

A APPENDIX: LEARNING DIVERSE AND EFFECTIVE POLICIES WITH NON-MARKOVIAN REWARDS

A.1 DDM PESUDO CODE

Here we present the pseudocode for DDM. DDM employs a diversity matrix to quantify policy diversity and includes the determinant of diversity matrix to policy objective function. To improve the efficiency of the algorithm, M learners (i.e., M diverse policies) execute parallelly in different copies of the same environment and share a common replay buffer.

Algorithm 1 DDM

Require: number of learners (i.e., policies) M , maximum time steps in an episode T , size of batch N , max iteration Q , period to train policy representation module U

Initialize: M policies $\{v_m\}_{m=1}^M$, pseudo task function ϕ , environments $\{E_1, \dots, E_m\}$, local experience replay buffer $\{D_1, \dots, D_m\}$, shared replay buffer D

```

1: while  $q < Q$  do
2:   for  $m = 1, 2, \dots, M$  in parallel do
3:     if  $t < T$  then
4:        $a_m^t \sim \pi_m(s_m^t)$ 
5:        $r_m^t, s_m^{t+1} \sim E_m(s_m^t, a_m^t)$ 
6:       Store experience  $(s_m^t, a_m^t, r_m^t, s_m^{t+1})$  to the local replay buffer  $D_m$ 
7:       if environment  $E_m$  done then
8:         Store trajectory  $\tau_m = \{s_m^t, a_m^t, r_m^t\}_{t=1}^T$  to the shared replay buffer  $D$ 
9:       end if
10:    end if
11:    if  $q \bmod U == 0$  then
12:      Sample batch of trajectories  $\{\tau_m^i\}_{i=1, m=1}^{N, M}$  from shared replay buffer  $D$ 
13:      Obtain pseudo labels  $\{y_m\}_{m=1}^M$  using Eq. (3)
14:      Predict pseudo labels  $\{\xi_{i,m}\}_{i=1, m=1}^{N, M} = \phi(\{\tau_m^i\}_{i=1, m=1}^{N, M})$ 
15:      Update  $\phi$  by minimizing  $\mathcal{L} = -\frac{1}{N} \frac{1}{M} \sum_{i=1}^N \sum_{m=1}^M (y_m^T \log(\xi_{i,m}))$ 
16:    end if
17:    Randomly select one trajectory  $\tau_m$  for each policy  $\pi_m$ 
18:    Obtain policy embeddings  $\{v_m\}_{m=1}^M$  using Eq. (5)
19:    Stack policy embeddings  $\{v_m\}_{m=1}^M$  into a matrix  $V$ 
20:    Obtain diversity Matrix  $\mathbf{S} = F(V)$  by the Definition 3.1
21:    Compute Policy Diversity  $\text{Div}(\{v_m\}_{m=1}^M) = \det(\mathbf{S})$ 
22:    Update policies with policy diversity objective function:  $\tilde{J}(\pi_m) = (1 - \beta)J(\pi_m) + \beta \text{Div}(\{v_m\}_{m=1}^M)$ 
23:  end for
24: end while

```

A.2 ADDITIONAL EXPERIMENT DETAILS

A.2.1 VISUALIZATION OF FROZENLAKE-v1

We set the number of policies to be 5 for our method and all the baseline methods, and visualize the trajectories of these policies in the FrozenLake-v1 environment (see Fig. 5). The environment is an 8×8 grid with 28 holes, where square S in upper left corner and square G in lower right corner represent the start position and target position, respectively. Squares marked with H represent holes, and blank squares represent ice. We use different colors of polylines to represent routes of different policies.

As shown in Fig. 5, all policies of our method reach the target in the FrozenLake-v1 environment. Besides, our DDM method learned three types of routes to reach the target, including the most tortuous route in the middle. P3S-DQN only learns one type of route to the target, and one of the policies falls into a hole. This demonstrates the P3S-DQN can only learn similar policies and can not properly handle non-Markovian rewards. QD-DQN also only learns one type of route to the target. Although QD-DQN explores the other two kinds of routes, all these policies fail to reach the target and fall into holes. DvD-DQN learns two types of routes to the target, and a policy still drives the agent to a hole. Only a part of the policies in the policy sets of DvD-PPO, QD-PPO, and P3S-PPO can reach the target, and these policies can only generate one or two types of routes. In comparison, DDM-DQN can explore three types of routes, which demonstrates the effectiveness of our method in learning diverse policies.

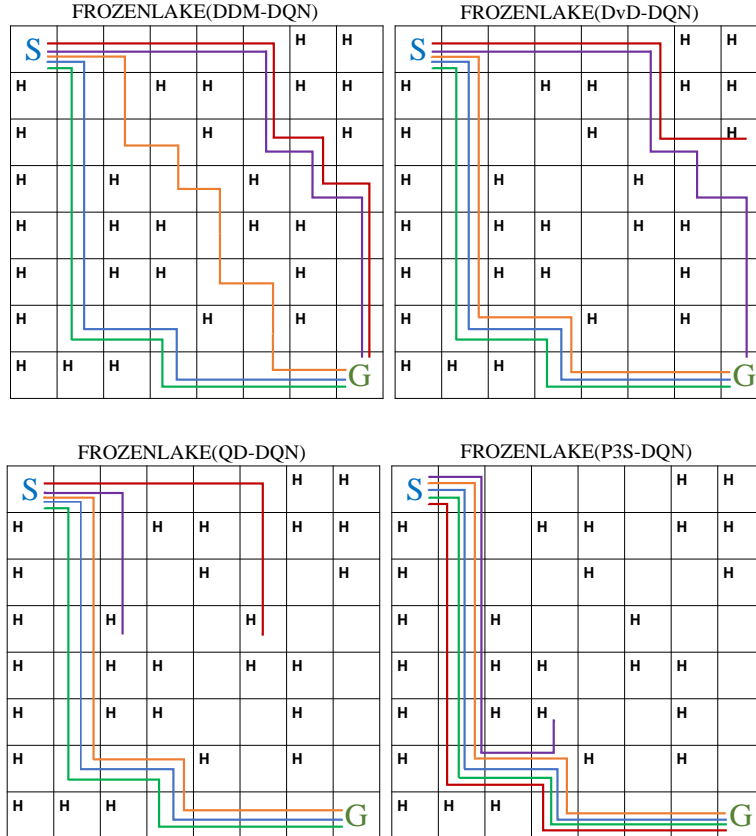


Figure 5: Trajectory visualization of our proposed DDM and baselines after 4000 training episodes in the FrozenLake-v1 environment with a 8×8 grid and 28 holes.

A.2.2 ABLATION STUDIES

In this section, we conduct an ablation study to provide insights into the relative contribution of our policy representation module.

The contribution of our policy representation module. One of the crucial parts of this work is the policy representation module. To demonstrate its importance, we replace it with two policy embedding methods: auto-encoder and behavioral embedding. The auto-encoder method takes as input the trajectories generated by π_m and outputs the embedding for π_m . The behavioral embedding method is an action-based behavior characterization method. Following the implementation in (Parker-Holder et al., 2020b), we randomly select 20 states and concatenate the actions of the selected states into the policy embedding.

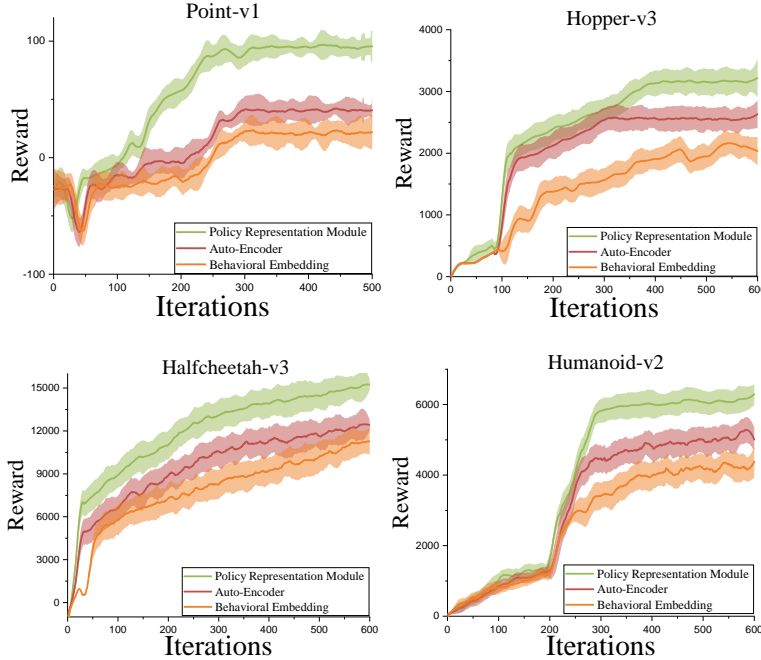


Figure 6: Performance of DDM with our policy representation module, auto-encoder and behavior embedding.

As shown in Fig. 6 and Table 3, using the policy embeddings generated by the auto-encoder method and the behavioral embedding method will significantly reduce the performance of DDM. The behavior embedding method can not extract the mutual dependencies between states and actions, and the auto-encoder can not properly characterize the long-horizon dependencies of states and actions. Moreover, both of these two methods can not directly use the pseudo tasks to embed policies, which also leads to their deficiencies in performance. The pseudo task makes the algorithm obtain high rewards in every case.

Table 3: The number of times to reach the target of DDM with our policy representation module, auto-encoder and behavior embedding in three FrozenLake-v1 environments.

Environment	Pseudo Task	Auto-Encoder	Behavior Embedding
FrozenLake-v1 4×4 (4holes)	2175	1257	848
FrozenLake-v1 5×5 (10holes)	1895	808	653
FrozenLake-v1 8×8 (28holes)	1040	688	301

Influence of Policy set size M The policy set size M refers to the number of policies used to construct the *Diversity Matrix*. If the policy set size is too small, it is less likely to find most of the

optimal policies with diverse behaviors. If the policy set size is too large, some policies with similar behavior will be selected and updated, harming the efficiency of DDM. Thus, it is important to select a proper policy set size M .

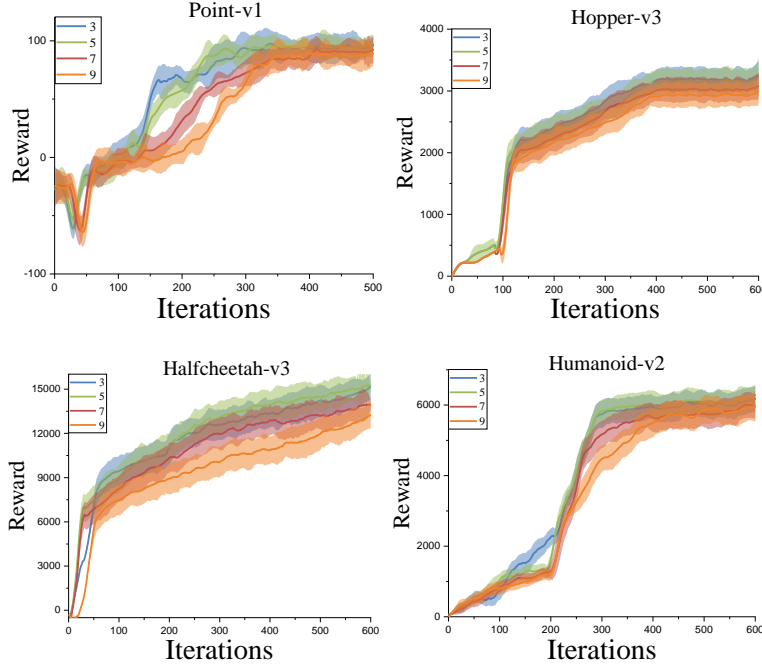


Figure 7: Performance of DDM with different policy set sizes M .

Fig. 7 and Table 4 shows the performance of DDM with different policy set sizes. We report the results under the same number of iterations. It can be observed that the DDM can achieve better performance than the baseline methods when M takes 3, 5, and 7, which shows our method is not quite sensitive to the policy set size.

Table 4: The number of times to reach the target of DDM with different policy set sizes M in three FrozenLake-v1 environments.

Environment	M = 3	M = 5	M = 7	M = 9
FrozenLake-v1 4×4 (4holes)	2243	2175	2120	2092
FrozenLake-v1 5×5 (10holes)	1812	1895	1862	1809
FrozenLake-v1 8×8 (28holes)	1021	1040	1012	995

A.2.3 HYPERPARAMETERS

Our method combine with different reinforcement learning methods (PPO, DQN, TD3) in the experiments. For all method, Adam id used as the gradient optimizer. In every experiment, out method trains policy embeddings. The hyperparameters and details are reported below.

Policy embedding In Table 5 we show the hyperparameters of the policy representation module of DDM. Three FrozenLake-v1 environments use the same settings. Since the state dimensions of the Point-v1 environment and the Hopper environment are smaller than other environments, the embedding dimensions and attention depths of these two environments are also set to be smaller.

Table 5: Policy embedding parameter configurations for every environment.

Hyperparameters	Point-v1	FrozenLake-v1	Humanoid-v2	HalfCheetah-v3	Hopper-v3
embedding dim	30	125	120	85	40
attention heads	4	8	8	8	4
learning rate	3e-4	3e-4	3e-4	3e-4	3e-4
attention depth	4	6	6	6	4
dropout	0.1	0.1	0.1	0.1	0.1

DDM-PPO In Table 6 we show the hyperparameters used in PPO when combined with DDM. The networks for policy and critic have two hidden layers of dimension 64 . The non-linearity function of the hidden layers is tangent.

Table 6: Parameter configurations for the Point-v1 environment.

Hyperparameters	Point-v1
batch size	2048
minibatch size	256
λ	0.97
γ	0.995
learning rate	3e-4

DDM-DQN In Table 7 we show the hyperparameters used in DQN when combined with DDM. Three FrozenLake-v1 environments used the same settings. The network has a hidden layer of dimension 10, and the non-linearity function of the hidden layers is ReLU.

Table 7: Parameter configurations for three FrozenLake-v1 environments.

Hyperparameters	FrozenLake-v1
learning rate	0.001
γ	0.9
epsilon greedy	0.9
batch size	32

DDM-TD3 In Table 8 we show the hyperparameters used in TD3 when combined with DDM. The networks for two Q-functions and the policy have 2 hidden layers. Both the first and second layers have a dimension of 256. The non-linearity function of the hidden layers is ReLU. The last layer of the actor network has a tangent activation function, whereas the last layer of the critic networks has a linear activation function.

Table 8: Parameter configurations for the three continuous control environments.

Hyperparameters	Humanoid-v2	HalfCheetah-v3	Hopper-v3
learning rate	3e-4	3e-4	3e-4
γ	0.99	0.99	0.99
batch size	256	256	128
target noise	0.2	0.2	0.2
buffer size	1e6	1e6	1e6

A.3 THEORETICAL RESULTS

A.3.1 PROOF OF THEOREM 3.1

Proof. Recall that $\text{Div}(\{v_m\}_{m=1}^M) = \det(\mathbf{S})$. Since \mathbf{S} is positive definite, according to Hadamard's inequality (Hadamard, 1893), the following bounds hold:

$$0 < \text{Div}(\{v_m\}_{m=1}^M) = \det(\mathbf{S}) \leq \prod_{i=1}^K s_{ii} = \Lambda, \quad (9)$$

where s_{ii} is the (i, i) -th element of \mathbf{S} . Let $\{\tilde{\pi}_m\}_{m=1}^M$ be a set of policies with at least one suboptimal policy and $\{\pi_m\}_{m=1}^M$ be a set of optimal policy. We suppose optimal policy π_m achieves a cumulative reward of R and suboptimal policy $\tilde{\pi}$ achieves a cumulative reward of $R(\tilde{\pi})$ with $R(\tilde{\pi}) + \Delta < R$ for some $\Delta > 0$, the following formula holds:

$$(1 - \beta) \sum_{m=1}^M \tilde{J}(\tilde{\pi}_m) + \beta \text{Div}(\{\tilde{v}_m\}_{m=1}^M) \leq (1 - \beta)MR - (1 - \beta)\Delta + \beta\Lambda. \quad (10)$$

For the set of optimal policies $\{\pi_m\}_{m=1}^M$, we have:

$$(1 - \beta) \sum_{m=1}^M \tilde{J}(\pi_m) + \beta \text{Div}(\{v_m\}_{m=1}^M) \geq MR. \quad (11)$$

From 10 and 11, if $\Delta > \frac{\beta}{1-\beta}\Lambda$, we have:

$$(1 - \beta) \sum_{m=1}^M \tilde{J}(\tilde{\pi}_m) + \beta \text{Div}(\{\tilde{v}_m\}_{m=1}^M) < (1 - \beta) \sum_{m=1}^M \tilde{J}(\pi_m) + \beta \text{Div}(\{v_m\}_{m=1}^M). \quad (12)$$

Since $\text{Div}(\{v_m\}_{m=1}^M) > 0$, there exist at least M distinct solutions, we conclude that whenever $\Delta > \frac{\beta}{1-\beta}\Lambda$, the objective in Eq. (8) can only be maximized when all policies are optimal. \square

A.4 EXPLANATION FOR GENERALIZED VARIANCE

Consider M policy embeddings $v = (v_m)_{m=1}^M$ with dimension k . We stack these vectors as a matrix:

$$V = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1K} \\ v_{21} & v_{22} & \cdots & v_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ v_{M1} & v_{M2} & \cdots & v_{MK} \end{bmatrix}. \quad (13)$$

We then construct the covariance matrix \mathbf{S} of $(v_m)_{m=1}^M$, where (i, k) -th element is:

$$s_{ik} = \frac{1}{M-1} \sum_{m=1}^M (v_{mi} - \bar{v}_i)(v_{mk} - \bar{v}_k), \quad (14)$$

where $\bar{v}_i \triangleq \frac{1}{M} \sum_{m=1}^M v_{mi}$, $1 \leq i \leq K$. The covariance matrix \mathbf{S} is denoted as:

$$\mathbf{S} = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1K} \\ s_{12} & s_{22} & \cdots & s_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ s_{1K} & s_{2K} & \cdots & s_{KK} \end{bmatrix}. \quad (15)$$

The determinant of \mathbf{S} is called generalized variance (Wilks, 1932), which measures the degree of scatter of the sample data. The larger the determinant of \mathbf{S} is, the more dispersed the data points are. Next we explain it in terms of both entropy and geometry.

From the perspective of entropy, we assume the distribution of the policy embeddings follow the Gaussian distribution, and their differential entropy is formulated as:

$$H = \frac{K}{2}(1 + \ln(2\pi)) + \frac{1}{2} \ln(\det(\mathbf{S})), \quad (16)$$

where K is the dimension of the space. We can observe the entropy of the policy embeddings monotonically increases with the determinant of \mathbf{S} . According to the nature of entropy, the larger the determinant of \mathbf{S} is, the larger H is, which means that the policy embeddings are more dispersed in the policy space.

From a geometric perspective, the generalized variance of the samples is proportional to the volume of the ellipsoid (Anderson, 1962; Johnson et al., 2014). Recall that $\bar{v} = (\bar{v}_1, \bar{v}_2, \dots, \bar{v}_K)$ are the means of the rows of \mathbf{V} . Then a hyperellipsoid (an ellipse if $K = 2$) centered at \bar{v} is defined as

$$\{v : (v - \bar{v})' \mathbf{S}^{-1} (v - \bar{v}) = c^2\}, \quad (17)$$

where $v \triangleq [v_1, v_2, \dots, v_K]$ are the coordinates of a data point in the hyperellipsoid. Then the volume of this hyperellipsoid is expressed as:

$$\text{Volume of } \{v : (v - \bar{v})' \mathbf{S}^{-1} (v - \bar{v}) \leq c^2\} = a_x (\det(\mathbf{S}))^{\frac{1}{2}} c^x \quad (18)$$

where a_x is a constant scalar. We can observe that the volume of the hyperellipsoid monotonically increases with $\det(\mathbf{S})$. In other words, $\det(\mathbf{S})$ can be used to measure the volume of policy embedding space.