STATIONARY DEEP REINFORCEMENT LEARNING WITH QUANTUM K-SPIN HAMILTONIAN REGULARIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Instability is a major issue of deep reinforcement learning (DRL) algorithms — high variance of performance over multiple runs. It is mainly caused by the existence of *many local minimas* and worsened by the *multiple fixed points* issue of Bellman's equation. As a fix, we propose a quantum K-spin Hamiltonian regularization term (called *H-term*) to help a policy network converge to a high-quality local minimum. First, we take a quantum perspective by modeling a policy as a *K-spin Ising model* and employ a Hamiltonian to measure the *energy* of a policy. Then, we derive a novel Hamiltonian policy gradient theorem and design a generic actor-critic algorithm that utilizes the H-term to regularize the policy network. Finally, the proposed method reduces the variance of cumulative rewards by $65.2\% \sim 85.6\%$ on six MuJoCo tasks, compared with existing algorithms over 20 runs.

1 INTRODUCTION

Instability is a major issue of deep reinforcement learning (DRL) (Sutton & Barto, 2018) algorithms — 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., 2018, Zoph & Le, 2016) empirically reported a high variance over multiple runs. Hence, in practice, it requires training tens of agents and picking the best one. Such a high variance largely contributes to the RL community's dispute of reliability and reproducibility (Dulac-Arnold et al., 2019; 2020), limiting the wider adoption in real-world tasks. The instability issue is mainly caused by the existence of *many local minimas* 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. A, we adapt dynamic programming examples (Bertsekas, 2019, Piunovskiy, 2013) into reinforcement learning settings and provide detailed explanations on the aforementioned issues.

The instability problem 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 that significantly reduces the variance of the advantage function. However, existing methods randomly converge to different local minima. For practical usage, we often expect a DRL algorithm stably converges to a certain policy independent of initialization and noises.

As a fix, we propose a quantum K-spin Hamiltonian regularization term (*H-term*) to help a policy network converge to a high-quality local minimum. We take a novel quantum perspective by modeling a policy as a *K-spin Ising model* (Kirkpatrick et al., 1983, Denchev et al., 2016) and employ a Hamiltonian equation to measure the *energy* of a policy, namely an *H-term*. We hypothesize that *a stationary policy would have low energy*.

Related Works Different from our quantum perspective, several recent papers utilized the (classical) Hamiltonian equation to endow RL agents with the capability of inductive biases. For example, (Greydanus et al., 2019, Toth et al., 2019) used Hamiltonian mechanics to train an agent that learns and respects *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 policy as a quantum K-spin Ising model.

Policy in (3)		Quantum K-spin Ising model in (1)	
State-action	$\mu_k \in \mathcal{S} \times \mathcal{A}$, $k = 0, ..., K$	Spins	$j_k \in \{1, \cdots, N\}$, $k = 0, ..., K$
Policy	$\pi_{\mu_0} \times \cdots \times \pi_{\mu_{K-1}} \in [0,1]^K$	Configuration	$\sigma_{j_0} \times \cdots \times \sigma_{j_{K-1}} \in [-1,+1]^K$
Optimal policy	$\pi^*_{\mu_0} \times \cdots \times \pi^*_{\mu_{K-1}} \in \{0,1\}^K$	Optimal configuration	$\sigma^*_{j_0} \times \cdots \times \sigma^*_{j_{K-1}} \in \{-1,+1\}^K$
Discounted reward	$L_{\mu_0 \ldots \mu_{K-1}}$	Density function	$L_{j_0 \cdots j_{K-1}}$
Energy of policy	$H(\pi_{\mu_0}, ..., \pi_{\mu_{K-1}})$	Energy	$H(\sigma_{j_0}, \cdots, \sigma_{j_{K-1}})$

2 Modeling Policy as K-spin Ising Model

We take a novel quantum perspective by modeling a policy as a *K-spin Ising model* and employ a Hamiltonian to measure the *energy* of a policy, namely an *H-term*. Our modeling policy as a quantum K-spin Ising model is inspired by the **simulated annealing** algorithms and the analogy in Table 1.

2.1 Quantifying Energy of a Policy via K-spin Hamiltonian Equation

The Hamiltonian of a quantum K-spin Ising model (Kirkpatrick et al., 1983, Denchev et al., 2016) measures the energy of a particular configuration, which takes the following form

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

where N is the number of spins in the k-th configuration, $\sigma_{j_k} = \pm 1$ are spin variables, and $L_{j_0 \cdots j_k}$ is an energy density function for k nearest spins' configuration $(\sigma_{j_0}, \ldots, \sigma_{j_k})$.

Analogy in Table 1. A state-action pair $\mu_k = (S_k, A_k)$ corresponds to a spin j_k, we map an optimal policy $\pi^*(\mu_k) \in \{0,1\}$ to the optimal single-qubit spin operator $\sigma^*_{j_k} \in \{-1,1\}$ via $\pi^*(\mu_k) \longleftrightarrow (\mathbb{1}_{\mu_k} - \sigma^*_{\mu_k})/2$, where $\pi(\mu_k)$ denotes the probability of taking action A_k at state S_k. The energy density function $L_{j_0 \ldots j_k}$ corresponds to the discounted reward on a path $(\mu_0, \cdots, \mu_{k-1})$ of length k,

$$L_{\mu_0, \ldots, \mu_k} = \gamma^k \cdot R(\mu_k) \cdot d_0(s_0) \cdot \prod_{\ell=0}^{k-1} \mathbb{P}(s_{\ell+1} | \mu_\ell), \text{(obtained via Monte Carlo simulation)} \tag{2}$$

where $d_0(s_0)$ denotes the distribution of initial state s_0.

We express the objective of reinforcement learning into a K-spin Hamiltonian (given in Appx. D)

$$H(\theta) \triangleq -\mathbb{E}_{S_0, A_0}\left[Q^{\pi_\theta}(S_0, A_0)\right] = -\lim_{K \to \infty} \sum_{k=0}^{K-1} \sum_{\mu_0}^{\mathcal{S} \times \mathcal{A}} \cdots \sum_{\mu_k}^{\mathcal{S} \times \mathcal{A}} L_{\mu_0, \ldots, \mu_k} \pi_\theta(\mu_0) \cdots \pi_\theta(\mu_k), \tag{3}$$

where expectation is taken over $S_0 \sim d_0(\cdot)$, $A_0 \sim \pi_\theta(S_0, \cdot)$, and $L_{\mu_0, \ldots, \mu_k}$ is given in (2).

3 Stationary Deep Reinforcement Learning

3.1 Hamiltonian Policy Gradient and Monte Carlo-based Gradient Estimator

Theorem 1. *The Hamiltonian gradient of (3) is*

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

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) employs an estimate of future rewards, while Hamiltonian policy gradient uses trajectories in replay buffer.

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 a 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 $O(BT + B'K(K+1)/2)$. We provide the pseudocode in Alg. 1.

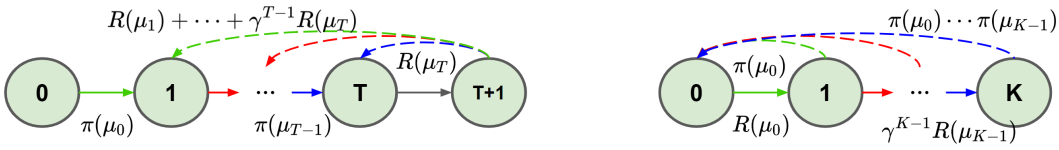


Figure 1: REINFORCE's policy gradient (left) vs. Hamiltonian's policy gradient (right).

3.2 Stationary Actor-Critic Algorithm with H-term

We propose a novel H-term for actor-critic algorithms. Similar to the entropy term in (Haarnoja et al., 2018), the proposed H-term is an add-on term to regularize the actor network and help it converge to a stationary policy. Specifically, the objective functions of actor and critic networks become:

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

where a target Q-value is $y(S_k, A_k) = R(S_k, A_k) + \gamma Q_\phi(S_{k+1}, A_{k+1})$, and $\lambda > 0$ is a temperature parameter. As an interpretation, the second term $-\lambda H(\theta)$ in the maximization objective function of actor network aims to find a minimum energy configuration for the MDP problem.

In Alg. 1, an agent interacts with an environment and alternatively updates its actor network and critic network. The algorithm has M episodes and each episode consists of a (Monte Carlo) simulation process and a learning process (gradient estimation) as follows:

- During the (Monte Carlo) simulation process (lines 5-10 of Alg. 1), an agent takes action a_t according to a policy $\pi_\theta(\cdot|s_t)$, $t = 0, \cdots, T-1$, generating a trajectory of T steps/transitions. Then, these T transitions are stored into a replay buffer $\mathcal{D}_1$, while the full trajectory $\tau = (s_0, a_0, r_0, s_1, \cdots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$ is stored in replay buffer $\mathcal{D}_2$.
- During the learning process ($G \geq 1$ updates in one episode) (lines 11-16 of Alg. 1), a mini-batch of B transitions $\{(s_i, a_i, r_i, s_{i+1})\}_{i=1}^{B}$ and a mini-batch of B' trajectories (of length K) $\{\tau_j = (s_0^j, a_0^j, r_0^j, s_1^j, \cdots, s_{K-1}^j, a_{K-1}^j, r_{K-1}^j, s_K^j)\}_{j=1}^{B'}$ are sampled from $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively. The critic network is updated by a conventional method, e.g., minimizing the mean squared error (MSE) between an estimated Q-value and a target value. The actor is updated by a Monte Carlo gradient estimator over B transitions and B' trajectories.

4 Performance Evaluation

We consider six MuJoCo tasks (Todorov et al., 2012). These tasks have high-dimensional continuous state space and action space, in which there exist multiple locally optimal policies as revealed in Section B.1. 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. J) the same and make sure that the obtained results reproduce existing benchmark tests (Duan et al., 2016).

H-term converges 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 reduces 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 $65.2\% \sim 85.6\%$,

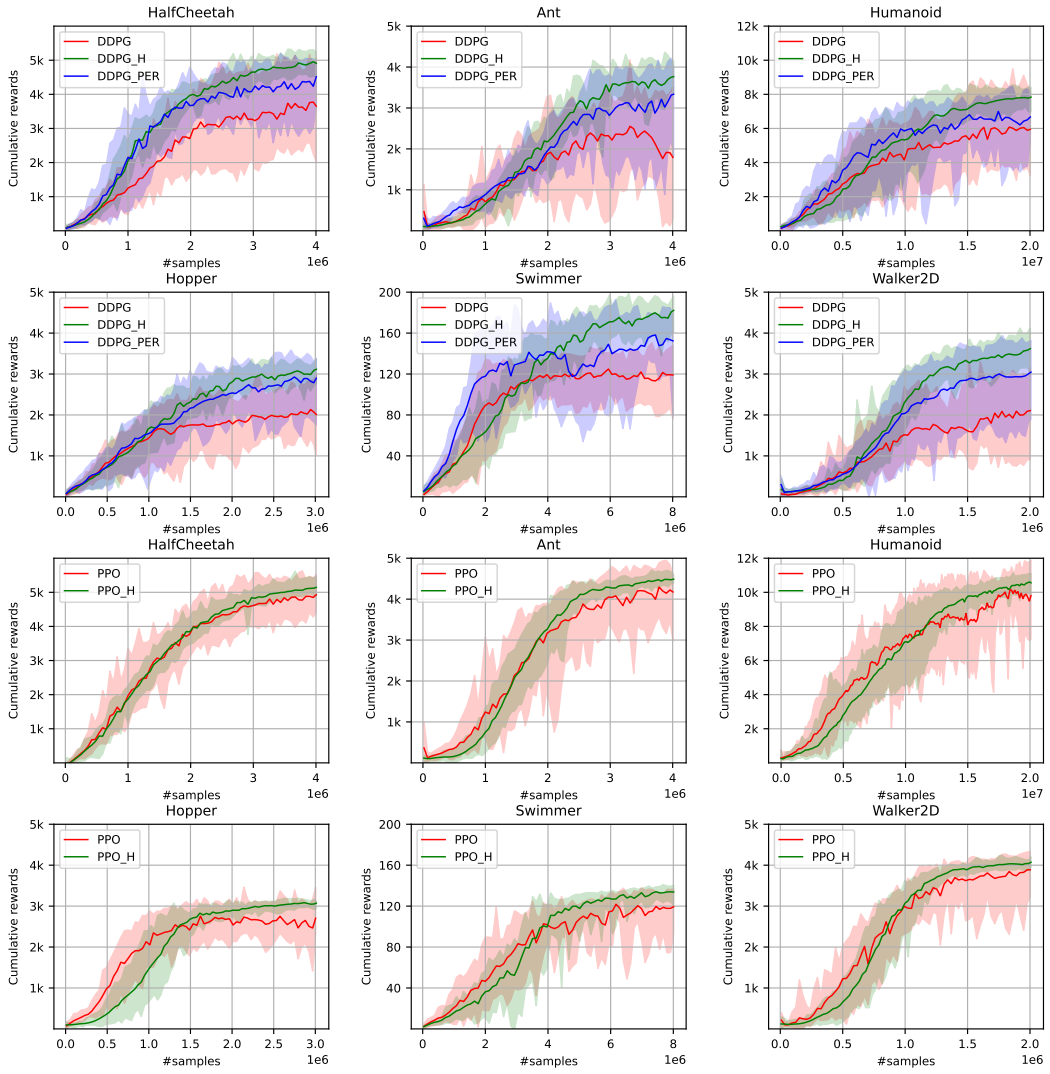


Figure 2: Cumulative rewards vs. #samples for compared DRL algorithms on six MuJoCo tasks.

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 drives to physically stationary policy. A key question needs to be answered: *is H-term really guiding the agent converge to a physically stationary policy?* Similar to Section B.1, 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 can converge 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 7, 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. J.2 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.

Peter Auer, Mark Herbster, and Manfred KK Warmuth. Exponentially many local minima for single neurons. *Advances in Neural Information Processing Systems*, 8, 1995.

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.

Hanjun Dai, Elias Khalil, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *Advances in Neural Information Processing Systems*, 30, 2017.

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 preprint arXiv:2003.11881*, 2020.

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

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.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870. PMLR, 2018.

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.

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

Wouter Kool, Herke Van Hoof, and Max Welling. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.

Ke Li and Jitendra Malik. Learning to optimize. *arXiv preprint arXiv:1606.01885*, 2016.

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.

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.

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.

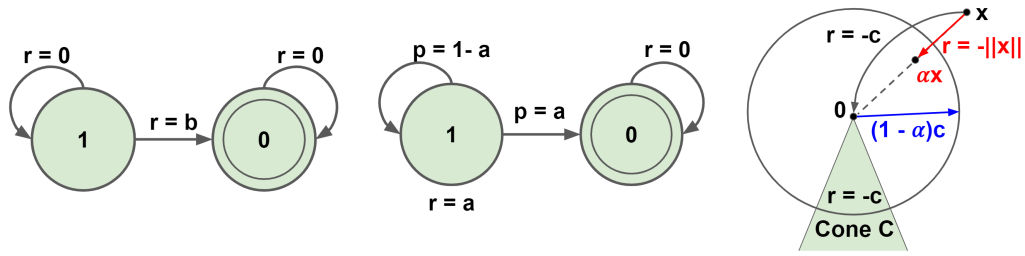
Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

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. *arXiv preprint arXiv:1611.01578*, 2016.



(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. 3 of Appx. A.

A More Examples with 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.

A.1 Cases of $\gamma = 1$

Consider MDP examples with an terminal state 0, as shown in Fig. 3 (we adapt dynamic programming examples citepbertsekas2019reinforcement, piunovskiy2013examples 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.

A.2 Cases of $\gamma \in (0, 1)$

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

$$\begin{aligned}
&V_1(1) = c + \gamma \cdot V_0(1) \geq c + \gamma \cdot b > b \\
&V_2(1) = c + \gamma \cdot c + \gamma^2 \cdot V_0(1) \geq (1 + \gamma)c + \gamma^2 b > b \\
&V_3(1) = c + \gamma \cdot c + \gamma^2 c + \gamma^3 \cdot V_0(1) \geq (1 + \gamma + \gamma^2)c + \gamma^3 b > b \\
&\cdots \\
&V_k(1) = \sum_{i=0}^{k-1} \gamma^i \cdot c + \gamma^k \cdot V_0(1) \geq \sum_{i=0}^{k-1} \gamma^i \cdot c + \gamma^k b > b \\
&\cdots \\
&V^*(1) = \sum_{i=0}^{\infty} \gamma^i \cdot c = \frac{1}{1 - \gamma} c > b
\end{aligned} \tag{7}$$

The values of $H(0)$ and $H(1)$ are as follows:

$$H(0) = 0, \quad H(1) = -b - \sum_{k=2}^{\infty} (\sum_{i=1}^{k-1} \gamma^{i-1} \cdot c + \gamma^k b) = -\infty. \tag{8}$$

Adding the above H-values to state 1 and 0, respectively, we have

$$\begin{aligned}
&V^*(1) + H(1) = \sum_{i=0}^{\infty} \gamma^i \cdot c - \infty = -\infty \\
&V^*(0) + H(0) = b.
\end{aligned} \tag{9}$$

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).

The values of $H(0)$ and $H(1)$ are as follows:

$$H(0) = 0, \quad H(1) = -\sum_{k=1}^{\infty} \sum_{i=1}^{k-1} \gamma^{i-1} \cdot a = -\infty. \tag{10}$$

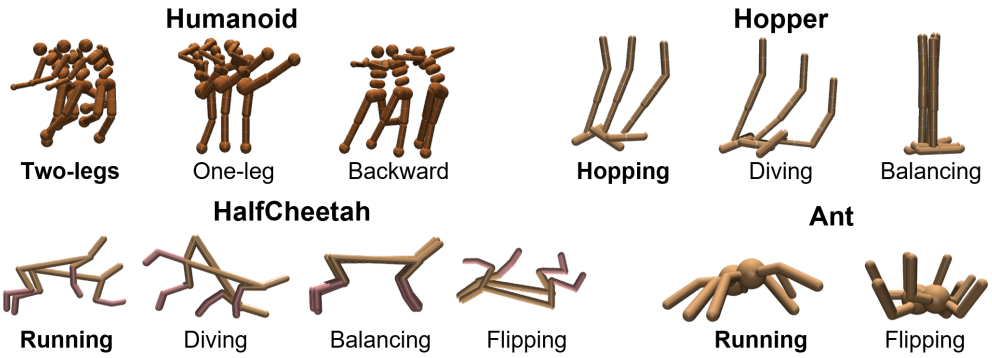


Figure 4: Different policies for MuJoCo tasks Todorov et al. (2012). The bold ones are physically stationary policies.

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)**

Any policy that takes infinite steps will have

$$H(x) = -c - \sum_{k=2}^{\infty} \left[\sum_{i=1}^{k-1} \gamma^i \cdot \alpha^i \cdot \|x\| + \gamma^k \cdot (-c) \right] = -\infty \tag{11}$$

and a direct jumping policy will have $H(x) = -c$. Therefore, the H-term drives to a terminating policy.

B The Problem of Many Local Minimas

First, we show the existence of many local minimas in many tasks. Then, we provide observational experiments to empirically verify the existence of multiple policies.

B.1 Existence of Many Local Minimas

We point out that combinatorial optimization and non-convex optimization have many local minima.

- MuJoCo tasks Todorov et al. (2012): agents randomly converge to policies of different gaits, as shown in Fig. 4.
- Travelling salesman problem (TSP) Kool et al. (2018): a case of 8 cities has 2 local minimas (in Appx. L).
- Graph max-cut Dai et al. (2017): an example graph of 20 nodes has 390 local minimas (in Appx. L).
- MIMO beamforming Björnson et al. (2013): a case of 2 users and 2 antennas has 3 local minimas (in Appx. **??**).
- Non-convex deep learning classifier Li & Malik (2016): an example problem has 25 local minimas Auer et al. (1995).

B.2 Multiple Policies of Existing DRL Algorithms

We provide observational experiments on four challenging MuJoCo tasks Todorov et al. (2012), namely, Humanoid, Hopper, HalfCheetah, and Ant (details given in Appx. C.1), which are typical examples of the locomotion control of a robot. 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. 4, 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. 4 (top-left); another

interesting example is HalfCheetah, in which an agent can run normally or in a flipped manner, as shown in Fig. 4 (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).

C MuJoCo Tasks with Multiple Policies

C.1 Description of MuJoCo Taks

We selected six challenging robotic locomotion tasks from MuJoCo, namely, Swimmer-v3, Hopper-v3, Walker2D-v3, HalfCheetah-v3, Ant-v3, and Humanoid-v3. Table 2 lists the action space and state space of each task.

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

C.2 Multiple policies in MuJoCo tasks

In the supplementary files, we include rendered videos of different policies, as given 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.

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

D Reinforcement Learning and Bellman Equation

A reinforcement learning (RL) citepsutton2018reinforcement agent interacts with an unknown environment and learns an optimal policy that maximizes the cumulative reward. Mathematically, the en-

vironment 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

$$Q^\pi(s,a) = \mathbb{E}_{S_{k+1}\sim\mathbb{P}(\cdot|S_k,A_k),A_{k+1}\sim\pi(S_{k+1},\cdot)}\left[\sum_{k=0}^{\infty}\gamma^k \cdot R(S_k, A_k, S_{k+1})|S_0 = s, A_0 = a\right], \quad (12)$$

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 conventional objective function $J(\theta)$ of reinforcement learning citepsutton2018reinforcement takes the following form

$$J(\theta) \triangleq \mathbb{E}_{S_0,A_0}[Q^{\pi_\theta}(S_0, A_0)] = \mathbb{E}_{\tau\sim\pi}\left[R(\tau) \cdot P(\tau|\pi_\theta)\right], \quad (13)$$

where τ is a trajectory, i.e., $\tau = (S_0, A_0, \cdots)$, and
$P(\tau|\pi_\theta) = d_0(s_0) \cdot \prod_{k=0}^{T} \mathbb{P}(s_{k+1}|s_k, a_k)\pi_\theta(a_k|s_k)$.

The Bellman equation citepsutton2018reinforcement converts (12) into a recursive form as follows

$$\begin{aligned} Q^\pi(s,a) &= \sum_{s'\in\mathcal{S}} \mathbb{P}(s' \mid s,a)\left[R(s,a,s') + \gamma \sum_{a'\in\mathcal{A}} \pi(s',a')Q^\pi(s',a')\right] \\ &= R(s,a) + \gamma \sum_{s'\in\mathcal{S}} \mathbb{P}(s' \mid s,a) \sum_{a'\in\mathcal{A}} \pi(s',a')Q^\pi(s',a'), \end{aligned} \quad (14)$$

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 citepsutton2018reinforcement is

$$Q^*(s,a) = \sum_{s'\in\mathcal{S}} \mathbb{P}(s' \mid s,a)\left[R(s,a,s') + \gamma \max_{a'} Q^*(s',a')\right]\cdot. \quad (15)$$

The optimal policy π^* is given by a greedy strategy such that $\pi^* = \arg\max_\pi \; Q^\pi(s, a)$.

E Quantum K-Spin Hamiltonian Formulation of Reinforcement Learning

We provide the detailed steps of reformulating (12) into a K-spin Hamiltonian equation

$$\begin{aligned} H(\theta) &\triangleq -\mathbb{E}_{S_0,A_0}\left[Q^{\pi_\theta}(S_0, A_0)\right] \\ &= -\mathbb{E}_{S_0,A_k\sim\pi_\theta(S_k,\cdot),S_{k+1}\sim\mathbb{P}(\cdot|S_k,A_k)}\left[\sum_{k=0}^{\infty}\gamma^k \cdot R(S_k, A_k)\right] \\ &= -\sum_{k=0}^{K-1} \mathbb{E}_{S_0,A_0,\cdots,S_k\sim\mathbb{P}(\cdot|S_{k-1},A_{k-1}),A_k\sim\pi_\theta(S_k,\cdot)}\left[\gamma^k \cdot R(S_k, A_k)\right] \\ &= -\sum_{k=0}^{K-1}\sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}}\cdots\sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}} \gamma^k \cdot R(\mu_k) \cdot d_0(S_0) \cdot \pi_\theta(\mu_0)\prod_{i=0}^{k-1}\left[\mathbb{P}(S_{i+1}|\mu_i) \cdot \pi_\theta(\mu_{i+1})\right] \\ &= -\sum_{k=0}^{K-1}\sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}}\cdots\sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}} \left[\gamma^k \cdot R(\mu_k) \cdot d_0(S_0) \cdot \prod_{i=0}^{k-1}\mathbb{P}(S_{i+1}|\mu_i)\right] \cdot \pi_\theta(\mu_0)\cdots\pi_\theta(\mu_k) \\ &= -\sum_{k=0}^{K-1}\sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}}\cdots\sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}} L_{\mu_0,\ldots,\mu_k} \cdot \pi_\theta(\mu_0)\cdots\pi_\theta(\mu_k), \end{aligned} \quad (16)$$

where $K \to \infty$, and the density function is

$$L_{\mu_0,\ldots,\mu_k} = \gamma^k \cdot R(\mu_k) \cdot d_0(S_0) \cdot \prod_{i=0}^{k-1} \mathbb{P}(S_{i+1}|\mu_i). \tag{17}$$

F Derivation Steps for Hamiltonian's Policy Gradients

We provide the policy gradient of the quantum K-spin Hamiltonian equation in (3) for both stochastic and deterministic cases, which are variants of the well-known policy gradient theorem citepsutton2018reinforcement.

Theorem 2. ***(Hamiltonian's stochastic policy gradient)*** *The stochastic gradient of the K-spin Hamiltonian equation (3) w.r.t. parameter θ is*

$$\nabla_\theta H(\theta) = -\mathbb{E}_{\mu_0,\ldots,\mu_{K-1}} \left[\sum_{k=0}^{K-1} \gamma^k \cdot R(\mu_k) \cdot \nabla_\theta \log \left(\pi_\theta(\mu_0) \cdot \pi_\theta(\mu_1) \cdots \pi_\theta(\mu_k) \right) \right]. \tag{18}$$

Corollary 1. *When $K \to \infty$, the Hamiltonian's stochastic policy gradient $\nabla_\theta H(\theta)$ in (18) is equal to the stochastic policy gradient $\nabla_\theta J(\theta)$ in citepsutton1999policy,*

$$\lim_{K\to\infty} \nabla_\theta H(\theta) = -\nabla_\theta J(\theta) = -\mathbb{E}_{s\sim d_\theta, a\sim\pi_\theta} \left[Q^{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(s,a) \right]. \tag{19}$$

Let $\eta_\theta(\cdot) : \mathcal{S} \to \mathcal{A}$ denote a deterministic policy, while we use $\widetilde{\pi}_{\theta,\delta}(\mu)$ to represent that a Gaussian noise (a.k.a, an exploration noise) with standard deviation $\delta > 0$ is added in the exploration process.

Theorem 3. ***(Hamiltonian's deterministic policy gradient)*** *The deterministic gradient of the K-spin Hamiltonian equation (3) w.r.t. parameter θ is*

$$\nabla_\theta H'(\theta) = -\mathbb{E}_{\mu_0,\ldots,\mu_{K-1}} \left[\sum_{k=0}^{K-1} \gamma^k \cdot R(\mu_k) \cdot \nabla_\theta \log \left(\widetilde{\pi}_{\theta,\delta}(\mu_0) \cdot \widetilde{\pi}_{\theta,\delta}(\mu_1) \cdots \widetilde{\pi}_{\theta,\delta}(\mu_k) \right) \right]. \tag{20}$$

Corollary 2. *When $K \to \infty$, the Hamiltonian's deterministic policy gradient $\nabla_\theta H'(\theta)$ in (20) is equal to the deterministic policy gradient $\nabla_\theta J'(\theta)$ in citepsilver2014deterministic,*

$$\lim_{K\to\infty} \nabla_\theta H'(\theta) = -\nabla_\theta J'(\theta) = -\mathbb{E}_{s\sim d_\theta} \left[\nabla_a Q^{\widetilde{\pi}_{\theta,\delta}}(s,a)|_{a=\eta_\theta} \nabla_\theta \eta_\theta(s) \right]. \tag{21}$$

Corollary 3. *When the variance of the exploration noise approaches zero, i.e., $\delta \to 0$, the deterministic policy gradient $\nabla_\theta H'(\theta)$ is the limiting case of the stochastic policy gradient $\nabla_\theta H(\theta)$,*

$$\nabla_\theta H'(\theta) = \lim_{\delta\to 0} \nabla_\theta H(\theta). \tag{22}$$

F.1 Proof of Theorem 2: Hamiltonian's Stochastic Policy Gradient

Proof.

$$\begin{aligned}
\nabla_\theta H(\theta) &= -\sum_{k=0}^{K-1} \sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}} \cdots \sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}} L_{\mu_0,\ldots,\mu_k} \nabla_\theta \left[\pi_\theta(\mu_0) \cdots \pi_\theta(\mu_k) \right] \\
&= -\sum_{k=0}^{K-1} \sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}} \cdots \sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}} L_{\mu_0,\ldots,\mu_k} \left[\pi_\theta(\mu_0) \cdots \pi_\theta(\mu_k) \right] \nabla_\theta \log \left[\pi_\theta(\mu_0) \cdots \pi_\theta(\mu_k) \right] \\
&= -\sum_{k=0}^{K-1} \sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}} \cdots \sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}} \gamma^k \cdot R(\mu_k) \cdot d_0(S_0) \cdot \pi_\theta(\mu_0) \prod_{i=0}^{k-1} \left[\mathbb{P}(S_{i+1}|\mu_i) \cdot \pi_\theta(\mu_{i+1}) \right] \cdot \nabla_\theta \log \left[\pi_\theta(\mu_0) \cdots \pi_\theta(\mu_k) \right] \\
&= -\mathbb{E}_{\mu_0,\ldots,\mu_{K-1}} \left[\sum_{k=0}^{K-1} \gamma^k \cdot R(\mu_k) \cdot \nabla_\theta \log \left[\pi_\theta(\mu_0) \cdots \pi_\theta(\mu_k) \right] \right],
\end{aligned} \tag{23}$$

where $\mu_k = (S_k, A_k)$, $S_0 \sim d_0(\cdot)$, $A_k \sim \pi_\theta(S_k, \cdot)$, $S_{k+1} \sim \mathbb{P}(\cdot \mid S_k, A_k)$ for $k = 0 \cdots K$. □

F.2 PROOF OF COROLLARY 1

Proof.

$$
\begin{aligned}
\nabla_\theta H(\theta) &\overset{(a)}{=} -\sum_{k=0}^{K-1}\sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}}\cdots\sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}} L_{\mu_0,\ldots,\mu_k}\nabla_\theta\left[\pi_\theta(\mu_0)\cdots\pi_\theta(\mu_k)\right] \\
&\overset{(b)}{=} -\sum_{k=0}^{K-1}\sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}}\cdots\sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}} L_{\mu_0,\ldots,\mu_k}\sum_{i=0}^{k}\pi_\theta(\mu_0)\cdots\pi_\theta(\mu_{i-1})\pi_\theta(\mu_{i+1})\cdots\pi_\theta(\mu_k)\nabla_\theta\pi_\theta(\mu_i) \\
&\overset{(c)}{=} -\sum_{k=0}^{K-1}\sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}}\cdots\sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}}\gamma^k\cdot R(\mu_k)\cdot d_0(S_0)\prod_{i=0}^{k-1}\mathbb{P}(S_{i+1}|\mu_i)\sum_{i=0}^{k}\left[\prod_{j=0}^{i-1}\pi_\theta(\mu_j)\cdot\nabla_\theta\pi_\theta(\mu_i)\cdot\prod_{j=i+1}^{k}\pi_\theta(\mu_j)\right] \\
&\overset{(d)}{=} -\sum_{k=0}^{K-1}\sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}}\cdots\sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}}\sum_{i=0}^{k}d_0(S_0)\left[\gamma^i\prod_{j=0}^{i-1}\pi_\theta(\mu_j)\mathbb{P}(S_{j+1}|\mu_j)\right]\nabla_\theta\pi_\theta(\mu_i)\left[\prod_{j=i+1}^{k-1}\pi_\theta(\mu_j)\mathbb{P}(S_{j+1}|\mu_j)\pi_\theta(\mu_k)\gamma^{k-i}R(\mu_k)\right] \\
&\overset{(e)}{=} -\sum_{k=0}^{K-1}\sum_{i=0}^{k}\sum_{S_0}^{\mathcal{S}}d_0(S_0)\sum_{S_i}^{\mathcal{S}}\rho(S_0,S_i,i)\sum_{A_i}^{\mathcal{A}}\nabla_\theta\pi_\theta(S_i,A_i)\cdot\sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}}\rho(S_i,S_k,k-i)\cdot\pi_\theta(\mu_k)\cdot R(\mu_k) \\
&\overset{(f)}{=} -\sum_{S_0}^{\mathcal{S}}d_0(S_0)\sum_{S}^{\mathcal{S}}\sum_{i=0}^{K-1}\rho(S_0,S,i)\sum_{A}^{\mathcal{A}}\nabla_\theta\pi_\theta(S,A)\cdot\left[\sum_{S'}^{\mathcal{S}}\sum_{k=i}^{K-1}\rho(S,S',k-i)\cdot\sum_{A'}^{\mathcal{A}}\pi_\theta(S',A')\cdot R(S',A')\right] \\
&\overset{(g)}{=} -\sum_{S_0}^{\mathcal{S}}d_0(S_0)\sum_{S}^{\mathcal{S}}\sum_{i=0}^{\infty}\rho(S_0,S,i)\sum_{A}^{\mathcal{A}}\nabla_\theta\pi_\theta(S,A)\cdot Q^{\pi_\theta}(S,A) \\
&\overset{(h)}{=} -\left[\sum_{S}^{\mathcal{S}}\sum_{S_0}^{\mathcal{S}}d_0(S_0)\sum_{i=0}^{\infty}\rho(S_0,S,i)\right]\cdot\sum_{S}^{\mathcal{S}}\frac{\sum_{S_0}^{\mathcal{S}}d_0(S_0)\sum_{i=0}^{\infty}\rho(S_0,S,i)}{\sum_{s}^{\mathcal{S}}\sum_{S_0}^{\mathcal{S}}d_0(S_0)\sum_{i=0}^{\infty}\rho(S_0,S,i)}\sum_{A}^{\mathcal{A}}\nabla_\theta\pi_\theta(S,A)\cdot Q_\theta(S,A) \\
&\overset{(i)}{\propto} -\sum_{S}^{\mathcal{S}}d_{\pi_\theta}(S)\sum_{A}^{\mathcal{A}}\nabla_\theta\pi_\theta(S,A)\cdot Q^{\pi_\theta}(S,A) \\
&\overset{(j)}{=} -\mathbb{E}_{S\sim d_\theta,A\sim\pi_\theta(S,\cdot)}[Q^{\pi_\theta}(S,A)\nabla_\theta\log\pi_\theta(S,A)],
\end{aligned}
\tag{24}
$$

where $\rho(S,S',i)$ denotes the probability of state S transfer to S' in i steps.

We provide detailed explanations step-by-step:

- Equality (a) holds by definition.
- In equality (b), using the chain rule, we take derivative of $\nabla_\theta[\pi_\theta(\mu_0)\cdots\pi_\theta(\mu_k)]$ with respect to $\pi_\theta(\mu_i)$, $i=1,...,k$.
- In equality (c), we plug in $L_{\mu_0,\cdots,\mu_k}$ in (2).
- In equality (d), we insert $\mathbb{P}(S_{i+1}|\mu_i)\ \mathbb{P}(S_{i+1}|\mu_i)$ between $\pi_\theta(\mu_i)$ and $\pi_\theta(\mu_{i+1})$, $i=1,...,k$.
- In equality (e), we split trajectory $\mu_0,\cdots,\mu_i,\cdots,\mu_k$ into two trajectories $\mu_0,\cdots,\mu_i$ and $\mu_i,\cdots,\mu_k$. Therefore, we can classify all trajectories $\mu_0,\cdots,\mu_k$ by $\mu_0,\mu_i,\mu_k,$ and i.
- In equality (f), we reorganize $\sum_{k=0}^{K-1}\sum_{i=0}^{k}$ into $\sum_{i=0}^{K-1}\sum_{k=i}^{K-1}$. The former one first traverses the length k of a trajeoctory, and then traverses the i-th step on it.The latter one first traverses the i-th step of a trajectory, and then traverses the length k of it.
- In equality (g), we calculate the limit of (f) when K approaches ∞.
- In equality (h), we normalize $\sum_{S_0}^{\mathcal{S}}d_0(S_0)\sum_{i=0}^{\infty}\rho(S_0,S,i)$ to be a probability distribution.

- In equality (i), we remove the constant $\sum_S^{\mathcal{S}} \sum_{S_0}^{\mathcal{S}} d_0(S_0) \sum_{i=0}^{\infty} \rho(S_0, S, i)$ and replace the fraction with $d_{\pi_\theta}(S)$, the stationary distribution of state S under policy π_θ.

- In equality (j), we reformulate (i) as expectation.

□

F.3 Proof of Theorem 3: Hamiltonian's Deterministic Policy Gradient

Proof. Let $\eta_\theta(\cdot) : \mathcal{S} \to \mathcal{A}$ denote a deterministic policy, while we use $\widetilde{\pi}_{\theta,\delta}(\mu)$ to represent that a Gaussian noise (a.k.a, an exploration noise) with standard deviation $\delta > 0$ is added in the exploration process. In the inference stage, there is no exploration noise, the policy is deterministic, i.e., $\delta = 0$ and $A_k = \eta_\theta(S_k)$.

$$
\begin{aligned}
H'(\theta) &\triangleq -\mathbb{E}_{S_0 \sim d_0, A_0 \sim \widetilde{\pi}_{\theta,\delta}} \left[Q^{\widetilde{\pi}_{\theta,\delta}}(S_0, A_0) \right] \\
&= -\mathbb{E}_{S_0, A_k \sim \widetilde{\pi}_{\theta,\delta}(S_k,\cdot), S_{k+1} \sim \mathbb{P}(\cdot|S_k, A_k)} \left[\sum_{k=0}^{\infty} \gamma^k \cdot R(S_k, A_k) \right] \\
&= -\sum_{k=0}^{K} \mathbb{E}_{S_0, A_k \sim \widetilde{\pi}_{\theta,\delta}(S_k,\cdot), S_{k+1} \sim \mathbb{P}(\cdot|S_k, A_k)} \left[\gamma^k \cdot R(S_k, A_k) \right] \\
&= -\sum_{k=0}^{K} \sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}} \cdots \sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}} \gamma^k \cdot R(\mu_k) \cdot d_0(S_0) \cdot \widetilde{\pi}_{\theta,\delta}(\mu_0) \prod_{i=0}^{k-1} \left[\mathbb{P}(S_{i+1}|\mu_i) \cdot \widetilde{\pi}_{\theta,\delta}(\mu_{i+1}) \right] \\
&= -\sum_{k=0}^{K} \sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}} \cdots \sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}} \left[\gamma^k \cdot R(\mu_k) \cdot d_0(S_0) \cdot \prod_{i=0}^{k-1} \mathbb{P}(S_{i+1}|\mu_i) \right] \cdot \widetilde{\pi}_{\theta,\delta}(\mu_0) \cdots \widetilde{\pi}_{\theta,\delta}(\mu_k) \\
&= -\sum_{k=0}^{K} \sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}} \cdots \sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}} L_{\mu_0,\ldots,\mu_k} \cdot \widetilde{\pi}_{\theta,\delta}(\mu_0) \cdots \widetilde{\pi}_{\theta,\delta}(\mu_k),
\end{aligned}
\tag{25}
$$

where $K \to \infty$, and

$$
L_{\mu_0,\ldots,\mu_k} = \gamma^k \cdot R(\mu_k) \cdot d_0(S_0) \cdot \prod_{i=0}^{k-1} \mathbb{P}(S_{i+1}|\mu_i). \tag{26}
$$

□

F.4 Proof of Corollary 2

Proof.

$$
\begin{aligned}
\nabla_\theta H'(\pi_\theta) &= -\sum_{k=0}^{K}\sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}}\cdots\sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}}\left(L_{\mu_0,\ldots,\mu_k}\cdot\nabla_\theta\left[\widetilde{\pi}_\theta(\mu_0)\cdots\widetilde{\pi}_\theta(\mu_k)\right]+\nabla_\theta L_{\mu_0,\cdots,\mu_k}\cdot\widetilde{\pi}_\theta(\mu_0)\cdots\widetilde{\pi}_\theta(\mu_k)\right)\\
&= -\sum_{k=0}^{K}\sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}}\cdots\sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}}\left[\widetilde{\pi}_\theta(\mu_0)\cdots\widetilde{\pi}_\theta(\mu_k)\right]\cdot\nabla_\theta L_{\mu_0,\ldots,\mu_k}\\
&= -\sum_{k=0}^{K}\sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}}\cdots\sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}}\nabla_\theta\left[\gamma^k\cdot R(\mu_k)\cdot d_0(S_0)\cdot\prod_{i=0}^{k-1}\mathbb{P}(S_{i+1}|\mu_i))\right]\\
&= -\sum_{k=0}^{K}\sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}}\cdots\sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}}\nabla_A\left[\gamma^k\cdot R(\mu_k)\cdot d_0(S_0)\cdot\prod_{i=0}^{k-1}\mathbb{P}(S_{i+1}|\mu_i))\right]\nabla_\theta\eta_\theta(S)\\
&= -\sum_{S_0}^{\mathcal{S}}d_0(S_0)\nabla_A\mathbb{E}_{S_{t+1}\sim\mathbb{P}(\cdot|S_t,A_t)}\left[\sum_{t=0}^{\infty}\gamma^k R(S_t,A_t)\right]\cdot\nabla_\theta\eta_\theta(S)\\
&= -\sum_{S_0}^{\mathcal{S}}d_0(S_0)\nabla_A Q(S_0,A_0)\cdot\nabla_\theta\eta_\theta(S)\\
&= -\mathbb{E}_{S_0\sim d_0(\cdot)}\left[\nabla_A Q(S_0,A_0)\cdot\nabla_\theta\eta_\theta(S)\right]
\end{aligned}
\tag{27}
$$

where $\mu_k = (S_k, A_k)$, $S_0 \sim d_0(\cdot)$, $A_k \sim \pi_\theta(S_k, \cdot)$, $S_{k+1} \sim \mathbb{P}(\cdot \mid S_k, A_k)$, for $k = 0 \cdots K$. □

F.5 Proof of Corollary 3

Proof. In Corollary 2 and Corollary 1, we have

$$
\begin{aligned}
\nabla_\theta H'(\theta) &= -\nabla_\theta J'(\theta),\\
\nabla_\theta H(\theta) &= -\nabla_\theta J(\theta),
\end{aligned}
\tag{28}
$$

when $K \to \infty$.

citepsilver2014deterministic proved that

$$
\nabla_\theta J'(\theta) = \lim_{\delta\to 0}\nabla_\theta J(\theta), \tag{29}
$$

where δ is the standard deviation of the Gaussian noise of stochastic policy π_θ.

Therefore,

$$
\nabla_\theta H'(\theta) = \lim_{\delta\to 0}\nabla_\theta H(\theta) \tag{30}
$$

□

G Variance Reduction with Quantum K-Spin Hamiltonian Equation

G.1 Monte Carlo Estimator of Quantum K-Spin Hamiltonian Equation

Monte Carlo Estimator citepmohamed2020monte: Consider a general probabilistic objective $\mathcal{F}$ of the form:

$$
\mathcal{F} \triangleq \mathbb{E}_{p(\boldsymbol{x};\theta)}[f(\boldsymbol{x};\phi)], \tag{31}
$$

in which a function f of an input variable $\boldsymbol{x}$ with *structural parameters* ϕ is evaluated on average with respect to an input distribution $p(\boldsymbol{x};\theta)$ with *distributional parameters* θ.

A Monte Carlo method evaluates the function by first drawing independent samples $\hat{\boldsymbol{x}}^{(1)}, ..., \hat{\boldsymbol{x}}^{(N)}$ from the distribution $p(\boldsymbol{x};\theta)$, and then computing the average:

$$
\widehat{\mathcal{F}}_N = \frac{1}{N}\sum_{i=1}^{N} f(\hat{\boldsymbol{x}}^{(i)}), \quad \text{where } \hat{\boldsymbol{x}}^{(i)} \sim p(\boldsymbol{x};\theta) \text{ for } i = 1, ..., N. \tag{32}
$$

The Monte Carlo estimator for (13) is

$$\widehat{J}(\theta) = \frac{1}{N}\sum_{i=1}^{N} R(\tau^{(i)}), \text{ where } \tau^{(i)} \sim P(\tau^{(i)}|\pi_\theta) \text{ for } i = 1, ..., N, \tag{33}$$

and

$$P(\tau^{(i)}|\pi_\theta) = d_0(s_0^{(i)}) \cdot \prod_{k=0}^{T} \mathbb{P}(s_{k+1}^{(i)}|s_k^{(i)}, a_k^{(i)})\pi_\theta(a_k^{(i)}|s_k^{(i)}). \tag{34}$$

The Monte Carlo estimator for (3) is

$$\widehat{H}(\theta) = \frac{1}{N'}\sum_{i=1}^{N'}\sum_{k=0}^{K-1} L_{\mu_0^{(i)},\ldots,\mu_k^{(i)}}, \text{ for } i = 1, ..., N', \tag{35}$$

and

$$L_{\mu_0^{(i)},\ldots,\mu_k^{(i)}} = \gamma^k \cdot R(\mu_k^{(i)}) \cdot d_0(s_0^{(i)}) \cdot \prod_{\ell=0}^{k-1} \mathbb{P}(s_{\ell+1}^{(i)}|\mu_\ell^{(i)}). \tag{36}$$

Remark: The above two Monte Carlo estimators are quite different in the simulation process. (34) samples a random trajectory by following an environment's stochastic transition and a policy. In contrast, (36) measures a random path's discounted reward (the "energy") without following any policy, and the Hamiltonian equation (3) combinatorially enumerates all possible paths of length K over the state-action space. In other words, the simulation process of the Hamiltonian term does not rely on any policy. Therefore, the Hamiltonian term is a suitable regularizer for both on-policy and off-policy algorithms.

This fundamental difference is due to the Ising model in (1), which combinatorially enumerates all paths and separates the environment and the policy.

G.2 Variance Reduction

For the general function in (31), one simple but effective variance reduction technique is to subtract a baseline term as follows:

$$\mathbb{E}_{p(\boldsymbol{x};\theta)}\left[(f(\boldsymbol{x}) - \beta)\nabla_\theta \log p(\boldsymbol{x};\theta)\right], \tag{37}$$

where β is the baseline term.

Our reasoning logic:

1). We first briefly describe a high-level idea citepgreensmith2001variance that adding a baseline term, like the proposed H-term, will help reduce the gradient variance.

2). We sketch the steps to show how the proposed H-term will mathematically reduce the gradient variance, following the framework in Section 5.2 of citepgreensmith2004variance.

High-level IDEA. One generic approach to reduce the variance of Monte Carlo estimates is to use an additive control variate. Suppose we wish to estimate the integral of the function $f : \mathcal{X} \to \mathbb{R}$, and we know the value of the integral of another function on the same space $\phi : \mathcal{X} \to \mathbb{R}$.
We have

$$\int_{\mathcal{X}} f(x) = \int_{\mathcal{X}} (f(x) - \phi(x)) + \int_{\mathcal{X}} \phi(x), \tag{38}$$

and the integral of $f(x) - \phi(x)$ can be estimated. If $\phi(x) = f(x)$, meaning that , then we have managed to reduce our variance to zero citepgreensmith2001variance. More generally,

$$\text{Var}(f - \phi) = \text{Var}(f) - 2\text{Cov}(f, \phi) + \text{Var}(\phi). \tag{39}$$

If ϕ and f are strongly correlated, so that the covariance term on the right hand side is greater than the variance of ϕ, i.e., $-2\text{Cov}(f, \phi) + \text{Var}(\phi) \leq 0$. then a variance reduction has been made over the original estimation problem citepgreensmith2001variance, i.e., $\text{Var}(f - \phi) \leq \text{Var}(f)$.

Our reasoning. Then, we present our reasoning.

Note that the gradient of the new objective function of the actor network in (6) consists of two components, namely $\nabla_\theta J(\theta)$ and $\nabla_\theta H(\theta)$. Here, we consider

$$\nabla_\theta J(\theta) - \lambda\, \nabla_\theta H(\theta), \quad \text{where } \lambda > 0 \text{ is a temperature parameter,} \tag{40}$$

where $\nabla_\theta J(\theta)$ in (19) is the above function $f(\cdot)$ and $\lambda\, \nabla_\theta H(\theta)$ in (4) is the above function $\phi(\cdot)$.

The Hamiltonian stochastic gradient in (4) has the optimal value

$$\nabla_\theta H^*(\theta) = -\lim_{K\to\infty} \mathbb{E}_{\mu_0,\ldots,\mu_{K-1}} \left[\sum_{k=0}^{K-1} \gamma^k \cdot R(\mu_k) \cdot \nabla_\theta \log\left(\pi_\theta(\mu_0)\cdot \pi_\theta(\mu_1)\cdots \pi_\theta(\mu_k)\right)\right]. \tag{41}$$

According to Theorem 8 of citepgreensmith2001variance that is proved via (39), we have

$$\begin{aligned}\mathrm{Var}\left[\nabla_\theta J(\theta) - \lambda\, \nabla_\theta H^*(\theta)\right] &= \mathrm{Var}[\nabla_\theta J(\theta)] - \frac{1}{\lambda}\mathbb{E}_{s\sim d_\theta, a\sim\pi_\theta}\left[\frac{\left(\mathbb{E}_{s\sim d_\theta, a\sim\pi_\theta}\left[(\nabla_\theta \log \pi_\theta(s,a))^2 \nabla_\theta J(\theta)\right]\right)^2}{\mathbb{E}_{s\sim d_\theta, a\sim\pi_\theta}\left[(\nabla_\theta \log \pi_\theta(s,a))^2\right]}\right] \\ &\le \mathrm{Var}[\nabla_\theta J(\theta)],\end{aligned} \tag{42}$$

where the second term is positive and

$$\nabla_\theta H^*(\theta) = \frac{\mathbb{E}_{s\sim d_\theta, a\sim\pi_\theta}\left(\left[\nabla_\theta \log \pi_\theta(s,a))^2 \nabla_\theta J(\theta)\right]\right.}{\mathbb{E}_{s\sim d_\theta, a\sim\pi_\theta}\left[(\nabla_\theta \log \pi_\theta(s,a))^2\right]}. \tag{43}$$

In Alg. 1 and Alg. 2, we used a general H-term $\nabla_\theta H(\theta)$, not the optimal one in (41). Next, we provide a general characterization for this case.

According to Theorem 10 of citepgreensmith2001variance, we have

$$\begin{aligned}&\mathrm{Var}\left[\nabla_\theta J(\theta) - \lambda\, \nabla_\theta H(\theta)\right] - \mathrm{Var}\left[\nabla_\theta J(\theta) - \lambda\, \nabla_\theta H^*(\theta)\right] \\ &= \lambda^2\, \mathbb{E}_{s\sim d_\theta, a\sim\pi_\theta}\left[(\nabla_\theta \log \pi_\theta(s,a))^2 (\nabla_\theta H(\theta) - \nabla_\theta H^*(\theta))^2\right]\end{aligned} \tag{44}$$

Assume Lipschiz continuity of the graident $\nabla_\theta H(\theta)$ such that

$$||\nabla_\theta H(\theta) - \nabla_\theta H^*(\theta)||_2 \le 2L(H(\theta) - H^*(\theta)) \le 2L\epsilon, \tag{45}$$

given $K \ge \log_\gamma \epsilon$ with $L > 0, \epsilon > 0$, as pointed out in the end of Section 3.2.

Therefore, combining (44), (46) with (46), we obtain that

$$\begin{aligned}\mathrm{Var}\left[\nabla_\theta J(\theta) - \lambda\, \nabla_\theta H(\theta)\right] &= \mathrm{Var}[\nabla_\theta J(\theta)] - \frac{1}{\lambda}\mathbb{E}_{s\sim d_\theta, a\sim\pi_\theta}\left[\frac{\left(\mathbb{E}_{s\sim d_\theta, a\sim\pi_\theta}\left[(\nabla_\theta \log \pi_\theta(s,a))^2 \nabla_\theta J(\theta)\right]\right)^2}{\mathbb{E}_{s\sim d_\theta, a\sim\pi_\theta}\left[(\nabla_\theta \log \pi_\theta(s,a))^2\right]}\right] \\ &+ \lambda^2 (2L\epsilon)^2\, \mathbb{E}_{s\sim d_\theta, a\sim\pi_\theta}\left[(\nabla_\theta \log \pi_\theta(s,a))^2\right] \\ &\le \mathrm{Var}[\nabla_\theta J(\theta)],\end{aligned} \tag{46}$$

when both $|\nabla_\theta \log \pi_\theta(s,a)|$ and $|\nabla_\theta J(\theta)|$ are upper bounded, e.g., $|\nabla_\theta \log \pi_\theta(s,a)| < C_1$ and $|\nabla_\theta J(\theta)| < C_2$; and we set ϵ, λ properly such that

$$\begin{aligned}-\frac{1}{\lambda}C_2^2 + 4\lambda^2 L^2 \epsilon^2 C_1^2 &< 0 \\ \lambda^3 \epsilon^2 &< \frac{C_2^2}{4L^2 C_1^2},\end{aligned} \tag{47}$$

which can be easily satisfied by properly selecting λ and $K \ge \log_\gamma \epsilon$.

Conclusion:

To sum up, we show that it is easy to achieve $\mathrm{Var}\left[\nabla_\theta J(\theta) - \lambda\, \nabla_\theta H(\theta)\right] \le \mathrm{Var}[\nabla_\theta J(\theta)]$, which means adding the H-term can lead to smaller variance than that of the conventional gradient.

H ACTOR-CRITIC ALGORITHMS FOR DEEP REINFORCEMENT LEARNING

The gradient of (13) is citepsutton2018reinforcement

$$\nabla_\theta J(\theta) \triangleq \sum_S^{\mathcal{S}} d_{\mathcal{S},\theta}(S) \sum_A^{\mathcal{A}} Q_\theta(S,A)\, \nabla_\theta \pi_\theta(S,A). \tag{48}$$

Since Q_θ in (48) is unknown citepwen2021characterizing (the stationary distribution d_θ is unknown), one can plug in a critic network with parameter ϕ as an estimator of Q_θ and obtain

$$\nabla_\theta^\phi J(\theta,\phi) = \sum_S^{\mathcal{S}} d_{\mathcal{S},\theta}(S) \sum_A^{\mathcal{A}} Q_\phi(S,A)\, \nabla_\theta \pi_\theta(S,A), \tag{49}$$

where $d_{\mathcal{S},\theta} \in \mathbb{R}_+^{|\mathcal{S}||\mathcal{A}|\times 1}$ denotes the stationary distribution over the states instead of state-action pairs.

(49) is a bi-level optimization problem citepchen2021closing, and a natural solution is an iterative algorithm that alternates between estimating Q_ϕ with parameter ϕ and improving policy π_θ with parameter θ. Therefore, a family of actor-critic algorithms are proposed with following objective functions:

$$\begin{cases} \text{Actor}: \max\limits_\theta J_\pi(\theta,\phi) = (1-\gamma)\mathbb{E}_{S_0\sim d_0, A_0\sim\pi_\theta(S_0,\cdot)}\left[Q_\phi(S_0,A_0)\right] \\ \text{Critic}: \max\limits_\phi J_Q(\theta,\phi) = \dfrac{1}{2}\mathbb{E}_{S\sim d_\theta(\cdot), A\sim\pi_\theta(S,\cdot)}\left[(Q_\phi(S,A)-y(S,A))^2\right]. \end{cases} \tag{50}$$

The gradient of (50) can be estimated as follows

$$\begin{aligned} \nabla_\theta \widehat{J}_\pi(\theta,\phi) &= \frac{1}{N}\sum_{i=1}^N Q_\phi(\mu)\cdot\nabla_\theta \log \pi_\theta(\mu) \\ \nabla_\phi \widehat{J}_Q(\theta,\phi) &= \frac{1}{N}\sum_{i=1}^N \left[Q_\phi(S,A)-y(S,A)\right]\cdot\nabla_\phi Q_\phi(S,A) \end{aligned} \tag{51}$$

The parameters ϕ and θ are updated as follows:

$$\begin{cases} \text{Actor}: \quad \theta \leftarrow \theta + \alpha\, \nabla_\theta^\phi \widehat{J}_\pi, \\ \text{Critic}: \quad \phi \leftarrow \phi - \alpha\, \nabla_\phi \widehat{J}_Q. \end{cases} \tag{52}$$

I DETERMINISTIC POLICY GRADIENT ALGORITHM WITH H-TERM

For completeness, we present the details of the deterministic actor-critic algorithm with H-term.

We apply the proposed Hamiltonian equation (3) to regularize the actor network. Specifically, $H'(\theta)$ in (3) is added to the actor's objective with weight $\lambda > 0$. The objective functions of actor and critic networks become:

$$\begin{cases} \text{Actor}: \max\limits_\theta J'_\pi(\theta,\phi) = (1-\gamma)\mathbb{E}_{S_0\sim d_0, A_0=\eta_\theta(S_0)}\left[Q_\phi\left(S_0,A_0\right)\right] - \lambda H'(\theta), \\ \text{Critic}: \min\limits_{\boldsymbol{\phi}} J_Q(\theta,\phi) = \dfrac{1}{2}\mathbb{E}_{S\sim d_\theta(\cdot), A=\eta_\theta(S)}\left[(Q_\phi(S,A)-y(S,A))^2\right]. \end{cases} \tag{53}$$

The gradient of (53) is

$$\nabla_\theta J'_\pi(\theta,\phi) = (1-\gamma)\sum_S^{\mathcal{S}} d_{\mathcal{S},\theta}(S)\nabla_A Q_\phi(S,A)\cdot\nabla_\theta \eta_\theta(S) - \lambda\nabla_\theta H'(\theta), \tag{54}$$

$$\nabla_\phi J_Q(\theta,\phi) = \sum_S^{\mathcal{S}} d_{S,\theta}(S)\cdot\left[Q_\phi(S,A)-y(S,A)\right]\cdot\nabla_\phi Q_\phi(S,A)|_{A=\eta_\theta(S)}. \tag{55}$$

Algorithm 1 Stationary Actor-Critic Algorithm with H-term

Input: learning rate α, temperature λ, look-ahead step K, and parameters M, T, G, B, B'
Initialize actor network π and critic network Q with parameters θ, ϕ, and replay buffers $\mathcal{D}_1$, $\mathcal{D}_2$
for episode $= 1, \cdots, M$ **do**
    Initialize state s_0
    for $t = 0, \cdots, T-1$ **do**
        Select action $a_t \sim \pi_\theta(\cdot|s_t)$
        Execute action a_t, receive reward r_t, and observe new state s_{t+1}
        Store a transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}_1$
    end
    Store a trajectory τ of length T in $\mathcal{D}_2$
    for $g = 1, \cdots, G$ **do**
        Randomly sample a mini-batch of B transitions $\{(s_i, a_i, r_i, s_{i+1})\}_{i=1}^{B}$ from $\mathcal{D}_1$
        Randomly sample a mini-batch of B' trajectories (of length K) $\{\tau_j\}_{j=1}^{B'}$ from $\mathcal{D}_2$
        Update critic network using a conventional method
        Update actor network as $\theta \leftarrow \theta + \alpha \left(\nabla_\theta \widehat{J}(\theta) - \lambda \nabla_\theta \widehat{H}(\theta)\right)$.
    end
end

Algorithm 2 Stationary Actor-Critic Algorithm with H-term (Deterministic Version)

Input: learning rate α, temperature λ, look-ahead step K, and parameters δ, M, T, G, B, B'
Initialize actor network η and critic network Q with parameters θ, ϕ, and replay buffers $\mathcal{D}_1$, $\mathcal{D}_2$
for episode $= 1, \cdots, M$ **do**
    Initialize state s_0
    for $t = 0, \cdots, T-1$ **do**
        Take action $a_t = \eta_\theta(s_t) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \delta^2)$
        Execute action a_t, receive reward r_t, and observe new state s_{t+1}
        Store a transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}_1$
    end
    Store a trajectory τ of length T in $\mathcal{D}_2$
    for $g = 1, \cdots, G$ **do**
        Randomly sample a mini-batch of B transitions $\{(s_i, a_i, r_i, s_{i+1})\}_{i=1}^{B}$ from $\mathcal{D}_1$
        Randomly sample a mini-batch of B' trajectories (of length K) $\{\tau_j\}_{j=1}^{B'}$ from $\mathcal{D}_2$
        Update critic network using a conventional method
        Update actor network as $\theta \leftarrow \theta + \alpha \left(\nabla_\theta \widehat{J}'(\theta) - \lambda \nabla_\theta \widehat{H}'(\theta)\right)$.
    end
end

To estimate $\nabla_\theta H'(\theta)$, the Monte Carlo gradient estimator in (5) is used. Therefore, (54) and (55) can be estimated as follows:

$$\nabla_\theta \widehat{J}'_\pi(\theta, \phi) = \frac{1}{N} \sum_{i=1}^{N} \left[\nabla_A Q_\phi(S, A)|_{A=\eta_\theta(S)} \nabla_\theta \eta_\theta(S)\right] - \frac{1}{N'} \sum_{i=1}^{N'} \left[\lambda \sum_{k=0}^{K} \gamma^k R(\mu_k) \nabla_\theta \log\left[\widetilde{\pi}_\theta(\mu_0) \cdots \widetilde{\pi}_\theta(\mu_k)\right]\right], \tag{56}$$

$$\nabla_\phi \widehat{J}_Q(\theta, \phi) = \frac{1}{N} \sum_{i=1}^{N} \left[Q_\phi(S, A) - y(S, A)\right] \cdot \nabla_\phi Q_\phi(S, A)|_{A=\eta_\theta(S)}. \tag{57}$$

Table 4: Hyperparameters used for the PPO and PPO + H in MuJoCo tasks

Parameters	Values
Optimizer	Adam
Learning rate	$3 \cdot 10^{-4}$
Discount (γ)	0.99
GAE parameter	0.95
Number of hidden layers for all networks	3
Number of hidden units per layer	256
Mini-batch size	32
Importance rate of H-term (λ)	2^{-3}
Truncation step of H-term (K)	16

Table 5: Hyperparameters used for the DDPG and DDPG + H in MuJoCo tasks

Parameters	Values
Optimizer	Adam
Learning rate	$5 \cdot 10^{-4}$
Target Update Rate (τ)	10^{-3}
Discount (γ)	0.995
Replay buffer size	10^6
Number of hidden layers for all networks	3
Number of hidden units per layer	256
Batch size	64
Importance rate of H-term (λ)	2^{-3}
Truncation step of H-term (K)	16

J EXPERIMENTS: HYPERPARAMETERS AND MORE RESULTS

J.1 HYPERPARAMETERS IN EXPERIMENTS

J.2 MORE RESULTS

Fig. 5 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 shows more performance of the PPO+H algorithm, for $K = 8, 16, 24$. We run each experiment with 20 random seeds and each run we test 100 episodes.

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

K HAMILTONIAN POLICY NETWORK

K.1 HAMILTONIAN POLICY NETWORK

Since Hamiltonian equation in (3) is a functional of policy π_θ, a natural question would be: can we use the Hamiltonian equation replace existing Bellman's equation (14) or the policy gradient's objective function (13)?

As a verification, we test the capability of Hamiltonian equation in (3) as a loss function to train a policy network. The algorithm is first given as follows.

In Alg. 3, an agent interacts with an environment and updates its policy network. The algorithm has M episodes and each episode consists of a (Monte Carlo) simulation process and a learning process (gradient estimation) as follows :

- During the (Monte Carlo) simulation process (lines 5-9 of Alg. 3), an agent takes action a_t according to a policy $\pi_\theta(\cdot|s_t)$, $t = 0, \cdots, T-1$, generating a trajectory of T steps/transitions.

Table 6: Hyperparameters used for the REINFORCE and REINFORCE + H in TSP, graph maxcut, MIMO beamforming, and non-convex deep learning classifier tasks

Parameters	Values
Optimizer	Adam
Learning rate	$5 \cdot 10^{-5}$
Discount (γ)	0.99
Number of hidden layers for all networks	3
Number of hidden units per layer	512
Batch size	4096
Importance rate of H-term (λ)	2^{-3}
Truncation step of H-term (K)	6

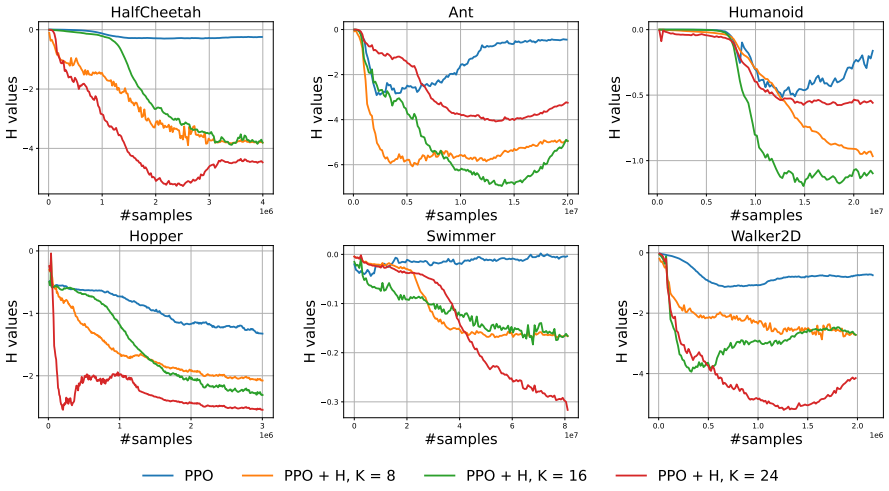


Figure 5: H values during the training process.

Then, the full trajectory $\tau = (s_0, a_0, r_0, s_1, \cdots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$ is stored in replay buffer $\mathcal{D}$.

- During the learning process ($G \geq 1$ updates in one episode) (lines 10-12 of Alg. 1), a mini-batch of B trajectories (of length K) $\{\tau_j = (s_0^j, a_0^j, r_0^j, s_1^j, \cdots, s_{K-1}^j, a_{K-1}^j, r_{K-1}^j, s_K^j)\}_{j=1}^B$ are sampled from $\mathcal{D}$, respectively. The policy network is updated by a Monte Carlo gradient estimator over B trajectories.

Implementation of replay buffer $\mathcal{D}$. After a full trajectory τ of length T is generated, it is partitioned into $T - K + 1$ trajectories of length K. We rank them according to the cumulative reward and store the top portion, say 80%, into a new replay buffer $\mathcal{D}$ (line 9 of Alg. 3). We randomly sample a mini-batch of B trajectories from $\mathcal{D}$ (line 11 of Alg. 3) to compute the H-term.

K.2 Frozenlake Task

Environment: Frozenlake 8×8, a game in OpenAI Gym.

Rules: As shown in Fig. 7 (left), the Frozenlake task has 8×8 states with 4 optional actions to move around. The agent needs to go from the start point and find the way to the destination in limited steps. There are 8 holes which can cause the agent to fail the game.

Experiment settings: We take Deep Q-learning (DQN) citepmnih2013playing as our baseline and use the implementation from the ElegantRL library. We use a 4-layer fully connected neural network as the deep policy network both in DQN and DHN. We use the Adam optimizer with a learning rate 1×10^{-3} and a batch size 100.

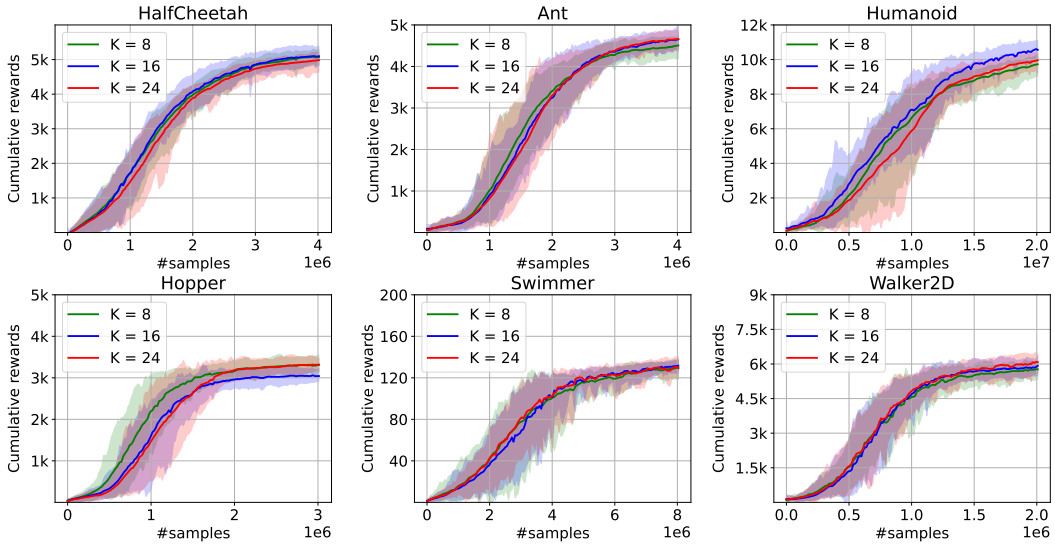


Figure 6: For the proposed PPO+H algorithm, the performance with different K values.

Table 7: Experimental results on six challenging 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	

Evaluation: We evaluate the performance of policy by computing the success rate, in which we use 50 agents to walk 100 steps and compute the rates of agents who successfully arrive the destination.

Results for the Frozenlake task: Fig. 8 (left) shows the success rate of agents with increasing the number of transitions learned by the network. compared with DQN, DHN has a more stable training process. It is easy for DQN to quickly obtain a good policy to win the game. But with increasing the number of transitions fed to the network, the performance of DQN shows a large and frequent shock while the performance of DHN shows the strong stability.

K.3 GRIDWORLD TASK

Environment: a Gridworld of size 10×10, a game available in our code.

Rules: As shown in the Fig. 7 (right), the Gridworld has 10×10 states with 4 optional actions to move around. The agent will initialize at a random locations and it needs to find the smiley as many as possible which has 10 reward in turn. It should be noted that there are some endpoints which may cause the agent game over and some transfer-points which transfer the agent to certain location.

Experiment settings and evaluation: Both the experiment settings and evaluation method are the same with that on Frozenlake 8×8 game.

Algorithm 3 Hamiltonian Policy Network

Input: learning rate α, look-ahead step K, and parameters M, T, G, B
Initialize policy network with parameters θ, and replay buffer $\mathcal{D}$
for episode $= 1, \cdots, M$ **do**
    Initialize state s_0
    for $t = 0, \cdots, T-1$ **do**
        Select action $a_t \sim \pi_\theta(\cdot|s_t)$
        Execute action a_t, receive reward r_t, and observe new state s_{t+1}
    end
    Store a trajectory τ of length T in $\mathcal{D}$
    for $g = 1, \cdots, G$ **do**
        Randomly sample a mini-batch of B trajectories (of length K) $\{\tau_j\}_{j=1}^{B}$ from $\mathcal{D}$
        Update pocliy network as $\theta \leftarrow \theta - \alpha \nabla_\theta \widehat{H}(\theta)$.
    end
end

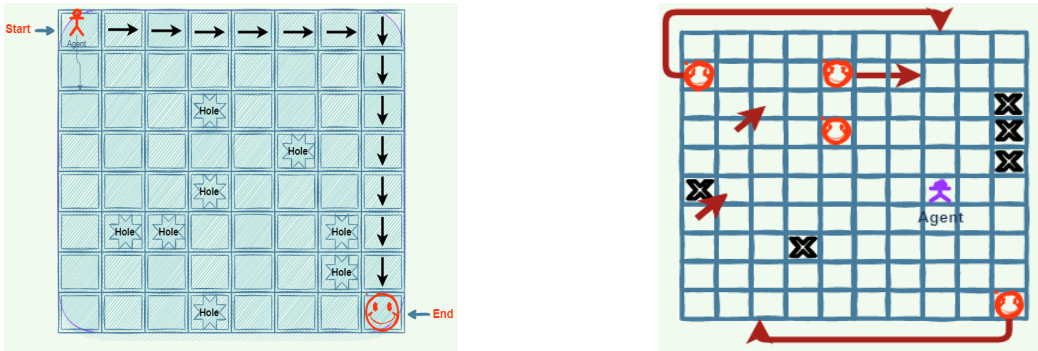


Figure 7: The Frozenlake task (left) and Gridworld task (right).

Results for the Gridworld task: Fig. 8 (right) shows the mean reward obtained by the agents with increasing the training time. Compared with DQN, DHN has a faster training process. It only needs massive random parallel samples of trajectories and do not need any policy for guided sampling while DQN needs guided exploration in the training process which costs a large time consumption.

L Combinatorial Optimization Problems

L.1 Problem Formulation and MDP formulation

- **Graph maxcut**: Given a graph $G = (V, E)$, where V is the set of nodes and E the set of edges, find a subset $S \subseteq V$ that maxmizes the weight of the cut-set $\sum_{u \in S, v \in V \setminus S, (u,v) \in E} w(u, v)$.
- **Traveling salesman problems (TSP)**: Given a fully connected graph $G = (V, E)$, find a tour J that minimizes the edge weights $\sum_{e \in J} w(e)$. A tour starts and ends at a specific node after having visited each node exactly once.
- **MDP formulation**: The MDP formulations of graph maxcut and TSP are given in Table 8. Note, J is a partially ordered set, (J, v) means add node v to the end of J.

L.2 Existence of Many Local Minimas

- Travelling salesman problem (TSP): the case of 8 cities (Fig. 9) has 2 local minimas.
- Graph maxcut: a fully connected graph of 20 nodes (Appx. L) has 390 local minimas.

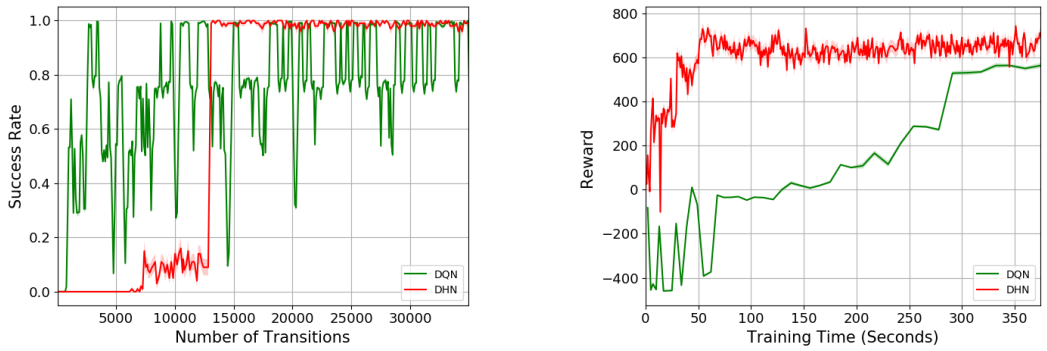


Figure 8: Comparison between the DQN and DHN algorithms. The Frozenlake task (left) and Gridworld task (right).

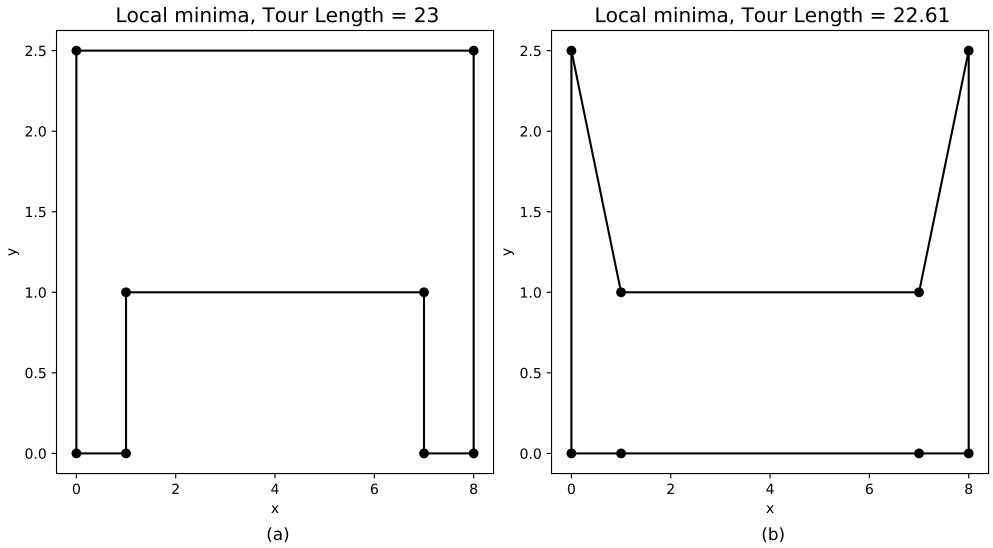


Figure 9: TSP problem has two local minimas in this case.

L.3 Experimental Settings

- **Environments (tasks)**. We train the proposed algorithms on TSP and graph maxcut tasks with number of nodes $N = 100$. For TSP, all nodes in training instances are generated from an uniform distribution within an unit square. For graph maxcut, training instances are randomly generated from Erdős–Rényi model. We use the same test cases in citepkool2018attention,khalil2017learning.
- **Compared algorithms**. To evaluate policy gradient algorithm, we choose REINFORCE citepsutton1999policy. Since the H-term is compatible with existing variance reduction techniques, we implement the REINFORCE algorithm with baseline reduction. For a fair comparison, we keep the hyperparameters (listed in Appx. J) the same and make sure that the obtained results reproduce existing benchmark tests.
- **Performance metrics**. We employ the metric: Frequency vs. Approximation Ratio. We run each experiment with 20 random seeds and in each run we test 100 episodes.The approximation ratio ϵ is defined as

$$\epsilon = \max(\frac{Optimal}{Obj}, \frac{Obj}{Optimal}), \tag{58}$$

where $Optimal$ is the optimal objective value by Branch and Bound citeplawler1966branch, Obj is the objective value of the best solution obtained from RL.

Table 8: MDP formulation of combinatorial optimization: Graph maxcut and TSP

Problem	State	Action	Transition	Reward	Termination
Graph maxcut	S	select $v \in V \backslash S$	$S' \leftarrow S \cup \{v\}$	$\text{cut}(S') - \text{cut}(S)$	$\forall S', \text{cut}(S') < \text{cut}(S)$
TSP	J	select $v \in V \backslash J$	$J' \leftarrow (J, v)$	$cost(J') - cost(J)$	$\forall v \in V, v \in J$

Algorithm 4 Simulated Annealing

Input: M, temperature function $t(\cdot)$, energy function $E(\pi)$, transition probability $P(E(\pi'), E(\pi), T)$, neighbour $N(\pi)$
Initialize policy π_0
for episode $i = 1, 2, \cdots, M$ **do**
    Update temperature $T_i = t(\frac{i}{M})$
    Pick a random neighbour $\pi_{new} \in N(\pi_{i-1})$

    Estimate the energy $E(\pi_{new})$
    if $P(E(\pi_{i-1}), E(\pi_{new}), T_i) \geq random(0, 1)$ **do**
        $\pi_i = \pi_{new}$
    else do
        $\pi_i = \pi_{i-1}$
end
Output: the final policy π_M.

L.3.1 Learn to Simulated Annealing with REINFORCE

Simulated Annealing Formulation

- Energy function $E(\pi) := H(\pi_{\mu_0}, \pi_{\mu_1}, \cdots, \pi_{\mu_{K-1}})$, where $H(\pi_{\mu_0}, \pi_{\mu_1}, \cdots, \pi_{\mu_{K-1}})$ is defined in (3). $E(\pi)$ can be estimated by performing N rollouts of length-K with policy π, as shown in Theorem 1.
- Neighbour $N(\pi) := \pi', \forall \pi'$.
- Transition probability $P(E(\pi), E(\pi'), T) := \exp(-(E(\pi') - E(\pi))/T)$.
- Temperature function $t(x) := 2 - x, x \in [0, 1]$.

M MIMO Beamforming

MIMO beamforming is an NP-hard problem in 5G/6G communication systems citepbjornson2013optimal. Given a channel $\boldsymbol{H} = [\boldsymbol{h}_1, \cdots, \boldsymbol{h}_K] \in \mathbb{C}^{N \times K}$, and a power constraint $P \in \mathbb{R}$, find a beamformer $\boldsymbol{W} = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_K] \in \mathbb{C}^{N \times K}$ that maximizes the summation of rates:

$$\begin{aligned} \max_{\boldsymbol{W}} \quad & \sum_{k=1}^{K} \log_2 \left(1 + \text{SINR}_k\right), \\ \text{s.t.} \quad & \text{Tr}\left(\boldsymbol{W}\boldsymbol{W}^H\right) \leq P, \\ \text{where} \quad & \text{SINR}_k = \frac{\left|\boldsymbol{h}_k^H \boldsymbol{w}_k\right|^2}{\sum_{j=1, j \neq k}^{K} \left|\boldsymbol{h}_k^H \boldsymbol{w}_j\right|^2 + \sigma^2}. \end{aligned} \tag{59}$$