This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2024.3455257

Diversifying Policies with Non-Markov Dispersion to Expand the Solution Space

Bohao Qu*, Xiaofeng Cao*, *Member, IEEE*, Yi Chang, *Senior Member, IEEE,* Ivor W.Tsang, *Fellow, IEEE* and Yew-Soon Ong, *Fellow, IEEE*

Abstract—Policy diversity, encompassing the variety of policies an agent can adopt, enhances reinforcement learning (RL) success by fostering more robust, adaptable, and innovative problem-solving in the environment. The environment in which standard RL operates is usually modeled with a Markov Decision Process (MDP) as the theoretical foundation. However, in many real-world scenarios, the rewards depend on an agent's history of states and actions leading to a non-MDP. Under the premise of policy diffusion initialization, non-MDPs may have unstructured expanding solution space due to varying historical information and temporal dependencies. This results in solutions having non-equivalent closed forms in non-MDPs. In this paper, deriving diverse solutions for non-MDPs requires policies to break through the boundaries of the current solution space through gradual dispersion. The goal is to expand the solution space, thereby obtaining more diverse policies. Specifically, we first model the sequences of states and actions by a transformer-based method to learn policy embeddings for dispersion in the solution space, since the transformer has advantages in handling sequential data and capturing long-range dependencies for non-MDP. Then, we stack the policy embeddings to construct a dispersion matrix as the policy diversity measure to induce the policy dispersion in the solution space and obtain a set of diverse policies. Finally, we prove that if the dispersion matrix is positive definite, the dispersed embeddings can effectively enlarge the disagreements across policies, yielding a diverse expression for the original policy embedding distribution. Experimental results of both non-MDP and MDP environments show that this dispersion scheme can obtain more expressive diverse policies via expanding the solution space, showing more robust performance than the recent learning baselines.

Index Terms—Policy diversity, non-Markov Decision Process, reinforcement learning, solution space, policy dispersion.

1 Introduction

Reinforcement learning (RL) has achieved great success in learning effective policies for a given task, including board games [1], [2], poker games [3], [4], video games [5], [6], [7], [8], [9], autonomous control [10], [11], [12], [13], and robotic manipulation [14], [15], [16], [17]. Policy diversity significantly contributes to the success of reinforcement learning (RL) [18], [19], referring to the variety or heterogeneity of policies that an agent or a set of agents can adopt during the learning process. It also encompasses the range of policies an agent can utilize within the same environment. This diversity leads to more robust and generalizable solutions, as exposure to various policies enables an agent to adapt to different situations and environments, enhancing its resilience to changes and uncertainties.

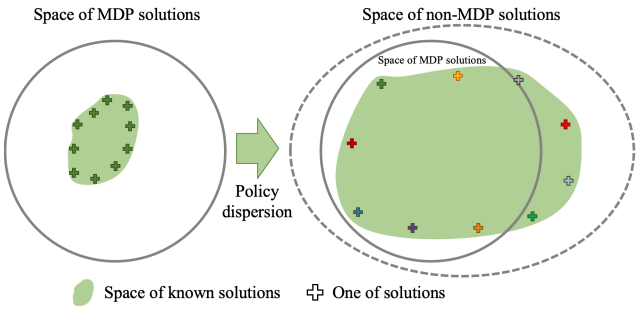


Fig. 1: Non-MDPs exhibit unstructured expanding solution space. The left circle represents the space of the MDP solutions, and the middle green area in the circle represents the coverage of consistent solutions (green plus signs) in the solution space. Non-MDPs may have unstructured expanding solution space due to varying historical information and temporal dependencies. The dashed ellipse on the right represents the expanded solution space of non-MDP solutions, and the green area in the ellipse represents the coverage of a series of dispersed different solutions (multiple color plus signs) in the space after policy dispersion.

The environment in which standard RL operates is usually modeled with a Markov Decision Process (MDP) as the theoretical foundation [20], [21]. An MDP is limited by the Markov property that a reward only depends on the current state and action. In many real-world problems, the rewards depend on the agent's visited states and performed actions. In such environments, the decision process violates

- *Bohao Qu, Xiaofeng Cao, and Yi Chang are with the School of Artificial Intelligence, Jilin University, Changchun, Jilin 130012, China, and also with the Engineering Research Center of Knowledge-Driven Human-Machine Intelligence, Ministry of Education, China. Yi Chang is also with the International Center of Future Science, Jilin University, Changchun, Jilin, 130012, China. E-mail: qubohao@126.com, {xiaofengcao, yichang}@jlu.edu.cn.*
- *Ivor W. Tsang are with the Institute of High Performance Computing (IHPC) and Centre for Frontier AI Research (CFAR), Agency for Science, Technology and Research (A*STAR), Singapore. He is also with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. E-mail: Ivor_Tsang@cfar.a-star.edu.sg.*
- *Yew-Soon Ong is with Nanyang Technological University, Singapore. E-mail: asysong@ntu.edu.sg. He is also with A*STAR Centre for Frontier AI Research, Singapore. E-mail: Ong_Yew_Soon@hq.a-star.edu.sg.*
- *Bohao Qu and Xiaofeng Cao contributed equally to this work.*
- *Xiaofeng Cao and Yi Chang are the corresponding authors.*

Manuscript received April 19, 2005; revised August 26, 2015.

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on September 19,2024 at 05:05:22 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2024.3455257

the Markov property leading to a non-Markov Decision Process (non-MDP) [22], [23], [24], [25], [26] where the reward has a temporal nature [27], that the policy receives its rewards for complex, temporally-extended behaviors sparsely. Under the premise of policy diffusion initialization, non-MDPs may have unstructured expanding solution space [1] due to varying historical information and temporal dependencies. This results in solutions having non-equivalent closed forms in non-MDPs.

Proposal/Motivation We thus seek to learn diverse policies in non-Markov environments. Deriving diverse solutions for non-MDPs requires policies to break through the boundaries of the current solution space through gradual dispersion. The goal is to expand the solution space, thereby obtaining more diverse policies, as shown in Fig. 1. This means that diverse policies should cover the solution space as much as possible. To this end, an effective approach is to learn representations of policies and ensure that these representations sufficiently diffuse throughout the space, thereby inducing diverse policies. Initially, the candidate policies of the agents typically start from similar initial states, with their embeddings clustered in a unified region of the policy embedding space. As the diffusion process unfolds, these embeddings are repeatedly reconstructed during policy updates, forming distinct diffusion paths. Ultimately, by maximizing the diffusion divergence of the dynamic policy update trajectory, a diverse set of diffusion trajectories is achieved, as illustrated in Fig. 2.

Technical statement Motivate by the above dispersion perspective, this paper proposes **D**iscovering **D**iverse **P**olicy via **E**mbedding **D**ispersion (D2PED) method, which can efficiently learn high-quality policies with diverse behaviors in non-Markov environments. Specifically, we design a policy dispersion scheme to disperse policy embeddings along different trajectories as the policy update progresses. Then the policy embeddings are employed to construct a *dispersion matrix*, which is used to measure the diversity of the candidate policies and guides the update of the policies. The key to learning effective policy embeddings in non-Markov environments lies in capturing essential historical information through the integration of past states and actions. Modeling trajectories, sequences of states and actions, is the most direct method to capture long-horizon dependencies for histories, which has been studied in the context of game theory [28], [29], [30] and non-Markov tasks [31], [32], [33]. In this context, we identify a strong isomorphism with the approach used by the transformer [34] to model trajectories and obtain policy embeddings. The transformer's self-attention mechanism is adept at capturing long-range dependencies and can adaptively prioritize historical information crucial for specific task demands. This capability enables the transformer to dynamically adjust its dependency on historical data according to varying scenarios, providing substantial flexibility that is particularly advantageous in non-Markov environments. Moreover, in non-Markov environments, the agent receives its reward sparsely for complex actions over a long period [27], which is not conducive to training the transformer-based policy representation model. We construct a sample categorical distribution to sample higher cumulative reward trajectories with a higher probability. We also give the theoretical conditions such that the policy diversity will not reduce their effectiveness.

1. Note that the solution space is an approximate concept of the policy diversity exploration space, which inspires us to learn diverse policies in non-Markovian environments.

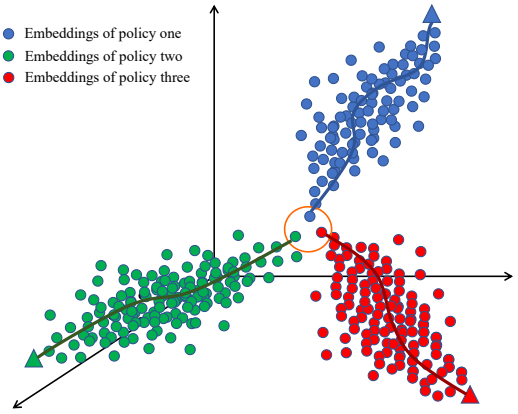


Fig. 2: Policy embeddings dispersion. There are three dispersion trajectories over different policy embeddings, where their initial positions are marked by a red circle. In the dispersion process, the embeddings are repeatedly constructed as the policy update progresses, forming three different dispersion trajectories with direction guidelines.

Contributions Our major contributions are summarized as follows:

- **New Insight.** Under the premise of policy diffusion initialization, non-MDPs may have unstructured expanding solution space due to varying historical information and temporal dependencies. This results in solutions having non-equivalent closed forms in non-MDPs.
- **New Scheme.** Motivated by our new insight, we design a policy dispersion mechanism to discover diverse policies in non-Markov environments. We incorporate the historical information into the policy representations and use the representations to construct a diversity measure. This diversity measure guides the policies to disperse in different directions, enabling the exploration of the unknown boundaries of the expanded solution space in non-Markov environments so that diverse policies can cover the solution space as much as possible.
- **Effective Analysis.** Our method proves that the *dispersion matrix* is positive definite (refer to Proposition 4.1), thereby ensuring the existence of diversity among the policies. Furthermore, by maximizing the objective function, we facilitate the achievement of a group of diverse, approximate optimal solutions.
- **Efficient Performance.** With our effective module design and theoretical analysis, our policy diversity learning scheme can be combined with any off-policy reinforcement learning algorithm, and the experimental results of both non-MDP and MDP environments show that this dispersion scheme can obtain more expressive diverse policies via expanding the solution

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on September 19,2024 at 05:05:22 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2024.3455257

space, showing more robust performance than the recent learning baselines.

The rest of this paper is organized as follows. In Section 2, we introduce the related work. In Section 3, we introduce the preliminary knowledge about the non-Markov decision process and policy diversity learning. The methodology of policy dispersion is presented in Section 4, where 4.1 presents the policy representation module which is employed to generate effective and diverse embeddings for different policies, and 4.2 describes the process of dispersing the dynamically updating trajectories over embedding reconstruction to learn diverse policies. In Section 5, we conduct various experiments to demonstrate the effectiveness and robustness of D2PRD. We conclude this paper in Section 6.

2 Related Work

In this paper, we discuss the related work in three aspects: from non-Markovian task scenarios to the specific tasks where we learn diverse policies, and the crucial technical approach of learning policy representations for policy diversity.

Non-Markov Environment. In a standard Reinforcement Learning (RL) problem setting of an MDP, rewards depend only on the most recent state-action pair. In a non-Markov reward decision process [35], rewards depend on the history of state-action pairs. Building in temporal logics over finite traces, [26] adopt linear dynamic logic on finite traces for specifying non-Markov rewards and provide an automaton construction algorithm for building a Markovian model. In another paper [36], the authors are concerned with both the specification and effective exploitation of non-Markov rewards in the context of MDPs. They specify non-Markov reward-worthy behavior in Linear time Temporal Logic (LTL). Similarly, [37] use truncated LTL as a reward specification language, and [38] use LTL_f to specify desired complex behavior. Because temporal formulas are evaluated over an entire trace, it is difficult to guide the RL agent locally toward desirable behaviors. Data-driven approaches to learning non-MDP problems [39] often make use of domain-specific propositions and temporal logic operators [40], [41], [42]. The work [28] introduces the indispensable role of game theory in understanding decision-making in scenarios involving multiple entities, where decisions are dependent on historical information. Another work [30] delves into the pivotal role of information structures in Multi-Agent Learning emphasizing the strategic decision-making processes among agents within dynamic environments. The core focus is on how agents adapt and make optimal decisions based on their historical experiences which interactions with other agents. A mainstream approach addresses the challenge of extending multi-agent reinforcement learning to complex real-world problems [29], which by the agents adjust their policies based on historical action trajectories.

Learning diverse policies. Our method can be grouped into this category. A series of existing Quality-Diversity (QD) [18] related evolutionary algorithms have achieved good performance in exploring diverse behaviors and diverse policies [43], [44], [45], [46], [47], [48], [49]. The MAP-Elites algorithms [50], [51] solve this problem by discretizing the behavior description space into a grid of cells. Some theoretical conclusions are also developed to ensure that diverse policies are not obtained by sacrificing their effectiveness. The DvD [19] algorithm proves that under tabular MDP, multiple distinct optimal solutions can be obtained by maximizing the proposed loss function. The ridge rider algorithm [52] proposes to use the eigenvectors of the Hessian matrix to discover diverse local optima with theoretical guarantees. [53] theoretically shows that maximizing the diversity metric based on the decision point process can guarantee to enlarge the convex polytopes spanned by the policies of agents. The work [54] proposes a method that generates a set of diverse and well-performing policies by measuring the differences between policies through a new metric and employing constrained optimization techniques. Some Reinforcement Learning (RL) based methods have been developed to explore diverse behaviors [54], [55], [56]. The GEP-PG algorithm [57] uses Goal Exploration Processes [58] to generate diverse policies and combines them with the Deep RL algorithm DDPG [8], which performs well in continuous control tasks. The RSPO [59] explores diverse policies by solving a filtering-based objective, which restricts RL policies from converging to a solution that differs from a set of local optimal policies. P3S-TD3 [60] method periodically determines the best policy among all learners and assigns the best policy parameters to all learners so that the learner can search for a better policy under the guidance of the best policy.

Policy representation learning. A generative method is proposed in [61], which proposed an encoder-decoder method for modeling the agent's policy. The encoder learns a point-based representation of different agent trajectories, and the decoder learns to reconstruct the modeled agent's policy. Two meta-learning methods are proposed in [62], [63], and they both regard the latent generative representation of learning model parameters as the policy representation, and the method in [63] shows more stable performance. [64] proposed relational forward models to model agents using graph neural networks. [65] uses a VAE for agent modeling for fully-observable tasks. [66] proposed the Theory of mind Network (TomNet), which learns embedding-based representations of modeled agents for meta-learning.

3 Preliminaries

In this section, we present the necessary background relevant to the problem setting of this work.

Markov Decision Process (MDP) is a mathematical framework to describe an environment in reinforcement learning. An MDP is a tuple $\mathcal{M} = \langle S, A, P, R, \gamma \rangle$, where S and A represent state space and action space, respectively. The state transition dynamic function is given by $P : S \times A \to S$, which is a mapping from the current state $s \in S$ to the next state $s' \in S$. The reward function is given by $R : S \times A \to \mathbb{R}$, mapping from state $s \in S$ and action $a \in A$ to reward $r \in \mathbb{R}$. $\gamma \in [0, 1)$ denotes the discount factor.

A policy $\pi : S \to A$ is a mapping from S to A. A trajectory $\tau \in \mathcal{T}$ is a sequence of state-action pairs, $\tau = ((s_0, a_0), ..., (s_{T-1}, a_{T-1}))$. In deep RL, policy π is typically a neural network, encoded by parameter vectors θ, and the goal is to optimize parameters θ of π such that an agent equipped with policy π_θ in the environment described by a fixed MDP maximizes $R(\tau) = \sum_{t=1}^{T} \gamma^{t-1} r_t$, the expected cumulative reward of a trajectory τ over an

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on September 19,2024 at 05:05:22 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2024.3455257

episode time-step horizon T (assumed to finite). The typical objective function of policy π_θ is as follows:

$$J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]. \quad (1)$$

Non-Markov Decision Process [25], [35], [40], [67] extends MDP and assumes transition and reward functions depend on the history of state-action pairs. Formally, an non-MDP is a tuple $M = \langle S, A, tr, R^+, \gamma\rangle$, where S, A are as in an MDP, $tr : S^* \times A \times S \to \Pi(S)$ is the transition function, i.e., $tr((s_0, \ldots, s_k), a, s')$ is the probability of reaching state s' when executing action a given history $s_0, ..., s_k$. Here S^* is the set of finite, non-empty, state sequences. A non-Markov Reward function is a mapping from the finite history of states and actions to reward $\mathbb{R}$, denoted as $R^+ : (S \times A)^* \to \mathbb{R}$. In the following, for the convenience of representation, we will simplify π_{θ^i} into π_i.

4 METHODOLOGY

In this paper, to derive diverse solutions in non-MDP environments, the policies need to gradually diffuse and break through the boundary of the existing solution space, we model the policy update process as the policy embedding dispersion, deriving different paths to explore diverse policies. Furthermore, we decompose the objective of policy update into two parts: one part is the objective of policy optimization, which updates the policy towards the direction of obtaining the maximum expected cumulative reward, and the other part is the objective of policy diversity, which updates the policy towards the direction of diversity development. The two objectives provide gradients in two different directions for policy updates. The combination of the two gradients enables the policy update to not only obtain the maximum expected cumulative reward but also consider dispersing in different directions than other policies. The objective of the first part, policy optimization, can directly adopt the optimization objective of standard reinforcement learning algorithms. For the second part, we construct a policy diversity objective suitable for non-Markov environments.

To guide the development of diverse policies, an effective method is to learn the representations of the policies and then promote dispersion in the representation space, thereby encouraging policy diversity. In non-Markov environments, the key to learning effective policy embeddings is capturing the characteristics of non-MDP, namely, the embeddings need to capture reward-relevant historical information, and also consider the sparse supervision signals. We present a transformer-based policy representation module, which is capable of learning effective policy embeddings in non-Markov environments. Meanwhile, to avoid the lack of supervision signal problem caused by non-Markov rewards, we design a sample categorical distribution to sample trajectories for training the policy representation model. Further, we propose a policy dispersion scheme to disperse policy embeddings along different trajectories as the policy update progresses and use the embeddings to construct a dispersion matrix that guides the learning of diverse policies.

Our proposed method, D2PED (Discovering Diverse Policies via Embedding Dispersion), comprises two primary submodules: the policy representation module and the policy dispersion module. The schematic illustration of our method is shown in Fig. 3. In our method, M parallel learners learn M distinct policies and share a common replay buffer D. The M learners execute parallelly in different copies of the same environment and employ a common base algorithm which can be any off-policy RL algorithm.

4.1 Non-MDP Policy Representation Module

The policy representation module (shown in Fig. 3(b)) is employed to generate effective embeddings for different policies. Specifically, for the m-th ($m \in \{1, ..., M\}$) policy π_m, the representation module inputs a trajectory collected by π_m and outputs a policy embedding v_m for π_m.

4.1.1 Generative Non-MDP Policy Representations

Using trajectories to learn policy embeddings in non-Markov environments is effective and suitable, as the state-action pairs within the trajectories directly reflect the characteristics of the policy [61]. We leverage the architecture of the transformer [34], which is well-suited to model trajectories to generate policy embeddings. The transformer's self-attention mechanism is particularly proficient at detecting and interpreting long-range dependencies within these sequences. It effectively discerns the significance of various historical points, allowing it to adaptively prioritize information that is most relevant to the task at hand. This dynamic adjustment capability is crucial, as it enables the transformer to modify its focus on different segments of historical data based on the current scenario and the demands of the environment.

Such flexibility is invaluable in non-Markov settings where the relevance of historical information may shift dramatically across different states and actions. By using the transformer, we can ensure that our policy embeddings are not only reflective of the deep temporal structure of the environment's dynamics but are also responsive to the changing priorities and requirements of different tasks.

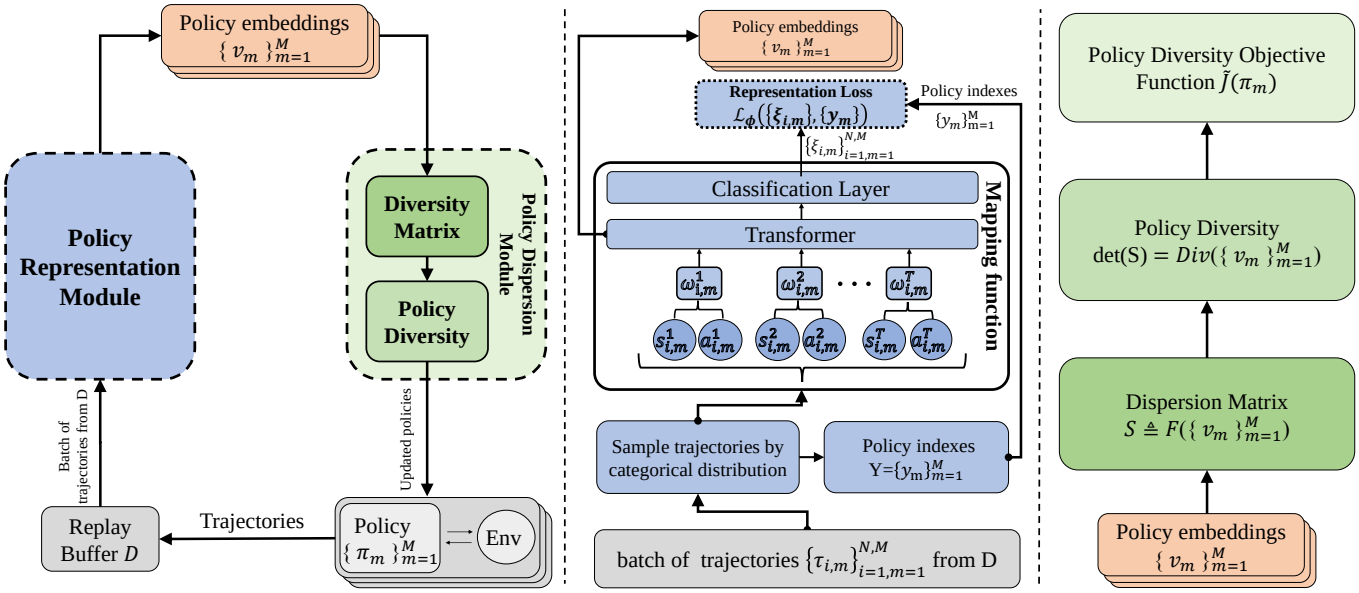
Specifically, we design the policy representation process as a mapping function $\mathcal{T} \to \mathcal{V} \to Y$ parameterized by ϕ, which is the combination of Eq. (2) and Eq. (3), where $\mathcal{T} = \{\tau_{i,m}\}_{i=1,m=1}^{N,M}$ denotes the set of trajectories collected by policies $\{\pi_m\}_{m=1}^{M}$, $\mathcal{V} = \{v_m\}_{m=1}^{M}$ is the policy embeddings, and $Y = \{y_m\}_{m=1}^{M}$ is the set of the policy indexes. The mapping function inputs trajectories and outputs policy embeddings which are the essential ingredients to learning diverse policies. Then, the embeddings are classified into policy indexes to train this module.

The transformer architecture can efficiently model sequential data. This model consists of stacked self-attention layers with residual connections. Each self-attention layer receives embeddings generated by tokens and outputs embeddings that preserve the input dimensions. For i-th T time-step horizon trajectory collected by policy π_m, $\tau_{i,m} = (s^1_{i,m}, a^1_{i,m}, ..., s^T_{i,m}, a^T_{i,m})$, we let $\omega^t_{i,m} = (s^t_{i,m}, a^t_{i,m})$ for each time step $t \in \{1, ..., T\}$ as a token. Subsequently, we regard $(\omega^t_{i,m})_{t=1}^T$ as a sequence, and input to the transformer encoder (i.e., a Layernorm (LN), a multiheaded self-attention (MSA) layer, and residual connections [68], [69]), to obtain the policy embedding:

$$v_{i,m} = \mathrm{MSA}(\mathrm{LN}(\omega^{\mathrm{t}}_{\mathrm{i,m}})^{\mathrm{T}}_{\mathrm{t=1}} + \mathrm{E}_{\mathrm{pos}}) + (\omega^{\mathrm{t}}_{\mathrm{i,m}})^{\mathrm{T}}_{\mathrm{t=1}}, \quad (2)$$

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on September 19,2024 at 05:05:22 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2024.3455257



(a) Overview (b) Policy Representation Module (c) Policy Dispersion Module

Fig. 3: **A schematic illustration of D2PED: (a)** The overall structure of D2PED. D2PED comprises the policy representation module and the policy dispersion module. M learners execute parallelly and store trajectories into a shared replay buffer D. **(b)** The policy representation module samples a batch of trajectories from replay buffer D and then constructs a sample distribution via Eq. (5) to select high-quality trajectories for training the module. The transformer encoder inputs the trajectories and outputs policy embeddings, then the embeddings are classified into policy indexes (classification labels) by the classification layer. **(c)** The policy dispersion module utilizes the policy embeddings to construct a dispersion matrix. The determinant of the diversity matrix is regarded as a policy diversity measure, which is further as a term of the policy diversity objective $\tilde{J}(\pi_m)$ (Eq. (6)).

where $v_{i,m} \in \mathbb{R}^K$ represents the i-th embedding for policy π_m and E_{pos} denotes the positional encoding. In our approach, T general represents the entire episode duration from the starting point to the final step of the agent's trajectory, using this full trajectory to effectively capture hidden patterns and temporal dependencies.

To train the transformer-based policy representation module, we model the problem as a trajectory classification task. First, we sample N trajectories for each of the M policies and shuffle all the trajectories. Then, we regard the policy indexes of trajectories $Y = \{y_m\}_{m=1}^{M}$ (the trajectories collected by which policy) as class labels, and assign the class labels to the policy embeddings. Specifically, $v_{i,m}$ are processed with a classification layer (CL), which is formulated by:

$$\xi_{i,m} = \mathrm{CL}(v_{i,m}), \tag{3}$$

where $\xi_{i,m}$ denotes a vector in which the j-th element ($j \in \{1, 2, ..., M\}$) represents the probability that the policy embedding $v_{i,m}$ is assigned to the j-th policy. Finally, we train the policy representation module by minimizing the loss function:

$$\mathcal{L}_\phi = -\frac{1}{N}\frac{1}{M}\sum_{i=1}^{N}\sum_{m=1}^{M}({y_m}^T \log(\xi_{i,m})), \tag{4}$$

where y_m is the policy index (classification label) of $\tau_{i,m}$. Trajectory sampling relies on the categorical distribution introduced in Section 4.1.2.

4.1.2 Sample Trajectories with Categorical Distribution

In non-Markov environments, rewards are usually sparse since they often involve delayed gratification, where the reward is only received after a series of correct actions are performed over time. The environment complexity also increases the difficulty for an agent to stumble upon the correct series of actions that lead to a reward, thus making the rewards sparse as they are less frequently encountered.

Non-Markov environments are characterized by dependencies on past states and actions. By focusing on higher-reward trajectories, the learned embeddings are more likely to capture the essential dynamics and dependencies that lead to successful outcomes. This focus ensures that the embeddings reflect characteristics of decisions that are more rewarded and embed the underlying policy features that achieve these outcomes. In particular, different policies have different decision preferences (certain actions in specific situations). Without a reward signal, it is difficult to identify the policy's preferences. Trajectories with higher rewards are more likely to reflect the policy's preferences.

To train the policy representation module with sparse supervision signals, we design a sample categorical distribution by the cumulative rewards of the trajectories to select high-quality (relatively high rewards) trajectories. Specifically, let $\tau_{i,m}$ be the i-th trajectory collected by policy π_m, and $r_{i,m}$ denotes the accumulated reward of $\tau_{i,m}$. We regard $r_{i,m}$ as a parameter and construct the sample categorical distribution,

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on September 19,2024 at 05:05:22 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2024.3455257

$$i_m \sim \text{Cat}\left(\left(\sigma(r_{i,m})^{\frac{1}{\epsilon}} / \sum_{j=1}^{N} \sigma(r_{j,m})^{\frac{1}{\epsilon}}\right)_{i=1}^{N}\right), \quad (5)$$

where σ is the sigmoid function, N is the batch size that sample trajectories from replay buffer D for policy m, and N is equal for each policy. A hyperparameter ϵ is used to adjust the "temperature" of the sampling distribution.

With the sample categorical distribution, trajectories with higher cumulative rewards will be sampled to train the model at a higher frequency, and M policies are sampled multiple times using the distribution to obtain samples for learning M policy embeddings. We regard the trajectory indexes sampled by Eq. (5) as the classification labels, which will be used to train the representation model by minimizing the loss Eq. (4) in Section 4.1.1.

The policy representation module is utilized to obtain N policy embeddings for each of the M policies. Then the policy embeddings are used to construct the diversity measure of the policy set and guide the policy dispersion into the boundary of the solution space.

4.2 Policy Dispersion Module

The candidate policies typically have consistent initializations, where their embeddings are concentrated in a uniform region. Subsequently, the policies are continuously updated during the optimization process, and the policy embeddings are repeatedly constructed along with the policy updates. The repeated construction process of policy embeddings for each policy forms a reconstruction path of policy embeddings. To obtain diverse policies, maximizing the disagreements in the reconstruction paths of policy embeddings is effective, that is, letting the reconstruction paths of policy embeddings for different policies extend in different directions in the representation space. This also means that policy embeddings are as dispersed as possible in the representation space. In this case, different policy representations on different dispersed paths can provide diversity-oriented update directions for policy updates. As shown in Fig. 2.

To maximize the disagreements in the reconstruction paths of policy embedding, we designed a policy dispersion module as shown in Fig. 3(c). The degree of dispersion of policy embeddings in the space will directly reflect the diversity of the policies. Further, the key is how to measure policy diversity. For this, we use policy embeddings to construct a *dispersion matrix* as a measure of policy diversity, guiding the policies to update toward the direction of diversity. Specifically, the *dispersion matrix* stacks the embeddings of the candidate policies, modeling the process trajectory dispersion with direction guidelines. Following the Determinantal Point Processes (DPPs) [70], the determinant of the dispersion matrix that proportionally draws the dimensional span over the associated geometric region of the matrix, could be specified as a quantification of the disagreement of dispersion trajectories, that is, characterizing the diversity of the candidate policies.

To construct the *dispersion matrix*, we randomly select an embedding for each of the M policies from $\{v_{i,m}\}_{i=1,m=1}^{N,M}$, constituting $\{v_m\}_{m=1}^{M}$. And stack the M selected embeddings as the *dispersion matrix*. Further, we propose the following definitions to construct the *dispersion matrix* and compute the policy diversity measure as follows:

Definition 4.1. ***(Dispersion Matrix)*** *Consider M policies $\{\pi_m\}_{m=1}^{M}$, and their embeddings are denoted as $\{v_m\}_{m=1}^{M}$, where $v_m \in \mathbb{R}^K$. Let $V \triangleq [v_1, ..., v_M]$, where $[\cdot]$ is the operator that stacks vectors into a Matrix. We define the Dispersion Matrix of policies as $\mathbf{S} \triangleq F(V)$, where $F(\cdot)$ is a function that transforms the $M \times K$ matrix V to a $K \times K$ positive-definite matrix.*

Definition 4.2. ***(Policy Diversity)*** *We define the diversity of policies as the determinant of their dispersion matrix, denoted as* $\mathrm{Div}(\{\mathrm{v_m}\}_{\mathrm{m=1}}^{\mathrm{M}}) = \det(\mathbf{S})$.

The matrix V constructed by Definition 4.1 is an $M \times K$ matrix. Suppose the number of candidate policies M equals the dimension K of the policy embeddings, i.e., $M = K$, the determinant of the square matrix V that represents the dispersion disagreement of dynamically updating trajectories over embedding reconstruction, could be a feasible measurement for the diversity of the M policies. However, for the scenarios of $M \neq K$, the function $F(\cdot)$ could be adopted to transform the polyhedron spanned by M K-dimensional embeddings to a parallelepiped spanned by a K-by-K positive-definite matrix in another embedding space. Then the determinant of the square matrix still could measure the diversity of the M candidate policies.

Now consider the D2PED total objective function $J(\Pi)$ of the candidate policies $\Pi = (\pi_1, ..., \pi_m)$, which seek to find a set of policies with both high rewards and diverse behaviors. The total objective function $J(\Pi)$ explicitly augments the RL objective function with an additional policy diversity term $\mathrm{Div}(\{\mathrm{v_m}\}_{\mathrm{m=1}}^{\mathrm{M}})$, as follows:

$$J(\Pi) = \left\{ \sum_{m=1}^{M} \left[(1-\beta) J(\pi_m) + \beta \mathrm{Div}(\{\mathrm{v_m}\}_{\mathrm{m=1}}^{\mathrm{M}}) \right] \right\} \quad (6)$$

where $\beta \in (0, 1)$ controls the trade-off between $J(\pi_m)$ and $\mathrm{Div}(\{\pi_\mathrm{m}\}_{\mathrm{m=1}}^{\mathrm{M}})$.

Applying our D2PED algorithm would lead to the following proposition, which implies that after sufficient policy dispersion, we will obtain a set of approximate optimal diverse policies.

Proposition 4.1. *Consider M policies for an environment characterized by finite non-MDP. Suppose optimal policy π^* achieves a cumulative reward of $R(\pi_m^*)$ and approximate-optimal policy $\tilde{\pi}$ achieves a cumulative reward of $R(\tilde{\pi})$ with $R(\tilde{\pi}) + \Delta < R(\pi_m^*)$ for some $\Delta > 0$. Since $\mathbf{S}$ is positive definite, according to Hadamard's inequality [71], the following bounds hold:*

$$0 < \mathrm{Div}(\{\mathrm{v_m}\}_{\mathrm{m=1}}^{\mathrm{M}}) = \det(\mathbf{S}) \leq \prod_{\mathrm{i=1}}^{\mathrm{K}} \mathrm{s_{ii}} = \Lambda, \quad (7)$$

where s_{ii} is the (i, i)-th element of $\mathbf{S}$, based on the objective Eq. 6, the following formula holds:

$$\sum_{m=1}^{M} \left[(1-\beta) R(\tilde{\pi}_m) + \beta \mathrm{Div}(\{\tilde{\mathrm{v}}_\mathrm{m}\}_{\mathrm{m=1}}^{\mathrm{M}}) \right] \quad (8)$$

$$< \sum_{m=1}^{M} (1-\beta) R(\pi_m^*) - (1-\beta) M\Delta + \beta M \Lambda. \quad (9)$$

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on September 19,2024 at 05:05:22 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2024.3455257

then maximizing the objective in Eq. (6) can yield a set of diverse approximate-optimal policies.

Remark 4.1. *From Proposition 4.1 and Eq. 7, we conclude that our method ensures the existence of diversity within the policies.*

Remark 4.2. *From Proposition 4.1 and Eq. 8, we conclude that by maximizing the objective function, we approximate the maximization of the bound, enabling the attainment of a group of diverse, approximate-optimal solutions.*

The proof of Proposition 4.1 is in Appendix A.1. According to Eq. 7, the diversity term is greater than 0, which provides a diversity gradient during the optimization of the objective equation, it inevitably leads to the occurrence of diversity, making it impossible to obtain a situation where the entire set consists of optimal solutions. Therefore, this upper bound is only a theoretical upper bound, and diversity must exist. Maximizing the objective equation approximates maximizing the bound, which can yield a set of diverse approximate-optimal solutions.

Our objective is to find multiple solutions to the problem, and the significance of this upper bound is to ensure the existence of policies that are close to optimal. This upper bound is relatively soft, providing a more precise constraint compared to the more rigid upper bound expressed as $\sum_{m=1}^{M} R(\pi_m^*) + \Lambda$. Therefore, approximating this upper bound allows us to achieve approximate-optimal diverse solutions. It's important to note that as the M policies become closer to the optimal solution, their diversity decreases. Thus, our upper bound also effectively balances the trade-off between achieving high-quality solutions and maintaining diversity among the policies.

A method to construct a dispersion matrix. The theory holds if the dispersion matrix is positive definite, and we give a method to construct a positive definite dispersion matrix. In general, we compute the covariance matrix of the policy embeddings and then regard the determinant of the covariance matrix as the diversity of policies in Definition 4.2. In statistical analysis, the determinant of the covariance matrix is termed the generalized variance [72], which is proportional to the sum of squares of the volumes of all the different parallelotopes formed by the policy embeddings using as principal edges [73], [74]. Note that the volume of the enclosed geometric region of those policy embeddings also can reflect the policy diversity, where the detailed explanation is presented in Appendix A.2. Specifically, we construct the covariance matrix $\mathbf{S}$ of $(v_m)_{m=1}^M$, where $v_m \triangleq (v_{m,k})_{k=1}^K$. The (i,k)-th element of $\mathbf{S}$ is $s_{ik} = \frac{1}{M-1}\sum_{m=1}^{M}(v_{mi} - \bar{v}_i)(v_{mk} - \bar{v}_k)$, where M is the number of policies, and $\bar{v}_i \triangleq \frac{1}{M}\sum_{j=1}^{M} v_{ji}$.

Since the generalized variance is by definition semi-positive definite, ensuring that the determinant is greater than or equal to zero, we use the Gershgorin Circle Theorem [75] to fine-tune in extreme cases (when the generalized variance is zero) to make its determinant greater than zero, thus ensuring that the diversity matrix is positive definite. Since the matrix is reconstructed with each update, a single adjustment does not cause a lasting impact. Specifically, if $\det(\mathbf{S}) = 0$, we replace $\mathbf{S}$ with $\tilde{\mathbf{S}}$, where (i,i)-th element is $\tilde{s}_{ii} = s_{ii} + \sum_{j \neq i} |s_{ij}|$. Therefore, $\det(\mathbf{S})$ could be specified as the quantification of the disagreement of dispersion trajectories, that is, the diversity measure to the policies.

Algorithm 1 D2PED

Require: number of learners M, batch size N for each policy, max iteration Q, period to train policy representation module U.
Initialize: M policies $\{v_m\}_{m=1}^M$, parameters of policy representation module: ϕ, environments $\{E_1, \cdots, E_m\}$, local experience replay buffer $\{D_1, \cdots, D_m\}$, shared replay buffer D_s.

while $q < Q$ **do**
  for $m = 1, 2, \cdots, M$ learners in parallel **do**
    Reset the environment and get state s_t.
    for each running time step t **do**
      Sample action a_m^t from policy $\pi_m(a_m^t|s_m^t)$.
      Apply the action a_m^t to the environment.
      Get next state s_m^{t+1} and the reward r_m^t.
      Store the transition $(s_m^t, a_m^t, r_m^t, s_m^{t+1})$ into the local replay buffer D_m.
      if environment E_m done **then**
        Store trajectory $\tau_m = \{s_m^t, a_m^t, r_m^t\}_{t=1}^T$ into the shared replay buffer D_s.
      end if
    end for
    if $q \mod \mathrm{U} == 0$ **then**
      for each training step **do**
        Randomly sample N trajectories for each of the M policies from shared replay buffer D_s, denoted as $\{\tau_m^i\}_{i=1,m=1}^{N,M}$.
        Construct sample categorical distribution via Eq. (5) to select training trajectories.
        Compute the policy embeddings $\{v_{i,m}\}_{i=1,m=1}^{N,M}$ via Eq. (2).
        Predict classification labels $\{\xi_{i,m}\}_{i=1,m=1}^{N,M}$ via Eq. (3).
        Compute $\mathcal{L}_\phi$ via Eq. (4).
        Update ϕ for policy representation module.
      end for
    end if
    Randomly samples N trajectories for each of the M policies and shuffles all the trajectories $\{\tau_m^i\}_{i=1,m=1}^{N,M}$.
    Compute policy embeddings $\{v_m\}_{m=1}^M$ via Eq. (2).
    Stack policy embeddings $\{v_m\}_{m=1}^M$ to construct a matrix V.
    Compute a Dispersion Matrix $\mathbf{S} = F(V)$ based on the Definition 4.1.
    Compute Policy Diversity $\mathrm{Div}(\{\mathrm{v_m}\}_{\mathrm{m=1}}^{\mathrm{M}}) = \det(\mathbf{S})$ based on the Definition 4.2.
    Compute $J(\Pi)$ and update candidate policies Π with objective function: 6.
  end for
end while

The D2PED algorithm is summarized in Alg. 1. D2PED employs a dispersion matrix to measure the diversity of the candidate policies and adds the determinant of the dispersion matrix to the policy objective function. To improve the efficiency of the algorithm, M learners (i.e., M diverse

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on September 19,2024 at 05:05:22 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2024.3455257

policies) execute parallelly in different copies of the same environment and share a shared replay buffer.

5 Experiments

In this section, we present and analyze our experimental results. First, we provide an overview of the experimental setup and the environments we used to evaluate our models in Section 5. Then, in Sections 5.2 $\sim$ 5.3, we report results in several different environments respectively. Finally, we further provide an ablative analysis of the proposed methodology in Section 5.4. We also summarize hyperparameters used in experiments in Appendix A.3.

5.1 Experimental Setup

We conducted experiments in both non-MDP and MDP environments to verify the expansion of the solution space by non-MDP decision processes. We also performed experiments in continuous and discrete action spaces in non-MDP environments to validate the capability of D2PED to learn diverse policies and the performance of the policies in both action spaces. Additionally, we designed single modal and multi modal experiments in MDP environments to verify the performance of policies under MDPs and the ability to learn diverse policies with D2PED. Finally, in the ablation Study, we separately verified the contribution of the policy representation module and the sensitivity of the policy dispersion module to the number of policies.

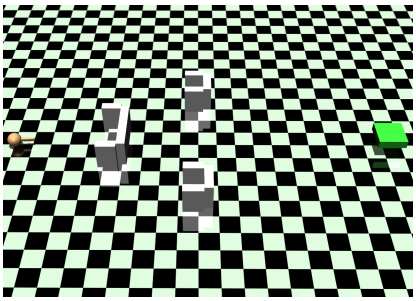


Fig. 4: The Point-v1 environment.

5.1.1 Environments

Point. To explicitly examine whether our method can efficiently find high-quality policies with diverse behaviors in the non-Markov environment, we create the Point-v1 environment with non-Markov rewards and multi-solution, which is modified from the Point-v0 [19] and has continuous state and action space. In the Point-v1 environment, an agent and a target (i.e., green cuboid) are separated by three U-shape walls (see: Fig. 4). When an episode starts, the target randomly sends a signal to the agent, and the agent needs to reach the goal within a specified number of time steps. When the agent receives the signal and reaches the target, it will get an event reward of 100. If there is no signal at the target, the agent starts to move, or if the agent hits a wall while moving, it will get an event reward of -100. In this case, the context or sequence of events (whether the agent receives the signal and what actions it takes after receiving the signal) leading to the agent reaching the target can influence the reward, that is, the past events can influence the current reward, making the process history-dependent. The current episode ends immediately if the agent hits the wall or reaches the target. The agent only obtains a total reward at the end of each episode. The total reward is the sum of the event reward and the distance reward, which is the negative distance between the agent and the target at the end of the episode.

FrozenLake. For discrete control tasks, we perform experiments in the FrozenLake-v1 environments with non-Markov rewards, which is a grid world game, some grids are walkable, and some grids are holes that will make the agent fall into the water (see: Fig. 5). At the beginning of an episode, the agent will stand by in the upper left corner of the map, with the target grid in the lower left corner. The task randomly sends a start signal to the agent. If the agent receives the signal and reaches the goal grid, a reward of 1 is obtained. If the agent starts moving without receiving the signal, a reward of 0 is obtained. If the target is not reached within a finite time step after receiving the signal or the agent falls into the hole, the obtained reward is still 0. As the size of grids and the density of holes increase, it becomes increasingly difficult for agents to reach the target grid.

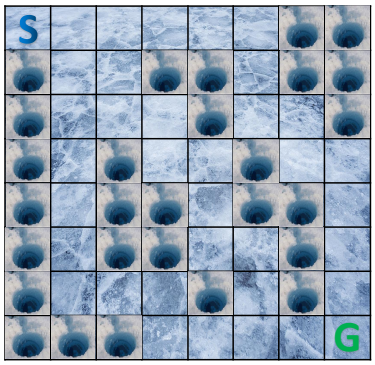
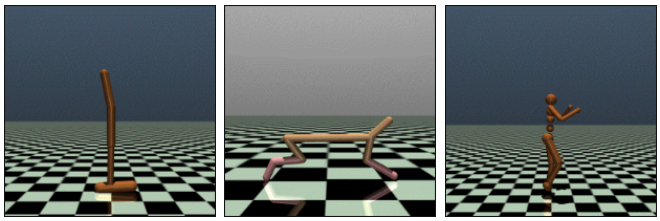


Fig. 5: The Frozenlake 8×8 environment.

MuJoCo. To examine the effectiveness of D2PED expanding the solution space, we conduct experiments in various tasks, Hopper (Fig. 6(a)), Halfcheetah (Fig. 6(b)), and Humanoid (Fig. 6(c)), from the *OpenAI Gym* library [76]. In each task, the agent takes as input a vector of physical states and generates a vector of action values to manipulate the robots in the environment. We also create two multi-modal environments based on the HalfCheetah and Ant, where we assign rewards for both Forward and Backward tasks to examine our method's effectiveness in finding high-quality and diverse policies in Markov reward environments.



(a) Hopper (b) Halfcheetah (c) Humanoid

Fig. 6: The three standard MuJoCo environments.

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on September 19,2024 at 05:05:22 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2024.3455257

5.1.2 Baseline Methods

The baseline methods adopted for comparison are the same within different environments. We select DvD [19], QD-RL [77], and P3S [60] as the baselines. For a fair comparison, we combine our method and baselines with the same base algorithm (i.e., DQN [78], PPO [12], TD3 [13]) and use the same hyperparameters for each environment. The number of candidate policies M of these algorithms is always set to 5. We report the mean and standard deviations across five identical seeds for all algorithms and all tasks. The experiments are performed using the ray [79] library for multi-process parallel computation.

To make policy optimization algorithms still work in non-MDP environments, we incorporate Long Short-Term Memory (LSTM) [80] networks into the PPO and DQN architecture allowing the algorithm to maintain a form of internal state that captures information from the past environment states. This approach effectively transforms the MDP environment into a non-MDP one from the algorithm's perspective, allowing it to make decisions that consider the history of states and actions.

5.2 Evaluations in Non-MDP Environments

We evaluate D2PED in non-MDP environments with continuous and discrete action spaces, focusing on two aspects: policy performance and the ability to learn diverse policies.

5.2.1 Continuous Control

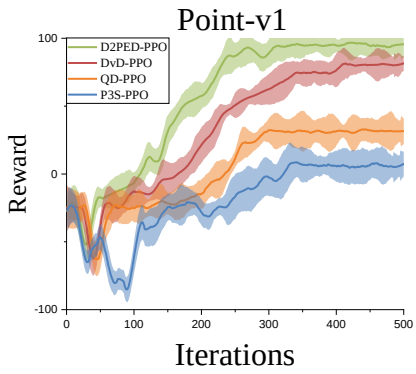


Fig. 7: Performance of our algorithms and three baselines on the Point-v1 environment. The solid line and shaded regions represent the mean and standard deviation, respectively, across ten seeds.

Performance. In the Point-v1 environment experiments, we evaluate D2PED against several baselines, and all methods select PPO [12] as the policy optimization algorithm. As shown in Fig. 7, our method (D2PED-PPO) demonstrates superior performance in terms of reward over iterations when compared to other baselines, including DvD-PPO, QD-PPO, and P3S-PPO. Specifically, D2PED-PPO achieves a higher reward earlier in the training process, indicating more efficient exploration and exploitation of the environment. This advantage is maintained throughout the training, with D2PED-PPO consistently outperforming other methods. DvD-PPO achieves the second-best performance but is still 20% worse than our method. The results highlight the effectiveness of our dispersion mechanism in generating diverse policies that enable better exploration and lead to improved performance in non-MDP tasks. Furthermore, the divergence in performance between our method and the baselines becomes more pronounced as the number of iterations increases, showcasing the long-term benefits of policy diversity. The consistent superiority of D2PED-PPO across various stages of training underscores the importance of incorporating diverse policies to enhance adaptability and robustness in complex environments.

Diverse policies. To intuitively measure the diversity of different methods, Fig. 8 visualize the movement paths of five candidate policies of each method which are learned after 500 training iterations in the Point-v1 environment. Each subplot, from (a) to (d), visualizes the distinct behavior of different policy enhancement methods developed from our proposed D2PED framework and other baseline approaches.

In subfigure (a), D2PED-PPO exhibits a diverse set of trajectories with policies exploring various regions of the solution space, indicating a wide coverage and inherent robustness in the decision-making process. This aligns with our hypothesis that policy diversity can greatly enhance the ability of an agent to adapt to and solve complex tasks in non-MDP environments. In subfigure (b), DvD-PPO shows a focused exploration pattern, with policies converging towards a common set of trajectories. While this might imply efficient exploration in some contexts, it may also suggest a potential for overfitting to specific paths and less adaptability in changing environments.

In subfigure (c), we observe that only two policies can bypass the wall from the gap between the middle wall and the upper wall, and the others are stuck in front of the walls. One of the policies hit the edge of the upper wall and stumble to the target. QD-PPO policies generally maintain proximity. Lastly, in subplot (d), the movement paths of P3S-PPO overlapped over a long period, and most policies got stuck in front of the walls, which demonstrates that P3S-PPO can not work well in the non-Markov environment.

Overall, the visualization in Fig. 8 supports our proposal that the dispersion matrix promotes diversity without affecting the performance of the policy. Our D2PED-PPO method has demonstrated a notably more expressive set of diverse policies, which are crucial for success in both non-MDP discrete action space environments, leading to a more robust and adaptable performance as compared to the baselines.

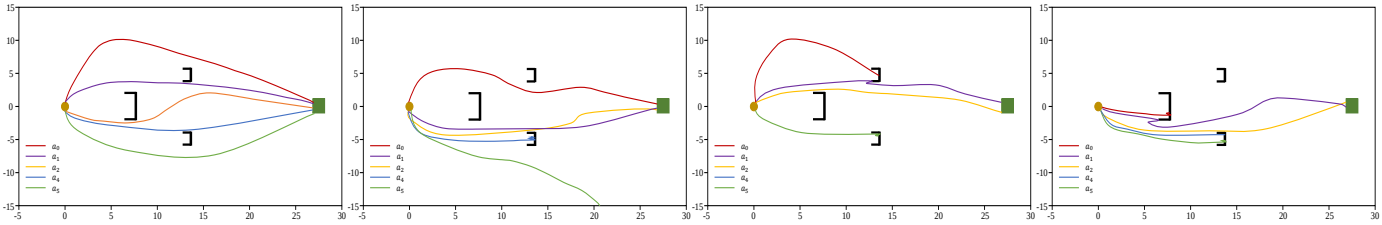
5.2.2 Discrete Control

Performance. For each policy of each method, we first train 4000 episodes, then Table 1 shows the average number of times the five candidate policies for each method reach the target. We conduct experiments across three versions of the FrozenLake-v1 environment with grid sizes of 4×4, 5×5, and 8×8. In these experiments, our method, D2PED, and the baseline methods are all integrated with the DQN [78].

Our D2PED-DQN algorithm demonstrates superior efficiency compared to P3S-DQN, DvD-DQN, and QD-DQN across all tested environments. Specifically, in the simplest 4×4 grid with 4 holes, D2PED-DQN reaches the goal more times than other methods, highlighting its effectiveness in navigating with limited challenges. As the environment's

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on September 19,2024 at 05:05:22 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2024.3455257



(a) POINT(D2PED-PPO) (b) POINT(DvD-PPO) (c) POINT(QD-PPO) (d) POINT(P3S-PPO)

Fig. 8: Movement paths of five candidate policies visualization of D2PED and baselines after 500 training iterations in the Point-v1 environment. From (a) to (d), the lines of five different colors in each subfigure represent the movement paths of five candidate policies. The orange point and the green square represent the start position and target position, respectively, and the black polylines represent the walls separating the agent and the target.

TABLE 1: The average number of times each method reaches the target across 5 seeds in three versions of the FrozenLake-v1 environment. After 4000 training episodes for each seed, the average number of times the 5 candidate policies of each method reach the target is computed. Then, the averages of these averages for each method across the 5 seeds are taken to obtain the final average number of times each method reaches the target.

Environment	D2PED-DQN	P3S-DQN	DvD-DQN	QD-DQN
FrozenLake-v1 4×4 (4holes)	**2175**	2034	1874	1582
FrozenLake-v1 5×5 (10holes)	**1895**	1798	1603	1489
FrozenLake-v1 8×8 (28holes)	**1040**	770	689	634

complexity increases with a 5×5 grid and 10 holes, D2PED-DQN still maintains its advantage.

The most striking results are observed in the complex 8×8 grid with 28 holes, where D2PED-DQN outperforms its baselines by a significant margin, reinforcing the proposed theory's premise that policy diversity leads to more robust problem-solving strategies. This is evidenced by D2PED-DQN's ability to adapt to the highly stochastic nature of the environment, which is closer to real-world scenarios where rewards are influenced by an agent's historical states and actions—a key feature of non-MDPs.

These results underscore the benefits of policy diversity in RL, as D2PED-DQN's varied policy approach allows it to diffuse effectively into the boundary of the solution space, demonstrating enhanced adaptability and innovation in problem-solving. The empirical data support the hypothesis that expanding the solution space through policy diversity is not only theoretically sound but also practically advantageous, enabling RL algorithms to perform robustly across a spectrum of environmental complexities.

Diverse policies. Fig. 9 provides a compelling visual representation of the paths taken by five candidate policies within the complex 8×8 grid of the FrozenLake-v1 environment, replete with 28 hazardous holes. The visualization encompasses our proposed D2PED-DQN method alongside three baseline methods—DvD-DQN, QD-DQN, and P3S-DQN—after undergoing 4000 training episodes. The distinct paths traced by each policy signify their unique approach to navigating from the start ('S') to the goal ('G'), circumventing the holes ('H') scattered across the grid.

The D2PED-DQN demonstrates a preference for varied, exploratory paths, indicating a higher level of policy diversity and a more profound search within the solution space. This diversity is consistent with our theoretical framework that posits the benefits of diversifying solutions in non-Markov environments, where the policy's adaptability to historical information and temporal dependencies is critical. In comparison, the paths of the baselines, while effective in reaching the goal, offer less variability, which may translate to a narrower exploration of the available solution space. Notably, the P3S-DQN method shows a more direct approach to the goal, aligning with its performance metrics in Table 1.

These visual results affirm the theoretical implications proposed in our paper, evidencing that the dispersion matrix indeed broadens the policy disagreements, leading to a richer policy embedding distribution. This enhances the agent's ability to develop robust and versatile strategies, thereby improving its performance in non-MDP environments. The adaptability of the policies to a challenging and hazard-filled grid such as this one underscores the practical advantages of employing diverse policy learning in complex decision-making scenarios.

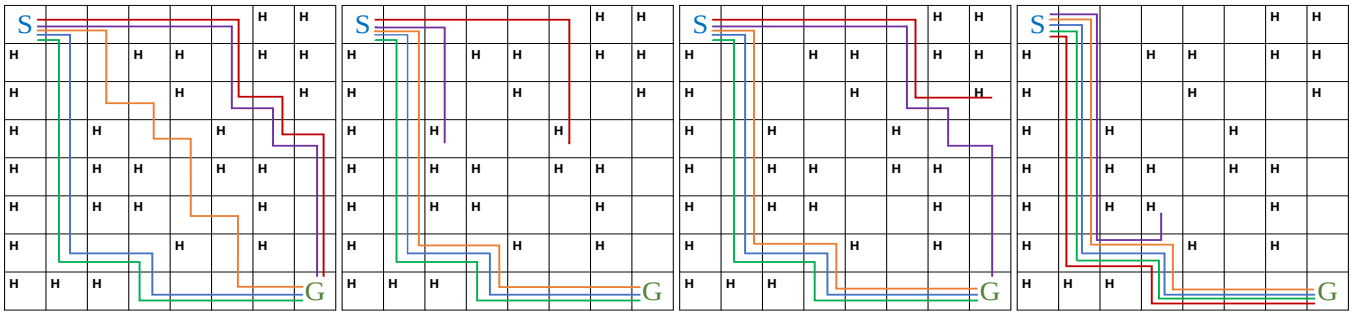
5.3 Evaluations in MDP Environments

The single modal evaluation verifies whether D2PED affects performance, while the multi modal evaluation confirms that D2PED can also learn diversity in MDP environments, indicating that our approach expands the solution space. We combine our method D2PED and baselines with TD3 [13].

Single modal. Fig. 10 illustrates the performance of different algorithms in the standard MuJoCo environments: Hopper-v3, Halfcheetah-v3, and Humanoid-v2. In the Hopper-v3 environment, D2PED-TD3 shows a steep initial learning curve, surpassing other algorithms early on and maintaining a slight edge throughout the iterations. DvD-TD3 and P3S-TD3 exhibit comparable performance, with P3S-TD3 showing slightly higher reward consistency as evidenced by its narrower confidence interval. QD-TD3,

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on September 19,2024 at 05:05:22 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2024.3455257



(a) FROZENLAKE(D2PED-DQN) (b) FROZENLAKE(DvD-DQN) (c) FROZENLAKE(QD-DQN) (d) FROZENLAKE(P3S-DQN)

Fig. 9: Movement paths of five candidate policies visualization of our proposed D2PED and baselines after 4000 training episodes in the FrozenLake-v1 environment with a 8×8 grid and 28 holes. From (a) to (d), the lines of five different colors in each subfigure represent the movement paths of five candidate policies. Each policy needs to navigate from the start ('S') to the goal ('G'), circumventing the holes ('H') scattered across the grids.

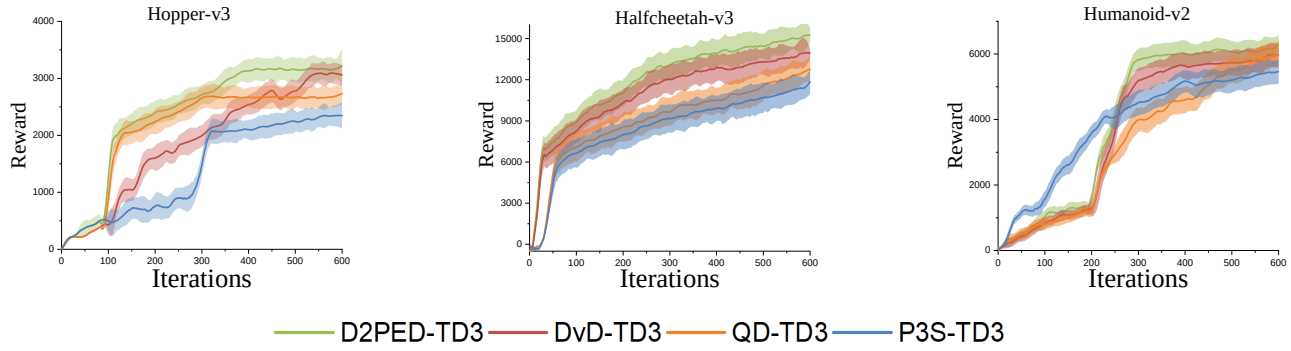


Fig. 10: Performance of our algorithms and baselines on the three standard MuJoCo environments. The solid line and shaded regions represent the mean and standard deviation, respectively, over five seeds.

while starting lower, demonstrates a steady improvement, suggesting effective learning over time.

The Halfcheetah-v3 environment reveals a more competitive dynamic between algorithms, with D2PED-TD3 and DvD-TD3 competing for the highest reward, but D2PED-TD3 consistently maintains the lead, indicating that D2PED-TD3 is a robust learning paradigm.

In the challenging Humanoid-v2 environment, all algorithms struggle to a certain extent, as reflected by broader confidence intervals, yet D2PED-TD3 stands out by eventually achieving the highest rewards. The increased complexity of this environment showcases the advantage of exploration provided by the policy diversity of D2PED-TD3.

These findings corroborate that policy diversity enhances RL success. At the same time, the empirical evidence provided here also validates that the proposed method has adaptive and robust performance in MDP environments, indicating that our method has expanded the solution space.

Multi modal. To assess the efficacy of D2PED in learning diverse and high-quality policies in MDP environments, we modified the original single-mode environments of HalfCheetah and Ant to support dual-mode tasks by assigning rewards for both Forward and Backward movements. We trained all methods within these dual-mode environments and conducted tests in single-mode settings to isolate performance capabilities. To facilitate a clearer comparison between different algorithms, we have systematically organized the experimental results into Table 2.

For the HalfCheetah environment, the D2PED-TD3 algorithm exhibits consistent performance, with a notable increase in average reward when the mode shifts from Forward to Backward, indicating its adaptability to multiple modes. P3S-TD3 shows a drastic drop in performance in the Backward mode, indicating a lack of robustness in policy diversity when adapting to the reverse direction. Similarly, DvD-TD3 performs poorly in the forward mode. QD-TD3 achieves moderate rewards in both modes.

In the Ant environment, all methods see a slight increase or roughly maintain performance when switching modes, with D2PED-TD3 showing a marginal improvement in the Backward mode. This pattern may highlight the effective policy diversity within these methods, allowing them to adapt to the mode change with minimal performance degradation.

Overall, these results emphasize the significance of policy diversity where it enhances RL success by enabling agents to tackle a variety of challenges within an environment. The data suggests that methods with a more diverse policy set, such as D2PED-TD3, are better equipped to handle directional changes within these MDP environments.

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on September 19,2024 at 05:05:22 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2024.3455257

TABLE 2: The average reward accrued by five candidate policies from each method on two multi-mode environments. First, calculate the average reward of the 5 candidate policies for each method in a single seed experiment. Then, the averages of these averages for each method across the 5 seeds are taken to obtain the final average reward of each method.

Envrionment	Mode	D2PED-TD3	P3S-TD3	DvD-TD3	QD-TD3
HalfCheetah	Forward	**5325**	5014	-3591	4897
	Backward	**6716**	-4452	6380	6045
Ant	Forward	**4740**	4454	4437	4033
	Backward	**4734**	-3273	4090	4108

5.4 Ablation Studies

In this section, we conduct ablation studies to analyze the relative contribution of our policy representation module and the sensitivity of the number of candidate policies M.

The contribution of our policy representation module. One of the crucial parts of this work is the policy representation module. To demonstrate the importance of our policy representation module, we replace it with two policy embedding methods: auto-encoder and behavioral embedding, respectively. The auto-encoder method takes the trajectories generated by π_m as input and outputs the embedding for π_m. The behavioral embedding method is an action-based behavior characterization method. Following the implementation in [19], we randomly select 20 states and concatenate the actions of the selected states as the policy embedding. The experimental results of 7 environments are shown in Fig. 11 and Table 3.

Fig. 11 presents the performance of the D2PED algorithm equipped with three different policy embedding techniques: the Policy Representation Module, Auto-Encoder, and Behavioral Embedding. It is evident across the environments of Point-v1, Hopper-v3, Halfcheetah-v3, and Humanoid-v2 that the Policy Representation Module yields a consistently higher reward over iterations compared to the other methods, suggesting a more effective policy learning process. The Auto-Encoder shows competitive performance but does not reach the efficiency of the Policy Representation Module.

Table 3 further corroborates these findings by indicating the average number of attempts required to reach the target in the three versions of FrozenLake-v1 environments with varying complexity. Policy Representation Module significantly outperforms the other two embedding techniques, achieving more times to reach the goal across all environments. This is particularly notable in the more complex 8×8 grid with 28 holes, where the number of times the Policy Representation Module reaches the target is more than three times that of the Behavioral Embedding.

The data collectively suggest that the Policy Representation Module enhances D2PED's ability to navigate and learn from the environment more effectively. By enabling diverse policy expressions, the D2PED with Policy Representation Module technique fosters robust performance and adaptability in both MDP and non-MDP settings, affirming that a diversified policy embedding can expand the solution space.

The sensitivity of the number of candidate policies. The number M of candidate policies directly influences the number of policy embeddings used to construct the *diversity matrix*. If M is too small, there is a reduced likelihood of identifying a comprehensive set of optimal policies with diverse behaviors. Conversely, if M is too large, it may lead to the selection and updating of some policies that exhibit similar behaviors, which can undermine efficiency.

Fig. 12 and Table 4 present the outcomes of varying the number of candidate policies for the D2PED algorithm across Point-v1, different standard MuJoCo environments, and versions of the FrozenLake-v1 environment. We report the results under the same number of iterations. The experiment results inform us about the influence of policy count on learning performance and problem-solving efficiency.

The performance curves in Fig. 12 depict a general trend where increasing or decreasing the number of candidate policies M has a negligible impact on the performance of the policy in the Point-v1, Hopper-v3, Halfcheetah-v3, and Humanoid-v2 environments. A higher or lower number of policies have some differences in the effectiveness of exploring the solution space, but the degree of difference is not significant, and it always produces better performance. This indicates that D2PED is not sensitive to the choice of policy number M.

Table 4 supplements this finding, showing that the average number of reaching the target in the FrozenLake-v1 environment varies little with the increase of M. The table suggests that the number of candidate policies has a minor impact on the efficiency of solving the problem.

The results from Fig. 12 and Table 4 indicate that the D2PED algorithm is not sensitive to the choice of M, demonstrating its good adaptability and robustness.

5.5 Discussion

We further discuss our method from three perspectives: its performance in non-stationary environments (Section 5.5.1), sample efficiency (Section 5.5.2), and time and space complexity (Section 5.5.3).

5.5.1 Non-stationary environments

We design the following experiments to validate the performance of our method in non-stationary environments.

We created a non-stationary version of the FrozenLake environment, as shown in Fig.13, and we added some slippery blocks. When an agent steps on these blocks, it quickly slides into the adjacent cells. The positions of the slippery blocks are randomized in each episode. This randomization introduces a dynamic element to the environment, as the agent must adapt to different slippery locations that affect movement patterns unpredictably in each episode. Similarly, the agents can only start moving after receiving a start signal, meaning the calculation of rewards is consistent with FrozenLake-v1.

Similarly, for the Point environment, we introduced non-stationary by randomizing the positions and numbers of

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on September 19,2024 at 05:05:22 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2024.3455257

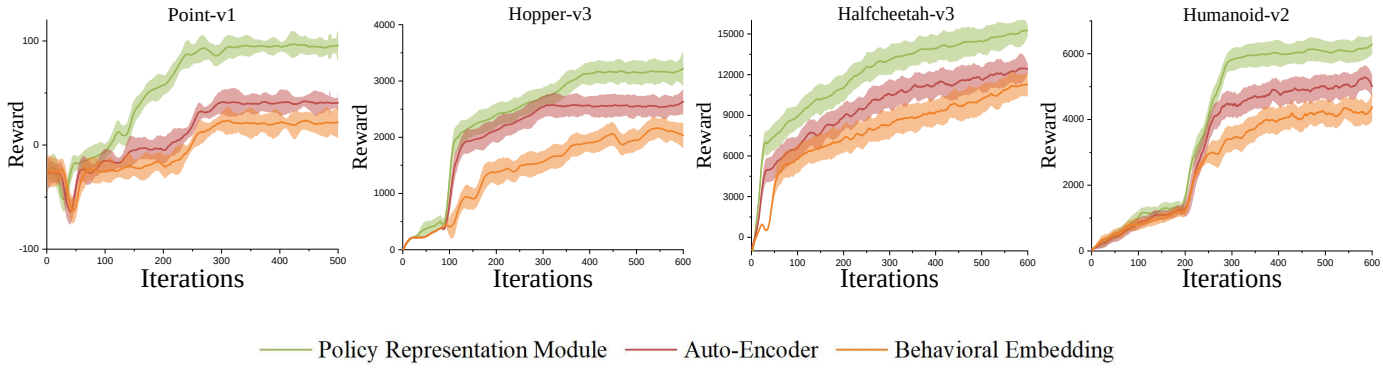


Fig. 11: Performance of D2PED equipped with different policy embedding algorithms. The solid line and shaded regions represent the mean and standard deviation, respectively, over five seeds.

TABLE 3: The average number of times to reach the target of D2PED equipped with different policy embedding algorithms across 5 seeds on the three versions of FrozenLake-v1 environments. After 4000 training episodes for each seed, the average number of times the 5 candidate policies of each method reach the target is computed. Then, the averages of these averages for each method across the 5 seeds are taken to obtain the final average number of times each method reaches the target.

Environment	Policy Representation Module	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

TABLE 4: The average number of times to reach the target of D2PED with different number of candidate policies M across 5 seeds on three FrozenLake-v1 environments. After 4000 training episodes for each seed, the average number of times the M candidate policies of each method reach the target is computed. Then, the averages of these averages for each method across the 5 seeds are taken to obtain the final average number of times each method reaches the target.

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

walls, as well as the starting position of the agent, as shown in Fig.14. These variations ensure that each episode presents a unique set of challenges, requiring the agent to continually adapt its policy to succeed. The agent can only start moving after receiving a start signal, meaning the calculation of rewards is consistent with Point-v1.

The results, as detailed in Fig. 15 and Table 5, show that our method performs effectively in these non-stationary settings. In the Point (non-stationary) environment, our method (D2PED-PPO) demonstrates superior performance in terms of reward over iterations when compared to other baselines, including DvD-PPO, QD-PPO, and P3S-PPO. Specifically, D2PED-PPO achieves a higher reward earlier in the training process, this advantage is maintained throughout the training. Furthermore, the divergence in performance between our method and the baselines becomes more pronounced as the number of iterations increases, showcasing the long-term benefits of policy diversity in non-stationary environments.

In the FrozenLake (non-stationary) environment, our D2PED-DQN algorithm demonstrates superior efficiency compared to P3S-DQN, DvD-DQN, and QD-DQN across all tested environments. Specifically, in the simplest 4×4 grid with 4 holes, D2PED-DQN reaches the goal more times than other methods, highlighting its effectiveness in navigating with limited challenges. As the environment's complexity increases with a 5×5 grid and 10 holes, D2PED-DQN still maintains its advantage. The most striking results are observed in the complex 8×8 grid with 28 holes, where D2PED-DQN outperforms its baselines by a significant margin, reinforcing the proposed theory's premise that policy diversity leads to more robust problem-solving policies. The ability of our algorithm to still reach or maintain high performance under varying conditions within each episode indicates robustness to environmental changes. The experiments conducted provide empirical evidence that our approach is not limited to static or merely non-Markovian scenarios but extends to non-stationary environments.

The experimental validation confirms that our approach effectively handles non-stationary environments, adapting to changes in the environment's dynamics from one episode to the next. This adaptability underscores the robustness and applicability of our method to a range of real-world scenarios where conditions can change unpredictably.

5.5.2 *Sample efficiency*

In this section, we evaluate and assert the sample efficiency of our approach.Sample efficiency in our context refers to

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on September 19,2024 at 05:05:22 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2024.3455257

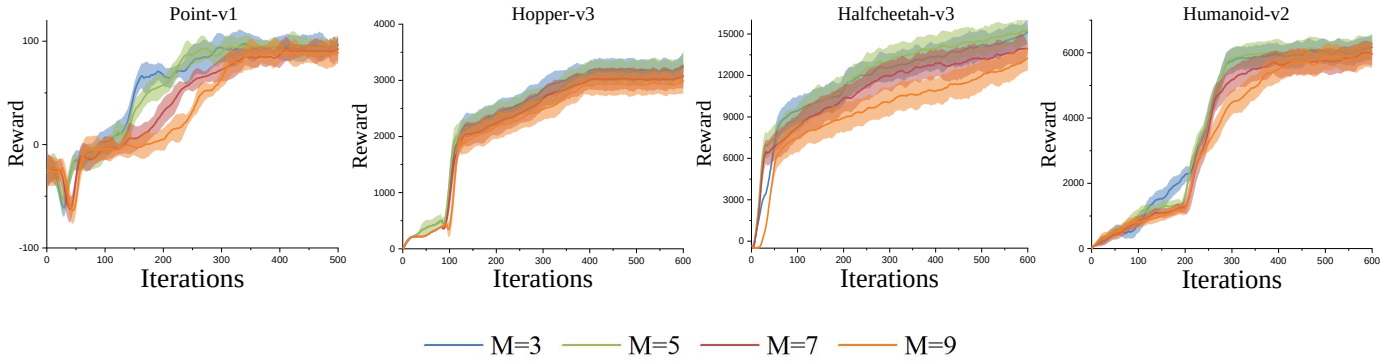


Fig. 12: Performance of D2PED with different number of candidate policies M.The solid line and shaded regions represent the mean and standard deviation, respectively, over five seeds.

TABLE 5: The average number of times each method reaches the target across 5 seeds in three versions of the FrozenLake (non-stationary) environment. After 4000 training episodes for each seed, the average number of times the 5 candidate policies of each method reach the target is computed. Then, the averages of these averages for each method across the 5 seeds are taken to obtain the final average number of times each method reaches the target.

Environment	D2PED-DQN	P3S-DQN	DvD-DQN	QD-DQN
FrozenLake-v1 4×4 (2holes, 2slippery)	**2175**	2036	1482	1690
FrozenLake-v1 5×5 (5holes, 5slippery)	**1895**	1751	1371	1582
FrozenLake-v1 8×8 (16holes, 12slippery)	**1040**	1016	881	907

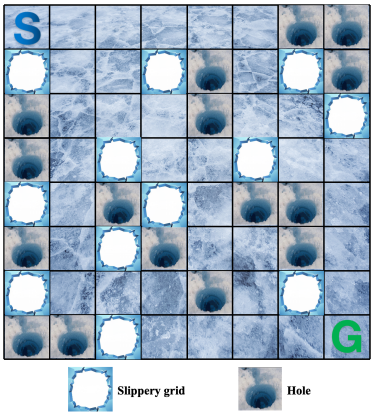


Fig. 13: The Frozenlake (non-stationary) 8×8 environment with 16 holes and 12 slippery.

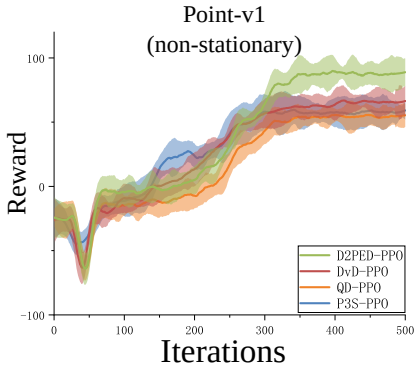


Fig. 15: Performance of our algorithms and three baselines on the Point (non-stationary) environment. The solid line and shaded regions represent the mean and standard deviation, respectively, across ten seeds.

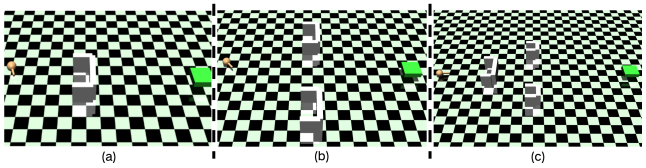


Fig. 14: The Point (non-stationary) with randomizing the positions and numbers of walls, as well as the starting position of the agent.

the ability of our algorithm to achieve significant learning or performance improvement from a limited number of interactions (samples) with the environment. We conducted experiments where our method was validated against the naive algorithms. The learning curves, derived from these experiments, provide a visual and quantitative measure of how quickly our algorithm reaches a certain level of performance relative to the number of samples used.

In these experiments, we measure performance metrics such as cumulative rewards and performance stability across different runs with the same number of samples. These metrics are compared to those of other established algorithms to ascertain relative sample efficiency.

The results are shown in Fig.16 and Table 6. From the experimental results, our method achieved similar performance to the naive algorithm at the same number of iterations, indicating that the sample utilization rate of our approach is acceptable. Our method optimizes both learning policy representations and the policies themselves using the same batch of samples collected from the environment. This dual

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on September 19,2024 at 05:05:22 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2024.3455257

TABLE 6: The sampling efficiency of D2PED with a naive algorithm by measuring the number of times each reaches the endpoint using the same number of iterations. This comparison is conducted across three versions of the FrozenLake-v1 environment, using 5 different seeds. After 4000 training episodes per seed, we compute the average number of times the 5 candidate policies of each method reach the target. Finally, we calculate the overall average for each method by averaging these results across all 5 seeds, yielding the final average number of times each method successfully reaches the target.

Environment	D2PED-DQN	DQN
FrozenLake-v1 4×4 (4holes)	2175	**2236**
FrozenLake-v1 5×5 (10holes)	1895	**1951**
FrozenLake-v1 8×8 (28holes)	**1040**	1028

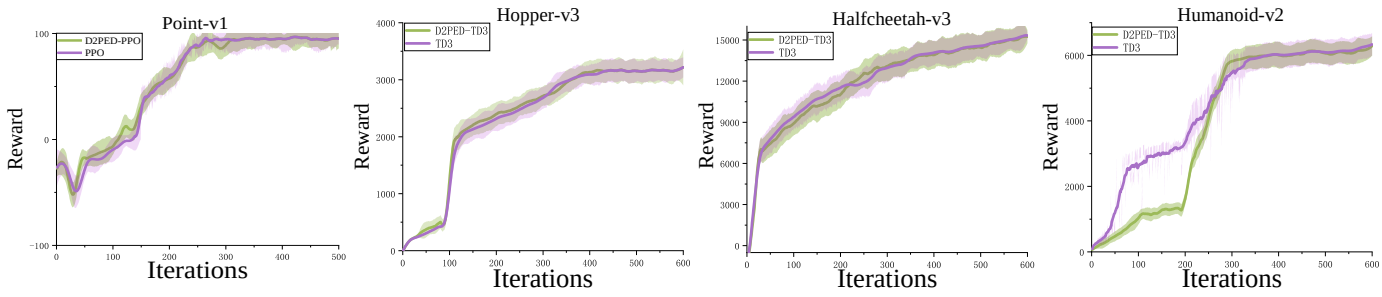


Fig. 16: The learning curves of our method and naive algorithms. The solid line and shaded regions represent the mean and standard deviation, respectively, over five seeds.

use of the same data set ensures that every sample collected is used to its fullest potential, enhancing our method's efficiency in sample utilization.

5.5.3 Time and space complexity

We evaluate the time and memory consumption of our algorithm across various experimental environments. Our experiments were conducted on a desktop with an 8 GB VRAM 2070s graphics card and 32 GB of RAM.

We conducted experimental tests on 5 candidate policies. During the experiments, the time and memory consumption corresponding to the number of iterations for each environment are shown in Table 7. D2PED utilizes parallel processing of multiple policy optimization and the policy representation module training. This parallelism inherently improves time efficiency, particularly when compared to unified policy solutions that may process sequentially. Our approach involves computing each policy on the GPU, utilizing some VRAM. The transformer architecture, which requires significant memory, is loaded into the system memory. This setup ensures that while policy optimization is ongoing, the transformer architecture can also be trained simultaneously. From the experimental results, the time and space complexity of our algorithm is found to be acceptable.

TABLE 7: Time consumption and memory consumption of D2PED under 5 candidate policies in 5 environments.

Env	Time	Memory
Point-v1 (500 iterations)	3072s	25.43G
FrozenLake-v1 8×8 (500 iterations)	3163s	26.41G
Hopper-v3 (600 iterations)	3646s	26.83G
Halfcheetah-v3 (600 iterations)	3458s	26.97G
Humanoid-v2 (600 iterations)	3792s	27.04G

6 Conclusion

In this paper, we overcome the limitations of MDPs and venture into a more expansive solution space, and also show that agents can develop more robust, adaptable, and innovative problem-solving capabilities, particularly in non-Markov Decision Process (non-MDP) environments. To that end, we introduced D2PED, a method for learning effective diverse policies for control tasks in non-Markov environments by incorporating temporal dependencies and historical information into the learning process. D2PED designs a policy dispersion scheme that repeatedly constructs policy embeddings as the policy update progresses, forming different dispersion trajectories and maximizing the dispersion disagreements of dynamically updating trajectories. We also analyze the relationship between diversity and policy performance. The experimental results underscore the effectiveness of our approach, demonstrating the capacity of our dispersion scheme to generate more expressive and diverse policy embeddings. Thus, our study not only reinforces the theoretical importance of policy diversity in reinforcement learning but also sets a new benchmark for future research in this field, particularly in complex real-world scenarios where adaptability and robustness are paramount.

References

[1] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[2] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[3] D. Zha, J. Xie, W. Ma, S. Zhang, X. Lian, X. Hu, and J. Liu, "Douzero: Mastering doudizhu with self-play deep reinforcement learning," in *international conference on machine learning*. PMLR, 2021, pp. 12 333–12 344.

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on September 19,2024 at 05:05:22 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2024.3455257

[4] E. Zhao, R. Yan, J. Li, K. Li, and J. Xing, "Alphaholdem: High-performance artificial intelligence for heads-up no-limit poker via end-to-end reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 4, 2022, pp. 4689–4697.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[6] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[7] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[8] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[9] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[10] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[11] M. G. Bellemare, S. Candido, P. S. Castro, J. Gong, M. C. Machado, S. Moitra, S. S. Ponda, and Z. Wang, "Autonomous navigation of stratospheric balloons using reinforcement learning," *Nature*, vol. 588, no. 7836, pp. 77–82, 2020.

[12] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[13] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1587–1596.

[14] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Advances in neural information processing systems*, vol. 21, 2008.

[15] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[16] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Conference on Robot Learning*. PMLR, 2018, pp. 651–673.

[17] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[18] J. K. Pugh, L. B. Soros, and K. O. Stanley, "Quality diversity: A new frontier for evolutionary computation," *Frontiers in Robotics and AI*, vol. 3, p. 40, 2016.

[19] J. Parker-Holder, A. Pacchiano, K. M. Choromanski, and S. J. Roberts, "Effective diversity in population based reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[20] M. L. Puterman, "Markov decision processes," *Handbooks in operations research and management science*, vol. 2, pp. 331–434, 1990.

[21] ——, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[22] A. Qin, F. Gao, Q. Li, S.-C. Zhu, and S. Xie, "Learning non-markovian decision-making from state-only sequences," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[23] B. Han, Z. Ren, Z. Wu, Y. Zhou, and J. Peng, "Off-policy reinforcement learning with delayed rewards," in *International Conference on Machine Learning*. PMLR, 2022, pp. 8280–8303.

[24] A. Camacho, O. Chen, S. Sanner, and S. A. McIlraith, "Decision-making with non-markovian rewards: From ltl to automata-based reward shaping," in *Proceedings of the Multi-disciplinary Conference on Reinforcement Learning and Decision Making (RLDM)*, 2017, pp. 279–283.

[25] M. Gaon and R. Brafman, "Reinforcement learning with non-markovian rewards," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 3980–3987.

[26] R. Brafman, G. De Giacomo, and F. Patrizi, "Ltlf/ldlf non-markovian rewards," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[27] D. Neider, J.-R. Gaglione, I. Gavran, U. Topcu, B. Wu, and Z. Xu, "Advice-guided reinforcement learning in a non-markovian environment," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 9073–9080.

[28] J. R. Marden and J. S. Shamma, "Game theory and control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 105–134, 2018.

[29] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," in *International conference on machine learning*. PMLR, 2017, pp. 1146–1155.

[30] T. Li, Y. Zhao, and Q. Zhu, "The role of information structures in game-theoretic multi-agent learning," *Annual Reviews in Control*, vol. 53, pp. 296–314, 2022.

[31] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[32] M. D. Pendrith and M. McGarity, "An analysis of direct reinforcement learning in non-markovian domains." in *ICML*, vol. 98. Citeseer, 1998, pp. 421–429.

[33] S. D. Whitehead and L.-J. Lin, "Reinforcement learning of non-markov decision processes," *Artificial intelligence*, vol. 73, no. 1-2, pp. 271–306, 1995.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[35] F. Bacchus, C. Boutilier, and A. J. Grove, "Rewarding behaviors," in *AAAI/IAAI, Vol. 2*, 1996.

[36] A. Camacho, O. Chen, S. Sanner, and S. McIlraith, "Non-markovian rewards expressed in ltl: guiding search via reward shaping," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 8, no. 1, 2017, pp. 159–160.

[37] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3834–3839.

[38] R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith, "Teaching multiple tasks to an rl agent using ltl," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2018, pp. 452–461.

[39] R. T. Icarte, T. Klassen, R. Valenzano, and S. McIlraith, "Using reward machines for high-level task specification and decomposition in reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2107–2116.

[40] F. Bacchus, C. Boutilier, and A. Grove, "Structured solution methods for non-markovian decision processes," in *AAAI/IAAI*. Citeseer, 1997, pp. 112–117.

[41] S. Thiébaux, C. Gretton, J. Slaney, D. Price, and F. Kabanza, "Decision-theoretic planning with non-markovian rewards," *Journal of Artificial Intelligence Research*, vol. 25, pp. 17–74, 2006.

[42] R. Toro Icarte, E. Waldie, T. Klassen, R. Valenzano, M. Castro, and S. McIlraith, "Learning reward machines for partially observable reinforcement learning," *Advances in neural information processing systems*, vol. 32, 2019.

[43] A. Cully and Y. Demiris, "Quality and diversity optimization: A unifying modular framework," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 245–259, 2017.

[44] T. Gangwani, J. Peng, and Y. Zhou, "Harnessing distribution ratio estimators for learning agents with quality and diversity," *arXiv preprint arXiv:2011.02614*, 2020.

[45] Z.-W. Hong, T.-Y. Shann, S.-Y. Su, Y.-H. Chang, T.-J. Fu, and C.-Y. Lee, "Diversity-driven exploration strategy for deep reinforcement learning," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 10 510–10 521.

[46] M. A. Masood and F. Doshi-Velez, "Diversity-inducing policy gradient: Using maximum mean discrepancy to find a set of diverse policies," *arXiv preprint arXiv:1906.00088*, 2019.

[47] Z. Peng, H. Sun, and B. Zhou, "Non-local policy optimization via diversity-regularized collaborative exploration," *arXiv preprint arXiv:2006.07781*, 2020.

[48] Y. Zhang, W. Yu, and G. Turk, "Learning novel policies for tasks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7483–7492.

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on September 19,2024 at 05:05:22 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2024.3455257

[49] E. Conti, V. Madhavan, F. P. Such, J. Lehman, K. O. Stanley, and J. Clune, "Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents," *arXiv preprint arXiv:1712.06560*, 2017.

[50] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.

[51] J.-B. Mouret and J. Clune, "Illuminating search spaces by mapping elites," *arXiv preprint arXiv:1504.04909*, 2015.

[52] J. Parker-Holder, L. Metz, C. Resnick, H. Hu, A. Lerer, A. Letcher, A. Peysakhovich, A. Pacchiano, and J. Foerster, "Ridge rider: Finding diverse solutions by following eigenvectors of the hessian," *Advances in Neural Information Processing Systems*, vol. 33, pp. 753–765, 2020.

[53] N. P. Nieves, Y. Yang, O. Slumbers, D. H. Mguni, Y. Wen, and J. Wang, "Modelling behavioural diversity for learning in open-ended games," *arXiv preprint arXiv:2103.07927*, 2021.

[54] H. Sun, Z. Peng, B. Dai, J. Guo, D. Lin, and B. Zhou, "Novel policy seeking with constrained optimization," *arXiv preprint arXiv:2005.10696*, 2020.

[55] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is all you need: Learning skills without a reward function," in *International Conference on Learning Representations*, 2018.

[56] K. Hartikainen, X. Geng, T. Haarnoja, and S. Levine, "Dynamical distance learning for semi-supervised and unsupervised skill discovery," in *International Conference on Learning Representations*, 2019.

[57] C. Colas, O. Sigaud, and P.-Y. Oudeyer, "Gep-pg: Decoupling exploration and exploitation in deep reinforcement learning algorithms," in *International conference on machine learning*. PMLR, 2018, pp. 1039–1048.

[58] S. Forestier, R. Portelas, Y. Mollard, and P.-Y. Oudeyer, "Intrinsically motivated goal exploration processes with automatic curriculum learning," *arXiv preprint arXiv:1708.02190*, 2017.

[59] Z. Zhou, W. Fu, B. Zhang, and Y. Wu, "Continuously discovering novel strategies via reward-switching policy optimization," *arXiv preprint arXiv:2204.02246*, 2022.

[60] W. Jung, G. Park, and Y. Sung, "Population-guided parallel policy search for reinforcement learning," in *International Conference on Learning Representations*, 2019.

[61] A. Grover, M. Al-Shedivat, J. Gupta, Y. Burda, and H. Edwards, "Learning policy representations in multiagent systems," in *International conference on machine learning*. PMLR, 2018, pp. 1802–1811.

[62] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-learning with latent embedding optimization," in *International Conference on Learning Representations*, 2018.

[63] D. Ghosh and M. G. Bellemare, "Representations for stable off-policy reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3556–3565.

[64] A. Tacchetti, H. F. Song, P. A. Mediano, V. Zambaldi, N. C. Rabinowitz, T. Graepel, M. Botvinick, and P. W. Battaglia, "Relational forward models for multi-agent learning," *arXiv preprint arXiv:1809.11044*, 2018.

[65] L. Zintgraf, S. Devlin, K. Ciosek, S. Whiteson, and K. Hofmann, "Deep interactive bayesian reinforcement learning via meta-learning," *arXiv preprint arXiv:2101.03864*, 2021.

[66] N. Rabinowitz, F. Perbet, F. Song, C. Zhang, S. A. Eslami, and M. Botvinick, "Machine theory of mind," in *International conference on machine learning*. PMLR, 2018, pp. 4218–4227.

[67] R. I. Brafman, G. De Giacomo *et al.*, "Regular decision processes: A model for non-markovian domains." in *IJCAI*, 2019, pp. 5516–5522.

[68] Q. Wang, B. Li, T. Xiao, J. Zhu, C. Li, D. F. Wong, and L. S. Chao, "Learning deep transformer models for machine translation," *arXiv preprint arXiv:1906.01787*, 2019.

[69] A. Baevski and M. Auli, "Adaptive input representations for neural language modeling," *arXiv preprint arXiv:1809.10853*, 2018.

[70] A. Kulesza, B. Taskar *et al.*, "Determinantal point processes for machine learning," *Foundations and Trends® in Machine Learning*, vol. 5, no. 2–3, pp. 123–286, 2012.

[71] J. Hadamard, "Étude sur les propriétés des fonctions entières et en particulier d'une fonction considérée par riemann," *Journal de mathématiques pures et appliquées*, pp. 171–216, 1893.

[72] S. S. Wilks, "Certain generalizations in the analysis of variance," *Biometrika*, pp. 471–494, 1932.

[73] T. W. Anderson, *An Introduction to Multivariate Statistical Analysis*. Job Wiley Sons, Inc. New York-London Sydney, 1962.

[74] R. A. Johnson, D. W. Wichern *et al.*, *Applied multivariate statistical analysis*. Pearson London, UK:, 2014, vol. 6.

[75] S. GERSCHGORIN, "Uber die abgrenzung der eigenwerte einer matrix," *Izv. Akad. Nauk. USSR. Otd. Fiz-Mat. Nauk*, vol. 7, pp. 749–754, 1931.

[76] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[77] G. Cideron, T. Pierrot, N. Perrin, K. Beguir, and O. Sigaud, "Qd-rl: Efficient mixing of quality and diversity in reinforcement learning," *arXiv preprint arXiv:2006.08505*, 2020.

[78] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *Computer Science*, 2013.

[79] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan *et al.*, "Ray: A distributed framework for emerging {AI} applications," in *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 2018, pp. 561–577.

[80] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

Bohao Qu is currently pursuing the Ph.D. degree with the School of Artificial Intelligence, Jilin University, Changchun, China. He is also a visiting student with the A*STAR Centre for Frontier AI Research, Singapore. Before that, he was an engineer of JD Mall Research, Beijing, China, from 2017 to 2019. His current research interests include deep reinforcement learning, multi-agent, machine learning, and hyperbolic (non-Euclidean) geometry.

Xiaofeng Cao received his Ph.D. degree at Australian Artificial Intelligence Institute, University of Technology Sydney, Australia, and was a visiting scholar at the Hong Kong University of Science and Technology. He is currently an Associate Professor at the School of Artificial Intelligence, Jilin University, China, and leading the Machine Perceptron Research Group with more than 20 members. He has more than 30 academic works published in IEEE T-PAMI, TNNLS, ICML, NeurIPS, ICLR, ECML, etc, and served as their PC members/reviewers. His research interests include the PAC learning theory, agnostic learning algorithm, generalization analysis, and hyperbolic (non-Euclidean) geometry.

Yi Chang is the dean of the School of Artificial Intelligence, Jilin University, Changchun, China. He was elected as a Chinese National Distinguished professor, in 2017 and an ACM distinguished scientist, in 2018. Before joining academia, he was the technical vice president with Huawei Research America, and the research director with Yahoo Labs. He is the author of two books and more than 100 papers in top conferences or journals. His research interests include information retrieval, data mining, machine learning, natural language processing, and artificial intelligence. He won the Best Paper Award on KDD'2016 and WSDM'2016. He has served as a Conference general chair for WSDM'2018 and SIGIR'2020.

Ivor W.Tsang (Fellow IEEE) is Director of A*STAR Centre for Frontier AI Research (CFAR) since Jan 2022. He is also an Adjunct Professor at School of Computer Science and Engineering (SCSE), Nanyang Technological University, Singapore. Previously, he was a Professor of Artificial Intelligence, at University of Technology Sydney (UTS), and Research Director of the Australian Artificial Intelligence Institute (AAII), the largest AI institute in Australia, which is the key player to drive the University of Technology Sydney to rank 10th globally and 1st in Australia for AI research, in the latest AI Research Index. Prof Tsang is working at the forefront of big data analytics and Artificial Intelligence. His research focuses on transfer learning, deep generative models, learning with weakly supervision, big data analytics for data with extremely high dimensions in features, samples and labels. His work is recognised internationally for its outstanding contributions to these fields.

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on September 19,2024 at 05:05:22 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2024.3455257

Yew-Soon Ong (M'99-SM'12-F'18) received the Ph.D. degree from the University of Southampton, U.K., in 2003. He is President's Chair Professor in Computer Science at Nanyang Technological University (NTU), and holds the position of Chief Artificial Intelligence Scientist of A*STAR, Singapore. At NTU, he serves as co-Director of the Singtel-NTU Cognitive & Artificial Intelligence Joint Lab. His research interest is in artificial and computational intelligence. He is founding EIC of the IEEE TETCI and AE of IEEE TNNLS, IEEE on Transactions on Cybernetics, IEEE Transactions on Artificial Intelligence and others. He has received several IEEE outstanding paper awards and was listed as a Thomson Reuters highly cited researcher and among the World's Most Influential Scientific Minds.

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on September 19,2024 at 05:05:22 UTC from IEEE Xplore. Restrictions apply.
© 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.