Lattice Protein Folding with Variational Annealing

Shoummo Ahsan Khandoker [1], **Estelle M. Inack**[2,3,4], **Mohamed Hibat-Allah**[5, 6]
[1] Department of Computer Science, Indiana University Bloomington, Bloomington, Indiana, USA
[2] Perimeter Institute for Theoretical Physics, Waterloo, Ontario, Canada
[3] yiyaniQ, Toronto, Ontario, Canada
[4] Department of Physics and Astronomy, University of Waterloo, Waterloo, Ontario, Canada
[5] Department of Applied Mathematics, University of Waterloo, Waterloo, Ontario, Canada
[6] Vector Institute, MaRS Centre, Toronto, Ontario, M5G 1M1, Canada
sakhando@iu.edu, einack@perimeterinstitute.ca, mhibatallah@uwaterloo.ca

Abstract

Understanding the principles of protein folding is a cornerstone of computational biology, with implications for drug design, bioengineering, and the understanding of fundamental biological processes. Lattice protein folding models offer a simplified yet powerful framework for studying the complexities of protein folding, enabling the exploration of energetically optimal folds under constrained conditions. However, finding these optimal folds is a computationally challenging combinatorial optimization problem. In this work, we introduce a novel upper-bound training scheme that employs masking to identify the lowest-energy folds in two-dimensional Hydrophobic-Polar (HP) lattice protein folding. By leveraging Dilated Recurrent Neural Networks (RNNs) integrated with an annealing process driven by temperature-like fluctuations, our method accurately predicts optimal folds for benchmark systems of up to 50 beads. Our approach also effectively masks invalid folds from being sampled without compromising the autoregressive sampling properties of RNNs. This scheme is generalizable to three spatial dimensions and can be extended to lattice protein models with larger alphabets. Our findings emphasize the potential of advanced machine learning techniques in tackling complex protein folding problems and a broader class of constrained combinatorial optimization challenges.

Introduction & Previous Work

Protein folding is a biological process in which a linear sequence of amino acids adopts a three-dimensional structure. A correct fold or a misfold can significantly affect the biological health of a living organism (Englander and Mayne 2014). As a result, an accurate understanding of how proteins fold is critical in biology and drug discovery (Dill et al. 2008). The curse of dimensionality of the protein folding space makes it challenging to address using standard computer simulations (som 2005). Lattice protein folding provides a simplified yet insightful framework for studying protein folding dynamics by reducing the complexity of the search space. In the regular lattice, each cell may house an amino acid. It is also common to further simplify this folding process by reducing all 20 types of amino acids to only two types: hydrophobic and polar amino acids, also called beads. These simplifications correspond to the Hydrophobic-polar (HP) lattice protein folding model (Dill 1985). Despite these simplifications, finding the fold with the lowest energy (global minima) is NP-complete for both 2D and 3D HP lattice models (Lau and Dill 1989).

Machine learning tools have already addressed the question of protein folding in different settings. In the continuous folding space, AlphaFold, a machine learning-based approach has achieved remarkable performance on the prediction of protein structures compared to state-of-the-art methods (Jumper et al. 2021). In the discrete folding space, machine learning approaches have also addressed the HP protein folding model in 2D. In particular, FoldingZero (Li et al. 2018) combines deep reinforcement learning (RL) with a two-head deep convolutional neural network (HPNet) and a modified tree search algorithm. This work investigated chain sizes up to 85 with successful runs matching the optimal energy for sizes up to 20. Another RL study (Yu, Schreck, and Ma 2020) also investigated HP chains up to 36 beads using various RL methods such as policy and value iteration, Monte Carlo Tree Search, and AlphaGo Zero with pretraining. According to their results, the adoption of the AlphaGo Zero algorithm exhibits superior performance in comparison to the other methods. Finally, by far the strongest RL work (Yang et al. 2023) obtains optimal folds for HP chains up to 50 beads. They largely attribute the success of their method to the incorporation of Long Short Term Memory (LSTM) architectures in their procedure, which capture long-range interactions in the folding process.

Our work demonstrates the ability of dilated recurrent neural networks (RNNs) (Chang et al. 2017) supplemented with temperature annealing to solve instances of the 2D HP lattice folding model. In all the previous studies, sampling invalid folds has been discouraged by introducing an energy penalty. In our work, we use masking to sample valid folds from RNNs autoregressively to enhance convergence and training stability. We also introduce a novel scheme by introducing a free energy upper bound that stabilizes and enhances RNNs training on the 2D HP model, while preserving their ability to generate folds in the valid folding space autoregressively.

The plan of this paper is as follows: in the methods section, we describe the mathematical details of the HP model and the variational annealing framework that we use in conjunction with dilated RNNs. Additionally, we present our

scheme for projecting dilated RNNs autoregressive sampling to valid folds and show our derived upper bound training which enhances the trainability of RNNs. Finally, in the Results and Discussion section, we highlight empirical evidence in favor of annealing and upper-bound training. We also highlight that our method can find ground state folds up to 50 beads, showing competitive results compared to other machine learning approaches in the literature.

Methods

The HP Model

A fully folded protein chain can be conveniently represented on the 2D Cartesian plane. Let $\Gamma = (\gamma_0, \ldots, \gamma_N) \in \{0,1\}^{N+1}$ represent an HP protein sequence having $N+1$ beads. Here, 0 and 1 denote the 'H' and 'P' beads respectively. A chain Γ having $N+1$ beads implies that N is the number of moves starting from bead γ_0. For a complete fold of Γ, let the Cartesian coordinates of each bead in Γ be $(x_0, y_0), \ldots, (x_N, y_N)$ respectively where $\forall i; x_i, y_i \in \mathbb{Z}$. A fold requires every pair of consecutive beads to be a unit distance away from each other, either on the x-axis or on the $y-$axis, but not both. In other words, the following condition

$$\forall i; |x_i - x_{i+1}| + |y_i - y_{i+1}| = 1 \tag{1}$$

is enforced. A hard constraint on this problem is that a fold must be a self-avoiding walk (SAW), meaning no two beads can have overlapping coordinates. Thus, the constraint

$$\forall i \neq j; (x_i, y_i) \neq (x_j, y_j) \tag{2}$$

must also hold true for a valid fold. To map a protein sequence Γ to some (valid or invalid) fold, we define a sequence of moves to be the solution $\boldsymbol{d} \in \{0,1,2,3\}^N$. A move $d_i \in \boldsymbol{d}$ dictates the coordinates of bead γ_i in a fold given the coordinates of the previous bead γ_{i-1} which are (x_{i-1}, y_{i-1}). Specifically, for $1 \leq i \leq N$, the next bead position is given as follows:

$$(x_i, y_i) = \begin{cases} (x_{i-1} - 1, y_{i-1}), & d_i = 0 \\ (x_{i-1} + 1, y_{i-1}), & d_i = 1 \\ (x_{i-1}, y_{i-1} + 1), & d_i = 2 \\ (x_{i-1}, y_{i-1} - 1), & d_i = 3. \end{cases}$$

The initial position (x_0, y_0) can be set to any reference coordinates, such as $(0,0)$, which we use in this work. Semantically, our chosen convention is such that moves $0, 1, 2$, and 3 corresponds to placing the current bead γ_i to the *'left of', 'right of', 'above',* and *'below'* the previous bead γ_{i-1} respectively on the Cartesian plane.

The energy of a fold $\boldsymbol{d}$ given by $E(\boldsymbol{d})$ is defined as the negative of the number of neighboring or adjacent 'H-H' pairs in the folding space that are not consecutive in the protein sequence itself, as illustrated in Fig. 1. We can denote this number as N_{HH}. To formulate $E(\boldsymbol{d})$ mathematically, we first define the function $M : \mathbb{Z} \times \mathbb{Z} \to \{0,1\}$ as

$$M(x,y) = \begin{cases} 1, & \text{if } (x,y) \text{ is occupied by an } H \text{ bead} \\ 0, & \text{otherwise.} \end{cases}$$

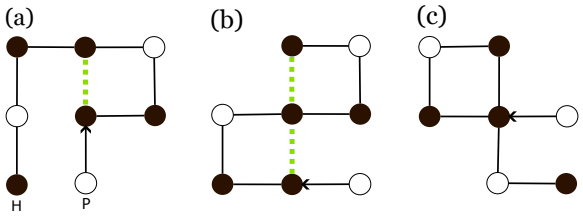

Figure 1: For the protein sequence 'PHHPHHPH', encoded as $\Gamma = (1,0,0,1,0,0,1,0)$, we may have three folds as shown above. The arrow indicates the start of the fold from the first bead and the dotted green line shows the 'H-H' pairs that contribute to energy. **(a)** $\boldsymbol{d} = (2,1,2,0,0,3,3)$ and $E(\boldsymbol{d}) = -1$. **(b)** $\boldsymbol{d} = (0,0,2,1,1,2,0)$ and $E(\boldsymbol{d}) = -2$. **(c)** $\boldsymbol{d} = (0,0,2,1,3,3,1)$ and since $\boldsymbol{d}$ breaks the self-avoiding walk constraint by overlapping the second and sixth beads, $E(\boldsymbol{d}) = 0$.

Using the function M, we define the energy of a *valid* protein sequence fold $\boldsymbol{d}$ as

$$\begin{aligned} E(\boldsymbol{d}) &\equiv -N_{\text{HH}} \\ &= \frac{1}{2} \sum_{i=0}^{N} \Big((1 - \gamma_i)(A_i - \widehat{M}_i) \Big), \end{aligned} \tag{3}$$

where

$$\begin{aligned} \widehat{M}_i =& M(x_{i-1}, y_i) + M(x_{i+1}, y_i) \\ &+ M(x_i, y_{i-1}) + M(x_i, y_{i+1}) \end{aligned}$$

and

$$A_i = (1 - \gamma_{i-1}) + (1 - \gamma_{i+1}).$$

Here, the energy function checks all four neighboring coordinates of γ_i in the fold. An energy contribution of -1 is added in proportion to the number of neighboring 'H' beads through the term proportional to $\widehat{M}_i$. However, we know for all beads, — except for the first and the last — two of the four neighboring beads γ_{i-1} and γ_{i+1} cannot contribute to the energy as they are consecutive to γ_i in the protein sequence. Therefore, we substrate these contributions by adding the term proportional to A_i. For the boundary cases of the first and the last bead, we use $\gamma_{-1} = 1 = \gamma_{N+1}$. Note that the double counting of 'H-H' adjacent pairs is taken into account by dividing by a factor of 2. Lastly, if a fold $\boldsymbol{d}$ is *invalid*, its energy is set to 0 by default.

Variational Learning

Given the NP-completeness of the folding process, it is natural to treat the problem of finding the optimal solution as a combinatorial problem. This leads us to our variational learning approach which consists of sampling solutions from a distribution $P_{\boldsymbol{\theta}}$ characterized by a probabilistic model with parameters $\boldsymbol{\theta}$. In particular, we want $P_{\boldsymbol{\theta}}$ to approximate the Boltzmann distribution at a given temperature T (Wu et al. 2019). To train the model parameters $\boldsymbol{\theta}$, we use the variational free energy

$$F_{\boldsymbol{\theta}}(T) = \mathbb{E}[E(\boldsymbol{d})] + T\mathbb{E}[\log(P_{\boldsymbol{\theta}}(\boldsymbol{d}))], \tag{4}$$

where $-\mathbb{E}[\log(P_{\boldsymbol{\theta}}(\boldsymbol{d}))]$ is the Shannon entropy. Here, $F_{\boldsymbol{\theta}}(T)$ computes the precise free energy over the entire state space $\boldsymbol{d} \in \{0,1,2,3\}^N$ which is intractable to compute exactly. To go around this challenge, we estimate $F_{\boldsymbol{\theta}}(T)$ by drawing M independent samples $\{\boldsymbol{d^{(i)}}\}_{i=1}^{M}$ from the RNN distribution $P_{\boldsymbol{\theta}}$ and we compute an estimate of the free energy as follows:

$$F_{\boldsymbol{\theta}}(T) \approx \frac{1}{M}\sum_{i=1}^{M}\Big(E(\boldsymbol{d}^{(i)}) + T\log(P_{\boldsymbol{\theta}}(\boldsymbol{d}^{(i)}))\Big). \quad (5)$$

Lastly, we note that by virtue of autoregressive sampling, the probability of sampling a fold $\boldsymbol{d} \sim P_{\boldsymbol{\theta}}$ is given by the probability chain rule

$$P_{\boldsymbol{\theta}}(\boldsymbol{d}) = \prod_{i=1}^{N} P_{\boldsymbol{\theta}}(d_i|d_1,\ldots,d_{i-1}), \quad (6)$$

where $P_{\boldsymbol{\theta}}(\boldsymbol{d})$ is the joint probability obtained by the product of all the conditional probabilities. Note that $P(d_i|d_{j<i})$ is the conditional probability of sampling the i^{th} move d_i given the realizations of all previous displacements $\{d_j\}_{j=1}^{i-1}$.

Variational Annealing

In Eq. (5), the term $T\log(P_{\boldsymbol{\theta}}(\boldsymbol{d}^{(i)}))$ can be seen as an entropy regularization term weighted by temperature T (Hibat-Allah et al. 2021; Wu, Wang, and Zhang 2019; Khandoker, Abedin, and Hibat-Allah 2023; Sanokowski et al. 2023). We use this entropy regularization to mitigate the effects of local minima in the optimization landscape (Hibat-Allah et al. 2021) and also to avoid mode collapse (Wu, Wang, and Zhang 2019). T is annealed or cooled from a starting temperature T_0 to a final temperature 0 with the possibility of varying curvatures in its descent depending on the annealing schedule – ranging from a steady, linear decay to a faster, nonlinear decay that follows the curvature of the inverse function for example. Selecting an annealing schedule is a design choice that dictates how fast T decays during the different stages of the annealing process. With these schedules, entropy regularization can be seen as a mechanism that encourages exploration in the folds landscape at high temperatures before exploitation by targeting the low energy folds near zero temperature.

Probabilistic Model

To model the probability distribution from which folds are sampled from $P_{\boldsymbol{\theta}}(\boldsymbol{d})$, we use a Dilated RNN architecture (Chang et al. 2017). The motivation behind using an RNN architecture is to enable autoregressive sampling, which is a form of perfect sampling that mitigates the challenges of Markov Chain sampling schemes of other neural network architectures (Goodfellow 2017). Furthermore, unlike the vanilla RNN model, Dilated RNNs have long recurrent skip connections that allow for the direct propagation of hidden state information from earlier inputs $\boldsymbol{x}_i$ to be utilized further down in the folding process. This is particularly useful in the context of folding as we may want to put an 'H' bead γ_i adjacent to an 'H' bead γ_j where $i-j$ is large. These long-term dependencies benefit from the introduced dilated recurrent connections (Chang et al. 2017).

The Dilated RNN architecture is composed of multiple layers of RNN cells stacked on top of each other as illustrated in Fig. 2. As a design choice, we use $L = \lceil \log_2(N) \rceil$ layers, and each layer has N RNN cells (Hibat-Allah et al. 2021; Khandoker, Abedin, and Hibat-Allah 2023). Here every RNN cell is indexed by layer l where $1 \leq l \leq L$ and column n where $1 \leq n \leq N$. Another design choice is that each of the $L \times N$ RNN cells has its own set of dedicated parameters compared to the traditional practice of using multiple RNN cells sharing the same set of parameters. We use non-weight sharing to take account of the randomness of the chain sequences Γ in a similar spirit to previous work (Khandoker, Abedin, and Hibat-Allah 2023; Hibat-Allah et al. 2021). Parameter notations are as follows: an RNN cell at layer l and column n has the set of weight parameters $W_n^{(l)}$ and $U_n^{(l)}$, bias vector parameter $\boldsymbol{b}_n^{(l)}$, and an associated hidden state vector $\boldsymbol{h}_n^{(l)}$. The hidden state is computed as

$$\boldsymbol{h}_n^{(l)} = \texttt{tanh}(W_n^{(l)}\boldsymbol{h}^{(l)}_{\max(0,n-2^{l-1})} + U_n^{(l)}\boldsymbol{h}_n^{(l-1)} + \boldsymbol{b}_n^{(l)}). \quad (7)$$

Here, $\boldsymbol{x}_{n-1}$ is the input (to the first layer of the Dilated RNN stack) that is a concatenation of the one-hot encoding of the protein bead for which we want to sample a fold for $\boldsymbol{q}_n$ and the one-hot encoding of the previously sampled output fold $\boldsymbol{d}_{n-1}$. More concretely, $\boldsymbol{q}_n$ is a one-hot encoding vector of $\{0,1\}$ where 0 represents 'H' and 1 represents 'P', and $\boldsymbol{d}_n$ is the one-hot encoding vector of integer $d_n \in \{0,1,2,3\}$. This gives us $\boldsymbol{x}_{n-1} = [\boldsymbol{q}_n \frown \boldsymbol{d}_{n-1}]$ where $\frown$ is the concatenation operation. Also, note that the initializations of the hidden state are defined as $\boldsymbol{h}_n^0 = \boldsymbol{x}_{n-1}$ and $\boldsymbol{h}_0^l = [\boldsymbol{q}_0 \frown \boldsymbol{0}]$. To get the output after the last layer of RNN cells, the n^{th} hidden state of the last layer $\boldsymbol{h}_n^{(L)}$ is fed into the respective dense layer having weight V_n and bias $\boldsymbol{c}_n$. As a result, we get the probability distribution for all the four folding directions of the n^{th} displacement. This probability distribution vector $\boldsymbol{P}_n^u \in [0,1]^4$ is computed as

$$\boldsymbol{P}_n^u = \texttt{Softmax}(V_n\boldsymbol{h}_n + \boldsymbol{c}_n). \quad (8)$$

Finally, the n^{th} move is sampled from this distribution with the unmasked conditional probability

$$P^u(d_n|d_{i<n}) = \boldsymbol{P}_n^u \cdot \boldsymbol{d}_n, \quad (9)$$

where $\cdot$ is the dot product. All N conditional probabilities are computed sequentially to compute the joint probability of the list of displacements $\boldsymbol{d}$ in Eq. (6).

Masking and Upper Bound Optimization

Sampling from the valid space of folds is crucial to stabilizing training the RNN architecture and getting low-energy folds. To ensure sampling of the RNN is within the valid space, we mask the invalid moves in each RNN conditional probability $P^u_{\boldsymbol{\theta}}(\boldsymbol{d}_i|\boldsymbol{d}_{j<i})$ as follows:

- If a direction d_i is invalid, then $\log\left(P^u_{\boldsymbol{\theta}}(\boldsymbol{d}_i|\boldsymbol{d}_{j<i})\right)$ is set to $-\infty$.

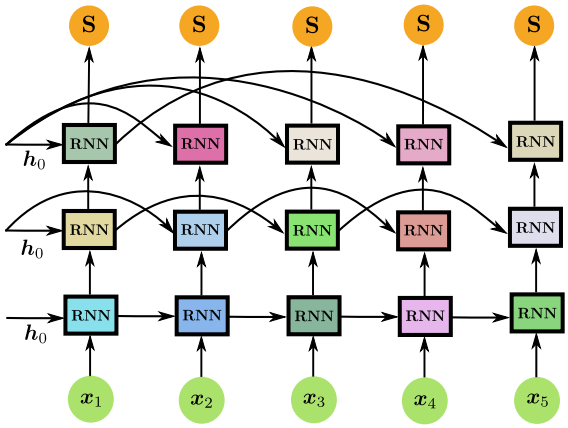

Figure 2: An illustration of a Dilated RNN architecture with $\lceil \log_2(N) \rceil$ layers, where N represents the system size. The architecture incorporates longer recurrent connections to address long-range interactions in the HP lattice protein folding model. The use of distinct colors indicates the absence of weight sharing across different RNN units in the layers. $\boldsymbol{h}_0$ is an initial hidden state initialized as a zero vector and $\boldsymbol{x}_n$ are the inputs which include information about the previous move $\boldsymbol{d}_{n-1}$ and the nature of the bead $\boldsymbol{q}_n$ to be added to the chain at step n.

- We renormalize the four-dimensional log conditional probability by applying the log-softmax activation and we denote it as $\log\left(P_{\boldsymbol{\theta}}(.|\boldsymbol{d}_{j<i})\right)$.

Note that there are dead-end folds, where at a certain step all the local moves are invalid. In this case, the masking procedure is forced to choose a random invalid direction which results in an invalid fold. In this scenario, we discourage the RNN from generating such folds by forcing an energy penalty $E = 0$. Note that the masking step is similar in spirit to other projection schemes explored in the literature (Solozabal, Ceberio, and Takáč 2020; Hibat-Allah et al. 2020).

Although we sample a fold $\boldsymbol{d}$ from the valid fold space, we use the unmasked RNN probability $P^u_{\boldsymbol{\theta}}(\boldsymbol{d})$ for training, which amounts to training an upper bound of the free energy. This choice allows us to stabilize training and obtain lower energy folds as demonstrated in the Results section. To show the upper bound claim, let us focus on the fake loss function used to estimate the gradients of the true free energy:

$$\mathcal{L}(T) = \sum_{\boldsymbol{d}} P^{\perp}_{\boldsymbol{\theta}}(\boldsymbol{d}) \log(P_{\boldsymbol{\theta}}(\boldsymbol{d})) \left(E(\boldsymbol{d}) + T \log(P^{\perp}_{\boldsymbol{\theta}}(\boldsymbol{d}))\right), \tag{10}$$

such that $P^{\perp}_{\boldsymbol{\theta}}$ is the masked RNN probability with a stop gradient assignment $\perp$ (Hibat-Allah et al. 2021; Zhang, Wan, and Yao 2023). Minimizing $\mathcal{L}(T)$ corresponds to the REINFORCE method (Sutton et al. 1999) with a vanilla policy gradient rule (Mohamed et al. 2020; Grooten et al. 2022) supplemented with an entropy term. To train our Dilated RNNs, we use the following cost function:

$$\tilde{\mathcal{L}}(T) = \sum_{\boldsymbol{d}} P^{\perp}_{\boldsymbol{\theta}}(\boldsymbol{d}) \log(P^{u}_{\boldsymbol{\theta}}(\boldsymbol{d})) \left(E(\boldsymbol{d}) + T \log(P^{u\perp}_{\boldsymbol{\theta}}(\boldsymbol{d}))\right) \tag{11}$$

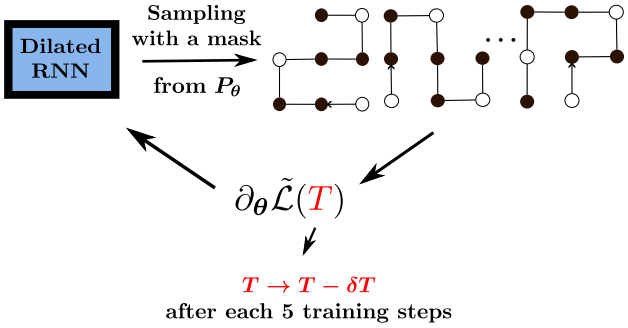

Figure 3: A diagram depicting a dilated recurrent neural network (RNN) training process applied to an HP lattice structure. The RNN samples folds from its parameterized probability distribution with a mask to generate valid folds.

where $P^u_{\boldsymbol{\theta}}$ is the unmasked RNN probability. The following inequality

$$\mathcal{L}(T) \leq \tilde{\mathcal{L}}(T)$$

follows from the observation

$$P_{\boldsymbol{\theta}}(\boldsymbol{d}) \geq P^u_{\boldsymbol{\theta}}(\boldsymbol{d})$$

for all possible valid folds $\boldsymbol{d}$, which implies that:

$$E(\boldsymbol{d}) \log P_{\boldsymbol{\theta}}(\boldsymbol{d}) \leq E(\boldsymbol{d}) \log P^u_{\boldsymbol{\theta}}(\boldsymbol{d})$$
$$\log^2 P_{\boldsymbol{\theta}}(\boldsymbol{d}) \leq \log^2 P^u_{\boldsymbol{\theta}}(\boldsymbol{d}).$$

The first inequality follows from the fact that $E(\boldsymbol{d}) \leq 0$ for all possible folds $\boldsymbol{d} \in \{0, 1, 2, 3\}^N$. Training using the upper bound loss function $\tilde{\mathcal{L}}(T)$ follows a similar spirit to the evidence lower bound (ELBO) when training variational autoencoders (Kingma and Welling 2022). The gradient of the free energy upper bound is given as:

$$\partial_{\boldsymbol{\theta}} \tilde{\mathcal{L}}(T) = \sum_{\boldsymbol{d}} P^{\perp}_{\boldsymbol{\theta}}(\boldsymbol{d}) \partial_{\boldsymbol{\theta}} \log(P^u_{\boldsymbol{\theta}}(\boldsymbol{d})) \left(E(\boldsymbol{d}) + T \log(P^{u\perp}_{\boldsymbol{\theta}}(\boldsymbol{d}))\right),$$

which can be estimated by sampling M folds autoregressively from the RNN as follows:

$$\partial_{\boldsymbol{\theta}} \tilde{\mathcal{L}}(T) \approx \frac{1}{M} \sum_{\boldsymbol{d} \sim P_{\boldsymbol{\theta}}} (\partial_{\boldsymbol{\theta}} \log P_{\boldsymbol{\theta}}(\boldsymbol{d})) \left(\overline{E(\boldsymbol{d})} + T \overline{\log(P_{\boldsymbol{\theta}}(\boldsymbol{d}))}\right). \tag{12}$$

Note that we used the notation $\overline{O(\boldsymbol{d})} \equiv O(\boldsymbol{d}) - \langle O \rangle$, where subtracting the average of the energies and log probabilities was shown to reduce the variance on the gradients as a control variate method (Mohamed et al. 2020; Hibat-Allah et al. 2020; Hibat-Allah, Melko, and Carrasquilla 2023).

Experiments & Results

Training Setup. The training strategy of our algorithm can be summarized by the following steps repeated until temperature T becomes 0:

HP Sequence		Length	$E(\boldsymbol{d}^*)$
20merA	HPHPPHHPHPPHPHHPPHPH	20	-9
20merB	HHHPPHPHPHPPHPHPHPPH	20	-10
24mer	HHPPHPPHPPHPPHPPHPPHPPHH	24	-9
25mer	PPHPPHHPPPPHHPPPPHHPPPPHH	25	-8
36mer	PPPHHPPHHPPPPPHHHHHHHPPHHPPPPHHPPHPP	36	-14
48mer	PPHPPHHPPHHPPPPPHHHHHHHHHHPPPPPHHPPHHPPHPPHHHHH	48	-23
50mer	HHPHPHPHPHHHHPHPPPHPPPHPPPPHPPPHPPPHPHHHHPHPHPHPHH	50	-21

Table 1: HP sequences from the Istrail Benchmark with their best known energies $E(\boldsymbol{d}^*)$.

HP Sequence	$E(\boldsymbol{d}^*)$	Folding-Zero	AlphaGo-Zero	DRL	DQN-LSTM	Variational Annealing (**ours**)
20merA	**-9**	**-9**	-8	-6	**-9**	**-9**
20merB	**-10**	-	-9	-8	**-10**	**-10**
24mer	**-9**	-8	-8	-6	**-9**	**-9**
25mer	**-8**	-7	-7	-	**-8**	**-8**
36mer	**-14**	-13	-13	-	**-14**	**-14**
48mer	**-23**	-18	-	-	**-23**	**-23**
50mer	**-21**	-18	-	-	**-21**	**-21**

Table 2: Lowest energy found by the following methods in order: Folding-Zero (Li et al. 2018), (best results among various methods of) Deep Reinforcement Learning (Yu, Schreck, and Ma 2020), AlphaGo Zero with pretraining (Yu, Schreck, and Ma 2020), Deep Q-Network using Long Short Term Memory (DQN-LSTM) (Yang et al. 2023), and our Variational Annealing method. Bold font represents the lowest energy of the corresponding sequence. '-' means energy was not reported by the corresponding authors.

- For each 5 training steps (i.e., one annealing step),
  - Sample $M = 200$ independent folds from the Dilated RNN.
  - Compute the gradients $\partial_{\boldsymbol{\theta}} \tilde{\mathcal{L}}(T)$ (Eq. 12).
  - Update the RNN parameters using the Adam optimizer (Kingma and Ba 2015).
- Anneal temperature: $T \leftarrow T - \delta T$.

The steps are also illustrated in Fig. 3. For the annealing process, we use a simple linear schedule where the temperature at the $N_{\text{steps}} \in \{1, \ldots, N_{\text{anneal}}\}$ stage in the training has a temperature $T = T_0(1 - N_{\text{steps}}/N_{\text{anneal}})$.

Benchmarks. To demonstrate the empirical value of annealing, we show in Fig. 4(a) a comparison between the expectation values of the energy $\langle E \rangle$ between variational learning with annealing (by setting $T_0 = 1$) and without annealing (by setting $T_0 = 0$). Our findings show that starting at a non-zero temperature during annealing allows our dilated RNN to find a lower energy fold compared to a plain optimization scheme that does not include an entropy regularization.

Additionally, we empirically support the use of masking in tandem with our derived upper bound loss based on the ablation study in the training process as shown in Fig. 4(b). We compare the results of masking with the upper bound loss $\tilde{\mathcal{L}}(T)$, masking with the fake free energy loss $\mathcal{L}(T)$, and no masking with the unmasked loss

$$\mathcal{L}^u(T) = \sum_{\boldsymbol{d}} P^{u\perp}_{\boldsymbol{\theta}}(\boldsymbol{d}) \log(P^u_{\boldsymbol{\theta}}(\boldsymbol{d})) \left(E(\boldsymbol{d}) + T \log(P^{u\perp}_{\boldsymbol{\theta}}(\boldsymbol{d}))\right).$$

From the results of Fig. 4(b), it is clear that enforcing the self-avoiding walk constraint Eq. (2) by masking is a crucial step for enhancing the training of our RNN model. More specifically, both $\tilde{\mathcal{L}}(T)$ and $\mathcal{L}(T)$ have lower values compared to $\mathcal{L}^u(T)$ throughout annealing. Furthermore, we also observe that optimizing the free energy upper bound loss $\mathcal{L}^u(T)$ can result in significantly lower energies compared to the free energy fake loss function $\mathcal{L}(T)$ as illustrated in Fig. 4(b). We believe that masking the RNN probability $P^u_{\boldsymbol{\theta}}$ to obtain $P_{\boldsymbol{\theta}}$ is constraining the optimization of $\mathcal{L}(T)$ when we apply the gradient on the $\log P_{\boldsymbol{\theta}}$ terms in Eq. (10). In contrast, the loss upper bound $\tilde{\mathcal{L}}(T)$ in Eq. (11) circumvents this limitation where the gradients are applied on $\log P^u_{\boldsymbol{\theta}}$ terms (Eq. (11)) that are not constrained by projection steps.

To compare our results with the machine learning literature, we report in Tab. 1 a list of HP sequences with lengths from 20 to 50 and their optimal energies that are often used to benchmark HP folding algorithms (Yang et al. 2023). We use these HP sequences to benchmark our algorithm and compare with recent machine learning methods on the criteria of the lowest energy fold found. The results are presented in Tab. 2. Our findings demonstrate that our variational annealing method with dilated RNNs can find the optimal fold

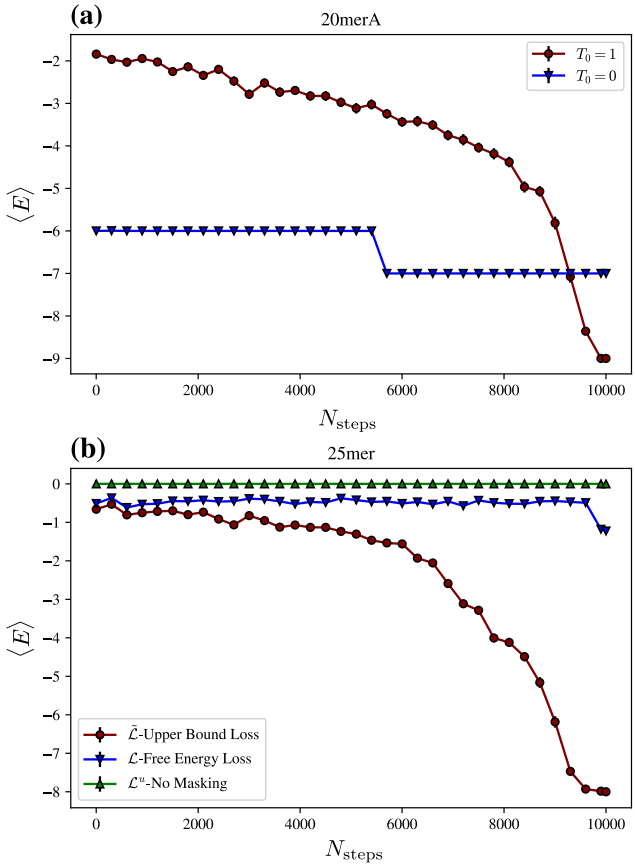

Figure 4: Figures of the training process of the variational annealing approach with a total number of annealing steps $N_{\text{anneal}} = 10000$. **(a)** Demonstrates the effect of annealing in training for the sequence 20merA of the expectation value of the energy $\langle E \rangle$ as a function of the number of temperature annealing steps N_{steps} where each step corresponds to 5 training steps at the same temperature. Note that the user-defined value N_{anneal} is the maximum value that N_{steps} can reach. **(b)** Demonstrates the training process using masking with the upper bound loss $\tilde{\mathcal{L}}(T)$, masking with the fake free energy loss $\mathcal{L}(T)$, and no masking with the unmasked loss $\hat{\mathcal{L}}(T)$ for the sequence 25mer.

for all the sequences up to 50 beads. We also highlight that our method performs better than previous studies that take inspiration from the AlphaGo Zero algorithm (Li et al. 2018; Yu, Schreck, and Ma 2020). Our method is also competitive with the results reported by Ref. (Yang et al. 2023), where we only use a simple policy gradient rule compared to the advanced policy gradient rules used in Ref. (Yang et al. 2023). Additionally, we note that we only use $d_h = 50$ as the size of the hidden state in our dilated RNN compared to the LSTM used in Ref. (Yang et al. 2023) with $d_h \in \{256, 512\}$. This observation highlights the computational efficiency of our method.

Conclusion

In this paper, we have introduced a novel upper-bound training scheme with masking to find the lowest energy fold of the 2D HP lattice protein folding. Using this scheme and by supplementing Dilated RNNs with annealing through temperature-like fluctuations, we found that our method can find the optimal folds of prototypical lattice folding benchmarks with system sizes up to 50 beads. We demonstrate that it is possible to mask moves that lead to invalid folds without compromising the autoregressive sampling feature of RNNs. We also devise a free energy upper bound loss function that enhances the trainability of RNNs that are prone to get compromised by masking the RNN probabilities.

Our scheme is generalizable to three spatial dimensions and also other lattice protein folding models with more than two alphabets such as the 20-letter Miyazawa-Jernigan model (Miyazawa and Jernigan 1996) by enlarging the sizes of the one-hot inputs $\boldsymbol{x}_i$ without compromising inference speed. We also expect inference time to be reduced by introducing an encoder network to enable just-in-time inference of low-energy folds (Sanokowski et al. 2023). For a broader scope, we also believe that our scheme could lead to a promising machine learning-based solution to a wide class of constrained combinatorial optimization problems.

Acknowledgments

EMI acknowledges support from the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Perimeter Institute for Theoretical Physics. Research at Perimeter Institute is supported in part by the Government of Canada through the Department of Innovation, Science and Economic Development Canada and by the Province of Ontario through the Ministry of Economic Development, Job Creation and Trade. We also want to thank Caleb Schultz Kisby for holding valuable discussions about the project. Computer simulations were made possible thanks to the Digital Research Alliance of Canada cluster.

References

2005. So Much More to Know . *Science*, 309(5731): 78–102.

Chang, S.; Zhang, Y.; Han, W.; Yu, M.; Guo, X.; Tan, W.; Cui, X.; Witbrock, M.; Hasegawa-Johnson, M. A.; and Huang, T. S. 2017. Dilated recurrent neural networks. *Advances in neural information processing systems*, 30.

Dill, K. A. 1985. Theory for the folding and stability of globular proteins. *Biochemistry*, 24(6): 1501–1509.

Dill, K. A.; Ozkan, S. B.; Shell, M. S.; and Weikl, T. R. 2008. The Protein Folding Problem. *Annu. Rev. Biophys.*, 37(1): 289–316.

Englander, S. W.; and Mayne, L. 2014. The nature of protein folding pathways. *Proceedings of the National Academy of Sciences*, 111(45): 15873–15880.

Goodfellow, I. 2017. NIPS 2016 Tutorial: Generative Adversarial Networks. arXiv:1701.00160.

Grooten, B.; Wemmenhove, J.; Poot, M.; and Portegies, J. 2022. Is Vanilla Policy Gradient Overlooked? Analyzing Deep Reinforcement Learning for Hanabi. arXiv:2203.11656.

Hibat-Allah, M.; Ganahl, M.; Hayward, L. E.; Melko, R. G.; and Carrasquilla, J. 2020. Recurrent neural network wave functions. *Physical Review Research*, 2(2).

Hibat-Allah, M.; Inack, E. M.; Wiersema, R.; Melko, R. G.; and Carrasquilla, J. 2021. Variational neural annealing. *Nature Machine Intelligence*, 3(11): 952–961.

Hibat-Allah, M.; Melko, R. G.; and Carrasquilla, J. 2023. Investigating topological order using recurrent neural networks. *Physical Review B*, 108(7).

Jumper, J.; Evans, R.; Pritzel, A.; Green, T.; Figurnov, M.; Ronneberger, O.; Tunyasuvunakool, K.; Bates, R.; Žídek, A.; Potapenko, A.; Bridgland, A.; Meyer, C.; Kohl, S. A. A.; Ballard, A. J.; Cowie, A.; Romera-Paredes, B.; Nikolov, S.; Jain, R.; Adler, J.; Back, T.; Petersen, S.; Reiman, D.; Clancy, E.; Zielinski, M.; Steinegger, M.; Pacholska, M.; Berghammer, T.; Bodenstein, S.; Silver, D.; Vinyals, O.; Senior, A. W.; Kavukcuoglu, K.; Kohli, P.; and Hassabis, D. 2021. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873): 583–589.

Khandoker, S. A.; Abedin, J. M.; and Hibat-Allah, M. 2023. Supplementing recurrent neural networks with annealing to solve combinatorial optimization problems. *Machine Learning: Science and Technology*, 4(1): 015026.

Kingma, D.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*. San Diega, CA, USA.

Kingma, D. P.; and Welling, M. 2022. Auto-Encoding Variational Bayes. arXiv:1312.6114.

Lau, K. F.; and Dill, K. A. 1989. A lattice statistical mechanics model of the conformational and sequence spaces of proteins. *Macromolecules*, 22(10): 3986–3997.

Li, Y.; Kang, H.; Ye, K.; Yin, S.; and Li, X. 2018. FoldingZero: Protein Folding from Scratch in Hydrophobic-Polar Model. arXiv:1812.00967.

Miyazawa, S.; and Jernigan, R. L. 1996. Residue – Residue Potentials with a Favorable Contact Pair Term and an Unfavorable High Packing Density Term, for Simulation and Threading. *Journal of Molecular Biology*, 256(3): 623–644.

Mohamed, S.; Rosca, M.; Figurnov, M.; and Mnih, A. 2020. Monte Carlo Gradient Estimation in Machine Learning. *Journal of Machine Learning Research*, 21(132): 1–62.

Sanokowski, S.; Berghammer, W.; Hochreiter, S.; and Lehner, S. 2023. Variational Annealing on Graphs for Combinatorial Optimization. arXiv:2311.14156.

Solozabal, R.; Ceberio, J.; and Takáč, M. 2020. Constrained Combinatorial Optimization with Reinforcement Learning. arXiv:2006.11984.

Sutton, R. S.; McAllester, D.; Singh, S.; and Mansour, Y. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In Solla, S.; Leen, T.; and Müller, K., eds., *Advances in Neural Information Processing Systems*, volume 12. MIT Press.

Wu, D.; Wang, L.; and Zhang, P. 2019. Solving statistical mechanics using variational autoregressive networks. *Physical review letters*, 122(8): 080602.

Wu, H.; Yang, R.; Fu, Q.; Chen, J.; Lu, W.; and Li, H. 2019. Research on predicting 2D-HP protein folding using reinforcement learning with full state space. *BMC Bioinformatics*, 20(25): 685.

Yang, K.; Huang, H.; Vandans, O.; Murali, A.; Tian, F.; Yap, R. H.; and Dai, L. 2023. Applying deep reinforcement learning to the HP model for protein structure prediction. *Physica A: Statistical Mechanics and its Applications*, 609: 128395.

Yu, H.; Schreck, J. S.; and Ma, W.-J. 2020. Deep Reinforcement Learning for Protein Folding in the Hydrophobic-Polar Model with Pull Moves.

Zhang, S.-X.; Wan, Z.-Q.; and Yao, H. 2023. Automatic differentiable Monte Carlo: Theory and application. *Physical Review Research*, 5(3).