:

$$\frac{\partial p^{\mathrm{u}}_i(a; \boldsymbol{\theta}^{\mathrm{u}}_i)}{\partial \theta^{\mathrm{u}}_{i,\texttt{A}}} = p^{\mathrm{u}}_i(a; \boldsymbol{\theta}^{\mathrm{u}}_i) \cdot (\mathbb{1}\left[a = \texttt{A} \right] - p^{\mathrm{u}}_i(\texttt{A}; \boldsymbol{\theta}^{\mathrm{u}}_i))$$

For other cases, such as pairs, mismatches, and trimismatches, the derivatives of the softmax function are provided below:

$$\frac{\partial p^{\mathrm{p}}_{i,j}(ab; \boldsymbol{\theta}^{\mathrm{p}}_{i,j})}{\partial \theta^{\mathrm{u}}_{i,j,a'b'}} = p^{\mathrm{p}}_{i,j}(ab; \boldsymbol{\theta}^{\mathrm{p}}_{i,j}) \cdot (\mathbb{1}\left[ab = a'b' \right] - p^{\mathrm{p}}_{i,j}(a'b'; \boldsymbol{\theta}^{\mathrm{p}}_{i,j})), \qquad \frac{\partial p^{\mathrm{m}}_{i,j}(ab; \boldsymbol{\theta}^{\mathrm{m}}_{i,j})}{\partial \theta^{\mathrm{u}}_{i,j,a'b'}} = p^{\mathrm{m}}_{i,j}(ab; \boldsymbol{\theta}^{\mathrm{m}}_{i,j}) \cdot (\mathbb{1}\left[ab = a'b' \right] - p^{\mathrm{m}}_{i,j}(a'b'; \boldsymbol{\theta}^{\mathrm{m}}_{i,j})),$$

$$\frac{\partial p^{\mathrm{tm}}_{i,j,k}(abc; \boldsymbol{\theta}^{\mathrm{tm}}_{i,j,k})}{\partial \theta^{\mathrm{u}}_{i,j,k,a'b'c'}} = p^{\mathrm{tm}}_{i,j,k}(abc; \boldsymbol{\theta}^{\mathrm{tm}}_{i,j,k}) \cdot (\mathbb{1}\left[abc = a'b'c' \right] - p^{\mathrm{tm}}_{i,j,k}(a'b'c'; \boldsymbol{\theta}^{\mathrm{tm}}_{i,j,k}))$$

Table S1. Results of different RNA Design methods on the shortest structures in Eterna100 (up to 50 and 104 nucleotides, respectively).

	Methods	Objective	$p(\boldsymbol{y}^{\star} \mid \boldsymbol{x})(\uparrow)$ mean	geom.†	NED$(\boldsymbol{x}, \boldsymbol{y}^{\star})$ $(\downarrow)$	$d(\text{MFE}(\boldsymbol{x}), \boldsymbol{y}^{\star})$ $(\downarrow)$	$\Delta\Delta G^{\circ}(\boldsymbol{x}, \boldsymbol{y}^{\star})$ $(\downarrow)$	# of MFE $(\uparrow)$	# of uMFE $(\uparrow)$
Length ≤ 50	Matthies et al. (16)	$-\log\left[\frac{\mathbb{E}_{\boldsymbol{x}}[e^{-\Delta G^{\circ}(x,y)/RT}]}{\mathbb{E}_{\boldsymbol{x}}[Q(x)]}\right]$	0.545	0.088	0.120	3.72	1.12	11	11
	SAMFEO	$1 - p(\boldsymbol{y}^{\star} \mid \boldsymbol{x})$	0.712	<u>0.314</u>	**0.046**	**0.56**	**0.49**	**16**	**16**
	This Work (*projection*)	$\mathbb{E}_{\boldsymbol{x}}[-\log p(\boldsymbol{y}^{\star} \mid \boldsymbol{x})]$	<u>0.715</u>	**0.317**	0.049	<u>0.78</u>	<u>0.50</u>	<u>15</u>	14
	This Work (*softmax*)	$\mathbb{E}_{\boldsymbol{x}}[-\log p(\boldsymbol{y}^{\star} \mid \boldsymbol{x})]$	**0.716**	**0.317**	<u>0.048</u>	0.83	0.52	<u>15</u>	<u>15</u>
Length ≤ 104	Matthies et al. (16)	$-\log\left[\frac{\mathbb{E}_{\boldsymbol{x}}[e^{-\Delta G^{\circ}(x,y)/RT}]}{\mathbb{E}_{\boldsymbol{x}}[Q(x)]}\right]$	0.447	0.135	*0.110*	*6.08*	*1.38*	*27*	*27*
	SAMFEO	$1 - p(\boldsymbol{y}^{\star} \mid \boldsymbol{x})$	0.675	0.699	**0.038**	**1.08**	0.37	**42**	40
	This Work (*projection*)	$\mathbb{E}_{\boldsymbol{x}}[-\log p(\boldsymbol{y}^{\star} \mid \boldsymbol{x})]$	**0.681**	**0.722**	<u>0.040</u>	<u>1.35</u>	**0.33**	42	<u>39</u>
	This Work (*softmax*)	$\mathbb{E}_{\boldsymbol{x}}[-\log p(\boldsymbol{y}^{\star} \mid \boldsymbol{x})]$	<u>0.680</u>	<u>0.717</u>	0.041	1.43	<u>0.34</u>	<u>41</u>	**40**

(a) $p(\boldsymbol{y}^{\star} \mid \boldsymbol{x})(\uparrow)$ of puzzles up to 104 nucleotides.

Fig. S1. $p(\boldsymbol{y}^{\star} \mid \boldsymbol{x})$ of solutions designed by this work vs. Matthies et al. (16) on the 51 shortest structures in Eterna100 (up to 104 nucleotides).

Table S2. Comparison of solutions designed by this work vs. SAMFEO on some long and hard-to-design Eterna100 puzzles.

Puzzle	Method	$p(\boldsymbol{y}^{\star} \mid \boldsymbol{x})$ $(\uparrow)$	NED$(\boldsymbol{x}, \boldsymbol{y}^{\star})$ $(\downarrow)$	$d(\text{MFE}(\boldsymbol{x}), \boldsymbol{y}^{\star})$ $(\downarrow)$	$\Delta\Delta G^{\circ}(\boldsymbol{x}, \boldsymbol{y}^{\star})$ $(\downarrow)$	is MFE	is uMFE
#73 (370 nt)	This Work	**0.005**	**0.055**	**0**	**0.0 kcal/mol**	**Yes**	**Yes**
Figure 6	SAMFEO	3×10^{-27}	0.417	138	33.2 kcal/mol	No	No
#76 (393 nt)	This Work	**0.035**	**0.054**	**0**	**0.0 kcal/mol**	**Yes**	**Yes**
Figure S4	SAMFEO	7×10^{-8}	0.239	26	4.2 kcal/mol	No	No
#78 (284 nt)	This Work	**0.001**	**0.123**	**4**	**2.6 kcal/mol**	No	No
Figure 7	SAMFEO	9×10^{-25}	0.452	140	29.2 kcal/mol	No	No
#91 (392 nt)	This Work	**0.0001**	**0.034**	**8**	**3.4 kcal/mol**	No	No
Figure S5	SAMFEO	2×10^{-20}	0.128	52	25.2 kcal/mol	No	No
#99 (364 nt)	This Work	$\mathbf{8 \times 10^{-11}}$	**0.111**	**20**	**9.8 kcal/mol**	No	No
Figure S6	SAMFEO	3×10^{-28}	0.197	80	34.4 kcal/mol	No	No

Table S3. Ablation studies. Results of this work (*softmax*) optimizing for $p(y^{\star} \mid x)$ without mismatch and trimismatch. Targeted initialization only. †: geometric mean without 18 undesignable puzzles.

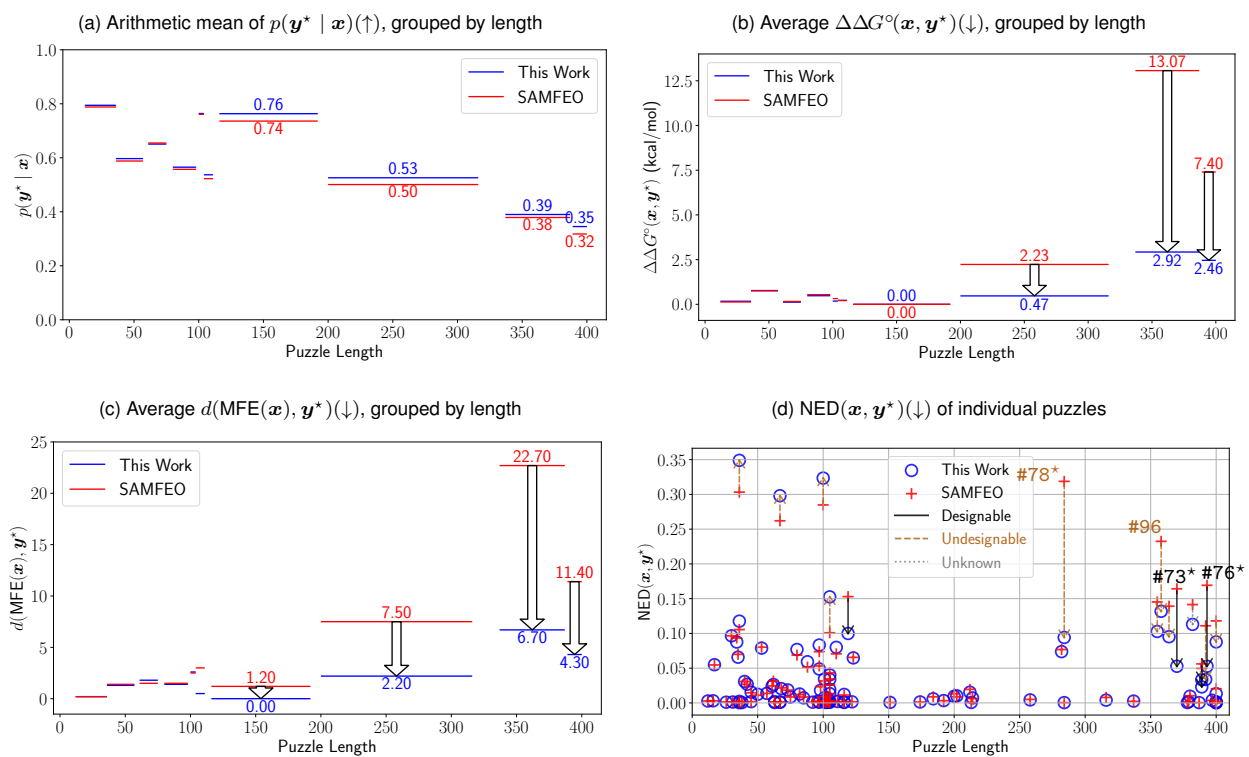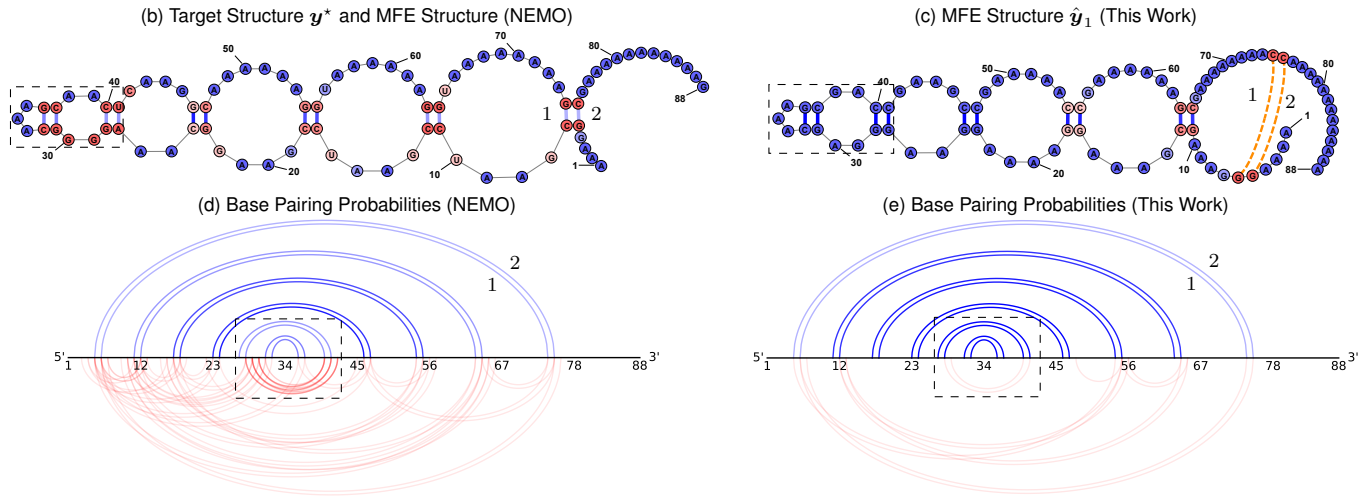| Methods	Distribution	$p(\boldsymbol{y}^{\star} \mid \boldsymbol{x})(\uparrow)$ mean	geom.†	NED$(\boldsymbol{x}, \boldsymbol{y}^{\star})$ $(\downarrow)$	$d(\text{MFE}(\boldsymbol{x}), \boldsymbol{y}^{\star})$ $(\downarrow)$	$\Delta\Delta G^{\circ}(\boldsymbol{x}, \boldsymbol{y}^{\star})$ $(\downarrow)$	# of MFE $(\uparrow)$	# of uMFE $(\uparrow)$
This Work	v3: default	**0.589**	**0.502**	**0.035**	**2.96**	**0.81**	**78**	**75**
	v2: base-pair & mismatch	0.578	0.467	0.039	3.19	0.94	75	72
	v1: only base-pair	0.566	0.422	0.042	3.93	0.97	75	70

Fig. S2. (a) – (c) Average of metrics when puzzles are grouped by length, with each group consisting of 10 puzzles. (d) NED($\boldsymbol{x}, \boldsymbol{y}^\star$) of solutions designed by this work vs. SAMFEO. Figure 5 displays similar grouped-by-length plots and scatterplots for other metrics.

(a)

#71 (88 nt)	$p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ ($\uparrow$)	NED($\boldsymbol{x}, \boldsymbol{y}^\star$) ($\downarrow$)	d(MFE($\boldsymbol{x}$), $\boldsymbol{y}^\star$) ($\downarrow$)	$\Delta\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star)$ ($\downarrow$)	is MFE	is uMFE
This Work	**0.211**	**0.059**	4	0.4 kcal/mol	No	No
NEMO	0.051	0.180	**0**	**0.0 kcal/mol**	Yes	Yes

correct base-pair probability

incorrect base-pair probability

nucleotide positional defect

(b) Target Structure $\boldsymbol{y}^\star$ and MFE Structure (NEMO)

(c) MFE Structure $\hat{\boldsymbol{y}}_1$ (This Work)

(d) Base Pairing Probabilities (NEMO)

(e) Base Pairing Probabilities (This Work)
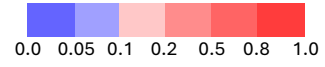
(f)

#79 (101 nt)	$p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ ($\uparrow$)	NED($\boldsymbol{x}, \boldsymbol{y}^\star$) ($\downarrow$)	d(MFE($\boldsymbol{x}$), $\boldsymbol{y}^\star$) ($\downarrow$)	$\Delta\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star)$ ($\downarrow$)	is MFE	is uMFE
This Work	**0.272**	**0.041**	4	**0.0 kcal/mol**	**Yes**	No
NEMO	0.092	0.113	**0**	**0.0 kcal/mol**	**Yes**	**Yes**

(g) Target Structure $\boldsymbol{y}^\star$ and MFE Structure (NEMO)

(h) MFE Structure $\hat{\boldsymbol{y}}_2$ (This Work)

(i) Base Pairing Probabilities (NEMO)

(j) Base Pairing Probabilities (This Work)

Fig. S3. Puzzles #71, #79 are solved by NEMO (but not by this work) under the UMFE criterion. MFE Structures: Base pairs are colored blue for correct pairs and red for incorrect pairs, with color intensity indicating pairing probability. Nucleotides are colored using a blue-to-red gradient to represent positional defects. Orange dashed lines in our MFE structure indicate the missing pairs from target structure. The dashed box highlights regions where NEMO's solution shows poor positional defects and many base-pairing competitions due to alternative structures. For puzzle #79, although our design is not a uMFE solution, it is still an MFE solution, meaning that $\hat{\boldsymbol{y}}_2 \in \text{MFE}_s(\boldsymbol{x})$ and $\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star) = \Delta G^\circ(\boldsymbol{x}, \hat{\boldsymbol{y}}_2)$.

#76 (393 nt)	$p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ (↑)	NED$(\boldsymbol{x}, \boldsymbol{y}^\star)$ (↓)	$d(\text{MFE}(\boldsymbol{x}), \boldsymbol{y}^\star)$ (↓)	$\Delta\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star)$ (↓)	is MFE	is uMFE
This Work	**0.035**	**0.054**	**0**	**0.0 kcal/mol**	**Yes**	**Yes**
SAMFEO	7×10^{-8}	0.239	26	4.2 kcal/mol	No	No

(a)

Fig. S4. Comparison of the best $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ solutions designed by this work vs. SAMFEO for Puzzle 76 ("Snowflake 3"). (b) – (c) MFE structures of the solutions from this work and SAMFEO. Base-pairs are colored as follows: blue for correct pairs, red for incorrect pairs, with the intensity indicating pairing probability. Nucleotide colors range from blue to red, indicating positional defect. (d) – (e) Base-pairing probabilities of this work and SAMFEO. Orange represents missing correct pairs (i.e. correct pairs with a pairing probability below 0.1).

#91 (392 nt)	$p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ ($\uparrow$)	NED($\boldsymbol{x}, \boldsymbol{y}^\star$) ($\downarrow$)	d(MFE($\boldsymbol{x}$), $\boldsymbol{y}^\star$) ($\downarrow$)	$\Delta\Delta G^\circ(\boldsymbol{x}, \boldsymbol{y}^\star)$ ($\downarrow$)	is MFE	is uMFE	$p(\tilde{\boldsymbol{y}} \mid \boldsymbol{x})$ ($\uparrow$)
This Work	**0.0001**	**0.034**	**8**	**3.4 kcal/mol**	No	No	**0.037**
SAMFEO	2×10^{-20}	0.128	52	25.2 kcal/mol	No	No	8×10^{-13}

correct base-pair probability

0.0 0.2 0.4 0.6 0.8 1.0

incorrect base-pair probability

0.01 0.2 0.4 0.6 0.8 1.0

nucleotide positional defect

0.0 0.05 0.1 0.2 0.5 0.8 1.0

(b) Target Structure $\boldsymbol{y}^\star$

(c) MFE Structure (This Work)

(d) MFE Structure (SAMFEO)

(d) Base-Pairing Probabilities (This Work)

(e) Base-Pairing Probabilities (SAMFEO)

Fig. S5. Comparison of the best $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ solution designed by this work vs. SAMFEO for Puzzle 91 ("Thunderbolt"). (b) Target structure: pink-filled regions highlight loops that belong to an undesignable motif, while orange base pairs represent the missing pairs in Sampling's MFE structure. (c) – (d) MFE structures of the best $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ solutions from this work and SAMFEO. (e) – (f) Base-pairing probabilities plots. Base-pairs are colored as follows: blue for correct pairs, red for incorrect pairs, with the intensity indicating pairing probability. Orange represents missing correct pairs (i.e. correct pairs with a pairing probability below 0.1). Nucleotide colors range from blue to red, indicating positional defect. $\tilde{\boldsymbol{y}}$ refers to the target structure with the (orange) base pairs from undesignable motifs removed (i.e. pairs 1, 2, 3 and 4 are removed).

Fig. S6. Comparison of the best $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ solution designed by this work vs. SAMFEO for Puzzle 99 ("Shooting Star"). (b) Target structure: pink-filled regions highlight loops that belong to an undesignable motif, while orange base pairs represent the missing pairs in Sampling's MFE structure. (c) – (d) MFE structures of the best $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ solutions from this work and SAMFEO. (e) – (f) Base-pairing probabilities plots. Base-pairs are colored as follows: blue for correct pairs, red for incorrect pairs, with the intensity indicating pairing probability. Orange represents missing correct pairs (i.e. correct pairs with a pairing probability below 0.1). Nucleotide colors range from blue to red, indicating positional defect. $\tilde{\boldsymbol{y}}$ refers to the target structure with the (orange) base pairs from undesignable motifs removed (i.e. pair 1 is removed).

Tang *et al.*

PNAS | **December 13, 2024** | vol. XXX | no. XX | **7**

(a) Results at different cutoff step

(b) Number of steps for each puzzle (sorted by length →)

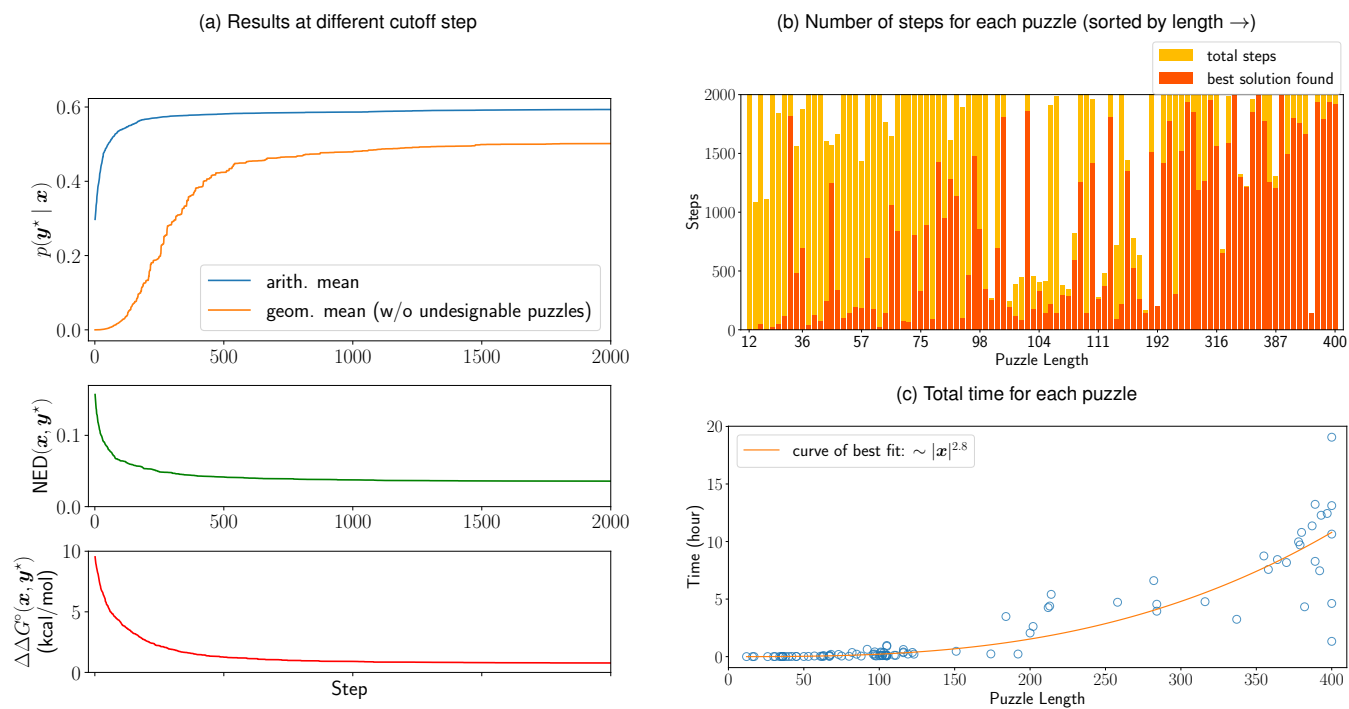**(c) Total time for each puzzle**

Fig. S7. (a) Metrics of the puzzles over different step cutoffs (up to 2000 steps). (b) Number of steps taken to solve each puzzle with the stopping criteria: 50 steps in which the objective function does not improved and the total number of steps is limited to 2000. (c) Total time taken to solve each puzzle, ran on a server with 28 physical cores.
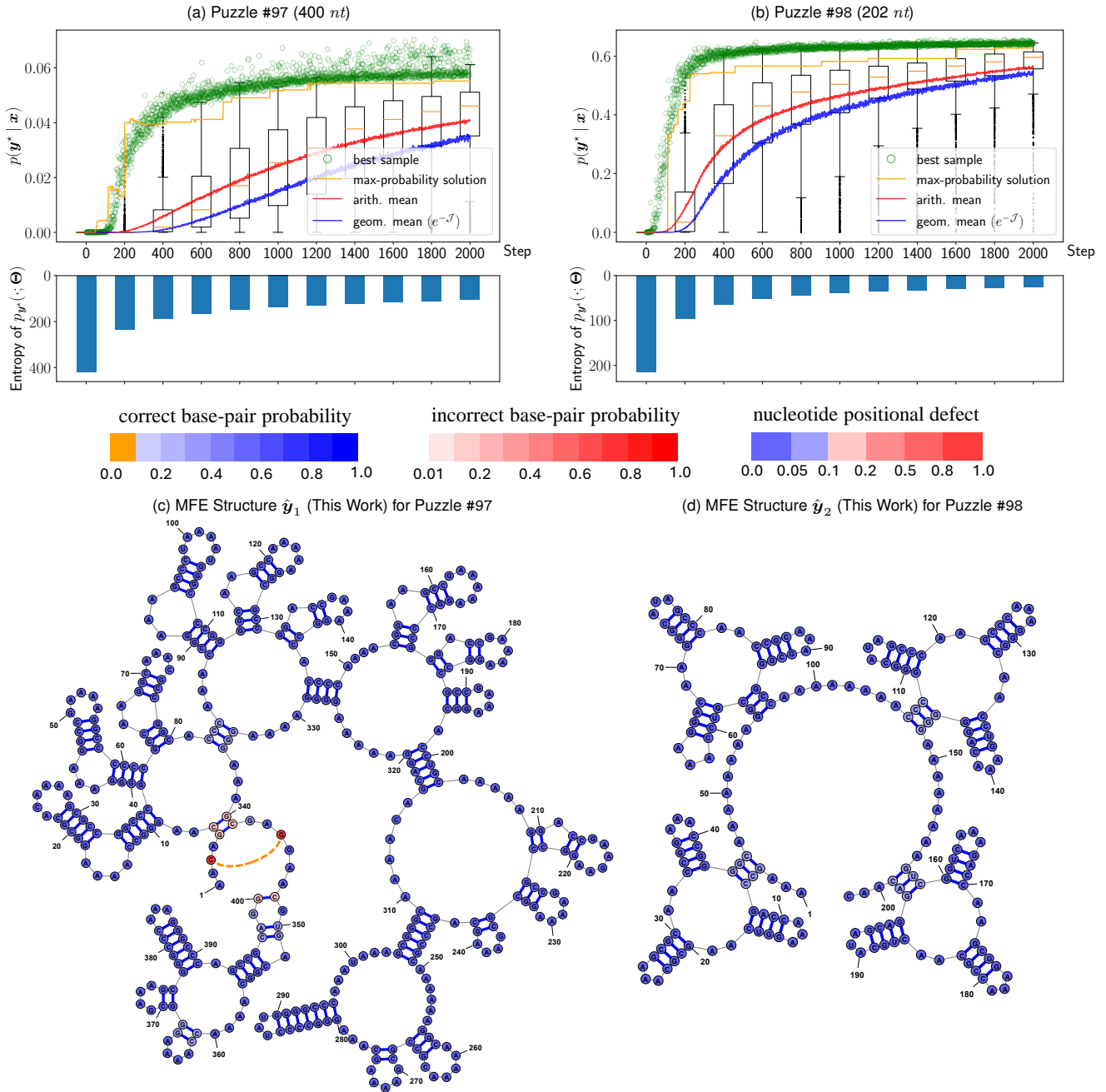
(a) Puzzle #97 (400 nt)

(b) Puzzle #98 (202 nt)

correct base-pair probability

incorrect base-pair probability

nucleotide positional defect

(c) MFE Structure $\hat{\boldsymbol{y}}_1$ (This Work) for Puzzle #97

(d) MFE Structure $\hat{\boldsymbol{y}}_2$ (This Work) for Puzzle #98

Fig. S8. (a) – (b) Learning curves of puzzles #97 and #98. Each step illustrates the $p(\boldsymbol{y}^{\star} \mid \boldsymbol{x})$ for both the best sample and the integral solution with the arithmetic and geometric means of $p(\boldsymbol{y}^{\star} \mid \boldsymbol{x})$ across all samples. The boxplots depict the distribution of $p(\boldsymbol{y}^{\star} \mid \boldsymbol{x})$ across all samples every 200 step, showing the interquartile range and the median of the samples. The barplots correspond to the entropy of the distribution every 200 step. (c) – (d) The MFE structures of the best $p(\boldsymbol{y}^{\star} \mid \boldsymbol{x})$ solution from our method. Puzzle #97 is "unknown" in the sense that it does not have a known uMFE solution and has not yet been proven to be undesignable. The MFE structure of puzzle #97, $\hat{\boldsymbol{y}}_1$, is missing a base pair from the target structure, denoted by the orange dashed line. For this puzzle, $p(\boldsymbol{y}^{\star} \mid \boldsymbol{x}) = 0.078$ and $p(\hat{\boldsymbol{y}}_1 \mid \boldsymbol{x}) = 0.337$.

Tang *et al.*

PNAS | **December 13, 2024** | vol. XXX | no. XX | **9**