2022 IEEE International Conference on Robotics and Automation (ICRA)
May 23-27, 2022. Philadelphia, PA, USA

# Promoting Quality and Diversity in Population-based Reinforcement Learning via Hierarchical Trajectory Space Exploration

Jiayu Miao[1†], Tianze Zhou[2†], Kun Shao[3], Ming Zhou[1],
Weinan Zhang[1*], Jianye Hao[3], Yong Yu[1], and Jun Wang[4]

***Abstract*— Quality Diversity (QD) algorithms in population-based reinforcement learning aim to optimize agents' returns and diversity among the population simultaneously. It is conducive to solving exploration problems in reinforcement learning and potentially getting multiple good and diverse strategies. However, previous methods typically define behavioral embedding in action space or outcome space, which neglect trajectory characteristics during the execution process. In this paper, we introduce a trajectory embedding model trained by Variational Autoencoder with similarity constraint to characterize trajectory features. Based on that, we propose a hierarchical trajectory-space exploration (HTSE) framework using Determinantal Point Processes (DPP) to generate high-quality and diverse solutions in the selection and mutation process. The experimental results show that our HTSE method effectively completes several simulated tasks, outperforming other Quality-Diversity Reinforcement Learning algorithms.**

## I. Introduction

Reinforcement Learning (RL) has achieved significant performance in several scenarios including video games, self-driving cars, data center cooling systems and recommendation systems [1]–[4]. RL methods require agents to learn an excellent policy to achieve high cumulative rewards in these fields. However, exploration remains a crucial problem in the RL domain. Algorithms with limited exploration ability can obstruct agents from seeking high-quality policies and make agents get stuck in poor local optima. Training a population of agents is a promising method to ameliorate this issue, where different agents are supposed to explore different parts of the behavior space.

Quality Diversity is a typical class of methods where agents are required to seek high-performance and diverse solutions. It is originated from evolutionary algorithms [5]–[8], which refer to population-based optimization methods inspired by natural selection. There are two main algorithms developed in Quality-Diversity family, Novelty-based approaches [6], [9]–[11] and MAP-Elites [7], [8]. NSLC [9] combines novelty search and local competition in behavioral niches, and maintains more functional morphological diversity. Novelty-search ES [6] encourages exploration by updating the parameters in the direction to maximize the novelty score among the population. The novelty score typically relies on the outcome space, like the final location of the agent in navigation tasks. It is measured as the mean of $k$-nearest neighbor distance between ego agent and other agents in the outcome space. Kim et al. [10] utilize novelty search and local adaptation to discover robot skills efficiently. To realize exploration in high dimensional space, MAP-Elites [7] splits the outcome space into different cells, and ME-ES [8] leverages it to maintain a population with high performance and diversity. Furthermore, Evolutionary Reinforcement Learning (ERL) algorithms [12]–[14] emerge through combining reinforcement learning and evolutionary algorithms. However, ERL methods tend to focus on improving quality rather than diversity. Recently, Quality Diversity has been introduced into reinforcement learning. Quality Diversity Reinforcement Learning (QD-RL) methods are supposed to train a population of RL agents and generate high-quality and diverse policies.

A vital issue in QD-RL is how to define the behavioral embedding space when pursuing diversity. DvD [15] defines behavioral embedding as the joint actions taken by an agent on a set of shared sampled states. It abandons the common L2-distance and uses the determinant of the kernel matrix in Determinantal Point Processes [16] as the diversity term. To improve the diversity, DvD directly adds the diversity term as an additional objective in the loss function. QD-TD3 [17] learns to improve diversity in the outcome space with the novelty score as an intrinsic reward. Here half of the policies in the population are trained for high performance with a shared quality critic, while the other half are trained for diversity with a shared diversity critic. Besides, a selection mechanism based on the Pareto optimality is used to select $N$ best policies from all past achieved agents.

However, defining the behavior space simply as the joint action space under several states or outcome space cannot represent the diversity of different strategies. For the former, it is restricted that different policies must have a certain number of coincident states. While for the latter, it ignores the execution process of the strategy, and the same outcome can also come from different policies. The trajectory is the most intuitive embodiment of strategy. Therefore, to enlarge the behavior space and consider the trajectory characteristics of the strategy, we propose a trajectory embedding model with Variational Autoencoder [18]. Besides, we also consider the constraint in the training process with contrastive learn-

[1]Shanghai Jiao Tong University {mg2015started,mingak, wnzhang}@sjtu.edu.cn;yyu@apex.sjtu.edu.cn
[2]Beijing Institute of Technology tianzezhou@bit.edu.cn
[3]Noah's Ark Lab, Huawei Technologies {shaokun2, haojianye}@huawei.com
[4]University College London jun.wang@cs.ucl.ac.uk
† Work done as an intern at Huawei Noah's Ark Lab
∗ The corresponding author Weinan Zhang is supported by "New Generation of AI 2030" Major Project (2018AAA0100900), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102) and National Natural Science Foundation of China (62076161).

2022 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9681-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICRA46639.2022.9811888

ing [19], such that similar trajectories will perform higher similarities in the embedding space while distinct trajectories should be further away in the embedding space.

To improve quality and diversity in the trajectory embedding space, we propose a hierarchical training framework, which uses Determinantal Point Processes [20] as the selection and mutation mechanism to handle the trade-off between exploration and exploitation in the trajectory space. The meta controller generates a set $Z$, composed of different hidden variables $z$. It requires each agent in the population to align with the hidden variable and its corresponding trajectories first and then do regular training. This encourages agents to train and explore in $z$-relevant trajectory directions. Then, top $K$ policies with high performance and diversity are selected based on a modified Determinantal Point Process. The left policies are mutated to explore other trajectory directions.

In summary, the technical contributions in this work are as follows:

- We propose a trajectory embedding model with Variational Autoencoder and a similarity constraint, which characterizes the strategy's execution process and generates distinguished and meaningful embeddings.
- We propose an efficient hierarchical training framework and a DPP-based selection and mutation mechanism to explore and exploit the trajectory space.
- We compare our hierarchical trajectory space exploration (HTSE) algorithm with two other QD-RL methods in MuJoCo and driving tasks to justify the superiority of our framework in terms of quality and diversity.

## II. Preliminaries

### A. Reinforcement Learning

Reinforcement learning [21] is used to describe and solve problems in which agents learn strategies to maximize rewards in the decision-making process when interacting with the environment. The interaction process can be modelled as a Markov Decision Process (MDP) [22]. A MDP can be denoted by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma \rangle$, where $\mathcal{S}$ is the set of states, $A$ is the set of actions, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(S)$ is the state transition function, $r : \mathcal{S} \times \mathcal{S}' \times \mathcal{A} \to \mathcal{P}(R)$ is the reward function and $\gamma$ is the discount factor. Reinforcement learning aims to maximize the sum of discounted rewards $R_t = \sum_{i=t}^{T} \mathbb{E}\left[\gamma^{(i-t)} r\left(s_i, s_{i+1}, a_i\right)\right]$, where $r\left(s_i, s_{i+1}, a_i\right)$ is the immediate reward received at timestep $i$. Policy gradient methods explicitly learn a policy to maximize cumulative return, and typical algorithms include Proximal Policy Optimization (PPO) [23] and Advantage Actor-Critic (A2C) [24].

### B. Determinantal Point Process

Determinantal Point Process (DPP) [16] is a random process to model subset selection and capture negative interactions. For a ground set $Y = \{1, 2, \ldots, N\}$, a DPP $\mathcal{P}$ is a probability measure on $2^Y$, the power set of $Y$. Suppose the nonzero probability is given to the empty set by a DPP $\mathcal{P}$. In that case, the probability of $S$ satisfies $\mathcal{P}(S) \propto \det\left(\mathbf{L}_S\right)$ for every subset $S \subseteq Y$, where $\mathbf{L}$ is a real, positive semidefinite (PSD) kernel matrix and $\mathbf{L}_S$ denotes the submatrix obtained by restricting to those rows and columns indexed by $S$. In practice, DPP is widely used in recommendation systems [25], [26] to generate diverse items for the user. Fast greedy MAP inference [25] provides a fast approach to select an item set $S$ that maximizes the MAP posteriori as

$$\arg\max_{S \subseteq Y} \det\left(\mathbf{L}_S\right). \tag{1}$$

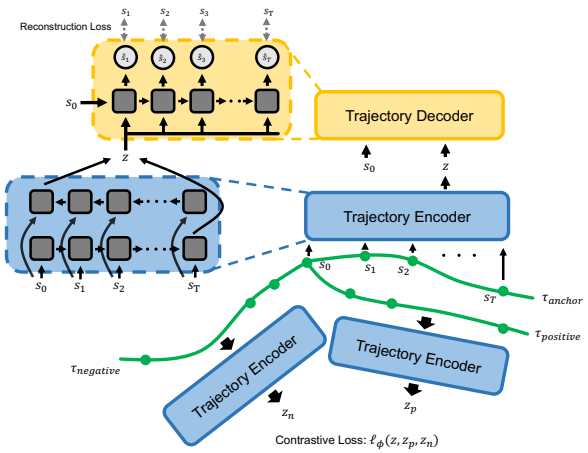



Fig. 1: The Structure of Trajectory Embedding Model. The upper half is VAE model structure and the lower half is contrastive learning process.

## III. Methodology

We will introduce the trajectory embedding model through VAE and contrastive learning in the first subsection. Then our hierarchical training framework and DPP-based selection and mutation mechanism are discussed in the following subsection. The whole hierarchical trajectory space exploration (HTSE) structure can be seen in Fig. 2.

### A. Trajectory Embedding Model

Unlike previous methods, which only focus on joint action space or outcome space, we aim to seek diversity in the trajectory embedding space in this work. The trajectory embedding model is shown in Fig. 1.

The trajectory embedding process is to map a trajectory sampled by a policy into a hidden representation. Our trajectory embedding is realized with Variational Autoencoder [18], [27]–[29] and LSTM [30]. The upper half part of Fig. 1 shows the trajectory encoder and trajectory decoder. The trajectory encoder is implemented by a double-layer bidirectional LSTM. A trajectory sequence input $\tau$:$s_0, s_1 ... s_T$ is fed into the trajectory encoder $q_\phi(z|\tau)$ and transformed into a hidden embedding vector $z$ with dimension $d$. Then with $z$ and $h_i$ as the last input except that the first input is $z$ and $s_0$, it will rollout to generate the next state $\hat{s}_{i+1}$ through trajectory decoder $p_\theta(\tau|z)$ implemented by the LSTM cell. Therefore, the decoded sequence $\tau'$:$s_0, \hat{s}_1 ... \hat{s}_T$ will be generated after $T$ timesteps' rollout.

In addition, as shown in the bottom half part of Fig. 1, we add a similarity constraint for the embedding $z$ with contrastive learning, unlike traditional trajectory embedding

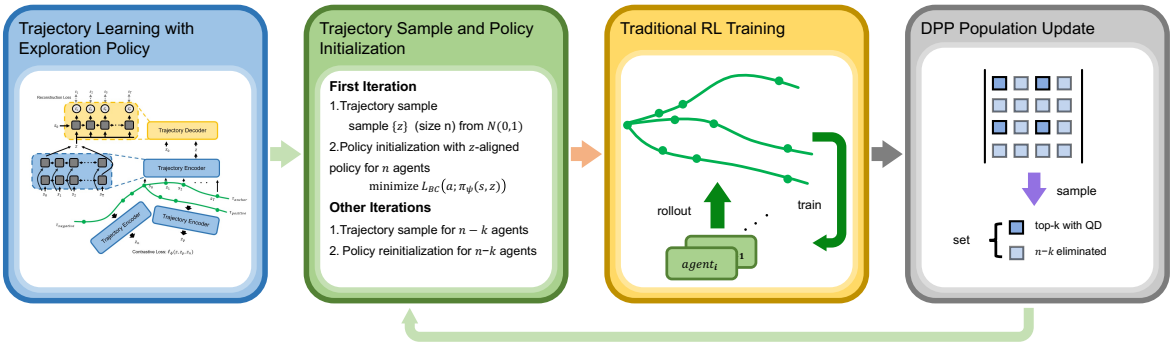



Fig. 2: Hierarchical trajectory space exploration process. The first graph is trajectory embedding model, the second one is policy initialization and reinitialization managed by meta controller, the third one is regular policy training along given trajectory directions, and the last one is DPP-based selection and mutation process.

approaches [31], [32] through VAE. This constraint is aimed to ensure that the representation of similar trajectories should be mapped close together, while distinct trajectories should be further away in the embedding space. For the anchor $\tau$, other trajectories will be sorted in an ascending order according to trajectory distance. The closest trajectory $\tau_p$ in the batch is chosen as the positive example and the latter half trajectories of the sorted trajectory batch $\tau_n$ are treated as negative examples. In practice, trajectory distance between $\tau_i$ and $\tau_j$ is measured as the sum of point location distance along the trajectories. Extending the idea of SimCLR [19], the formulation of contrastive loss can be written as

$$\begin{aligned}&\ell_\phi\left(\tau,\tau_p;\mathcal{T}_n\right)\\&=-\log\frac{e^{\lambda\mathrm{sim}\left(q_\phi(\tau),q_\phi(\tau_p)\right)}}{e^{\lambda\mathrm{sim}\left(q_\phi(\tau),q_\phi(\tau_p)\right)}+\sum_{\tau_n\in\mathcal{T}_n}e^{\lambda\mathrm{sim}\left(q_\phi(\tau),q_\phi(\tau_n)\right)}},\end{aligned}\tag{2}$$

with $\mathrm{sim}(u,v)=\frac{u^Tv}{\|u\|\|v\|}$ denoting the cosine similarity.

Furthermore, we have a $z$-aligned policy $\pi_\psi(s,z)$ which is required to generate a trajectory that aligns with $z$ and its corresponding trajectory. Hence given a hidden variable $z$, we can recover the $z$-corresponding trajectory through the $z$-aligned policy. In practice, it is trained by behavior cloning since we can obtain state and action information from the input trajectory directly. Given $\langle s,a\rangle$ pairs in the trajectory $\tau$ and trajectory embedding $z$, the behavior cloning loss is formulated as

$$\mathcal{L}_{BC}=\mathcal{L}_\psi\left(a;\pi_\psi(s,z)\right)=\frac{1}{2}\left(\pi_\psi(s,z)-a\right)^2.\tag{3}$$

The whole objective of the trajectory embedding model is written as

$$\max_{\theta,\phi,\psi}\ \log p_\theta(\tau)-\alpha\ell_\phi\left(\tau,\tau_p;\mathcal{T}_n\right)-\mathcal{L}_{BC}.\tag{4}$$

Introducing the evidence lower bound (ELBO) in place of the marginal likelihood $\log(p_\theta(\tau))$, we obtain the final **optimizing objective** as

$$\begin{aligned}&\log p_\theta(\tau)-\alpha\ell_\phi\left(\tau,\tau_p;\mathcal{T}_n\right)-\mathcal{L}_{BC}\\&\geq\mathbb{E}_{q_\phi}\left[\log p_\theta(\tau|z)\right]-D_{KL}\left(q_\phi(z|\tau)\|p(z)\right)\\&\quad-\alpha\ell_\phi\left(\tau,\tau_p;\mathcal{T}_n\right)-\mathcal{L}_{BC}.\end{aligned}\tag{5}$$

Intuitively, it aims to maximize the log-likelihood of the trajectory sequence $p(\tau)$ by optimizing the ELBO, and ensure the trajectories generated by $z$-aligned policy are aligned with the $z$-corresponding trajectory. At the same time, additional contrastive learning loss $\ell_\phi$ is added to make sure that similar trajectories are similar in the embedding space while distinct trajectories' representations should be further away in the embedding space. $\alpha$ is used to balance the VAE loss and contrastive learning loss.

*1) **Trajectory Encoder**:* Optimizing the objective in Eq. (5) with respect to $\phi$ maximizes $\mathbb{E}_{q_\phi}\left[\log p_\theta(\tau|z)\right]-D_{KL}\left(q_\phi(z|\tau)\|p(z)\right)-\alpha\ell_\phi\left(\tau,\tau_p;\mathcal{T}_n\right)$. By optimizing the reconstruction error loss and KL loss, the first and the second term encourage the trajectory encoder to encode the trajectory sequence to a distinguishable and characteristic space such that the decoder reconstructs it efficiently. The last term is to make sure the embedding space satisfies the similarity constraint.

*2) **Trajectory Decoder**:* Optimizing the objective with respect to $\theta$ maximizes the term $\mathbb{E}_{q_\phi}\left[\log p_\theta(\tau|z))\right]$. The term encourages the trajectory decoder to reconstruct the sequence to maximize the likelihood of the observed data.

*3) **z-aligned Policy**:* In practice, trajectory data from the environment consists of sequences of both states and actions. Behavior cloning loss $\mathcal{L}_{BC}$ can be directly applied to make sure trajectory generated by $\pi_\psi(s,z)$ is aligned with $z$ and its corresponding trajectory sequence.

Moreover, the training dataset of the trajectory embedding model is generated by another policy. To encourage it to continuously add unseen trajectories to the training dataset, we use the distance between trajectories newly generated by this policy and existing trajectories in the training dataset as the reward signal of this policy.

### *B. Hierarchical Trajectory Space Exploration*

The whole structure of hierarchical trajectory space exploration (HTSE) is shown in Fig. 2.

Assume that there are $N$ agents optimizing quality in the population, as shown in the second graph in Fig. 2. At the first iteration, the meta controller will generate $N$ different hidden variables $z$ from normal distribution into set $Z=$

$\{z_i, \forall i \in \{1, \ldots, N\}\}$ and their corresponding $z$-aligned policies into set $P = \{\pi_{\psi_i}, \forall i \in \{1, \ldots, N\}\}$. Agent policies are first initialized by cloning the $z$-aligned policy network weights or using behavior cloning methods. Then they are optimized towards maximizing the environment return with common reinforcement learning methods, as shown in the third graph. This is to encourage the agent $i$ to explore the $z_i$-corresponding trajectory space. Therefore, the population agents are expected to do training and simple exploration in different parts of the trajectory space.

After every $M$ training iterations, the meta controller will apply DPP-based selection and mutation mechanism to do exploitation and exploration in trajectory space, as shown in the fourth and the second graph. The selection process is to select the top $K$ policies with high quality and diversity. Furthermore, the mutation process is to mutate the remaining $N$-$K$ variables to encourage the policy to discover other possible directions in the trajectory space. In detail, top $K$ policies will continue training, and the meta controller will sample new hidden variables $z$ from normal distribution and reinitialize other $N$-$K$ policies with the newly generated $z$-aligned policies. More details about DPP-based selection and mutation mechanism can be seen in the following subsection.

*1) **DPP-based Selection and Mutation Mechanism**:* Given $N$ policies' scaled returns $\mathcal{R} = \{r_i, \forall i \in \{1, \ldots, N\}\}$ and their corresponding deterministic normalized trajectory embedding vectors $\mathcal{Z}' = \{z_i : z_i = \frac{q_\phi(\tau_i)}{\|q_\phi(\tau_i)\|}, \forall i \in \{1, \ldots, N\}\}$, we are supposed to select the subset $D$ composed of top $K$ policies with high quality and diversity.

We can first construct diagonal return matrix $\mathrm{Diag}(\mathbf{r})$ and similarity matrix $\mathbf{S}$, where $\mathrm{Diag}(\mathbf{r})$ is a diagonal matrix with entries $r_i$, and $S_{ij}$ means the cosine similarity between $z_i$ and $z_j$. Note that the deterministic trajectory here means the trajectories generated by the deterministic policy. In practice, deterministic policy always takes the action with the max likelihood for discrete action space and the action without noise for continuous action space.

The kernel matrix $\mathbf{L}$ in DPP can be formulated as a Gram matrix $\mathbf{L} = \mathbf{B}^\top \mathbf{B}$ [16]. Furthermore, by expressing column vector $B_i$ as the product of the expected return $r_i$ of policy and normalized embedding vector $z_i \in \mathbb{R}^d$, we have

$$L_{ij} = \langle B_i, B_j \rangle = \langle r_i z_i, r_j z_j \rangle = r_i r_j \langle z_i, z_j \rangle , \qquad (6)$$

where $L_{ij}$ is the entry of $\mathbf{L}$ and $\langle z_i, z_j \rangle$ is the cosine similarity between $z_i$ and $z_j$. Thus we can write the kernel matrix as $\mathbf{L} = \mathrm{Diag}(\mathbf{r}) \cdot \mathbf{S} \cdot \mathrm{Diag}(\mathbf{r})$. The log-probability of $D$ being selected is

$$\log\det(\mathbf{L}_D) = \sum_{i \in D} \log\left(r_i^2\right) + \log\det(\mathbf{S}_D) . \qquad (7)$$

When the trajectory embeddings in $D$ are orthogonal, the second term in Eq. (7) will achieve the maximum value. This objective considers both trajectories' return and differences, which is aligned with our QD-RL objective. To balance quality and diversity, we can construct a new DPP with kernel matrix $\mathbf{L}' = \mathrm{Diag}(\exp(\beta \mathbf{r})) \cdot \mathbf{S} \cdot \mathrm{Diag}(\exp(\beta \mathbf{r}))$, where $\beta = \theta'/(2(1-\theta'))$ and $\theta' \in [0, 1]$. Hence we have

$$\log \mathcal{P}(D) \propto \theta' \cdot \sum_{i \in D} r_i + (1 - \theta') \cdot \log\det(\mathbf{S}_D) . \qquad (8)$$

Finally, we are supposed to select subset $D$ to optimize the MAP inference via

$$\arg\max_D \det(\mathbf{L}_D) , \qquad (9)$$

which is equal to choosing a policy subset maximizing both quality and diversity. In practice, we use the fast greedy MAP algorithm [25] to choose $D$ composed of top $K$ quality-diversity policies. Finally, top $K$ policies considering quality and diversity will be selected and the other $N$-$K$ policies will be mutated in the meta controller as is revealed before.

## IV. Experiments

In this section, we compare HTSE with other QD-RL methods in three challenging environments in terms of quality and diversity. And then trajectory embedding model and DPP-based selection and mutation mechanism are discussed.

### A. Environment Tasks

*1) POINT Env:* As shown in Fig. 4a, POINT Environment is a MuJoCo environment for navigation. In the beginning, the agent is placed at a random place on the left, and it is required to navigate to the goal on the right. The reward is the negative distance away from the goal. However, there is a wall that separates the agent from the goal. And both the wall and the goal are not visible to the agent. So this is a deceptive reward task, and strong exploration ability is needed. In detail, The observation of the agent at time $t$ contains its coordinates $[x_t, y_t, z_t]^\mathsf{T}$ and its speed $[v_x, v_y, v_z]^\mathsf{T}$. The action of the agent is the position increment along axes $x$ and $y$. The reward is calculated as $r_t = -(x_t - x_{\text{goal}})^2 - (y_t - y_{\text{goal}})^2$. The maximum step of the environment is 200.

*2) MAZE Env:* MAZE is more complicated than POINT, as shown in Fig. 4b. In the beginning, the agent is placed at a random place, and it must learn to pass three corridors to reach the destination. In detail, the observation, reward and action are the same as those in the POINT Env. The maximum step of the environment is 400. This is a challenging task that requires strong exploration ability since agent cannot see the location of destination.

*3) HIWAY Env:* HIWAY is a collection of environments for autonomous driving tasks. We do experiments on the intersection scenario, where negotiation ability is required. In this task, the car is assigned a straight route while some cars drive in orthogonal lanes. The ego car is required to reach its destination without collision. The observation of the car contains its location $[x_t, y_t]^\mathsf{T}$, its speed $[v_x, v_y]^\mathsf{T}$ and nearby cars' locations and speeds. The reward is calculated as $r_t = r_{collision} + r_{distance}$, where $r_{distance} = -(x_t - x_{\text{goal}})^2 - (y_t - y_{\text{goal}})^2$ and $r_{collision} = -50$. The action of the agent is a continuous value between $-1$ and 1, which represents the magnitude of deceleration or acceleration relatively. In Fig. 4c, the yellow one is the ego car and blue ones are the social cars. Agents are expected to develop multiple driving styles on the premise of reaching the goal safely.

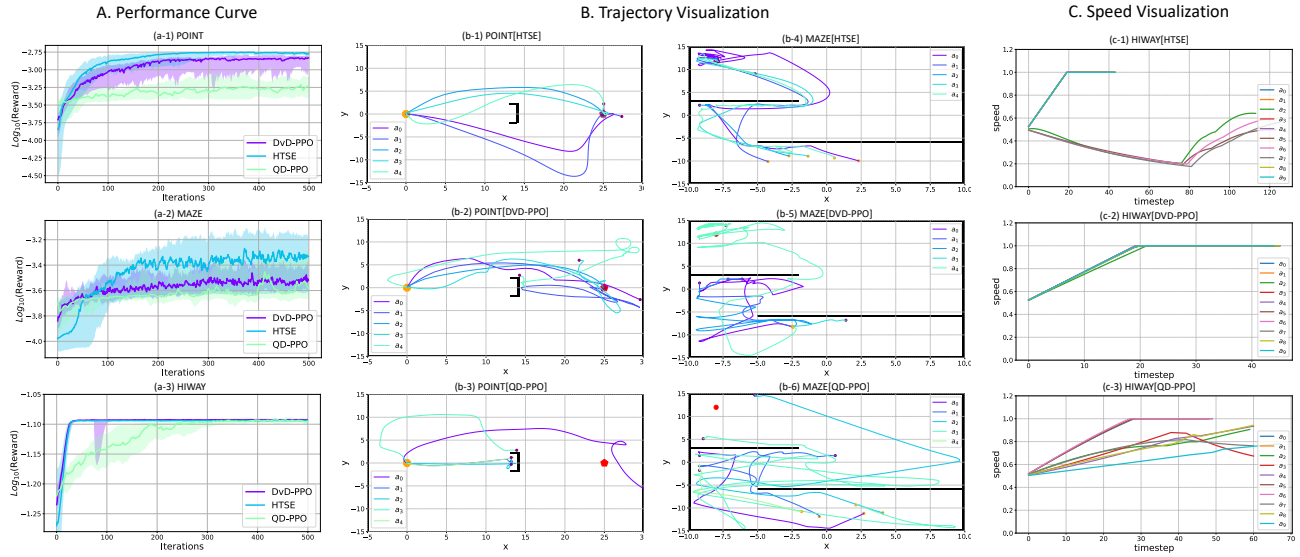
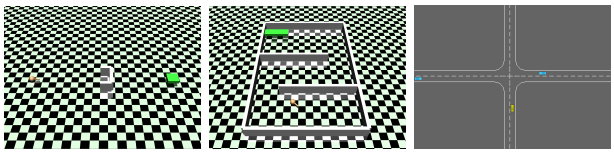



Fig. 3: Experimental results of performance and diversity. Figure $a$ depicts reward curves of median-performance agent across 5 seeds in three environments, where the shadow represents the $95\%$ confidence interval. Figure $b$ presents trajectories' visualization in POINT and MAZE, and figure $c$ presents speed vs. timestep curves in HIWAY with respect to three methods.



(a) POINT Env (b) MAZE Env (c) HIWAY Env

Fig. 4: MuJoCo and Driving Tasks

### B. Experimental Results

We compare the overall performance with the other two QD-RL methods, including DvD [15] and QD [17]. For a fair comparison, the RL algorithm in DvD, QD, and HTSE is PPO [23], and the hyperparameters are set to be the same. We run all the algorithms for $5$ seeds, and show the reward curves of median-performance agent across five seeds in Fig. 3a as [15], [17] do. Due to the extensive reward range, we transform reward $r$ to $sign(r)\cdot\log(abs(r))$. As we can see, our algorithm achieves the best performance in all three environments.

DvD-PPO's performance in POINT is close to ours, but the gap is enlarged in MAZE. This indicates that exploring in the action space is not enough for such deceptive reward. DvD-PPO cannot achieve higher performance since its training process is constantly disturbed by the Determinant objective gradients. Hence it sacrifices part of the performance for additional exploration ability. QD-PPO gets the lowest performance in three environments. We think there are two reasons. First, the shared critic is not stable during training because samples of different policies are mixed together in the buffer. Second, diverse agents can not learn well since the novelty reward signal from outcome space is weaken for the beginning states. In HIWAY, the final performance of all algorithms looks similar because HIWAY is not an environment requiring high exploration ability. However, there is a certain gap between the convergence speed of QD-PPO and the others, which reveals the instability and poor exploration ability of QD-PPO.

Our algorithm HTSE does not sacrifice the performance since the agents are purely required to seek high performance, and seeking diversity is decoupled into the DPP-based selection and mutation process. Also, behavior searching space is enlarged into trajectory embedding space, which allows diverse trajectory-level exploration.

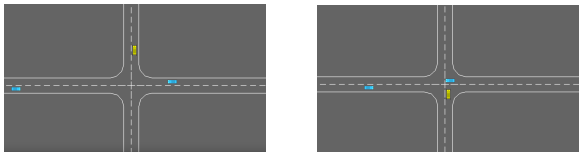
To measure the diversity between different methods, we choose the best $5$ agents in the population and visualize their trajectory locations in Fig. 3b for POINT and MAZE environment. The lines mean the trajectory point path, where the orange point, the purple point and the red point represent the start position, the terminal point and the goal, respectively. And the straight black lines represent the walls that block the agent. Besides, Fig. 3c presents the curve of agent's speed vs. timestep in HIWAY environment. And the initial speed of ego car is $0.5$ m/s.

As shown in Fig. 3b(1-3), all agents reach the destination successfully with our algorithm in POINT environment. Besides, bypassing the interval wall from the upper side and bottom side are both learned by our RL populations, which show excellent trajectory diversity. Even trajectories from the same side are also different from each other. However, the trajectories of DvD-PPO are mixed together and not so smooth, which demonstrates the agents trained by DvD-PPO are confused and disturbed by the diversity signal. In QD-PPO, most agents are stuck in the wall with only one agent bypassing the wall, and it shows the same impatience near the goal as DvD-PPO. Moreover, the agents in DvD-PPO and QD-PPO only learn to explore in the upper space of the wall, which demonstrates limited diversity in the population

of these two algorithms.

As for the MAZE environment shown in Fig. 3b(4-6), agents trained by our algorithms have better exploration ability with a higher success rate than the others. The trajectories also demonstrate diversity with different ways to pass the corridors. Most agents trained by DvD-PPO and QD-PPO are stuck by the second wall near $y = 2.5$ due to their poor exploration ability. This demonstrates that the superiority of doing exploration in the trajectory space over action space or outcome space.

As shown in Fig. 3c(1-3), all agents can reach the destination successfully with all three algorithms in HIWAY environment. However, only diverse driving styles emerge with the support of HTSE. All agents in DvD-PPO take the strategy that accelerates to the maximum speed at first and passes the intersection quickly. A similar strategy is taken by QD-PPO agents, but it differs that some agents do not achieve the maximum speed. However, HTSE develops two kinds of driving strategies. One is a rush strategy that passes the intersection as soon as possible to avoid collisions. The other is a wait strategy that slows the speed first to wait for other cars passing the intersection, and then accelerates to reach its destination. These two strategies can be observed in Fig. 5a and Fig. 5b, respectively. The results reveal that HTSE has the advantage of discovering diverse strategies than other algorithms.

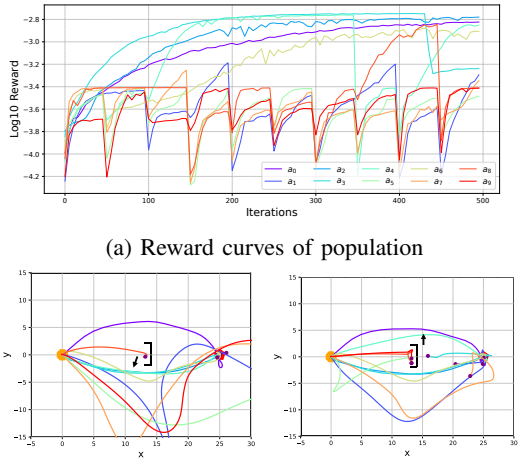


(a) Rush strategy (b) Wait strategy

Fig. 5: Driving strategies of HTSE

### C. DPP-based Selection and Mutation Process

In this subsection, we will figure out how the DPP-based selection and mutation process can help promote quality and diversity. As shown in Fig. 6a, we visualize all 10 agents' learning curves in the population in POINT environment. The mutation interval is 50, and the sharp drop of several curves indicates that the agent is mutated and required to explore other parts of trajectory space. Obviously, we can see at iteration 350, agent 4 is eliminated by the DPP framework although it has a very high performance, but then its performance increases near iteration 500. We visualize all trajectories at these two iterations in Fig. 6b and Fig 6c. The black arrow points to the trajectory of agent 4. We can see that at iteration 350, agent 4 is eliminated because of its high trajectory-level similarity to agent 2 and agent 3. They both bypass the wall from the bottom side and trajectories have many overlaps. However, we can see the mutated agent 4 later learns another policy that bypasses the wall through the upper space, which leads to higher population diversity. This example proves the significance of the DPP-based selection and mutation mechanism for promoting quality and diversity.




(a) Reward curves of population

(b) Trajectories after 350 training iterations (c) Trajectories after 500 training iterations

Fig. 6: Reward curves and trajectories visualization of DPP-based Selection and Mutation Process in POINT

### D. Training Details

In DvD-PPO, the hyperparameter $\lambda$, which balances expected return loss and kernel matrix determinants, is set to be $0.01$ after searching in $\{0.001, 0.01, 0.1, 0.5\}$. As for HTSE embedding model, $\lambda$ in contrastive loss is $0.07$, and $\alpha$ in total loss objective is $2$. The length of embedding vector $z$ is $8$. The default trajectory length $T$ for the embedding model is $50$ for POINT and MAZE while $30$ for HIWAY. The DPP interval $M$ is $50$ iterations for POINT, HIWAY and $100$ for MAZE. $\theta'$ in DPP kernel Matrix is $0.5$. The population size is $10$ for all algorithms. Moreover, $K$ is set to be the half population size for HTSE. Besides, As for PPO, the learning rate, batch size, $\gamma$ are set to be $0.0003$, $128$, $0.99$, respectively.

## V. Conclusion and Future Work

In this paper, we introduce HTSE, a hierarchical trajectory space exploration method in Quality Diversity Reinforcement Learning domain. Instead of defining behavior space as action space or outcome space like previous methods, which neglect trajectory characteristics during the execution process, we propose a trajectory embedding model with VAE and contrastive learning constraints. Besides, a DPP-based selection and mutation mechanism is used to help agents perform exploration and exploitation in the trajectory space. We conduct experiments across two MuJoCo deceptive reward tasks and an intersection negotiation driving task, where HTSE demonstrates strong exploration ability to achieve high-performance and diverse solutions.

In the future, we will try to extend our framework to long-horizon tasks which require outstanding exploration ability. Besides, how to extend QD framework to multi-agent reinforcement learning will also be an interesting direction.

## REFERENCES


[1] K. Shao, Z. Tang, Y. Zhu, N. Li, and D. Zhao, "A survey of deep reinforcement learning in video games," *arXiv preprint arXiv:1912.10944*, 2019.

[2] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[3] Y. Li, Y. Wen, D. Tao, and K. Guan, "Transforming cooling optimization for green data center via deep reinforcement learning," *IEEE transactions on cybernetics*, vol. 50, no. 5, pp. 2002–2013, 2019.

[4] M. M. Afsar, T. Crump, and B. Far, "Reinforcement learning based recommender systems: A survey," *arXiv preprint arXiv:2101.06286*, 2021.

[5] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv preprint arXiv:1703.03864*, 2017.

[6] 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.

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

[8] C. Colas, V. Madhavan, J. Huizinga, and J. Clune, "Scaling map-elites to deep neuroevolution," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 2020, pp. 67–75.

[9] J. Lehman and K. O. Stanley, "Evolving a diversity of virtual creatures through novelty search and local competition," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 2011, pp. 211–218.

[10] S. Kim, A. Coninx, and S. Doncieux, "From exploration to control: learning object manipulation skills through novelty search and local adaptation," *Robotics and Autonomous Systems*, vol. 136, p. 103710, 2021.

[11] 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.

[12] S. Khadka and K. Tumer, "Evolutionary reinforcement learning," *arXiv preprint arXiv:1805.07917*, vol. 223, 2018.

[13] C. Bodnar, B. Day, and P. Lió, "Proximal distilled evolutionary reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 3283–3290.

[14] A. Pourchot and O. Sigaud, "Cem-rl: Combining evolutionary and gradient-based methods for policy search," *arXiv preprint arXiv:1810.01222*, 2018.

[15] J. Parker-Holder, A. Pacchiano, K. Choromanski, and S. Roberts, "Effective diversity in population-based reinforcement learning," *arXiv preprint arXiv:2002.00632*, 2020.

[16] A. Kulesza and B. Taskar, "Determinantal point processes for machine learning," *arXiv preprint arXiv:1207.6083*, 2012.

[17] 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.

[18] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[19] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[20] A. Kulesza and B. Taskar, "Learning determinantal point processes," 2011.

[21] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

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

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

[24] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[25] L. Chen, G. Zhang, and H. Zhou, "Fast greedy map inference for determinantal point process to improve recommendation diversity," *arXiv preprint arXiv:1709.05135*, 2017.

[26] Y. Liu, Y. Zhang, Q. Wu, C. Miao, L. Cui, B. Zhao, Y. Zhao, and L. Guan, "Diversity-promoting deep reinforcement learning for interactive recommendation," *arXiv preprint arXiv:1903.07826*, 2019.

[27] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *International conference on machine learning*. PMLR, 2014, pp. 1278–1286.

[28] J. Walker, C. Doersch, A. Gupta, and M. Hebert, "An uncertain future: Forecasting from static images using variational autoencoders," in *European Conference on Computer Vision*. Springer, 2016, pp. 835–851.

[29] T. Duan and J. Lee, "Graph embedding vae: A permutation invariant model of graph structure," *arXiv preprint arXiv:1910.08057*, 2019.

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

[31] Z. Wang, J. S. Merel, S. E. Reed, N. de Freitas, G. Wayne, and N. Heess, "Robust imitation of diverse behaviors," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[32] J. Co-Reyes, Y. Liu, A. Gupta, B. Eysenbach, P. Abbeel, and S. Levine, "Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings," in *International conference on machine learning*. PMLR, 2018, pp. 1009–1018.