

A Algorithms

Algorithm 2: Hamiltonian Policy Optimization

```

1 Denote  $a_t, s_t$  as the action and state at time  $t$ ; Denote the replay buffer as  $\mathcal{B}$ ;
2 Initialize  $\theta, h$ ;
3 for  $t = 1, 2, \dots$  do
4   Sample  $a_t \sim \pi_{\theta_t}(\cdot | s_t)$ ;
5   Obtain  $a_t^K, \rho_t^K = f_{\text{HMC}}^K(s_t, a_t; \theta_t, h_t)$ ;
6   Apply  $a_t^K$ , and obtain next state  $s_{t+1}$ ;
7   Store the experience tuple  $(s_t, a_t^K, s_{t+1})$  into  $\mathcal{B}$ ;
8   Sample a minibatch of transitions  $\mathcal{D}_t$  from  $\mathcal{B}$ ;
9   Update the Q network by  $\mathcal{D}_t$ ;
10  Update  $\theta$  and  $h$  by optimizing  $\mathcal{J}(\theta, h)$  in (12) with minibatch  $\mathcal{D}_t$ ;
11 end

```

Algorithm 3: $f_{\text{HMC}}^K(s, a; \theta, h), \beta_0, \epsilon$

```

1 Sample  $\rho^0 \sim \mathcal{N}(0, I)$ ;
2 Set  $\rho_0 \leftarrow \rho^0 \cdot \frac{1}{\sqrt{\beta_0}}$ ;
3 for  $k = 1, \dots, K$  do
4   Obtain  $\rho_{k+1/2}$  by the first equation in (13);
5   Update  $a_k = a_{k-1} + \epsilon \odot \rho_{k+1/2}$ ;
6   Obtain  $\rho_{k+1}$  by the second equation in (13);
7 end
8 Return  $a_K, \rho_K$ ;

```

B Hyper-parameter Table

	α	$1/\sqrt{\beta_0^{\text{tr}}}$	$1/\sqrt{\beta_0^{\text{exp}}}$	ϵ	K	l	h_n	m	\mathcal{B} size
HalfCheetah-v2	0.2	1	0.5	0.2	3	1	32	256	10^6
Hopper-v2	0.2	0.1	1.5	0.15	2	1	32		
Walker2d-v2	0.2	0.2	1.5	0.15	3	1	32		
Ant-v2	0.2	0.1	1.0	0.1	3	1	32		
Humanoid-v2	0.05	1	1	0.1	3	1	64		
Humanoid PyB.	0.05	0.4	1.5	0.2	3	1	64		
Flagrun	0.05	0.2	1	0.15	3	1	32		
Flagrun Harder	0.05	0.2	1.5	0.15	3	1	64		

Table 1: Hyperparameters in SAC-HPO.

C Analysis

In this section, we conduct ablation study, sensitivity analysis and investigate the shape of the policy distributions evolved by HD.

C.1 Ablation Study

In ablation study, we first verify the effect of the proposed leapfrog operator (13) in comparison with the conventional leapfrog in (9). Then we study the differences of the effects of HMC in exploration and policy training, where the policy training (policy optimization) refers to the training step for policy network and neural networks T_h, σ_h in leapfrog (13).

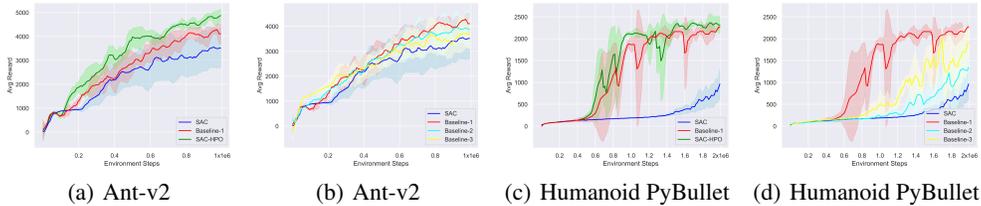


Figure 4: Performance Comparison in Ablation Study.

Specifically, we introduce three baselines adopting different leapfrog operators in exploration and policy training, which are summarized in Table 2. Here "Gaussian Policy" means that the same policy network in SAC is directly used without evolving actions by HMC. "Conv. Leapfrog" refers to that actions are evolved by the conventional leapfrog in (9), and "Prop. Leapfrog" denotes that actions are evolved by the proposed leapfrog in (13). We cannot use the proposed leapfrog only in exploration, since the neural networks T_h, σ_h in (13) need to be trained.

	Exploration	Policy Training
SAC-HPO	Prop. Leapfrog	Prop. Leapfrog
Baseline-1	Conv. Leapfrog	Conv. Leapfrog
Baseline-2	Conv. Leapfrog	Gaussian Policy
Baseline-3	Gaus. Policy	Prop. Leapfrog
SAC	Gaussian Policy	Gaussian Policy

Table 2: Exploration and Training Strategies in Baselines

The SAC-HPO and baselines are evaluated over Ant-v2 and HumanoidPyBulletEnv-v0, and learning curves are shown in Figure 4, where we use the same hyper-parameters as Section 5.1. It can be seen that in Figure 4(a) and 4(c), SAC-HPO outperforms the baseline-1 in terms of both performance and learning stability, showing the advantage of the proposed leapfrog operator over the conventional counterpart. And this advantage can also be observed in other environments.

In Figure 4(b) and 4(d), the baseline-1 outperforms both baseline-2 and baseline-3, showing the effects of leapfrog in both exploration and policy optimization. Further, we can see that the improvement of baseline-2 over SAC is higher than that of baseline-3 over SAC, meaning that using HMC in exploration can yield more performance improvement than that in policy training.

C.2 Sensitivity Analysis

Here we conducted sensitivity analysis on three important hyper-parameters. The first is the number of leapfrog steps in simulating HD, i.e., $K = 1, 2, 3$, shown in Figures 5(g), 5(h) and 5(i). We can see that in Hopper-v2, the cases of $K = 2$ and $K = 3$ perform similar, but much better than that of $K = 1$, showing that it is not meaningful to have larger K . The second parameter analyzed here is the variance of momentum vector ρ in exploration, i.e., $1/\sqrt{\beta_0^{\text{exp}}} = 0.5, 1.0, 1.5$, shown in Figure 5(d), 5(e) and 5(f). We can see that the performance is sensitive to the choice of β^{exp} , showing the variance of ρ in exploration is important to the learning performance. Although more randomness in exploration can encourage the agent to explore more unseen states, not the highest choice of β^{exp} leads to best performance, such as Figure 5(e). That is because too much variance of ρ may make the sampled state-action pairs deviate too much from the optimal trajectory during the exploration. In Figure 5(a), 5(b) and 5(c), we analyze three different values for the variances for momentum vector ρ in policy training, i.e., $1/\sqrt{\beta_0^{\text{tr}}} = 0.01, 0.1, 0.2$. However, we can see that the performance is not sensitive to β_0^{tr} . According to more evaluations, larger β_0^{tr} cannot lead to better performance. For example when $\beta_0^{\text{tr}} = 1$, the performance degrades significantly.

C.3 Effect of Random Momentum Vector

In Figure 6, we study the effect of the randomness of momentum vector ρ on the performance improvement. It is similar as the comparison with iterative amortized policy optimization (IAPO)

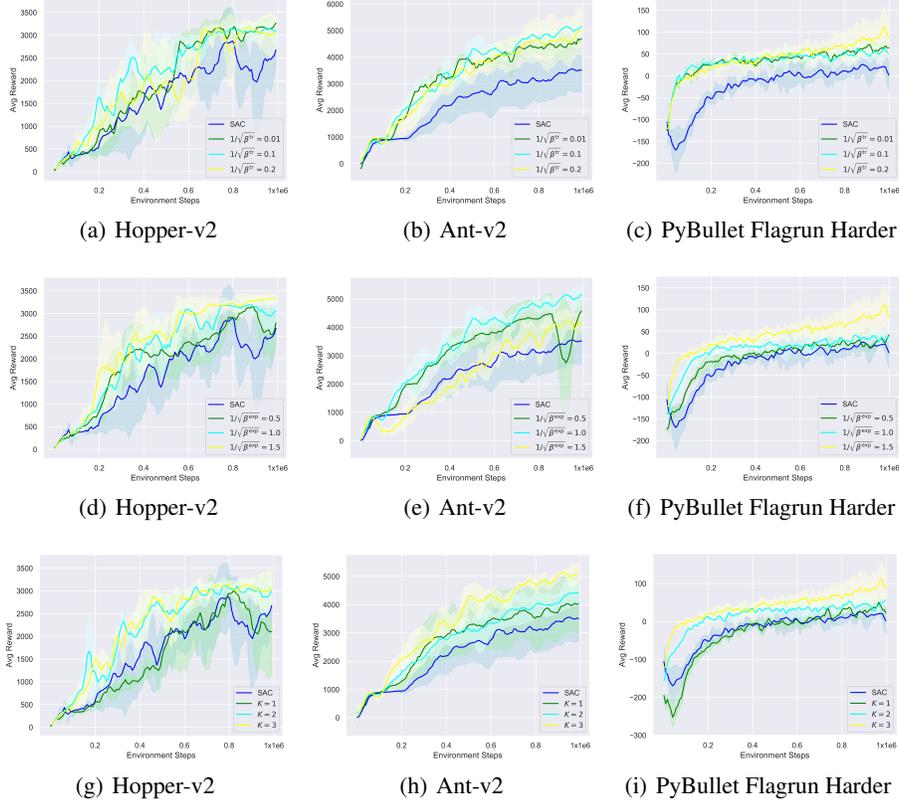


Figure 5: Sensitivity analysis on K , β^{exp} , and β^{tr} .

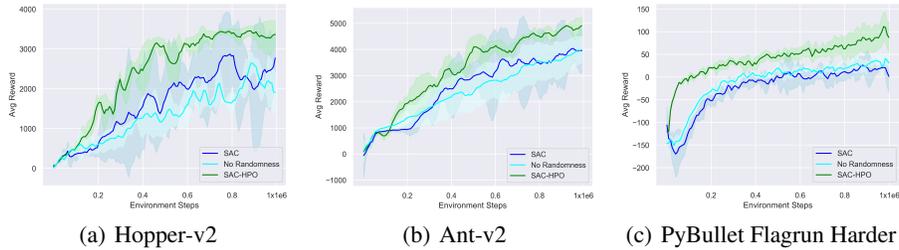


Figure 6: Performance comparison on the randomness of momentum vector. "No Randomness" refers to SAC-HPO without using random momentum vectors which is to set $\rho = 0$.

[26]. The baseline here is the SAC-HPO without random momentum vector, where we set $\rho = 0$ in both exploration and policy training. We can see that if no randomness in momentum vectors, the performance of SAC-HPO degrades significantly. Previous work such as IAPO [26] directly uses gradients to update the actions sampled from the base policy, without using any extra random variables, which is same as the baseline here. However, their performance is not good in high-dimensional environments [26]. The comparison in Figure 6 can explain the reason and show the advantage of our method. And similar advantage of our method can be observed in other environments as well.

C.4 Visualization of Policy Distribution

In Figure 7, we visualize the action distributions of Hamiltonian policy (actions evolved by HMC) at different environmental steps, where x and y axes represent two different action dimensions. Specifically, in Figure 7, the red dots represent 1000 actions sampled from the policy distribution evolved by leapfrog steps (13). For comparison, the blue dots represent 1000 actions sampled from

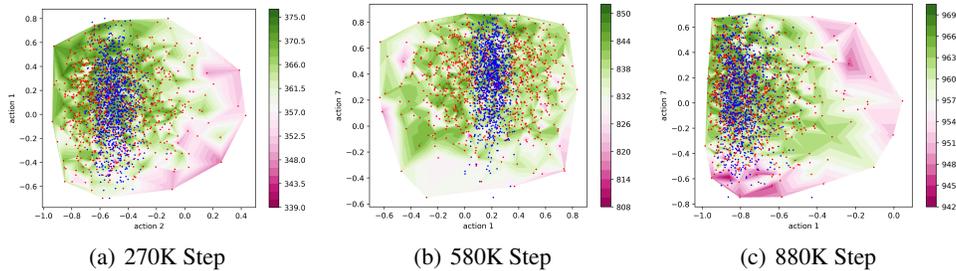


Figure 7: The shape of policy distribution in Ant-v2. The dimensions of action are shown as x and y labels. The color bar is for Q values. $\beta_0^{\text{exp}} = 1$ for every step.

the base policy network π_θ directly, which are Gaussian and are not evolved by HMC. The contour of Q values is shown as background for reference, which is drawn by triangular interpolation method. In Figure 7, comparing red and blue dots, we can see that HMC can evolve actions sampled from the base policy more towards regions with higher Q values, making sampled actions more directionally informed and hence improving exploration efficiency. We can also observe that policy distribution evolved by leapfrog operators can be highly non-Gaussian and have larger variance with much broader effective support. Besides, there are still some actions evolved to regions with similar or lower Q values, so that a reasonable trade-off of exploration and exploitation can be reached. That is why the exploration of RL agent can be boosted by Hamiltonian policy and the learning performance can be improved.