STATIONARY DEEP REINFORCEMENT LEARNING WITH QUANTUM K-SPIN HAMILTONIAN REGULARIZATION

Xiao-Yang Liu, Zechu Li, Shixun Wu, Xiaodong Wang
Department of Electrical Engineering, Department of Computer Science
Columbia University, New York, NY, USA
{xl2427, zl2993, sw3511, xw2008}@columbia.edu

ABSTRACT

Instability is a big issue of deep reinforcement learning (DRL) — high variance over multiple runs. It is mainly due to the existence of *many local minima* and worsened by the *multiple fixed points* issue of Bellman's equation. As a fix, we propose a *K-spin Ising model* and minimize its *K-spin Hamiltonian*, which searches for the *ground state* (a.k.a. lowest energy) of a quantum-mechanical system. First, we take a novel quantum perspective by modeling the target problem as a *K-spin Ising model* and employ a Hamiltonian as the objective function. Then, we derive a novel Hamiltonian policy gradient and design a generic actor-critic algorithm that utilizes the K-spin Hamiltonian measure to regularize the policy network, which guides a policy network to converge towards high-quality local minima. Finally, the proposed method reduces the variance of cumulative rewards by $65.2\% \sim 85.6\%$ on six MuJoCo tasks over 20 runs.

1 INTRODUCTION

Instability is a major issue of current deep reinforcement learning (DRL) (Sutton & Barto, 2018) — agents trained with different random seeds may have dramatically different performances. Existing works (Agarwal et al., 2021, Duan et al., 2016, Chan et al., 2019, Henderson et al., 2018, Kool et al., Zoph & Le, 2016) empirically reported a high variance over multiple runs. Such a high variance contributes to people questioning RL's reliability and reproducibility (Dulac-Arnold et al., 2019; 2020), limiting the broader adoption in real life. The instability issue is mainly due to the existence of *many local minima* and worsened by the *multiple fixed points* issue of Bellman's optimality equation (Bertsekas, 2019, Piunovskiy, 2013, Gupta et al., 2021, Eysenbach et al., 2019). In Appx. B, we adapt dynamic programming examples (Bertsekas, 2019, Piunovskiy, 2013) into reinforcement learning settings and provide detailed explanations on the aforementioned issues.

The instability issue has been partially addressed by ensemble methods (Anschel et al., 2017, Chen et al., 2021), regularization approaches (Thodoroff et al., 2018, Cheng et al., 2019), and baseline-correction approaches (Schulman et al., 2016, Wu et al., 2018). In particular, Generalized Advantage Estimation (GAE) (Schulman et al., 2016) is a widely used one to reduce the variance of an advantage function. However, existing methods randomly converge to different local minima. We expect a DRL algorithm to stably converge to a policy independent of initialization and noises for practical usage.

As a fix, we propose a K-spin Hamiltonian formulation, called *H-term*, which guides a policy network to converge towards high-quality local minima. We take a novel quantum perspective by using a *K-spin Ising model* (Kirkpatrick et al., 1983, Denchev et al., 2016) and employ a Hamiltonian to measure a policy's *energy*. We will demonstrate that *a stationary policy would have low energy*.

Related Works Different from our quantum perspective, several recent papers utilized a (classical) Hamiltonian equation to endow RL agents with the capability of inductive biases. (Greydanus et al., 2019, Toth et al., 2019) used Hamiltonian mechanics to train an agent that learns *conservation laws*, (Xu & Fekri, 2021) applied a Hamiltonian Monte Carlo (HMC) simulator to approximate the posterior action probability, and (Loizou et al., 2020) proposed an unbiased estimator for the stochastic Hamiltonian gradient methods for min-max optimization problems.

Table 1: Modeling a task as a K-spin Ising model.

Hamiltonian formulation		Quantum K-spin Ising model	
Transition	$\mu_k \in \mathcal{S} \times \mathcal{S}, k = 0, ..., K-1$	Spin	$j_k \in \{1, \cdots, N\}, k = 0, ..., K-1$
Path prob.	$\pi(\mu_0) \cdots \pi(\mu_{K-1}) \in \{0, 1\}^K$	Config.	$\sigma^*_{j_0} \times \cdots \times \sigma^*_{j_{K-1}} \in \{-1, +1\}^K$
Path reward	$L_{\mu_0 \cdots \mu_{K-1}} = \gamma^{K-1} R(\mu_{K-1})$	Density	$L_{j_0 \cdots j_{K-1}}$
Objective	Eqn. (2)	Energy	Eqn. (3)

2 Hamiltonian Formulation Using K-spin Ising Model

We take a novel quantum perspective by modeling the target problem as a ***K-spin Ising model*** and employ a Hamiltonian measure as the objective function.

Let $\mathcal{S}$ denote the state space and $\mu = (s, s') \in \mathcal{S} \times \mathcal{S}$ denote a (possible) transition with associated reward $R(\mu) \in \mathbb{R}$, where $R(\mu) = 0$ is the transition μ is not feasible. Let a deep neural network with parameter θ approximate a policy $\pi_\theta : \mathcal{S} \to \mathcal{S}$. $\pi(s, \cdot)$ is a probability distribution over $\mathcal{S}$. For $\mu(s, s') \in \mathcal{S} \times \mathcal{S}$, let $\pi(\mu) = \pi(s, s')$, where $\pi(s, s') = 0$ is the transition from s to s' is not feasible. The conventional objective function $J(\theta)$ of reinforcement learning (Sutton & Barto, 2018) is

$$J(\theta) \triangleq \mathbb{E}_{\tau \sim \pi_\theta} \left[R(\tau) \cdot \mathbb{P}(\tau | \pi_\theta) \right], \tag{1}$$

where $\mathbb{P}(\tau|\pi_\theta)$ is the probability of trajectory $\tau = (S_0, \cdots, S_T)$ following policy π_θ.

We propose to use a Hamiltonian measure as the objective function:

$$H(\theta) = \sum_{k=0}^{K-1} \sum_{\mu_0}^{\mathcal{S} \times \mathcal{S}} \cdots \sum_{\mu_k}^{\mathcal{S} \times \mathcal{S}} L_{\mu_0, ..., \mu_k} \pi_\theta(\mu_0) \cdots \pi_\theta(\mu_k), \qquad L_{\mu_0, ..., \mu_k} = \gamma^k R(\mu_k), \tag{2}$$

where $L_{\mu_0, ..., \mu_k}$ is the (discounted) reward on a path segment $(\mu_0, \cdots, \mu_k)$.

Remark: 1). There is an implicit action space $\mathcal{A}$; 2). (2) is equal to (1) when $K \to \infty$; and 3). K can be much smaller than T, e.g., $K = 5, 10$ and $T = 100, 1000$.

Analogy in Table 1. The Hamiltonian in (2) draws an analogy from a quantum K-spin Ising model (Kirkpatrick et al., 1983, Denchev et al., 2016) whose Hamiltonian is

$$H(\sigma) = -\sum_{k=0}^{K-1} \sum_{j_0=1}^{N} \cdots \sum_{j_k=1}^{N} L_{j_0 \cdots j_k} \sigma_{j_0} \cdots \sigma_{j_k}, \tag{3}$$

where N is the number of spins, $\sigma_{j_k} = \pm 1$ are spin variables, and $L_{j_0 \cdots j_k}$ is an energy density function for $(\sigma_{j_0}, \ldots, \sigma_{j_k})$. A transition $\mu_k \in \mathcal{S} \times \mathcal{S}$ corresponds to a spin j_k, we map an optimal policy $\pi^*(\mu_k) \in \{0, 1\}$ to the optimal spin configuration $\sigma^*_{j_k} \in \{1, -1\}$, i.e., $\pi^*(\mu_k) \longleftrightarrow (1 - \sigma^*_{\mu_k})/2$. The energy density $L_{\mu_0, ..., \mu_k} = \gamma^k R(\mu_k)$ is the (discounted) reward on a path segment $(\mu_0, \cdots, \mu_k)$.

3 Stationary Deep Reinforcement Learning

3.1 Hamiltonian Policy Gradient and Monte Carlo-based Gradient Estimator

Theorem 1. *The Hamiltonian gradient of (2) is*

$$\nabla_\theta H(\theta) = -\mathbb{E}_{\mu_0, ..., \mu_{K-1}} \left[\sum_{k=0}^{K-1} \gamma^k \cdot R(\mu_k) \cdot \nabla_\theta \log \left(\pi_\theta(\mu_0) \cdot \pi_\theta(\mu_1) \cdots \pi_\theta(\mu_k) \right) \right]. \tag{4}$$

We obtain the Monte Carlo gradient estimator of $\nabla_\theta H(\theta)$, illustrated in Fig. 1 (right), as follows

$$\nabla_\theta \widehat{H}(\theta) = -\frac{1}{N'} \sum_{i=1}^{N'} \left[\sum_{k=0}^{K-1} \gamma^k \cdot R(\mu^i_k) \cdot \nabla_\theta \log \left[\pi_\theta(\mu^i_0) \cdots \pi_\theta(\mu^i_k) \right] \right]. \tag{5}$$

As shown in Fig. 1, REINFORCE's policy gradient (Sutton et al., 1999) uses discounted rewards from future steps, while Hamiltonian policy gradient splits a trajectory into segments.

Computational complexity: we measure the computation complexity by the times of computing one $\nabla_\theta \log \pi_\theta(\mu)$. Assume $N = B$ and $N' = B'$, since most DRL algorithms use mini-batch stochastic gradient descent methods. REINFORCE's (Sutton et al., 1999) policy gradient takes $O(BT)$ computations, while we add $O(B'K(K+1)/2)$ computations in each gradient update step, thus a total complexity of $O(BT + B'K(K+1)/2)$.

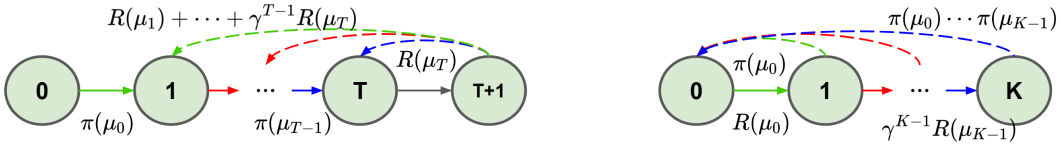


Figure 1: REINFORCE's policy gradient (left) vs. Hamiltonian's policy gradient (right).

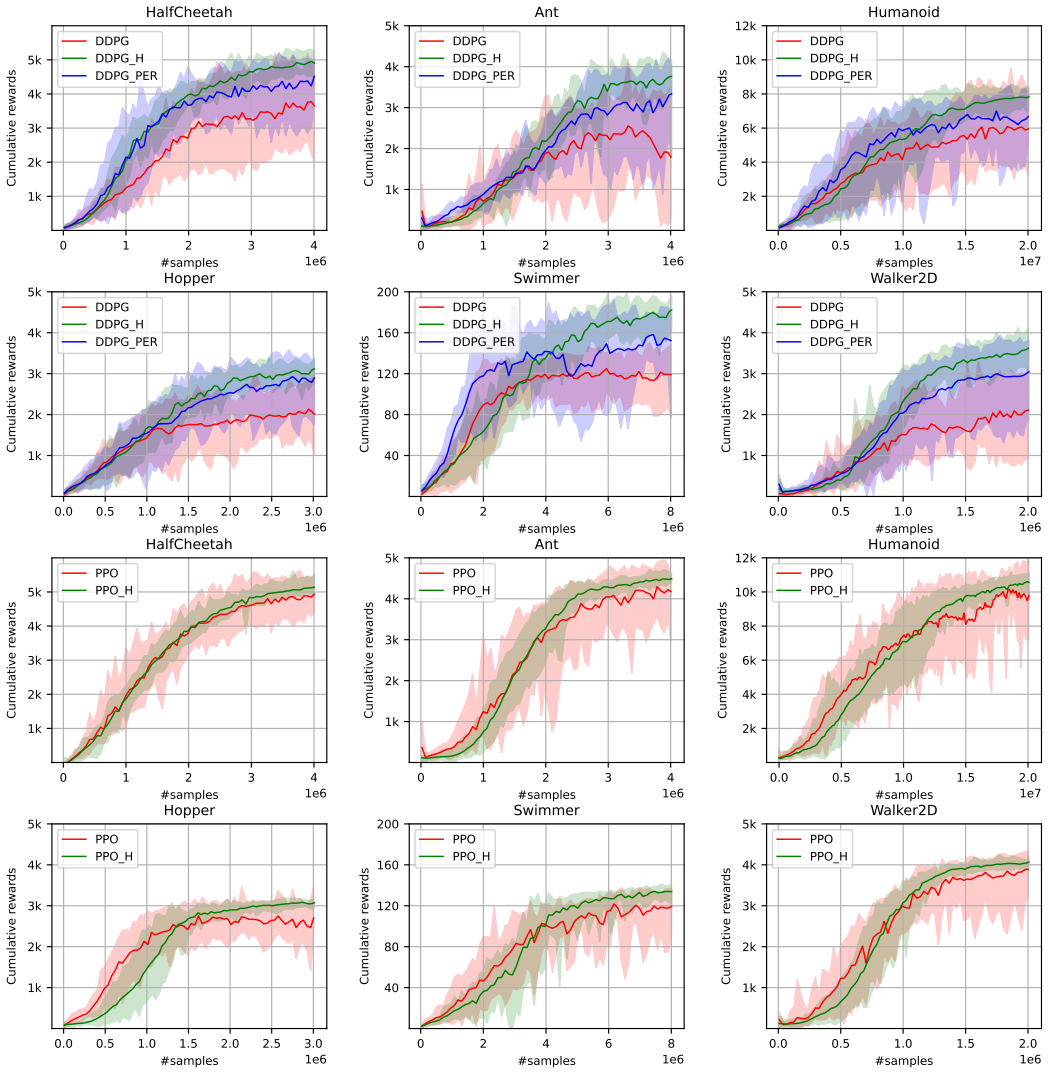


Figure 2: Cumulative rewards vs. #samples for compared DRL algorithms on six MuJoCo tasks.

3.2 Stationary Actor-Critic Algorithm with H-term

We use the proposed H-term as an add-on term to help the actor network converge to a stationary policy. Specifically, the objective functions of actor and critic networks become:

$$\begin{cases} \text{Actor}: \max\limits_{\theta} J_\pi(\theta, \phi) \triangleq (1-\gamma)\mathbb{E}_{S_0 \sim d_0, A_0 \sim \pi_\theta(S_0,\cdot)} \left[Q_\phi \left(S_0, A_0 \right) \right] - \lambda H(\theta), \\ \text{Critic}: \min\limits_{\phi} J_Q(\theta, \phi) \triangleq \dfrac{1}{2} \mathbb{E}_{S \sim d_\theta(\cdot), A \sim \pi_\theta(S,\cdot)} \left[\left(Q_\phi(S, A) - y(S, A) \right)^2 \right], \end{cases} \tag{6}$$

where $y(S_k, A_k) = R(S_k, A_k) + \gamma Q_\phi(S_{k+1}, A_{k+1})$ and $\lambda > 0$ is a temperature parameter.

4 PERFORMANCE EVALUATION

We consider six MuJoCo tasks (Todorov et al., 2012) with high-dimensional continuous state space and action space, in which multiple locally optimal policies exist as revealed in Appx. C. To evaluate both deterministic and stochastic algorithms, we choose Deep Deterministic Policy Gradient (Lillicrap et al., 2016) with Prioritized Experience Replay (Schaul et al., 2016) (DDPG+PER) and Proximal Policy Optimization (PPO) (Schulman et al., 2017) with the GAE trick (Schulman et al., 2016). For a fair comparison, we keep the hyperparameters (listed in Appx. C) the same and make sure that the obtained results reproduce existing benchmark tests (Duan et al., 2016).

H-term helps converge to a high-quality local minima. In Fig. 2, both DDPG+PER and DDPG+H achieve a substantial improvement in cumulative reward. In particular, DDPG+H achieves the highest cumulative rewards in all six tasks, which are comparable to PPO's performance in Fig. 2. It is worthwhile to point out the advantage of DDPG+H over DDPG+PER. DDPG+PER utilizes a prioritized replay strategy to obtain a more accurate critic network, while the H-term in DDPG+H is performed on the actor network. Our results indicate that an experience replay technique on the actor network may be more powerful.

H-term helps reduce variance. The PPO algorithm with GAE is regarded as the state-of-the-art algorithm in MuJoCo environments. However, it still has a very high variance (the shaded area) after the policies have converged, as shown in Fig. 2. Such a high variance is mainly due to the fact that the agent may converge to a random one of multiple policies.

In Fig. 2, the shaded areas of PPO+H ($K = 16$) are dramatically smaller, i.e., a variance of 228.4, 225.4, 683.7, 184.2, 31.6, and 296.8, respectively. The variance has been reduced by $65.2\% \sim 85.6\%$, which verifies the effectiveness of the proposed H-term. Therefore, we may conclude that the H-term guides the agent to search for a stationary policy among multiple feasible ones.

H-term helps drive to physically stationary policy. A key question needs to be answered: *does H-term help guide the agent converge to a physically stationary policy?* We perform observational experiments on MuJoCo tasks and measure the number of convergences to different policies over 20 runs. The vanilla PPO algorithm converges to the physically stationary policy (**bold**) with 13, 17, 7, 10, 14, and 5 times for the six tasks, while the PPO+H ($K = 16$) converges to the stationary policy with 20, 20, 16, 20, 20, and 16 times, respectively. From the empirical observation, we find that the PPO gets stuck in locally optimal policies, failing to find a consistent one. As expected, PPO+H converges to the stationary policy with a substantially higher ratio, which verifies the effectiveness of the proposed H-term in finding a physically stationary policy.

Impact of trajectory length K. We investigate the impact of trajectory length K. From (5), we know that a large K means a more accurate estimation of $\nabla_\theta H(\theta)$ but at a price of computations. Here, we evaluate PPO+H with $K = 8, 16$ and set the size of replay buffer $\mathcal{D}_2$ to $1,000$. In Table 5, we observe that the cumulative reward increases and the variance decrease as K increases from 8 to 16. However, for the case $K = 24$, both metrics get worse due to the out-of-memory issue and we reduce the replay buffer size to 800. The smaller replay buffer size hurts the diversity of the trajectories and may lead to a performance drop. Appx. C provides results for replay buffer size 800.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.

Oron Anschel, Nir Baram, and Nahum Shimkin. Averaged-DQN: Variance reduction and stabilization for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 176–185. PMLR, 2017.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual International Conference on Machine Learning*, pp. 41–48, 2009.

Una Benlic and Jin-Kao Hao. Breakout local search for the max-cut problem. *Engineering Applications of Artificial Intelligence*, 26(3):1162–1173, 2013.

Dimitri Bertsekas. *Reinforcement learning and optimal control*. Athena Scientific, 2019.

Emil Björnson, Eduard Jorswieck, et al. Optimal resource allocation in coordinated multi-cell systems. *Foundations and Trends® in Communications and Information Theory*, 9(2–3):113–381, 2013.

Stephanie CY Chan, Samuel Fishman, Anoop Korattikara, John Canny, and Sergio Guadarrama. Measuring the reliability of reinforcement learning algorithms. 2019.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double Q-learning: Learning fast without a model. *ICLR*, 2021.

Richard Cheng, Abhinav Verma, Gabor Orosz, Swarat Chaudhuri, Yisong Yue, and Joel Burdick. Control regularization for reduced variance reinforcement learning. In *International Conference on Machine Learning*, pp. 1141–1150. PMLR, 2019.

Changhui Choi and Yinyu Ye. Solving sparse semidefinite programs using the dual scaling algorithm with an iterative solver. *Manuscript, Department of Management Sciences, University of Iowa, Iowa City, IA*, 52242, 2000.

G Daniel, Johnnie Gray, et al. Opt_einsum-a python package for optimizing contraction order for einsum-like expressions. *Journal of Open Source Software*, 3(26):753, 2018.

Vasil S Denchev, Sergio Boixo, Sergei V Isakov, Nan Ding, Ryan Babbush, Vadim Smelyanskiy, John Martinis, and Hartmut Neven. What is the computational value of finite-range tunneling? *Physical Review X*, 6(3):031015, 2016.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pp. 1329–1338. PMLR, 2016.

Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. 2019.

Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. An empirical investigation of the challenges of real-world reinforcement learning. *arXiv e-prints*, pp. arXiv–2003, 2020.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *ICLR*, 2019.

Johnnie Gray and Stefanos Kourtis. Hyper-optimized tensor network contraction. *Quantum*, 5:410, 2021.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Agrim Gupta, Silvio Savarese, Surya Ganguli, and Li Fei-Fei. Embodied intelligence via learning and evolution. *Nature Communications*, 12(1):1–12, 2021.

Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2023. URL https://www.gurobi.com.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Cupjin Huang, Fang Zhang, Michael Newman, Xiaotong Ni, Dawei Ding, Junjie Cai, Xun Gao, Tenghui Wang, Feng Wu, Gengyan Zhang, et al. Efficient parallelization of tensor network contraction for simulating quantum computation. *Nature Computational Science*, 1(9):578–587, 2021.

Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

Gary A Kochenberger, Jin-Kao Hao, Zhipeng Lü, Haibo Wang, and Fred Glover. Solving large scale max cut problems via tabu search. *Journal of Heuristics*, 19:565–571, 2013.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *ICLR*, 2016.

Nicolas Loizou, Hugo Berard, Alexia Jolicoeur-Martineau, Pascal Vincent, Simon Lacoste-Julien, and Ioannis Mitliagkas. Stochastic Hamiltonian gradient methods for smooth games. In *International Conference on Machine Learning*, pp. 6370–6381. PMLR, 2020.

Eli Meirom, Haggai Maron, Shie Mannor, and Gal Chechik. Optimizing tensor network contraction using reinforcement learning. In *International Conference on Machine Learning*, pp. 15278–15292. PMLR, 2022.

Alexey B Piunovskiy. *Examples in markov decision processes*, volume 2. World Scientific, 2013.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *ICLR*, 2016.

Martin JA Schuetz, J Kyle Brubaker, and Helmut G Katzgraber. Combinatorial optimization with physics-inspired graph neural networks. *Nature Machine Intelligence*, 4(4):367–377, 2022.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *ICLR*, 2016.

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

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12, 1999.

Pierre Thodoroff, Audrey Durand, Joelle Pineau, and Doina Precup. Temporal regularization for markov decision process. *Advances in Neural Information Processing Systems*, 31, 2018.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Jan Toenshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Graph neural networks for maximum constraint satisfaction. *Frontiers in Artificial Intelligence*, 3:580607, 2021.

Peter Toth, Danilo Jimenez Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. Hamiltonian generative networks. *ICLR*, 2019.

Cathy Wu, Aravind Rajeswaran, Yan Duan, Vikash Kumar, Alexandre M Bayen, Sham Kakade, Igor Mordatch, and Pieter Abbeel. Variance reduction for policy gradient with action-dependent factorized baselines. *ICLR*, 2018.

Duo Xu and Faramarz Fekri. Improving actor-critic reinforcement learning via Hamiltonian Monte Carlo method. *Deep Reinforcement Learning Workshop at NeurIPS*, 2021.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *International Conference on Learning Representations*, 2016.

A Background of Reinforcement Learning and Bellman Equation

In reinforcement learning (RL) (Sutton & Barto, 2018), an agent learns by interacting with an unknown environment and obtains a policy by the maximization of cumulative rewards. Mathematically, it can be formulated as a Markov Decision Process (MDP) with the five-tuple $\langle \mathcal{S}, \mathcal{A}, \mathbb{P}, R, \gamma \rangle$. Here $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces; $\mathbb{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ denotes a transition probability function, where Δ is a probability simplex; $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ denotes a reward function; and $\gamma \in (0, 1]$ denotes a discount factor. The objective is to find an optimal policy $\pi^* : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes (discounted) expected reward. Consider a discrete, finite, discounted MDP with infinite horizon, one can define the Q-value function of a state-action pair (s, a) under policy π as follows

$$Q^\pi(s,a) = \mathbb{E}_{S_{k+1}\sim\mathbb{P}(\cdot|S_k,A_k),A_{k+1}\sim\pi(S_{k+1},\cdot)}\left[\sum_{k=0}^{\infty}\gamma^k \cdot R(S_k, A_k, S_{k+1})|S_0 = s, A_0 = a\right], \quad (7)$$

where $R(S_k, A_k, S_{k+1})$ denotes the immediate reward when taking action A_k at state S_k and arriving at state S_{k+1}, capital letters denote random variables and lowercase letters denote values.

The Bellman equation (Sutton & Barto, 2018) converts (7) into a recursive form as follows

$$\begin{aligned} Q^\pi(s,a) &= \sum_{s'\in\mathcal{S}} \mathbb{P}(s' \mid s, a)\left[R(s,a,s') + \gamma \sum_{a'\in\mathcal{A}} \pi(s',a')Q^\pi(s',a')\right] \\ &= R(s,a) + \gamma \sum_{s'\in\mathcal{S}} \mathbb{P}(s' \mid s, a) \sum_{a'\in\mathcal{A}} \pi(s',a')Q^\pi(s',a'), \end{aligned} \quad (8)$$

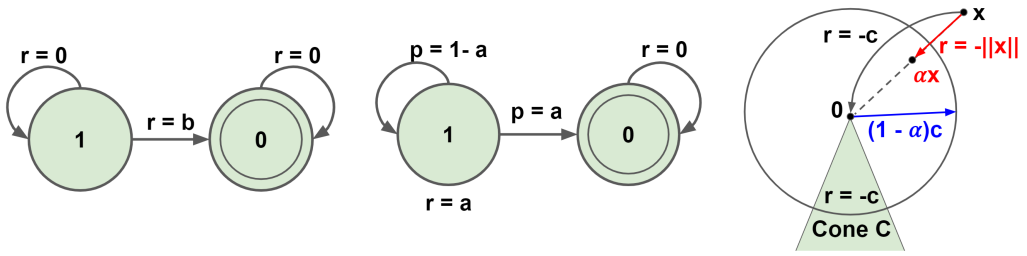
which expresses the expected reward as a summation of immediate reward $R(s, a)$ and discounted future rewards, and the immediate reward $R(s, a)$ is defined as $R(s,a) = \sum_{s'\in\mathcal{S}} \mathbb{P}(s' \mid s, a)R(s, a, s')$.

The Bellman's optimality equation (Sutton & Barto, 2018) is

$$Q^*(s,a) = \sum_{s'\in\mathcal{S}} \mathbb{P}(s' \mid s, a)\left[R(s,a,s') + \gamma \max_{a'} Q^*(s',a')\right]. \quad (9)$$

The optimal policy π^* is given by

$$\pi^* = \arg\max_{\pi}\ Q^\pi(s,a). \quad (10)$$



(a) Shortest path problem (b) Blackmailer's problem (c) minimal stopping problem

Figure 3: Examples with $\gamma = 1$. Examples with $\gamma < 1$ are given in Fig. 4.

B Bellman Equation has an Issue of Multiple Fixed Points

In Fig. 3, we adapt dynamic programming examples (Bertsekas, 2019, Piunovskiy, 2013) into reinforcement learning settings.

- **Shortest path problem (deterministic)** in Fig. 3(a): two policies, 1) transiting back to state 1; 2) driving to terminal state 0.
- **Blackmailer's problem (stochastic)** in Fig. 3(b): two policies, 1) demanding $a \to 0$ to keep the victim at state 1; 2) demanding $a = 1$ that drives the victim to terminate state 0.
- **Optimal stopping problem (terminating policies)** in Fig. 3(c): two polices, 1) continuing inside the sphere of radius $(1 - \alpha)c$ and stopping outside; 2) jumping to point 0 at any point in region C.

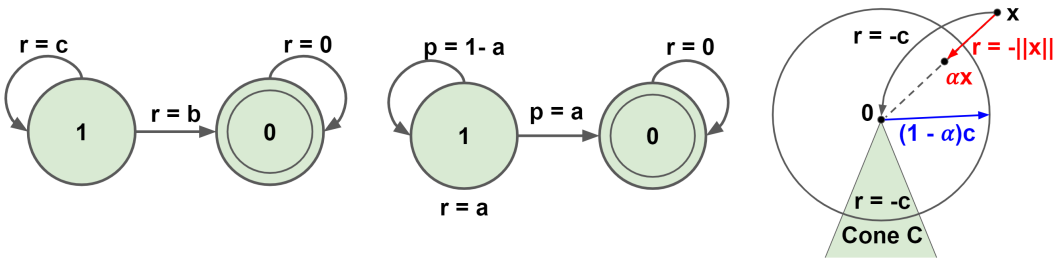
B.1 Case of $\gamma = 1$ in Fig. 3

Consider MDP examples with an terminal state 0, as shown in Fig. 3 (we adapt dynamic programming examples (Bertsekas, 2019, Piunovskiy, 2013) into reinforcement learning settings),

- **Shortest path problem (deterministic)** in Fig. 3(a): At state 1, an agent transits to either state 1 or 0 with reward 0 or b, respectively. Assume the value function for state 0 is $V(0) = 0$. The Bellman's optimality equation for state 1 is $V(1) = \max\{V(1), b\}$, where any $V(1) \geq b$ is a feasible solution. If initialize $V_0(1) \geq b$, a resulting policy is that an agent at state 1 always transits back to state 1; otherwise, drives to terminal state 0 (always returns back to itself with reward 0).
- **Blackmailer's problem (stochastic)** in Fig. 3(b): At state 1, a profit maximizing blackmailer demands a cash amount $a \in (0, 1]$; a victim transits to state 1 with probability a or state 0 with probability $1 - a$, respectively. At state 0, a victim always refuses to yield, i.e., $V(0) = 0$. The Bellman's optimality equation for state 1 is $V(1) = \max_a\{a + (1 - a)V(1)\}$, where any $V(1) \geq 1$ is a feasible solution. If initialize $V_0(1) > 1$, the blackmailer's policy is demanding $a \to 0$ to keep the victim at state 1; otherwise, demanding $a = 1$ that drives the victim to terminal state 0.
- **Optimal stopping problem (terminating policies)** in Fig. 3(c): In a space $\mathbb{R}^2$ with terminal state of point 0, an agent at point $x \neq 0$ moves to either point 0 with negative reward $-c$ or point αx with reward $-||x||$, respectively, where $\alpha \in (0, 1)$. The Bellman's optimality equation is $V(x) = \max\{-c, -||x|| + V(\alpha x)\}$ and the optimal policy is to continue inside the sphere of radius $(1 - \alpha)c$ and to stop outside. If add a cone region C within which an agent always receives a reward $-c$, a second policy is jumping to point 0 at any point in region C.

We elaborate how adding the energy measured by (2) onto each state can help drive to the terminal state (a stationary policy), which fixes the foundational issue of multiple fixed points in Fig. 3 where $\gamma = 1$. We have $H(0) = 0$ for the terminal state 0.

- (a) **Shortest path problem (deterministic)**: $H(1) = -\sum_{k=1}^{\infty} b = -\infty$. At state 1, the Bellman's optimality equation becomes $V(1) = \max\{V(1) + \lambda H(1), b\}$. Independent of the initial value $V_0(1)$, an agent obtains a policy that always transits back to terminal state 0.
- (b) **Blackmailer's problem (stochastic)**: $H(1) = -\infty$. The Bellman's optimality equation becomes $V(1) = \max_a\{a + (1 - a)(V(1) + \lambda H(1))\}$ for state 1. For any $V_0(1) < \infty$, the optimal policy becomes $a = 1$ that drives to the terminal state 0.
- (c) **Optimal stopping problem (terminating policies)**: any policy that takes infinite steps will have $H(x) = -\infty$, since at each step number k, there are always trajectories that jump to point 0



(a) Shortest path problem (b) Blackmailer's problem (c) Optimal stopping problem

Figure 4: Revisiting Fig. 3 for the discounted cases where $\gamma \in (0, 1)$.

with reward $-c$; and a direct jumping policy will have $H(x) = -c$. Therefore, adding $H(x)$ to each point $x \neq 0$ will lead to a policy of *jumping back to point* 0.

B.2 Case of $\gamma \in (0, 1)$ in Fig. 4

First, we consider the discounted formulations of the three examples (shown in Fig. 3), as shown in Fig. 3 where $\gamma \in (0, 1)$. The differences are marked in red.

- (a) **Shortest path problem (deterministic, discounted case)**: Given two states 1 and 0, an agent at state 1 transits to either state 1 or 0 with rewards $r = c$ or $r = b$, respectively. $c > (1 - \gamma) \cdot b$. At state 0, the value function is $V(0) = 0$. At state 1, the Bellman's optimality equation is $V(1) = \max\{c + \gamma \cdot V(1), b\}$, where any $V(1) \geq (b - c)/\gamma$ is a solution. If initialize $V_0(1) \geq b$, an agent obtains a policy that always transits back to state 1; otherwise, a result policy drives to terminal state 0.
- (b) **Blackmailer's problem (stochastic, discounted case)**: Different from (a), a profit maximizing blackmailer/agent at 1 demands a cash amount $a \in (0, 1]$ (an action), while a victim transits to state 1 with probability a or to state 0 with probability $1 - a$, respectively. At state 0, a victim always refuses to yield to the blackmailer's demand, i.e., $V(0) = 0$. The Bellman's optimality equation is $V(1) = \max_a\{a + \gamma \cdot (1 - a)V(1)\}$ for state 1, where any $V(1) \geq 1$ is a feasible solution. If initialize $V_0(1) = c > 1$, the blackmailer's policy is demanding $a \to 0$ at the k-th step to keep the victim stay at state 1, for any $k \leq K_0 = -\lfloor \log_\gamma c \rfloor$; and taking $a = 1$ to transit to terminal state 0 at the k-th step, for any $k \geq K_0 + 1$; otherwise initialize $V_0(1) = c \leq 1$, the result policy is demanding the maximum $a = 1$ that drives the victim to a refusal state 0 (a terminal state).
- (c) **Optimal stopping problem (terminating policies, discounted case)**: In a space $\mathbb{R}^2$ with terminating state at point 0, at point $x \neq 0$ an agent moves to either point 0 with negative reward $-c$ or point αx with reward $-||x||$, respectively, where $\alpha \in (0, 1)$. The Bellman's optimality equation is $V(x) = \max\{-c, -||x|| + \gamma \cdot V(\alpha x)\}$ and the optimal policy is to continue inside the sphere of radius $(1 - \alpha)c$ and to stop outside. If add a cone region C within which an agent always receives a reward $-c$, a second policy is jumping to point 0 at any point in region C.

Then, we elaborate how the proposed H-term fixes the problems in Fig. 3.

(a) Shortest path problem (deterministic, discounted case)

Assume $V_0(1) \geq b$ and $c > (1-\gamma)b$, we have $V_k(1) = \sum_{i=0}^{k-1} \gamma^i \cdot c + \gamma^k \cdot V_0(1) \geq \sum_{i=0}^{k-1} \gamma^i \cdot c + \gamma^k b > b$ and $V^*(1) = \sum_{i=0}^{\infty} \gamma^i \cdot c = \frac{1}{1-\gamma} c > b$. The values of $H(0)$ and $H(1)$ are $H(0) = 0$ and $H(1) = -b - \sum_{k=2}^{\infty}(\sum_{i=1}^{k-1} \gamma^{i-1} \cdot c + \gamma^k b) = -\infty$. Adding the above H-values to state 1 and 0, respectively, we have $V^*(0) + H(0) = b$, and $V^*(1) + H(1) = -\infty$. Therefore, $V^*(1) + H(1) < V^*(0) + H(0)$, independent of the initial value $V_0(1)$. That is, an agent always obtains a policy that drives to terminal state 0 at step 1.

(b) **Blackmailer's problem (stochastic, discounted case)**

If initialize $V_0(1) = c > 1$, the blackmailer's policy is demanding $a \to 0$ at the k-th step to keep the victim stay at state 1, for any $k \leq K_0 = -\lfloor \log_\gamma c \rfloor$; and taking $a = 1$ to transit to terminal state 0 at the k-th step, for any $k \geq K_0 + 1$; otherwise initialize $V_0(1) = c \leq 1$, the result policy is demanding the maximum $a = 1$ that drives the victim to a refusal state 0 (a terminal state).

We have $H(0) = 0$ and $H(1) = -\sum_{k=1}^{\infty} \sum_{i=1}^{k-1} \gamma^{i-1} \cdot a = -\infty$. For arbitrary initial value of $V_0(1)$, $V_1(1) = a + (1-a) \cdot \gamma(V_0(1) + H(1))$ take maximum $V_1(1) = 1$ when $a = 1$. Therefore, the policy always drives to terminal state 0 at step 1.

(c) **Optimal stopping problem (terminating policies, discounted case)**

We have $H(x) = -c - \sum_{k=2}^{\infty} \left[\sum_{i=1}^{k-1} \gamma^i \cdot \alpha^i \cdot \|x\| + \gamma^k \cdot (-c) \right] = -\infty$ for any policy that takes infinite steps. and $H(x) = -c$ for a direct jumping policy. Therefore, the H-term drives to a terminating policy.

Table 2: The state and action spaces of six challenging MuJoCo tasks.

Tasks	Agent	Action Space	State Space
Swimmer-v3	Three-link swimming robot	2	8
Hopper-v3	Two-dimensional one-legged robot	3	11
Walker2d-v3	Two-dimensional bipedal robot	6	17
HalfCheetah-v3	Two-dimensional robot	6	17
Ant-v3	Four-legged creature	8	111
Humanoid-v3	Three-dimensional bipedal robot	17	376

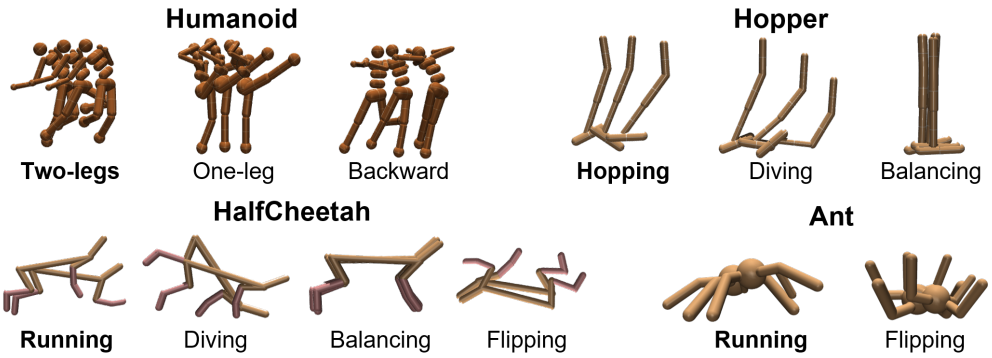


Figure 5: Different policies for MuJoCo tasks (Todorov et al., 2012). The bold ones are physically stationary policies.

C Experiments on MuJoCo Tasks

We selected six challenging robotic locomotion tasks from MuJoCo, namely, Swimmer-v3, Hopper-v3, Walker2D-v3, HalfCheetah-v3, Ant-v3, and Humanoid-v3, listed in Table 2.

C.1 MuJoCo tasks have multiple policies

Multiple Policies of MuJoCo Tasks: On four MuJoCo tasks (Todorov et al., 2012), namely, Humanoid, Hopper, HalfCheetah, and Ant, we were able to provide observational experiments. We render the obtained policies over multiple runs and then identify physically stationary ones. We observe various types of moving strategies, as shown in Fig. 5, which verifies that multiple policies are very common. For example, the Humanoid agent learns either jump with a single leg or run with two legs, as shown in Fig. 5 (top-left); another interesting example is HalfCheetah, in which an agent can run normally or in a flipped manner, as shown in Fig. 5 (bottom-left). Among the obtained policies, one can easily identify the physically stationary policies that control the robot moving forward with a *stable gait* (defined as gait that does not lead to fall).

In the supplementary files, we include rendered videos of different policies, listed in Table 3. Different policies are obtained over 20 runs of the PPO algorithm. We rendered these policies and classified them by physical gaits. The policies in bold texts are physically stationary.

C.2 Hyperparameters and Results

Fig. 6 shows the H-value (average over 20 runs) during the training process, which verified that the trained agents have converged to policies with small H-values. Fig. 6 compares the performance of $K = 8, 16, 24$. We run each experiment with 20 random seeds and for each run we test 100 episodes.

To verify the hypothesis that a smaller replay buffer hurts the performance, we rerun the trials of $K = 8, 16$ with a replay buffer size 800.

Table 3: List of video files for different policies.

Task	Different Policies	Video Name
Hopper	**hopping**	hopper_hopping.mp4
	diving	hopper_diving.mp4
	standing	hopper_standing.mp4
Ant	**running**	ant_running.mp4
	standing	ant_standing.mp4
	flipping	ant_flipping.mp4
Walker	**walking**	walker_walking.mp4
	diving	walker_diving.mp4
	standing	walker_standing.mp4
Humanoid	**two-legs**	humanoid_two_legs.mp4
	one-leg	humanoid_one_leg.mp4
	backward	humanoid_backward.mp4
HalfCheetah	**running**	halfcheetah_running.mp4
	diving	halfcheetah_diving.mp4
	flipping	halfcheetah_flipping.mp4
	standing	halfcheetah_standing.mp4
Swimmer	**moving**	swimmer_moving.mp4
	standing	swimmer_standing.mp4

Table 4: Hyperparameters used by PPO / PPO+H and DDPG / DDPG+H in MuJoCo tasks.

Parameters	PPO / PPO+H	DDPG / DDPG+H
Optimizer	Adam	Adam
Learning rate	$3 \cdot 10^{-4}$	$5 \cdot 10^{-4}$
Discount (γ)	0.99	0.995
GAE parameter	0.95	-
Replay buffer size	-	10^6
Target Update Rate (τ)	-	10^{-3}
Number of hidden layers for all networks	3	3
Number of hidden units per layer	256	256
Mini-batch size	32	64
Importance rate of H-term (λ)	2^{-3}	2^{-3}
Truncation step of H-term (K)	16	16

Table 5: Experimental results on six MuJoCo tasks.

Tasks	Policies	PPO		PPO+H ($K = 8$)		PPO+H ($K = 16$)		PPO+H ($K = 24$)	
HalfCheetah	**running**	**13**	4720.8 ±969.2	**19**	5028.4 ±211.3	**20**	5104.3 ±228.4	**20**	4995.1 ±383.3
	flipping	5		0		0		0	
	diving	1		1		0		0	
	balancing	1		0		0		0	
Ant	**running**	**17**	4164 ±1563.4	**20**	4505.3 ±253.6	**20**	4645.6 ±225.4	**20**	4662.5 ±277.5
	jumping	0		0		0		0	
	flipping	3		0		0		0	
Humanoid	**two-legs**	**7**	9433.4 ±2513.5	**17**	9670.3 ±497.2	**16**	10189.1 ±683.7	**17**	9942.2 ±538.4
	one-leg	12		3		4		3	
	backward	1		0		0		0	
Hopper	**hopping**	**10**	2659.3 ±905.3	**18**	3116.5 ±289.4	**20**	3300.1 ±184.2	**20**	3340.7 ±191.5
	diving	8		2		0		0	
	balancing	2		0		0		0	
Swimmer	**moving**	**14**	110.7 ±60.7	**20**	130.6 ±33.5	**20**	132.5 ±31.6	**19**	132.2 ±36.2
	balancing	6		0		0		1	
Walker	**walking**	**5**	5461.7 ±1290.1	**16**	5819.9 ±315.6	**16**	5927.2 ±296.8	**15**	6089.3 ±314.7
	diving	8		2		4		5	
	balancing	7		2		0		0	

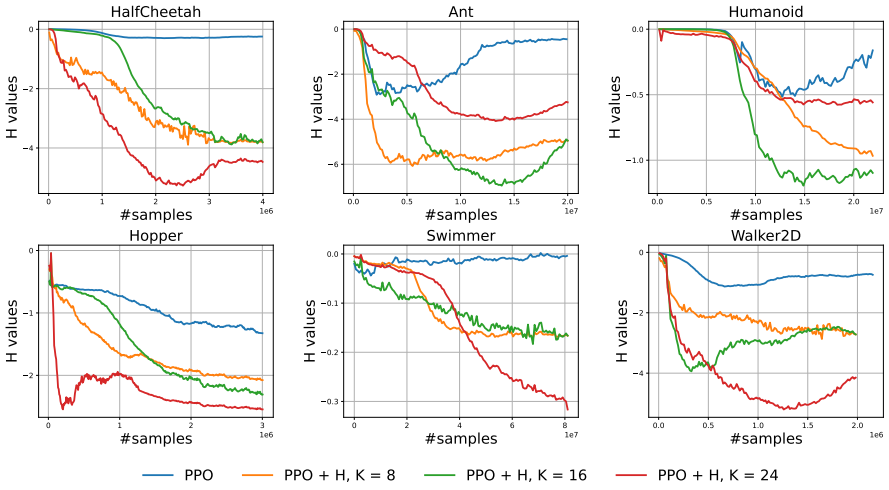


Figure 6: H values during the training process.

D GRAPH MAXCUT

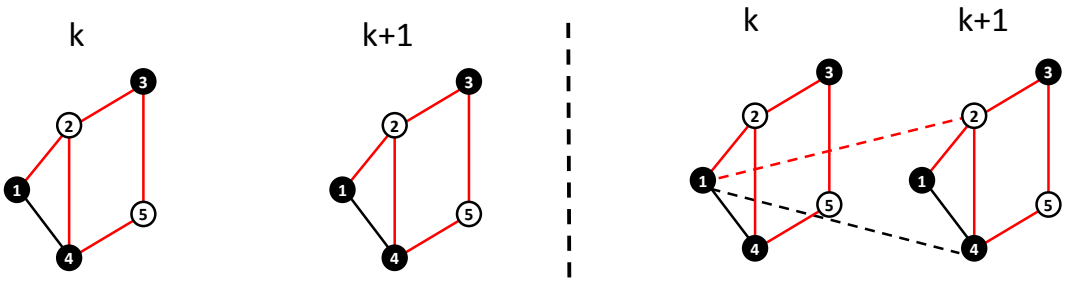


Figure 7: Illustration of graph maxcut using K-spin Ising model.

Table 6: Graph maxcut using K-spin Ising model.

Graph maxcut (original)		Graph maxcut using K-spin Ising model	
Nodes	$i \in V$	Nodes	$i_k \in V, k = 1, ..., K$
Solution	$\boldsymbol{x} \in [-1, +1]^N$	Solutions	$\boldsymbol{x}^1, \cdots, \boldsymbol{x}^K \in [-1, +1]^N$
Path	-	Path prob.	$\boldsymbol{x}^1_{i_1} \cdots \boldsymbol{x}^K_{i_K} \in [-1, +1]^K$
Reward	$J_{ij} = w(i, j)$	Path reward	$J_{i_1 \cdots i_K} = \sum_{k=2}^{K} \gamma^{K-k} w(i_1, i_2) \cdots w(i_{k-1}, i_k)$

We use the K-spin Ising model to solve the graph maxcut problem.

D.1 GRAPH MAXCUT USING K-SPIN ISING MODEL

We study the graph maxcut problem over weighted graphs. Let $G = (V, E, w)$ denote a weighted graph G with node set V and edge set E, i.e., $|V| = N$ and $|E| = M$, and edge weight $w : E \to \mathbb{R}_+$ such that

$$w(i, j) = \begin{cases} 1, & \text{if } (i, j) \in E, \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Graph maxcut using Ising model: for a solution $\boldsymbol{x} \in \{-1, +1\}^N$, the cut value can be expressed as the following Hamiltonian

$$H(\boldsymbol{x}) = \sum_{(i,j) \in E} w(i, j) \boldsymbol{x}_i \boldsymbol{x}_j = \sum_{i}^{N} \sum_{j>i}^{N} w(i, j) \boldsymbol{x}_i \boldsymbol{x}_j. \tag{12}$$

For the left graph of Fig. 7, we have $E = \{(1, 2), (1, 4), (2, 3), (2, 4), (3, 4), (3, 5)\}$ and $w(1, 2) = w(1, 4) = w(2, 3) = w(2, 4) = w(3, 5) = w(4, 5) = 1$. For the left graph in Fig. 7, the solution at the k-th step is $\boldsymbol{x}^k \in \{-1, +1\}^5$ and its Hamiltonian in (12) is

$$H(\boldsymbol{x}^k) = \boldsymbol{x}^k_1 \boldsymbol{x}^k_2 + \boldsymbol{x}^k_1 \boldsymbol{x}^k_4 + \boldsymbol{x}^k_2 \boldsymbol{x}^k_3 + \boldsymbol{x}^k_2 \boldsymbol{x}^k_4 + \boldsymbol{x}^k_3 \boldsymbol{x}^k_5 + \boldsymbol{x}^k_4 \boldsymbol{x}^k_5. \tag{13}$$

Graph maxcut using K-spin Ising model: Our solution takes K steps to solve the graph maxcut problem, i.e., $\boldsymbol{x}^1 \to \boldsymbol{x}^2 \cdots \to \boldsymbol{x}^K$. These K steps induce a new graph $G' = (V', E', w')$, as shown in the right graph in Fig. 7, with the following rules:

- V' consists of K replicas of V, i.e., $V^1, V^2, ..., V^K$. For node $i \in V$, we denote its K replicas as $i_1, i_2, ..., i_K$.
- The edge set E is kept for the k-th step, $E^k = E$. Each step of those K steps is a snapshot of $G = (V, E, w)$.
- We add new edges between the k-th and $(k + 1)$-th steps, i.e., $w'(i_k, j_{k+1}) = w(i, j)$, therefore

$$\begin{cases} (i_k, j_{k+1}) \in E', & \text{if } (i, j) \in E, \\ (i_k, j_{k+1}) \notin E', & \text{if } (i, j) \notin E, \end{cases} \qquad w'(i_k, j_{k+1}) = \begin{cases} 1, & \text{if } (i, j) \in E, \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

Table 7: Results for graph maxcut on synthetic instances

Nodes	Gurobi	RL-Hamiltonian	Improvement	Speedup
20	67 (5s)	71 (36s)	$+5.97\%$	$0.139\times$
30	132 (10s)	135 (93s)	$+2.27\%$	$0.108\times$
100	1408 (2000s)	1415 (33s)	$+0.49\%$	$60.6\times$
1000	128508 (4400s)	129714 (119s)	$+0.94\%$	$36.97\times$
5000	- ($>$ 8h)	3175813 (202s)	-	$-$

Table 8: Graph maxcut on the Gset dataset. The compared methods are BLS (Benlic & Hao, 2013), DSDP (Choi & Ye, 2000), KGLWG (Kochenberger et al., 2013), RUN-CSP (Toenshoff et al., 2021), PI-GNN (Schuetz et al., 2022).

Graph	Nodes	Edges	BLS	DSDP	KHLWG	RUN-CSP	PI-GNN	RL-Hamiltonian	Improvement
G14	800	4694	**3064**	-	2922	3061	2943	3003	-1.99%
G15	800	4661	**3050**	2938	**3050**	2928	2990	2965	-2.78%
G22	2000	19990	**13359**	12960	**13359**	13028	13181	12991	-2.75%
G49	3000	6000	**6000**	**6000**	**6000**	**6000**	5918	5790	-3.50%
G50	3000	6000	**5880**	**5880**	**5880**	**5880**	5820	5720	-2.72%
G55	5000	12468	**10294**	9960	10236	10116	10138	9830	-4.51%
G70	10000	9999	**9541**	9456	9458	-	9421	9091	-4.72%

For example, as shown in the right graph of Fig. 7, we have $(1_k, 2_{k+1}) \in E'$ and $(1_k, 4_{k+1}) \in E'$ since $(1, 2) \in E$ and $(1, 4) \in E$.

The Hamiltonian measure in (2) becomes:

$$H(\boldsymbol{x}^1, \cdots, \boldsymbol{x}^K) = \sum_{k=1}^{K} H(\boldsymbol{x}^k) + \sum_{k=1}^{K} \sum_{i_1 \in V^1} \cdots \sum_{i_k \in V^k} J_{i_1 \cdots i_k} \boldsymbol{x}_{i_1}^1 \cdots \boldsymbol{x}_{i_k}^k, \tag{15}$$

where $J_{i_1 \cdots i_k} = \sum_{k=2}^{K} \gamma^{K-k} w(i_1, i_2) \cdots w(i_{k-1}, i_k)$.

For the right graph in Fig. 7, the 1st and 2nd steps have

$$\begin{aligned} H(\boldsymbol{x}^1, \boldsymbol{x}^2) =& H(\boldsymbol{x}^1) + H(\boldsymbol{x}^2) + (\boldsymbol{x}_1^1 \boldsymbol{x}_2^2 + \boldsymbol{x}_1^1 \boldsymbol{x}_4^2 + \boldsymbol{x}_2^1 \boldsymbol{x}_1^2 + \boldsymbol{x}_2^1 \boldsymbol{x}_3^2 + \boldsymbol{x}_2^1 \boldsymbol{x}_4^2 \\ &+ \boldsymbol{x}_3^1 \boldsymbol{x}_2^2 + \boldsymbol{x}_3^1 \boldsymbol{x}_5^2 + \boldsymbol{x}_4^1 \boldsymbol{x}_2^2 + \boldsymbol{x}_4^1 \boldsymbol{x}_5^2 + \boldsymbol{x}_5^1 \boldsymbol{x}_3^2 + \boldsymbol{x}_5^1 \boldsymbol{x}_4^2). \end{aligned} \tag{16}$$

D.2 Experimental Results

Instead of using simple MLP networks, we use Long Short Term Memory (LSTM) to capture the dynamics over K iterations. One desirable property of LSTM is that we can leverage the momentum to overcome the existence of local minima, which is helpful in non-convex optimization. However, one challenge in solving combinatorial optimization problems is scalability due to the exponential complexity. To avoid this difficulty, we optimize coordinatewise on the dimensions of the solution and show that we can significantly scale each problem up, e.g., a graph with 10,000 nodes.

Gurobi Gurobi Optimization, LLC (2023) is a state-of-the-art mathematical programming solver that can be used to solve optimization problems. It is one of the most popular commercial optimization solvers used in both academia and industry, known for its speed, efficiency, and reliability. Therefore, we use it as an important comparison method.

Table 7 shows the result on synthetic data with $N = 20, 30, 100, 1000, 5000$. For the instance with 5000 nodes, Gurobi cannot return a solution within 8 hours. The RL-Hamiltonian method finds better (or same) solutions than the Gurobi for all test instances. When $N \geq 100$, our RL-Hamiltonian method has a speedup $36.97\times \sim 60.6\times$.

Table 8 presents results of RL-Hamiltonian and 6 compared methods for seven instances from Gset. The compared methods include Gurobi (Gurobi Optimization, LLC, 2023), SDP (DSDP) (Choi & Ye, 2000), breakout local search (BLS) (Benlic & Hao, 2013), Tabu search (KHLWG)

(Kochenberger et al., 2013), recurrent GNN (RUN-CSP) (Toenshoff et al., 2021), and physical-inspired GNN (PI-GNN) (Schuetz et al., 2022). We evaluate the solution via the metric of relative gap $= (H(\boldsymbol{x}) - H(\boldsymbol{x}^*)/H(\boldsymbol{x}^*)$. Compared to the best-known solution, our RL-Hamiltonian method has a gap from -1.99% to -4.72%.

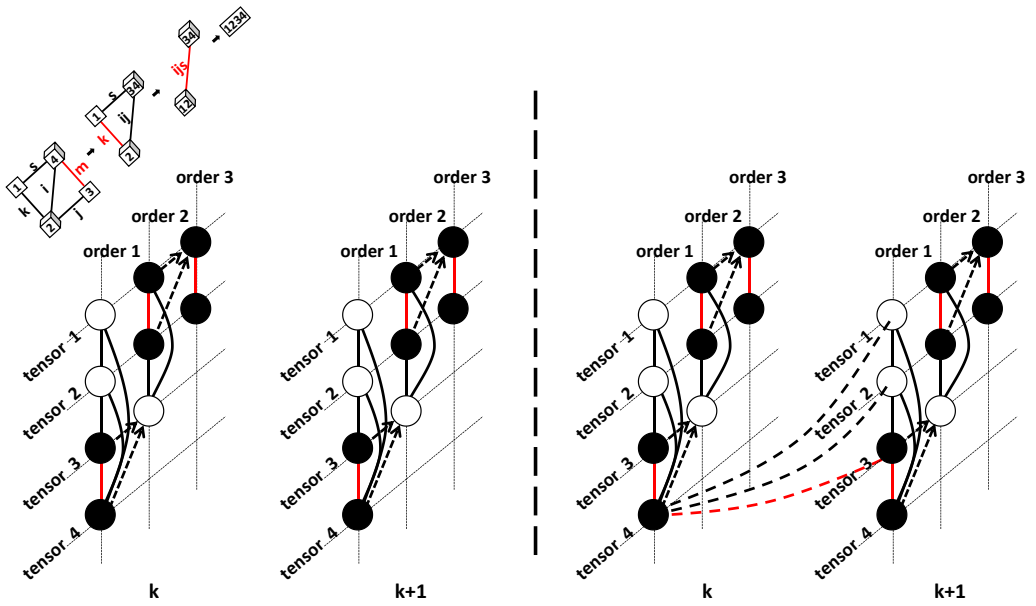


Figure 8: Illustration of TNCO using K-spin Ising model.

E TENSOR NETWORK CONTRACTION ORDERING

We use the K-spin Ising model to solve the tensor network contraction ordering (TNCO) problem.

Table 9: TNCO using K-spin Ising model.

TNCO (original)		TNCO using K-spin Ising model	
Spins	$\boldsymbol{x}_{u,j} \in V$	Spins	$\boldsymbol{x}_{u_k,j_k} \in V, k = 1, \cdots, K$
Solution	$\boldsymbol{x} \in \{0,1\}^{N^2}$	Solutions	$\boldsymbol{x}^1, \cdots, \boldsymbol{x}^K \in \{0,1\}^{N^2}$
Path	-	Path prob.	$\boldsymbol{x}^1_{i_1}, \cdots, \boldsymbol{x}^K_{i_K} \in \{0,1\}^K$
Reward	$J_{u_i,v_j} = w(u_i, v_j)$	Path reward	$J_{i_1,\cdots,i_K} = \sum_{k=2}^{K} \gamma^{K-k} w(i_1, i_2) \cdots w(i_{k-1}, i_k)$

E.1 PROBLEM FORMULATION

Given a tensor network, $G = (V, E, w)$, a contraction path $P = (e_1, \ldots, e_{n-1}), e_t \in E_t$, and a corresponding sequence of graphs $(G_1, \ldots G_{n-1})$. The goal is to find a path P with minimum cost,

$$\begin{aligned} P^*(G) &= \operatorname{argmin}_P c(P) \\ \text{s.t. } & P = (e_1, \ldots, e_{n-1}), e_t \in E_t, \end{aligned} \tag{17}$$

where the cost $c(P)$ is the sum of rewards along the contracted path P

$$c(P) = \sum_{t=1}^{n-1} w_t(e_t). \tag{18}$$

E.2 OUR SOLUTION

In the left of Fig. 8, we construct a new graph $G = (V, E, w)$ with $G_1, G_2, \cdots, G_{n-1}$ using the following rules:

- V consist of $V_1, V_2, \cdots, V_{n-1}$, node $u \in V_i$ are denoted with $u_i \in V$ in V.
- E consist of $E_1, E_2, \cdots, E_{n-1}$
- $$\begin{cases} (u_i, v_j) \in E, & \text{if } i = j \text{ and } (u, v) \in E_i, \\ (u_i, v_j) \notin E, & \text{otherwise,} \end{cases} \qquad w(u_i, v_j) = \begin{cases} w_i(u, v), & \text{if } i = j \text{ and } (u, v) \in E_i, \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

TNCO using Ising model: The Ising model of TNCO problem uses $N(N-1)$ spins $\boldsymbol{x}_{u,j}$, where u denotes the tensor and j denotes its order in the TNCO path. The energy of the original TNCO problem has three terms. The first term requires each tensor to appear at least once in the TNCO path. The second term requires there are exactly two tensors selected at order j along a path. The third term measures the contraction cost at order j. These are encoded in the following Hamiltonian:

$$H(\boldsymbol{x}) = \sum_{i=1}^{N-1}(2 - \sum_{u=1}^{N-i}\boldsymbol{x}_{u,i})^2 + \sum_{i=1}^{N-1}\sum_{u=1}^{N}\sum_{v=1}^{N} J_{u_i,v_i}\boldsymbol{x}_{u,i}\boldsymbol{x}_{v,i}. \tag{20}$$

$H(x) = (2 - \boldsymbol{x}_{1,1} - \boldsymbol{x}_{2,1} - \boldsymbol{x}_{3,1} - \boldsymbol{x}_{4,1})^2 + (2 - \boldsymbol{x}_{1,2} - \boldsymbol{x}_{2,2} - \boldsymbol{x}_{3,2})^2 + (2 - \boldsymbol{x}_{1,3} - \boldsymbol{x}_{2,3})^2 + w_1(1,4)\boldsymbol{x}_{1,1}\boldsymbol{x}_{4,1} + w_1(1,2)\boldsymbol{x}_{1,1}\boldsymbol{x}_{2,1} + w_1(2,3)\boldsymbol{x}_{2,1}\boldsymbol{x}_{3,1} + w_1(2,4)\boldsymbol{x}_{2,1}\boldsymbol{x}_{4,1} + w_1(3,4)\boldsymbol{x}_{3,1}\boldsymbol{x}_{4,1} + w_2(1,3)\boldsymbol{x}_{1,2}\boldsymbol{x}_{3,2} + w_2(1,2)\boldsymbol{x}_{1,2}\boldsymbol{x}_{2,2} + w_2(2,3)\boldsymbol{x}_{2,2}\boldsymbol{x}_{3,2} + w_3(1,2)\boldsymbol{x}_{1,3}\boldsymbol{x}_{2,3}$

TNCO using K-spin Ising model: Our solution takes K steps to solve the TNCO problem, i.e., $\boldsymbol{x}^1 \to \boldsymbol{x}^2 \to \cdots \to \boldsymbol{x}^K$. These K steps induce a new graph $G' = (V', E', w')$, as shown in the right graph in Fig. 7, with the following rules:

- V' consists of K replicas of V, i.e., $V^1, V^2, ..., V^K$. For node $u_i \in V$, we denote its K replicas as $u_i^1, u_i^2, ..., u_i^K$.
- The edge set E is kept for the k-th step, $E^k = E$. Each step of those K steps is a snapshot of $G = (V, E, w)$.
- We add new edges between the k-th and $(k+1)$-th steps, i.e., $w'(u_i^k, v_j^{k+1}) = w(u_i, v_j)$, therefore

$$\begin{cases}(u_i^k, v_j^{k+1}) \in E', & \text{if } (u_i, v_j) \in E,\\ (u_i^k, v_j^{k+1}) \notin E', & \text{if } (u_i, v_j) \notin E,\end{cases} \qquad w'(u_i^k, v_j^{k+1}) = \begin{cases}w(u_i^k, v_j^{k+1}), & \text{if } (u_i, v_j) \in E,\\ 0, & \text{otherwise.}\end{cases} \tag{21}$$

$$H(\boldsymbol{x}^1, \cdots, \boldsymbol{x}^K) = \sum_{k=1}^{K} H(\boldsymbol{x}^k) + \sum_{k=1}^{K}\sum_{i_1 \in V^1}\cdots\sum_{i_k \in V^k} J_{i_1\cdots i_k}\boldsymbol{x}_{i_1}^1 \cdots \boldsymbol{x}_{i_k}^k, \tag{22}$$

where $J_{i_1\cdots i_k} = \sum_{k=2}^{K}\gamma^{K-k}w(i_1,i_2)w(i_2,i_3)\cdots w(i_{k-1},i_k)$.

E.3 Experimental Results

Table 10: Total flop count in tensor-train network of various sizes. The compared methods are OE Greedy (Daniel et al., 2018), CTG Greedy (Gray & Kourtis, 2021), and CTG Kahypar (Gray & Kourtis, 2021).

Size	OE Greedy	CTG Greedy	CTG Kahypar	RL-Hamiltionian
N=100	30.927	30.705	30.710	**30.404**
N=200	61.030	60.808	60.810	**60.507**
N=400	121.236	121.014	121.010	**120.713**
N=600	181.442	181.220	181.220	**180.919**
N=800	241.648	241.426	241.430	**241.125**
N=1000	301.854	301.632	301.630	**301.331**
N=1500	-	-	452.150	**451.846**
N=2000	-	-	602.660	**602.361**

Table 11: Total flop count in Sycamore circuit of various cycles. The compared methods are OE Greedy (Daniel et al., 2018), CTG Greedy (Gray & Kourtis, 2021), CTG Kahypar (Gray & Kourtis, 2021), AC-QDP (Huang et al., 2021), and RL-TNCO (Meirom et al., 2022).

Cycles	OE Greedy	CTG Greedy	CTG Kahypar	AC-QDP	RL-TNCO	RL-Hamiltionian
m=12	17.795	17.065	13.408	13.037	12.869	**12.12**
m=14	19.679	19.282	14.152	**13.85**	14.411	14.731
m=16	25.889	23.151	17.012	17.06	-	**15.967**
m=18	26.793	23.570	17.684	17.41	-	**15.777**
m=20	26.491	25.623	18.826	18.82	18.544	**16.978**

The float counts in Table 10 and Table 11 are measured by taking $\log_{10}$.

Table 10 shows the contraction cost for the tensor-train network with nodes $N = 100, 200, ..., 20,000$. Our RL-Hamiltonian method achieves a reduction of 0.3 order over the state-of-the-art method CTG Kahypar (Gray & Kourtis, 2021).

Table 11 shows the contraction cost for the Sycamore network with 53 qubits and cycles $m = 12, 14, 16, 18, 20$. Our RL-Hamiltonian method achieves a reduction of orders 1.045, 1.907, and 1.748 for $m = 16, 18, 20$, over the state-of-the-art method CTG Kahypar (Gray & Kourtis, 2021).

F MASSIVE MIMO BEAMFORMING IN WIRELESS COMMUNICATION

We use the K-spin Ising model to solve the multiple input multiple output (MIMO) beamforming problem.

F.1 PROBLEM FORMULATION

The massive MIMO beamforming problem in 5G/6G wireless communication systems is a nonconvex optimization problem, which is NP-hard (Björnson et al., 2013).

Consider a base station with N antennas with a power budget $P \in \mathbb{R}$ and M single-antenna users, the channel matrix is a Gaussian matrix $\boldsymbol{S} = [\boldsymbol{s}_1, \cdots, \boldsymbol{s}_M] \in \mathbb{C}^{N \times M}$, where each entry is independent and identically distributed (i.i.d.), i.e., $\boldsymbol{S}_{ij} \sim \mathcal{CN}(0, \sigma^2)$. The goal is to find an optimal beamformer matrix $\boldsymbol{W} = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_M] \in \mathbb{C}^{N \times M}$ that maximizes the sum of *achievable rates* as follows:

$$\begin{aligned} \boldsymbol{W}^* = \underset{\boldsymbol{W} \in \mathbb{C}^{N \times M}}{\arg\max} \quad & \sum_{m=1}^{M} \log_2 \left(1 + \mathrm{SINR}_m\right), \\ \text{s.t.} \quad & \mathrm{Tr}\left(\boldsymbol{W}\boldsymbol{W}^H\right) \leq P, \end{aligned} \tag{23}$$

where $\log_2 \left(1 + \mathrm{SINR}_m\right)$ is the achievable rate for user m, and the signal-to-interference-plus-noise ratio (SINR) is given by

$$\mathrm{SINR}_m = \frac{\left|\boldsymbol{s}_m^H \boldsymbol{w}_m\right|^2}{\sum_{i=1, i \neq m}^{M} \left|\boldsymbol{s}_m^H \boldsymbol{w}_i\right|^2 + \sigma^2}. \tag{24}$$

F.2 OUR SOLUTION

For a random channel sample $\boldsymbol{S} \in \mathbb{C}^{N \times M}$, our policy network maps an input $(\boldsymbol{S}, \boldsymbol{W}^k)$ to an output $\boldsymbol{W}^{k+1} \in \mathbb{C}^{N \times M}$, for $k = 0, 1, 2, \cdots, K$, where $\boldsymbol{W}^0 \in \mathbb{C}^{N \times M}$ is a random (complex) matrix. The input of the k-th iteration is $(\boldsymbol{S}, \boldsymbol{W}^k)$, while the output of the k-th iteration is $\boldsymbol{W}^{k+1}$. K iterations are $(\boldsymbol{S}, \boldsymbol{W}^0) \to (\boldsymbol{S}, \boldsymbol{W}^1) \to (\boldsymbol{S}, \boldsymbol{W}^2) \to \cdots \to (\boldsymbol{S}, \boldsymbol{W}^K)$. The final output is $\widehat{\boldsymbol{W}}^* = \boldsymbol{W}^K$.

We study the MIMO beamforming problem over weighted graphs. Let $G = (V, E, w)$ denote a weighted graph G with node set $V = \{0, 1, \cdots M\}$ and edge set $E = \{(0, i) | i = 1, 2 \cdots, M\}$, i.e., $|V| = M + 1$ and $|E| = M$, node 0 denotes the base station, and edge weight $w : E \to \mathbb{R}_+$ such that

$$w(i, j) = \begin{cases} \mathrm{SINR}_j, & \text{if } i = 0, j > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{25}$$

$$H(\boldsymbol{x}) = -\sum_{i=1}^{M} \mathrm{SINR}_i(\boldsymbol{S}, \boldsymbol{W}) \boldsymbol{x}_0 \boldsymbol{x}_i \tag{26}$$

$$H(\boldsymbol{x}^1, \cdots, \boldsymbol{x}^K) = \sum_{k=1}^{K} H(\boldsymbol{x}^k) + \tag{27}$$

Our objective function (2) for this MIMO beamforming problem becomes

$$\mathbb{E}_{\boldsymbol{S} \sim \mathcal{CN}(0, \sigma^2)} \left[\sum_{k=1}^{K} \sum_{m=1}^{M} \log_2 \left(1 + \mathrm{SINR}_m\right) \right]. \tag{28}$$

The Hamiltonian's deterministic policy gradient of (28) corresponding to (4) becomes:

$$\nabla_\theta H'(\theta) = -\mathbb{E}_{\boldsymbol{S} \sim \mathcal{CN}(0, \sigma^2)} \left[\sum_{k=0}^{K-1} \gamma^k \cdot \sum_{m=1}^{M} \log_2 \left(1 + \mathrm{SINR}_m\right) \cdot \nabla_\theta \log \left(\widetilde{\pi}_{\theta, \delta}(\mu_0) \cdot \widetilde{\pi}_{\theta, \delta}(\mu_1) \cdots \widetilde{\pi}_{\theta, \delta}(\mu_k) \right) \right]. \tag{29}$$

To solve those optimization problems using deep learning, we have three major modifications compared to the original method.

Table 12: Performance comparison at different SNRs.

N	K	SNR (dB)	MMSE	BRB Björnson et al. (2013)	SLSQP	RL-Hamiltonian
4	4	10	8.31 (1.02ms)	9.8 (1000s)	9.76 (1.01s)	**9.8** (9.5ms)
4	4	20	17.03 (1.03ms)	19.41 (1000s)	19.23 (1.51s)	**19.26** (9.8 ms)
8	8	10	15.45 (1.50ms)	18.2 (2h)	17.95 (2.52s)	**18.01** (14ms)
8	8	20	31.13 (1.54)	34.4 (2h)	**36.26** (2.58s)	35.3 (15ms)
16	16	10	31.25 (2.01ms)	- (>24h)	35.57 (4.72s)	34.91(530ms)
16	16	20	36.54 (2.01ms)	- (>24h)	70.97 (4.86s)	69.41 (518ms)
32	32	10	60.58 (3.10ms)	- (>24h)	67.79 (58.7s)	66.79 (570ms)
32	32	20	111.71 (3.11ms)	- (>24h)	133.96 (61.4Ds)	127.69 (580ms)

State and action space Given that it is nontrivial to formulate combinatorial optimization problems as an MDP, we directly optimize the K-spin objective rather than relying on existing RL algorithms. Specifically, we have a neural network that takes the problem state S^t as input, outputs the solution W^t, and trains it to maximize the objective over K iterations. Note that the state S^t will include both the problem observation O^t and the last solution W^{t-1}.

Curriculum learning(Bengio et al., 2009) To overcome the local minima, we further propose to use curriculum learning as a warm-up. For example, in MIMO, we first train the neural network by substituting the non-convex objective with a convex one minimum mean square error (MMSE), which lowers the difficulty. After certain training steps, we substitute back the original objective.

F.3 Experimental Results

Experimental Results: Table 12 shows the sum rate for the MIMO beamforming task with $N = K = 4, 8$, and SNR=10 dB, 20 dB. For the first three cases, our RL-Hamiltonian method outperforms the compared methods. In other cases, our RL-Hamiltonian maintains an ignorable gap with the best solution while obtaining an $8.9 \sim 172$ times speedup compared to the state-of-the-art solver SLSQP.