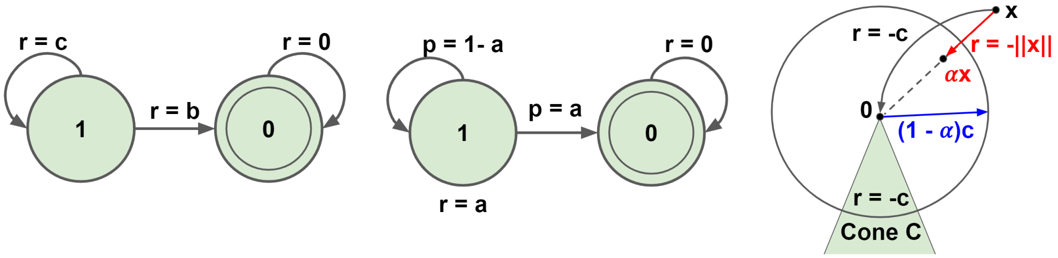

(a) Shortest path problem (b) Blackmailer's problem (c) Optimal stopping problem

Figure 5: Revisiting Fig. 1 for the discounted cases where $\gamma \in (0, 1)$.

A More Examples with Multiple Fixed Points

First, we consider the discounted formulations of the three examples (shown in Fig. 1), as shown in Fig. 5 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. 5.

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

Assume $V_0(1) \geq b$ and $c > (1 - \gamma)b$, we have

$$\begin{aligned}
V_1(1) &= c + \gamma \cdot V_0(1) \geq c + \gamma \cdot b > b \\
V_2(1) &= c + \gamma \cdot c + \gamma^2 \cdot V_0(1) \geq (1 + \gamma)c + \gamma^2 b > b \\
V_3(1) &= c + \gamma \cdot c + \gamma^2 c + \gamma^3 \cdot V_0(1) \geq (1 + \gamma + \gamma^2)c + \gamma^3 b > b \\
&\cdots \\
V_k(1) &= \sum_{i=0}^{k-1} \gamma^i \cdot c + \gamma^k \cdot V_0(1) \geq \sum_{i=0}^{k-1} \gamma^i \cdot c + \gamma^k b > b \\
&\cdots \\
V^*(1) &= \sum_{i=0}^{\infty} \gamma^i \cdot c = \frac{1}{1 - \gamma} c > b
\end{aligned} \tag{13}$$

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

$$H(0) = 0, \quad H(1) = -b - \sum_{k=2}^{\infty}(\sum_{i=1}^{k-1}\gamma^{i-1} \cdot c + \gamma^k b) = -\infty. \tag{14}$$

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

$$\begin{aligned} V^*(1) + H(1) &= \sum_{i=0}^{\infty}\gamma^i \cdot c - \infty = -\infty \\ V^*(0) + H(0) &= b. \end{aligned} \tag{15}$$

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:

$$H(0) = 0, \quad H(1) = -\sum_{k=1}^{\infty}\sum_{i=1}^{k-1}\gamma^{i-1} \cdot a = -\infty. \tag{16}$$

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

$$H(x) = -c - \sum_{k=2}^{\infty}\left[\sum_{i=1}^{k-1}\gamma^i \cdot \alpha^i \cdot \|x\| + \gamma^k \cdot (-c)\right] = -\infty \tag{17}$$

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

B MuJoCo Tasks with Multiple Policies

B.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, Humanoid-v3. Table 3 lists the action space and state space of each task.

Table 3: 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

B.2 Multiple policies in MuJoCo tasks

In the supplementary files, we includes rendered videos of different policies, as given in Table 4.

- Different policies are obtained over 20 runs of the PPO algorithm. We rendered theses polices and classified them by physical gaits.
- The policies in bold texts are physically stationary.

Table 4: 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

C Quantum K-Spin Hamiltonian Formulation of Reinforcement Learning

We provide the detailed steps of reformulating (1) into a K-spin Hamiltonian equation

$$\begin{aligned}
H(\theta) &\triangleq -\mathbb{E}_{S_0,A_0}\left[Q^{\pi_\theta}(S_0,A_0)\right] \\
&= -\mathbb{E}_{S_0,A_k\sim\pi_\theta(S_k,\cdot),S_{k+1}\sim\mathbb{P}(\cdot|S_k,A_k)}\left[\sum_{k=0}^{\infty}\gamma^k\cdot R(S_k,A_k)\right] \\
&= -\sum_{k=0}^{K-1}\mathbb{E}_{S_0,A_0,\cdots,S_k\sim\mathbb{P}(\cdot|S_{k-1},A_{k-1}),A_k\sim\pi_\theta(S_k,\cdot)}\left[\gamma^k\cdot R(S_k,A_k)\right] \\
&= -\sum_{k=0}^{K-1}\sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}}\cdots\sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}}\gamma^k\cdot R(\mu_k)\cdot d_0(S_0)\cdot\pi_\theta(\mu_0)\prod_{i=0}^{k-1}\left[\mathbb{P}(S_{i+1}|\mu_i)\cdot\pi_\theta(\mu_{i+1})\right] \\
&= -\sum_{k=0}^{K-1}\sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}}\cdots\sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}}\left[\gamma^k\cdot R(\mu_k)\cdot d_0(S_0)\cdot\prod_{i=0}^{k-1}\mathbb{P}(S_{i+1}|\mu_i)\right]\cdot\pi_\theta(\mu_0)\cdots\pi_\theta(\mu_k) \\
&= -\sum_{k=0}^{K-1}\sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}}\cdots\sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}}L_{\mu_0,\ldots,\mu_k}\cdot\pi_\theta(\mu_0)\cdots\pi_\theta(\mu_k),
\end{aligned} \tag{18}$$

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

$$L_{\mu_0,\ldots,\mu_k}=\gamma^k\cdot R(\mu_k)\cdot d_0(S_0)\cdot\prod_{i=0}^{k-1}\mathbb{P}(S_{i+1}|\mu_i). \tag{19}$$

Table 5: Revisiting the analogy between MDP and quantum K-spin Ising model.

MDP (Our formulation in (7))		Quantum K-spin Ising Model [23, 12] in (5)	
State-action pairs	$\mu_0, ..., \mu_{K-1}$	Spins	$j_0, \cdots, j_{K-1}$
Optimal policy	$\pi^*_{\mu_0}\times\pi^*_{\mu_1}\times\cdots\times\pi^*_{\mu_{K-1}}$	Quantum field	$\sigma_{j_0}\times\sigma_{j_1}\times\cdots\times\sigma_{j_{K-1}}$
Cumulative rewards	$L_{\mu_0\ldots\mu_{K-1}}$	Density function	$L_{j_0\cdots j_{K-1}}$
Functional of policy	$H(\pi_{\mu_0}, ..., \pi_{\mu_{K-1}})$	Functional of spins	$H(\sigma_{j_0},\cdots,\sigma_{j_{K-1}})$
Stationary condition	$\frac{\delta H(\pi_{\mu_0},\cdots,\pi_{\mu_{K-1}})}{\delta\pi_\mu}=0$	Stationary condition	$\frac{\delta H(\sigma_{j_0},\cdots,\sigma_{j_{K-1}})}{\delta\sigma_j}=0$

D Derivation Steps for Section 4.2: Hamiltonian's Policy Gradients

We provide the policy gradient of the quantum K-spin Hamiltonian equation in (7) for both stochastic and deterministic cases, which are variants of the well-known policy gradient theorem [32].

Theorem 1. ***(Hamiltonian's stochastic policy gradient)*** *The stochastic gradient of the K-spin Hamiltonian equation (7) w.r.t. parameter θ is*

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

Corollary 1. *When $K \to \infty$, the Hamiltonian's stochastic policy gradient $\nabla_\theta H(\theta)$ in (20) is equal to the stochastic policy gradient $\nabla_\theta J(\theta)$ in [33],*

$$\lim_{K\to\infty} \nabla_\theta H(\theta) = -\nabla_\theta J(\theta) = -\mathbb{E}_{s\sim d_\theta, a\sim\pi_\theta}\left[Q^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(s,a)\right]. \tag{21}$$

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 2. ***(Hamiltonian's deterministic policy gradient)*** *The deterministic gradient of the K-spin Hamiltonian equation (7) w.r.t. parameter θ is*

$$\nabla_\theta H'(\theta) = -\mathbb{E}_{\mu_0,\ldots,\mu_{K-1}}\left[\sum_{k=0}^{K-1} \gamma^k \cdot R(\mu_k) \cdot \nabla_\theta \log\left(\widetilde{\pi}_{\theta,\delta}(\mu_0)\cdot\widetilde{\pi}_{\theta,\delta}(\mu_1)\cdots\widetilde{\pi}_{\theta,\delta}(\mu_k)\right)\right]. \tag{22}$$

Corollary 2. *When $K \to \infty$, the Hamiltonian's deterministic policy gradient $\nabla_\theta H'(\theta)$ in (22) is equal to the deterministic policy gradient $\nabla_\theta J'(\theta)$ in [31],*

$$\lim_{K\to\infty} \nabla_\theta H'(\theta) = -\nabla_\theta J'(\theta) = -\mathbb{E}_{s\sim d_\theta}\left[\nabla_a Q^{\widetilde{\pi}_{\theta,\delta}}(s,a)|_{a=\eta_\theta}\nabla_\theta \eta_\theta(s)\right]. \tag{23}$$

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 $\nabla_\theta H(\theta)$,*

$$\nabla_\theta H'(\theta) = \lim_{\delta\to 0} \nabla_\theta H(\theta). \tag{24}$$

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

Proof.

$$\begin{aligned}
\nabla_\theta H(\theta) &= -\sum_{k=0}^{K-1}\sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}}\cdots\sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}} L_{\mu_0,\ldots,\mu_k}\nabla_\theta\left[\pi_\theta(\mu_0)\cdots\pi_\theta(\mu_k)\right]\\
&= -\sum_{k=0}^{K-1}\sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}}\cdots\sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}} L_{\mu_0,\ldots,\mu_k}\left[\pi_\theta(\mu_0)\cdots\pi_\theta(\mu_k)\right]\nabla_\theta\log\left[\pi_\theta(\mu_0)\cdots\pi_\theta(\mu_k)\right]\\
&= -\sum_{k=0}^{K-1}\sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}}\cdots\sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}} \gamma^k\cdot R(\mu_k)\cdot d_0(S_0)\cdot\pi_\theta(\mu_0)\prod_{i=0}^{k-1}\left[\mathbb{P}(S_{i+1}|\mu_i)\cdot\pi_\theta(\mu_{i+1})\right]\cdot\nabla_\theta\log\left[\pi_\theta(\mu_0)\cdots\pi_\theta(\mu_k)\right]\\
&= -\mathbb{E}_{\mu_0,\ldots,\mu_{K-1}}\left[\sum_{k=0}^{K-1}\gamma^k\cdot R(\mu_k)\cdot\nabla_\theta\log\left[\pi_\theta(\mu_0)\cdots\pi_\theta(\mu_k)\right]\right],
\end{aligned} \tag{25}$$

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 $k = 0\cdots K$. $\square$

D.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,\dots,\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,\dots,\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{26}
$$

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 (6).
- 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 μ_0,μ_i,μ_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.

□

D.3 Proof of Theorem 2: 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 $A_k = \eta_\theta(S_k)$.

$$\begin{aligned}
H'(\theta) &\triangleq -\mathbb{E}_{S_0 \sim d_0, A_0 \sim \widetilde{\pi}_{\theta,\delta}} \left[Q^{\widetilde{\pi}_{\theta,\delta}}(S_0, A_0) \right] \\
&= -\mathbb{E}_{S_0, A_k \sim \widetilde{\pi}_{\theta,\delta}(S_k,\cdot), S_{k+1} \sim \mathbb{P}(\cdot|S_k,A_k)} \left[\sum_{k=0}^{\infty} \gamma^k \cdot R(S_k, A_k) \right] \\
&= -\sum_{k=0}^{K} \mathbb{E}_{S_0, A_k \sim \widetilde{\pi}_{\theta,\delta}(S_k,\cdot), S_{k+1} \sim \mathbb{P}(\cdot|S_k,A_k)} \left[\gamma^k \cdot R(S_k, A_k) \right] \\
&= -\sum_{k=0}^{K} \sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}} \cdots \sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}} \gamma^k \cdot R(\mu_k) \cdot d_0(S_0) \cdot \widetilde{\pi}_{\theta,\delta}(\mu_0) \prod_{i=0}^{k-1} \left[\mathbb{P}(S_{i+1}|\mu_i) \cdot \widetilde{\pi}_{\theta,\delta}(\mu_{i+1}) \right] \\
&= -\sum_{k=0}^{K} \sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}} \cdots \sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}} \left[\gamma^k \cdot R(\mu_k) \cdot d_0(S_0) \cdot \prod_{i=0}^{k-1} \mathbb{P}(S_{i+1}|\mu_i) \right] \cdot \widetilde{\pi}_{\theta,\delta}(\mu_0) \cdots \widetilde{\pi}_{\theta,\delta}(\mu_k) \\
&= -\sum_{k=0}^{K} \sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}} \cdots \sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}} L_{\mu_0,\ldots,\mu_k} \cdot \widetilde{\pi}_{\theta,\delta}(\mu_0) \cdots \widetilde{\pi}_{\theta,\delta}(\mu_k),
\end{aligned} \tag{27}$$

where $K \to \infty$, and

$$L_{\mu_0,\ldots,\mu_k} = \gamma^k \cdot R(\mu_k) \cdot d_0(S_0) \cdot \prod_{i=0}^{k-1} \mathbb{P}(S_{i+1}|\mu_i). \tag{28}$$

□

D.4 Proof of Corollary 2

Proof.

$$
\begin{aligned}
\nabla_\theta H'(\pi_\theta) &= -\sum_{k=0}^{K}\sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}}\cdots\sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}} \left(L_{\mu_0,\ldots,\mu_k}\cdot\nabla_\theta\left[\widetilde{\pi}_\theta(\mu_0)\cdots\widetilde{\pi}_\theta(\mu_k)\right] + \nabla_\theta L_{\mu_0,\cdots,\mu_k}\cdot\widetilde{\pi}_\theta(\mu_0)\cdots\widetilde{\pi}_\theta(\mu_k)\right)\\
&= -\sum_{k=0}^{K}\sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}}\cdots\sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}} \left[\widetilde{\pi}_\theta(\mu_0)\cdots\widetilde{\pi}_\theta(\mu_k)\right]\cdot\nabla_\theta L_{\mu_0,\ldots,\mu_k}\\
&= -\sum_{k=0}^{K}\sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}}\cdots\sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}} \nabla_\theta\left[\gamma^k\cdot R(\mu_k)\cdot d_0(S_0)\cdot\prod_{i=0}^{k-1}\mathbb{P}(S_{i+1}|\mu_i))\right]\\
&= -\sum_{k=0}^{K}\sum_{\mu_0}^{\mathcal{S}\times\mathcal{A}}\cdots\sum_{\mu_k}^{\mathcal{S}\times\mathcal{A}} \nabla_A\left[\gamma^k\cdot R(\mu_k)\cdot d_0(S_0)\cdot\prod_{i=0}^{k-1}\mathbb{P}(S_{i+1}|\mu_i))\right]\nabla_\theta\eta_\theta(S)\\
&= -\sum_{S_0}^{\mathcal{S}} d_0(S_0)\nabla_A\mathbb{E}_{S_{t+1}\sim\mathbb{P}(\cdot|S_t,A_t)}\left[\sum_{t=0}^{\infty}\gamma^k R(S_t,A_t)\right]\cdot\nabla_\theta\eta_\theta(S)\\
&= -\sum_{S_0}^{\mathcal{S}} d_0(S_0)\nabla_A Q(S_0,A_0)\cdot\nabla_\theta\eta_\theta(S)\\
&= -\mathbb{E}_{S_0\sim d_0(\cdot)}\left[\nabla_A Q(S_0,A_0)\cdot\nabla_\theta\eta_\theta(S)\right]
\end{aligned}
\tag{29}
$$

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 $k = 0\cdots K$. $\square$

D.5 Proof of Corollary 3

Proof. In Corollary 2 and Corollary 1, we have

$$
\begin{aligned}
\nabla_\theta H'(\theta) &= -\nabla_\theta J'(\theta),\\
\nabla_\theta H(\theta) &= -\nabla_\theta J(\theta),
\end{aligned}
\tag{30}
$$

when $K \to \infty$.

[31] proved that

$$\nabla_\theta J'(\theta) = \lim_{\delta\to 0}\nabla_\theta J(\theta), \tag{31}$$

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

Therefore,

$$\nabla_\theta H'(\theta) = \lim_{\delta\to 0}\nabla_\theta H(\theta) \tag{32}$$

$\square$

E Conventional Actor-Critic Algorithms for Deep Reinforcement Learning

The gradient of (2) is [32]

$$\nabla_\theta J(\theta) \triangleq \sum_{S}^{\mathcal{S}} d_{\mathcal{S},\theta}(S) \sum_{A}^{\mathcal{A}} Q_\theta(S,A)\ \nabla_\theta \pi_\theta(S,A). \tag{33}$$

Since Q_θ in (33) is unknown [37] (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

$$\nabla_\theta^\phi J(\theta,\phi) = \sum_{S}^{\mathcal{S}} d_{\mathcal{S},\theta}(S) \sum_{A}^{\mathcal{A}} Q_\phi(S,A)\ \nabla_\theta \pi_\theta(S,A), \tag{34}$$

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.

(34) is a bi-level optimization problem [7], 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:

$$\begin{cases} \text{Actor} : \max\limits_{\theta} J_\pi(\theta,\phi) = (1-\gamma)\mathbb{E}_{S_0\sim d_0, A_0\sim\pi_\theta(S_0,\cdot)}\left[Q_\phi(S_0,A_0)\right] \\ \text{Critic} : \max\limits_{\phi} J_Q(\theta,\phi) = \dfrac{1}{2}\mathbb{E}_{S\sim d_\theta(\cdot), A\sim\pi_\theta(S,\cdot)}\left[(Q_\phi(S,A) - y(S,A))^2\right]. \end{cases} \tag{35}$$

The gradient of (35) can be estimated as follows

$$\begin{aligned} \nabla_\theta \widehat{J}_\pi(\theta,\phi) &= \frac{1}{N}\sum_{i=1}^{N} Q_\phi(\mu)\cdot\nabla_\theta \log\pi_\theta(\mu) \\ \nabla_\phi \widehat{J}_Q(\theta,\phi) &= \frac{1}{N}\sum_{i=1}^{N} \left[Q_\phi(S,A) - y(S,A)\right]\cdot\nabla_\phi Q_\phi(S,A) \end{aligned} \tag{36}$$

The parameters ϕ and θ are updated as follows:

$$\begin{cases} \text{Actor} : & \theta \leftarrow \theta + \alpha\ \nabla_\theta^\phi \widehat{J}_\pi, \\ \text{Critic} : & \phi \leftarrow \phi - \alpha\ \nabla_\phi \widehat{J}_Q. \end{cases} \tag{37}$$

F Stationary Deterministic Policy Gradient Algorithm with H-term

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

Algorithm 2 Stationary Actor-Critic Algorithm with H-term

```
Input: learning rate $\alpha$, temperature $\lambda$, look-ahead step $K$, and parameters $\delta, M, T, G, B, B'$
Initialize actor network $\eta$ and critic network $Q$ with parameters $\theta$, $\phi$, 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 $\tau$ 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
```

We apply the proposed Hamiltonian equation (7) to regularize the actor network. Specifically, $H'(\theta)$ in (7) is added to the actor's objective with weight $\lambda > 0$. The objective functions of actor and critic networks become:

$$\begin{cases} \text{Actor} : \max_{\theta} J'_\pi(\theta, \phi) = (1-\gamma)\mathbb{E}_{S_0 \sim d_0, A_0 = \eta_\theta(S_0)}\left[Q_\phi\left(S_0, A_0\right)\right] - \lambda H'(\theta), \\ \text{Critic} : \min_{\phi} J_Q(\theta, \phi) = \frac{1}{2}\mathbb{E}_{S \sim d_\theta(\cdot), A = \eta_\theta(S)}\left[\left(Q_\phi(S, A) - y(S, A)\right)^2\right]. \end{cases} \tag{38}$$

The gradient of (38) is

$$\nabla_\theta J'_\pi(\theta, \phi) = (1-\gamma)\sum_{S}^{\mathcal{S}} d_{\mathcal{S},\theta}(S)\nabla_A Q_\phi(S, A) \cdot \nabla_\theta \eta_\theta(S) - \lambda \nabla_\theta H'(\theta), \tag{39}$$

$$\nabla_\phi J_Q(\theta, \phi) = \sum_{S}^{\mathcal{S}} d_{\mathcal{S},\theta}(S) \cdot \left[Q_\phi(S, A) - y(S, A)\right] \cdot \nabla_\phi Q_\phi(S, A)|_{A=\eta_\theta(S)}. \tag{40}$$

To estimate $\nabla_\theta H'(\theta)$, the Monte Carlo gradient estimator in (11) is used. Therefore, (39) and (40) can be estimated as follows:

$$\nabla_\theta \widehat{J}'_\pi(\theta, \phi) = \frac{1}{N}\sum_{i=1}^{N}\left[\nabla_A Q_\phi(S, A)|_{A=\eta_\theta(S)} \nabla_\theta \eta_\theta(S)\right] - \frac{1}{N'}\sum_{i=1}^{N'}\left[\lambda \sum_{k=0}^{K} \gamma^k R(\mu_k) \nabla_\theta \log\left[\widetilde{\pi}_\theta(\mu_0)\cdots\widetilde{\pi}_\theta(\mu_k)\right]\right], \tag{41}$$

$$\nabla_\phi \widehat{J}_Q(\theta, \phi) = \frac{1}{N}\sum_{i=1}^{N}\left[Q_\phi(S, A) - y(S, A)\right] \cdot \nabla_\phi Q_\phi(S, A)|_{A=\eta_\theta(S)}. \tag{42}$$

G Experiments: Hyperparameters and More Results

G.1 Hyperparameters in Experiments

Table 6: 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
Replay buffer size	10^6
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 7: 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

G.2 More 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.

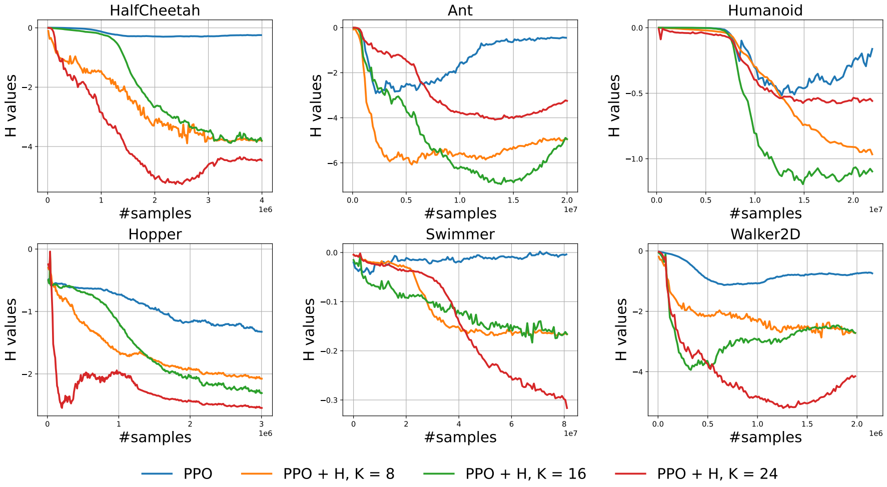

Figure 6: H values during the training process.

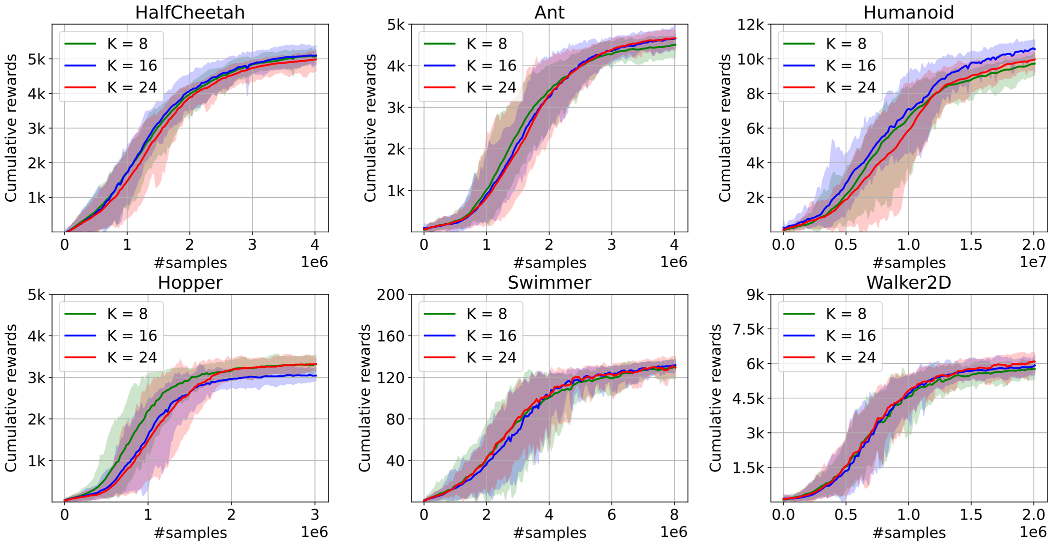

Figure 7: For the proposed PPO+H algorithm, the performance with different K values.

Fig. 7 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.

H Hamiltonian Policy Network

H.1 Hamiltonian Policy Network

Since Hamiltonian equation in (7) is a functional of policy π_θ, a natural question would be: can we use the Hamiltonian equation replace existing Bellman's equation (3) or the policy gradient's objective function (2)?

As a verification, we test the capability of Hamiltonian equation in (7) as a loss function to train a policy network. The algorithm is first given as follows.

Algorithm 3 Hamiltonian Policy Network

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

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. 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.

H.2 Frozenlake Task

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

Rules: As shown in Fig. 8 (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) [26] 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.

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. 9 (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.

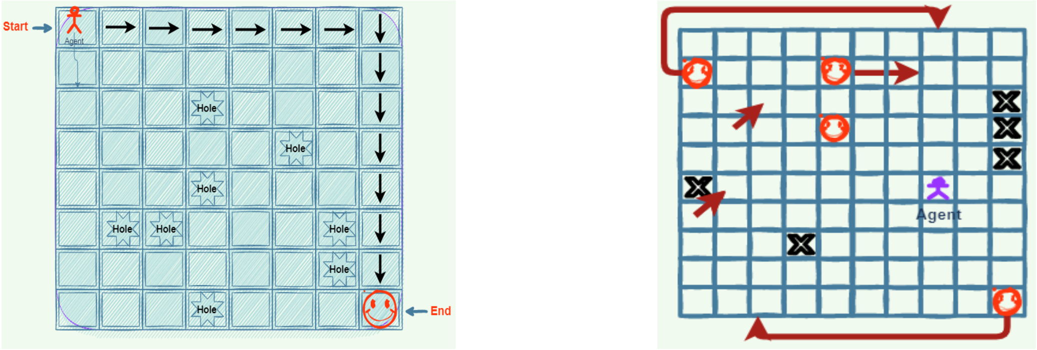

Figure 8: The Frozenlake task (left) and Gridworld task (right).

H.3 Gridworld Task

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

Rules: As shown in the Fig. 8 (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.

Results for the Gridworld task: Fig. 9 (rigt) 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.

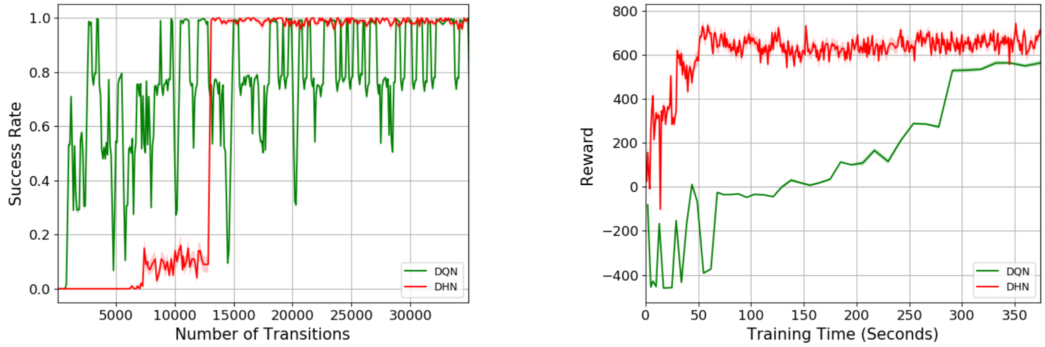

Figure 9: Comparison between the DQN and DHN algorithms. The Frozenlake task (left) and Gridworld task (right).