Model-based Reinforcement Learning for Parameterized Action Spaces

Renhao Zhang [*1] **Haotian Fu** [*1] **Yilin Miao** [1] **George Konidaris** [1]

Abstract

We propose a novel model-based reinforcement learning algorithm—Dynamics Learning and predictive control with Parameterized Actions (DLPA)—for Parameterized Action Markov Decision Processes (PAMDPs). The agent learns a parameterized-action-conditioned dynamics model and plans with a modified Model Predictive Path Integral control. We theoretically quantify the difference between the generated trajectory and the optimal trajectory during planning in terms of the value they achieved through the lens of Lipschitz Continuity. Our empirical results on several standard benchmarks show that our algorithm achieves superior sample efficiency and asymptotic performance than state-of-the-art PAMDP methods.[1]

1. Introduction

Reinforcement learning (RL) has gained significant traction in recent years due to its proven capabilities in solving a wide range of decision-making problems across various domains, from game playing (Mnih et al., 2015) to robot control (Schulman et al., 2015; Lillicrap et al., 2016). One of the complexities inherent to some RL problems is a discrete-continuous hybrid action space. For instance, in a robot soccer game, at each time step the agent must choose a discrete action (move, dribble or shoot) as well as the continuous parameters corresponding to that chosen discrete action, e.g. the location to move/dribble to. In this setting—the Parameterized Action Markov Decision Processes (PAMDP) (Masson et al., 2016)—each discrete action is parameterized by some continuous parameters, and the agent must choose them together at every timestep. PAMDPs model many applications of interest like RTS Games (Xiong et al., 2018) or Robot Soccer (Hausknecht & Stone, 2016b).

Compared to just discrete or continuous action space, Reinforcement Learning with Parameterized action space allows the agent to perform more structural exploration and solve more complex tasks with a semantically more meaningful action space (Hausknecht & Stone, 2016b). Recent papers provide various approaches for RL in the PAMDP setting (Bester et al., 2019; Fu et al., 2019; Fan et al., 2019; Li et al., 2022). However, to the best of our knowledge, all previous methods are model-free. By contrast, in continuous/discrete-only action spaces, deep model-based reinforcement learning has shown better performance than model-free approaches in many complex domains (Hafner et al., 2020; 2021; 2023; Hansen et al., 2022), in terms of both sample efficiency and asymptotic performance. We therefore seek to leverage the high sample efficiency of model-based RL in PAMDPs.

We propose a model-based RL framework tailored for PAMDPs: Dynamics Learning and predictive control with Parameterized Actions (DLPA). Our key innovations in terms of contextualizing model-based RL in PAMDPs are: 1. We propose three inference structures for the transition model considering the entangled parameterized action space. 2. We propose to update the transition models with H-step loss. 3. We propose to learn two separate reward predictors conditioned on the prediction for termination. 4. We propose an approach for PAMDP-specific MPPI. We empirically find that all these components significantly improves the algorithm's performance. We also provide theoretical analysis regarding DLPA's performance guarantee and sample complexity, which is the first theoretical performance bound for PAMDP RL algorithm, to the best of our knowledge. Our empirical results on 8 different PAMDP benchmarks show that DLPA achieves better or comparable asymptotic performance with significantly better sample efficiency than all the state-of-the-art PAMDP algorithms. We also find that DLPA succeeds in tasks with extremely large parameterized action spaces where prior methods cannot succeed without learning a complex action embedding space, and converges much faster. The proposed method even outperforms a method (Li et al., 2022) that has a customized action space compression algorithm as the original parameterized action space becomes larger. To the best of our knowledge, our work is the first method that successfully

[*]Equal contribution [1]Department of Computer Science, Brown University. Correspondence to: Renhao Zhang <rzhan160@cs.brown.edu>, Haotian Fu <hfu7@cs.brown.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

[1]Code available at https://github.com/Valarzz/DLPA.

applies model-based RL to PAMDPs.

2. Background

2.1. Parameterized Action Markov Decision Processes

Markov Decision Processes (MDPs) form the foundational framework for many reinforcement learning problems, where an agent interacts with an environment to maximize some cumulative reward. Traditional MDPs are characterized by a tuple $\{S, A, T, R, \gamma\}$, where S is a set of states, A is a set of actions, T denotes the state transition probability function, R denotes the reward function, and γ denotes the discount factor. Parameterized Action Markov Decision Processes (PAMDPs) (Masson et al., 2016) extend the traditional MDP framework by introducing the parameterized actions, denoted as a tuple $\{S, M, T, R, \gamma\}$, where M (unlike traditional MDPs) is the parameterized action space that can be defined as $M = \{(k, z_k)|z_k \in Z_k \text{ for all } k \in \{1, \cdots, K\}\}$, where each discrete action k is parameterized by a continuous parameter z_k, and Z_k is the space of continuous parameter for discrete action k and K is the total number of different discrete actions. Thus, we have the dynamic transition function $T(s'|s, k, z_k)$ and the reward function $R(r|s, k, z_k)$.

2.2. Model Predictive Control (MPC)

In Deep Reinforcement Learning, Model-free methods usually learn a policy parameterized by neural networks that learns to maximize the cumulative returns $\mathbb{E}_\tau[\sum_t \gamma^t R(s, a)]$. In the model-based domain, since we assume we have access to the learned dynamics model, we can use Model Predictive Control (Garcia et al., 1989) to plan and select the action at every time step instead of explicitly learn to approximate a policy function. Specifically, at time step t, the agent will first sample a set of action sequences with the length of horizon H for states $s_t : s_{t+H}$. Then it will use the learned dynamics model to take in actions as well as the initial states and get the predicted reward for each time step. Then the cumulative reward for each action sequence will be computed and the action sequence with the highest estimated return will be selected to execute in the real environment. Cross-Entropy Method (CEM) (Rubinstein, 1997) is often used together with this planning procedure, which works by iteratively fitting the parameters of the sampling distributions over the actions.

3. Related Work

Several model-free RL methods have been proposed in the context of deep reinforcement learning for PAMDPs. PADDPG (Hausknecht & Stone, 2016b) builds upon DDPG (Lillicrap et al., 2016) by letting the actor output a concatenation of the discrete action and the continuous parameters for each discrete action. Similar ideas are proposed in HPPO (Fan et al., 2019), which is based on the framework of PPO (Schulman et al., 2017). P-DQN (Xiong et al., 2018; Bester et al., 2019) is another framework based on the actor-critic structure that maintains a policy network that outputs continuous parameters for each discrete action. This structure has the problem of computational efficiency since it computes continuous parameters for each discrete action at every timestep. Hybrid MPO (Neunert et al., 2019) is based on MPO (Abdolmaleki et al., 2018) and it considers a special case where the discrete part and the continuous part of the action space are independent, while in this paper we assume the two parts are entangled. HyAR (Li et al., 2022) proposes to construct a latent embedding space to model the dependency between discrete actions and continuous parameters, achieving the best empirical performance among these methods. However, introducing another latent embedding space can be computationally expensive and the error in compressing the original actions may be significant in complex tasks. While all these methods can be effective in some PAMDP problems, to the best of our knowledge, no work has successfully applied model-based RL to PAMDPs, even though model-based approaches have achieved high performance and excellent sample efficiency in MDPs. Recent papers also make an effort to construct such parameterized action spaces from the primitive continuous action spaces (Fu et al., 2023b) leveraging the HiP-MDP assumption (Doshi-Velez & Konidaris, 2016; Killian et al., 2017; Fu et al., 2023a).

Most existing (Deep) Model-based Reinforcement Learning methods can be classified into two categories in terms of how the learned model is used. The first category of methods learns the dynamics model and plans for credit assignment (Ebert et al., 2018; Zhang et al., 2018; Janner et al., 2019; Hafner et al., 2019; Lowrey et al., 2019; Kaiser et al., 2020; Bhardwaj et al., 2020; Yu et al., 2020; Schrittwieser et al., 2020; Nguyen et al., 2021). A large number of algorithms in this category involves planning with random shooting (Nagabandi et al., 2018) or Cross-Entropy Method (Rubinstein, 1997; Chua et al., 2018). Okada & Taniguchi (2019) proposes a variational inference MPC method that introduces action ensembles with a Gaussian mixture model, considering planning with uncertainty-aware dynamics and continuous action space. However, our modification to the MPPI is to enable it to learn the optimal distribution for parameterized action space, which is composed of both discrete and continuous actions. The other way is to use the learned model to generate more data and explicitly train a policy based on that (Pong et al., 2018; Ha & Schmidhuber, 2018; Hafner et al., 2020; Sekar et al., 2020; Hafner et al., 2023), which is also known as Dyna (Sutton, 1990)-based methods. There are also algorithms combining model-free and model-based methods (Hansen et al., 2022). None of

these methods apply to the parameterized action (PAMDP) settings.

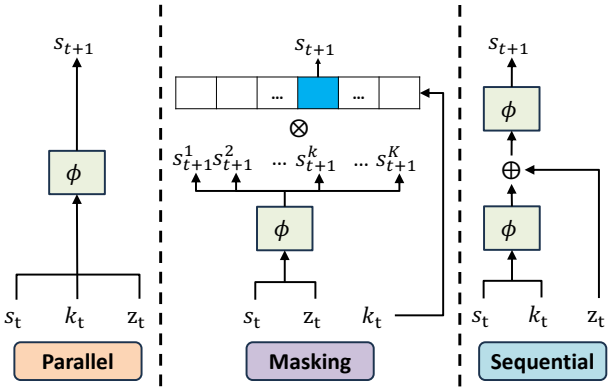

Figure 1. Three distinct inference architectures for the predictors. All the models are parameterized with ϕ.

4. Dynamics Learning and Predictive Control with Parameterized Actions

4.1. Dynamics Model with Parameterized Actions

To perform Model Predictive Control with Parameterized Actions, DLPA requires learning the following list of model components:

$$\text{Transition predictor: } \hat{s}_{t+1} \sim T_\phi(\hat{s}_{t+1}|s_t, k_t, z_{k_t}),$$

$$\text{Continue predictor: } \hat{c}_t \sim p_\phi(\hat{c}_t|s_{t+1}),$$

$$\text{Reward predictor: } \hat{r}_t \sim R_\phi(\hat{r}_t|s_t, k_t, z_{k_t}),$$

where we use c_t to denote the episode continuation flag. Given a state s_t observed at time step t, a discrete action k_t and the corresponding continuous parameter z_{k_t}, the transition predictor and the reward predictor T_ϕ and R_ϕ predict the next state $\hat{s}_{t+1}$ and reward $\hat{r}_{t+1}$ respectively. The Continue predictor outputs a prediction $\hat{c}_t$ for whether the trajectory continues at time step $t+1$ given s_{t+1}. All the components are implemented in the form of neural networks and we use ϕ to denote the combined network parameters. Specifically, the first three components are implemented with networks with stochastic outputs. i.e., we model the output distribution as a Gaussian and the outputs of the networks are mean and standard deviation of the distribution. We leverage reparameterization trick (Kingma & Welling, 2014) to allow computing gradients for the sampled $\hat{s}, \hat{r}, \hat{c}$'s.

PAMDP-specific inference models. In DLPA, we consider three distinct inference architectures for the predictors, as depicted in Figure 1. Specifically, the first model directly takes in the concatenation of s_t, k_t, z_{k_t} and infers the next state. The discrete component and the continuous component are treated in parallel. The second model first takes in s_t and z_{k_t} to infer all the K possible separate predictions for each discrete action, from which the relevant prediction is chosen based on k_t through masking. A similar framework is used in Xiong et al. (2018) as the policy. The third method treats the discrete component and continuous component sequentially by decoupling the prediction process. It first infers a latent embedding based on s_t and k_t, then concatenates the latent embedding with z_t and predicts s_{t+1}. Intuitively, the agent will be forced to predict a range of the outcome based on only the discrete action, and then predict the more precise outcome given the continuous parameters. Notably, the latter two methods are tailored to predict future outcomes by considering the intrinsic structure of parameterized actions. In the experimental section, we demonstrate that the third method exhibits the best overall performance.

H-step prediction loss. We train the components listed in section 4.1 through minimizing the loss below:

$$\begin{aligned}\mathcal{L}_{joint} = \mathbb{E}_{\{s_t,k_t,z_{k_t},r_t,s_{t+1},c_t\}_{t_0:t_0+H}} \sum_{t_0}^{t_0+H} \beta^{t-t_0} \Big\{ \\ \lambda_1 \|T_\phi(\hat{s}_{t+1}|\hat{s}_t, k_t, z_{k_t}) - s_{t+1}\|_2^2 + \\ \lambda_2 \|R_\phi(\hat{r}_{t+1}|\hat{s}_t, k_t, z_{k_t}) - r_{t+1}\|_2^2 + \\ \lambda_3 \|p_\phi(\hat{c}_t|\hat{s}_{t+1}) - c_t\|_2^2 \Big\},\end{aligned} \tag{1}$$

where $\hat{s}_{t_0} = s_{t_0}$. We use H to denote the planning horizon and t_0 to denote the start time step. β denotes the hyperparameter we use to control the weight of the loss term. The weight of the sum of the loss will be lower if t is closer to the end time step $t_0 + H$. λ denotes the weight of each prediction loss term.

Specifically, at each training step, we first sample a batch of trajectories $\{s_t, k_t, z_{k_t}, r_t, s_{t+1}, c_t\}_{t_0:t_0+H}$ from a replay buffer. Then we do the inference procedures as shown in Figure 2 Left. We give the dynamics model the sampled $s_{t_0}, k_{t_0}, z_{k_{t_0}}$ at first, and we get the predictions of next state $\hat{s}_{t_0+1}$, reward $\hat{r}_{t_0}$ and continuation flag $\hat{c}_{t_0}$ for the first time step. Then, we iteratively let our dynamics model predict the transitions with the sampled parameterized actions and **the predicted state from last time step**. At the end we take the weighted sum of all the prediction losses for state, reward and termination and update the model as described in Equation 1. And the gradients from the loss term for the last time step $t_0 + H$ will be backpropagated all the way to the first inference at time step t_0. That is to say, what we give our model as input during training is $\{s_{t_0}, k_{t_0}, z_{k_{t_0}}, k_{t_0+1}, z_{k_{t_0+1}}, \cdots, k_{t_0+H}, z_{k_{t_0+H}}\}$ which contains only the start state s_{t_0} without the other ground truth states in the trajectory. And we use this information to infer all the other information contained in the sampled

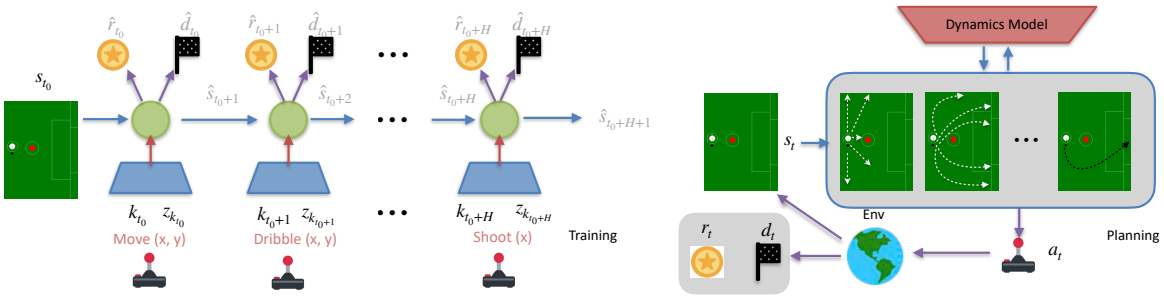

Figure 2. Left: Inference of dynamics during training. **Variables colored with default black are those we feed as input to the dynamics model. Variables colored with grey are those generated from the dynamics model.** Right: Planning and interacting with the environment. At each time step we execute only the first action from the sampled trajectory. White lines are example rollout trajectories from DLPA.The black line denotes the final selected rollout trajectory for one planning step.

trajectory and calculate the loss. As we will plan into the future with the exact same length H during planning, our choice allows gradients to flow back through time and assign credit more effectively than the alternative. We empirically find out that, by learning to infer several steps into the future instead of just the next step, the downstream planning tends to get better performance and thus benefits the whole training process.

By focusing on multi-step ahead predictions, the dynamic model develops a better understanding of the long-term consequences of actions, including the nuanced effects of different parameterized actions. Intuitively, in PAMDPs, the impact of actions can be highly dependent on the specific parameters chosen, and **the effects may only become apparent over several time steps**. i.e., choosing the same discrete action but different continuous parameters at current timestep may not result in much difference for the next state, but the compounding error will be significant after a few more steps.

Separate reward predictors. In practice, we find that learning two separate reward predictors can greatly improve Model-based RL algorithm's performance in the PAMDP context. Specifically, we learn one reward predictor for the case when the episode continuation flag $\hat{c}_t = 1$ and another predictor when $\hat{c}_t = 0$. The empirical performance comparison can be found in Table 2. We hypothesize that in PAMDP, the outcome of an action can be highly specific to the combination of its discrete and continuous components. i.e., with the same discrete action, only a small range of continuous parameters can lead to the terminal state. If we do not separate the prediction, the reward model is required first accurately predict the impact of very precise actions on the termination condition and also maintain a combined reward prediction for the two cases. Separate reward models allow for a more detailed understanding of how different action parameters influence outcomes in critical (terminal) versus normal (non-terminal) scenarios.

4.2. MPC with Parameterized Actions

Now we introduce the planning part for the proposed algorithm. The planning algorithm is based on Model Predictive Path Integral (Williams et al., 2015), adapted for the PAMDP setting (**PAMDP-specifc MPPI**). The main modification we make when applying MPPI to PAMDPs is that we keep a separate distribution over the continuous parameters for each discrete action and update them at each iteration instead of keeping just one independent distribution for the discrete actions and one independent distribution for the continuous parameters. In other words, we let the distribution of the continuous parameters condition on the chosen discrete action during the sampling process. This is an important change to make when using MPPI for PAMDPs as we wish to avoid discarding the established dependency between the continuous parameter and corresponding the discrete action in PAMDPs.

Specifically, we model the discrete action k follows a categorical distribution:

$$k \sim \text{Cat}(\theta_1^0, \theta_2^0, \cdots, \theta_K^0), \sum_{k=1}^{K} \theta_k^0 = 1,\ \theta_k^0 \geq 0, \tag{2}$$

where the probability for choosing each discrete action k is given by θ_k^0. For the continuous parameter z_k corresponding to each discrete action k, we model it as a multivariate Gaussian:

$$z_k \sim \mathcal{N}(\mu_k^0, (\sigma_k^0)^2 I),\ \mu_k^0, \sigma_k^0 \sim \mathbb{R}^{|z_k|}. \tag{3}$$

At the beginning of planning, we initialize a set of independent parameters $\mathcal{C}^0 = \{\theta_1^0, \cdots, \theta_K^0, \mu_1^0, \sigma_1^0, \cdots, \mu_K^0, \sigma_K^0\}_{t:t+H}$ for each discrete action and continuous parameter over a horizon with length H. Recall that K is the total number of discrete actions. Note that next we will update these distribution parameters for E iterations, so for each iteration j, we will have $\mathcal{C}^j = \{\theta_1^j, \cdots, \theta_K^j, \mu_1^j, \sigma_1^j, \cdots, \mu_K^j, \sigma_K^j\}_{t_0:t_0+H}$.

Then, for each iteration j, we independently sample N trajectories by independently sampling from the action distributions at every time step t and forwarding to get the trajectory with length H using the dynamics models T_ϕ, R_ϕ, p_ϕ as introduced in the last section. For a sampled trajectory $\tau = \{s_{t_0}, \hat{k}_{t_0}, \hat{z}_{k_{t_0}}, \hat{s}_{t_0+1}, \hat{r}_{t_0}, \hat{c}_{t_0}, \cdots, \hat{s}_{t_0+H}, \hat{k}_{t_0+H}, \hat{z}_{k_{t_0+H}}, \hat{s}_{t_0+1+H}, \hat{r}_{t_0+H}, \hat{c}_{t_0+H}\}$, we can calculate the cumulative return $\mathcal{J}_\tau$ with:

$$\mathcal{J}_\tau = \mathbb{E}_\tau[\sum_{t=t_0}^{t_0+H} \gamma^t c_t R_\phi(\hat{s}_t, \hat{k}_t, \hat{z}_{k_t})], \text{where } \hat{s}_{t_0} = s_{t_0}. \quad (4)$$

Let $\Gamma_{k,\tau} = \{\hat{k}_t\}_{t=t_0:t_0+H}$ denote the discrete action sequences and $\Gamma_{z,\tau} = \{\hat{z}_{k_t}\}_{t=t_0:t_0+H}$ denote the continuous parameter sequences within each trajectory τ. Then based on the cumulative return of each trajectory, we select the trajectories with top-n cumulative returns and update the set of distribution parameters $\mathcal{C}^j$ via:

$$\theta_k^j = (1-\alpha)\frac{\sum_{i=1}^n e^{\xi \mathcal{J}_{\tau_i}} \Gamma_{k,\tau_i}}{\sum_{i=1}^n e^{\xi \mathcal{J}_{\tau_i}}} + \alpha \theta_k^{j-1}, \quad (5)$$

$$\mu_k^j = (1-\alpha)\frac{\sum_{i=1}^n e^{\xi \mathcal{J}_{\tau_i}} \Gamma_{z,\tau_i} \mathbb{1}\{\Gamma_{k,\tau_i} == k\}}{\sum_{i=1}^n e^{\xi \mathcal{J}_{\tau_i}}} + \rho_k \mu_k^{j-1}, \quad (6)$$

$$\sigma_k^j = (1-\alpha)\sqrt{\frac{\sum_{i=1}^n e^{\xi \mathcal{J}_{\tau_i}} (\Gamma_{z,\tau_i} - \mu_k^j)^2 \mathbb{1}\{\Gamma_{k,\tau_i} == k\}}{\sum_{i=1}^n e^{\xi \mathcal{J}_{\tau_i}}}} + \rho_k \sigma_k^{j-1}, \quad (7)$$

where $\rho_k = 1 - \mathbb{1}\{\sum_{i=1}^n \mathbb{1}\{\Gamma_{k,\tau_i} == k\} > 0\}(1-\alpha)$, and we use ξ to denote the temperature that controls the trajectory-return weight. Intuitively, for each iteration j, we update our sampling distribution over the discrete and continuous actions weighted by the expected return as the returns come from these actions by forwarding our learned dynamics model. The discrete actions and continuous parameters that achieve higher cumulative return will be more likely to be chosen again during the next iteration. For updating the continuous parameters, we add a indicator function as we only want to update the corresponding distributions for those continuous parameters corresponding to the selected k. The updated distribution parameters at each iteration is calculated in the form of a weighted sum of the new value derived from the returns and the old value used at the last time step. We use a hyperparameter α to control the weights.

After E iterations of updating the distribution parameters, we sample a final trajectory from the updated distribution and execute only the first parameterized action, which is also known as receding-horizon MPC similar to previous work (Hansen et al., 2022). Then we move on to the next time step and do all the planning procedures again.

The overall algorithm is described in Algorithm 1. At each environment time step, the agent executes E steps of forward planning while updating the distribution parameters over discrete actions and continuous parameters. Then it uses the first action sampled from the final updated distribution to interact with the environment and add the new transitions into the replay buffer $\mathcal{B}$. Then if it is in training phase, the agent samples a batch of trajectories from the replay buffer, using the steps introduced in Section 4.1 to compute the loss and update the dynamics models.

5. Analysis

In this section, we provide some theoretical performance guarantees for DLPA. We quantify the estimation error between the cumulative return of the trajectory generated from DLPA and the optimal trajectory through the lens of Lipschitz continuity. All the proofs can be found in the appendix.

Definition 5.1. A PAMDP is (L_T^S, L_T^K, L_T^Z)-Lipschitz continuous if, for all $s \in S$, $k \in \{1, \cdots, K\}$, and $z \in Z$ where $z \sim \omega(\cdot|k)$[2]:

$$W(T(\cdot|s_1,k,\omega)), T(\cdot|s_2,k,\omega))) \leq L_T^S d_S(s_1,s_2)$$

$$W(T(\cdot|s,k_1,\omega)), T(\cdot|s,k_2,\omega))) \leq L_T^K d_K(k_1,k_2)$$

$$W(T(\cdot|s,k,\omega_1), T(\cdot|s,k,\omega_2)) \leq L_T^Z d_Z(\omega_1,\omega_2),$$

where W denotes the Wasserstein Metric and ω denotes the distribution over the continuous parameters given a discrete action type k. d_S, d_K, d_Z are the distance metrics defined on space S, K, Z. Similarly, we can define a (L_R^S, L_R^K, L_R^Z)-Lipschitz PAMDP. Note that as $k_1, k_2, \cdots$ are discrete variables, we use Kronecker delta function as the distance metric: $d_K(k_j, k_i) = 1, \forall i \neq j$. The definition can be seen as an extension of the Lipschitz continuity assumption for regular MDP (Rachelson & Lagoudakis, 2010; Pirotta et al., 2015; Asadi et al., 2018; Gelada et al., 2019) to PAMDP. Furthermore, in this paper, we follow the Lipschitz model class assumption (Asadi et al., 2018).

Assuming the error of the learned transition model $\hat{T}$ and reward model $\hat{R}$ are bounded by ϵ_T and ϵ_R respectively: $W(T(s,k,\omega), \hat{T}(s,k,\omega)) \leq \epsilon_T, |R(s,k,\omega) - \hat{R}(s,k,\omega)| \leq \epsilon_R$, for all s, k, ω. We can derive the following theorem:

Theorem 5.2. *For a $(L_R^S, L_R^K, L_R^Z, L_T^S, L_T^K, L_T^Z)$-Lipschitz PAMDP and the learned DLPA ϵ_T-accurate transition model $\hat{T}$ and ϵ_R-accurate reward model $\hat{T}$, let $L_{\hat{T}}^S = \min\{L_T^S, L_{\hat{T}}^S\}$, $L_{\hat{T}}^K = \min\{L_T^K, L_{\hat{T}}^K\}$, $L_{\hat{T}}^Z = \min\{L_T^Z, L_{\hat{T}}^Z\}$ and similarly define $L_{\hat{R}}^S$, $L_{\hat{R}}^K$, $L_{\hat{R}}^Z$. If $L_{\hat{T}}^S < 1$, then the regret of the rollout trajectory*

[2]We define the distribution over z given k and use it in all the following theoretical analysis to be consistent with the proposed method described in Section 4.2: for each k, we keep a distribution over z and sample from it to roll out the trajectory.

$\hat{\tau} = \{\hat{s_1}, \hat{k_1}, \hat{\omega_1}(\cdot|\hat{k_1}), \hat{s_2}, \hat{k_2}, \hat{\omega_2}(\cdot|\hat{k_2}), \cdots\}$ *from DLPA is bounded by:*

$$
\begin{aligned}
&|\mathcal{J}_{\tau^*} - \mathcal{J}_{\hat{\tau}}| := \sum_{t=1}^{H} \gamma^{t-1} |R(s_t, k_t, \omega_t(\cdot|k_t)) - \hat{R}(\hat{s}_t, \hat{k}_t, \hat{\omega}_t(\cdot|\hat{k}_t))| \\
&\leq \mathcal{O}\Big((L_R^K + L_R^S L_T^K)m \\
&+ H(\epsilon_R + L_R^S \epsilon_T + (L_R^Z + L_R^S L_T^Z)(\frac{m}{H}\Delta_{k,\hat{k}} + \Delta_{\omega,\hat{\omega}}))\Big),
\end{aligned} \tag{8}
$$

where $m = \sum_{t=1}^{H} \mathbb{1}(k_t \neq \hat{k}_t)$, $\Delta_{\omega,\hat{\omega}} = W(\omega(\cdot|k), \hat{\omega}(\cdot|k))$, $\Delta_{k,\hat{k}} = W(\omega(\cdot|k), \omega(\cdot|\hat{k}))$, N denotes the number of samples and H is the planning horizon.

Imagine you are navigating through a forest (the environment) to find the best path to a treasure (optimal trajectory). You have a map (DLPA's model) that predicts paths based on your current location and chosen direction. However, this map is not perfect—it sometimes inaccurately predicts where a path might lead (estimation errors). Theorem 5.2 essentially tells us that even with an imperfect map, if we know how "sensitive" the forest's paths are to changes in direction (Lipschitz continuity), and we understand the inaccuracies of our map (model errors), we can still confidently say that the path we choose won't be significantly worse than the best possible path, within a calculable margin.

Theorem 5.2 quantifies how the following categories of estimation error will affect the cumulative return difference between the rollout trajectories of DLPA and the optimal trajectories: 1. The estimation error m for the discrete action k. 2. The estimation error $\Delta_{k,\hat{k}} + \Delta_{\omega,\hat{\omega}}$ for the distribution ω over the continuous parameters. 3. The transition and reward model estimation error ϵ_T, ϵ_R. It also shows how the smoothness of the transition function and reward function will affect DLPA's performance. We also provide a bound for the multi-step prediction error (compounding error) of DLPA in Appendix Theorem B.1.

The following lemma further shows how the estimation error $\frac{m}{H}\Delta_{k,\hat{k}} + \Delta_{\omega,\hat{\omega}}$ for the continuous parameters changes with respect to the number of samples and the dimentionality of the space of continuous parameters.

Lemma 5.3. *Let $|Z|$ denote the cardinality of the continuous parameters space and N be the number of samples. Then, with probability at least $1 - \delta$:*

$$
\frac{m}{H}\Delta_{k,\hat{k}} + \Delta_{\omega,\hat{\omega}} \leq \frac{2m}{H}\sqrt{|Z|} + \frac{2}{N}\ln\frac{2|Z|}{\delta}. \tag{9}
$$

6. Experiments

We evaluated the performance of DLPA on eight standard PAMDP benchmarks, including Platform and Goal (Masson et al., 2016), Catch Point (Fan et al., 2019), Hard Goal and four versions of Hard Move. **Note that these 8 benchmarks are exactly the same environments tested in Li et al. (2022)**, which introduced the algorithm (HyAR) that reaches state-of-the-art performance in these environments. We have provided a short description for each environment in Appendix C. We also evaluated our method in Half-Field-Offense (HFO) (Hausknecht & Stone, 2016b), which is a complex domain with longer task horizon. The results can be found in Appendix I.

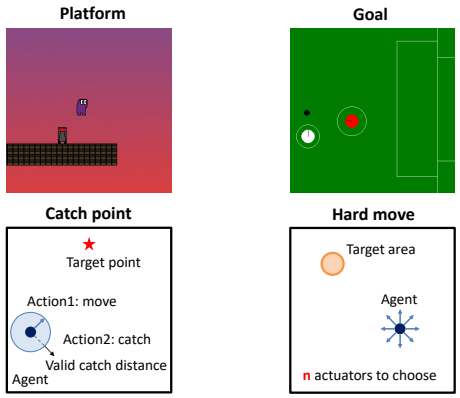

Figure 3. Visualization of the tested environments.

Implementation details. We parameterize all models using MLPs with stochastic outputs. The planning Horizon for all tasks is chosen between $\{5, 8, 10\}$. During planning, we select the trajectories with top-$\{100, 400\}$ cumulative returns, and we run 6 planning iterations. We provide more details about the hyperparameters in the appendix. **Baselines.** We evaluate DLPA against 5 different standard PAMDP algorithms across all the 8 tasks. HyAR (Li et al., 2022) and P-DQN (Xiong et al., 2018; Bester et al., 2019) are state-of-the-art RL algorithms for PAMDPs. HyAR learns an embedding of the parameterized action space which has been shown to be especially helpful when the dimensionality of the parameterized action space is large. P-DQN learns an actor network for every discrete action type. PADDPG (Hausknecht & Stone, 2016b) and HPPO (Fan et al., 2019) share similar ideas that an actor is learned to output the concatenation of discrete action and continuous parameter together. Following Li et al. (2022), we replace DDPG with TD3 (Fujimoto et al., 2018) in PADDPG and PDQN to make the comparison fair and rename PADDPG as PATD3.

6.1. Results

We show the evaluation results in Figure 4 and Table 1 (for asymptotic performance). A full timescale version of this plot can be found in Appendix E. We find that DLPA achieves significantly higher sample efficiency with better or comparable asymptotic performance across all the 8 different tasks. Among all the model-free RL algorithms, HyAR achieves the overall best performance among all the other

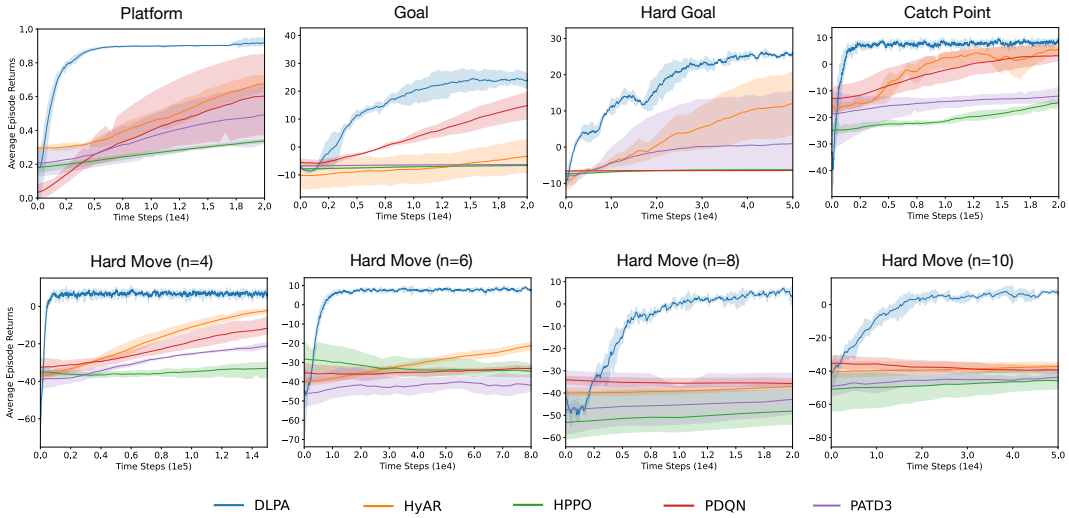

Figure 4. Comparison of different algorithms across the 8 PAMDP benchmarks. Our algorithm DLPA significantly outperforms state-of-the-art PAMDP algorithms in terms of sample efficiency. Note that HyAR has an additional 20000 environment steps pretraining for the action encoder which we do not include in the plot.

baselines, which is consistent with the results shown in their original paper. DLPA on average achieves $30\times$ **higher sample efficiency** compared to the best model-free RL method in each scenario. In all the 8 scenarios except Platform and Goal, DLPA reaches a better asymptotic performance, while in those two domains, the final performance is still close to HyAR. In Hard Move ($n \geq 6$), it has been shown in HyAR's original paper that, no regular PAMDP algorithms can learn a meaningful policy without learning a latent action embedding space. This happens because the action space is too large (i.e., 2^n). **However, we find that our algorithm DLPA—without learning such embedding space—can achieve even better performance just by sampling from the original large action space.** The table shows that, as the action space becomes larger (from 4 to 10), the gap between DLPA and HyAR also increases. HyAR's learned action embeddings are indeed useful in these cases, but it also sacrifices computational efficiency by making the algorithm much more sophisticated.

6.2. Ablation Study

In this section, we investigate the importance of some major components in DLPA: the planning algorithm, three PAMDP-specific inference models, H-step prediction loss, separate reward predictors and PAMDP-specifc MPPI. We show the experimental results on *Platform* and *Goal*.

We first investigate how the category of the planning algorithm will affect the performance of Model-based RL in PAMDP. We compare DLPA with a Dyna-like non-MPC model-based RL baseline, where we explicitly train a policy using the data generated from the learned model. As shown in Figure 5 first column, this non-MPC baseline generally performs better than the model-free baselines but is not as good as DLPA.

As we mentioned in Section 4, an important difference between our method and many prior model-based RL algorithms is the **H-step prediction loss**, that is, when updating the dynamics model, we only give the model the start state and the action sequences sampled from the replay buffer as input and let it predict the whole trajectory. We show an empirical performance comparison for these two ways of updating the dynamics models in Figure 5 second column. DLPA achieves significantly higher sample efficiency by predicting into several steps into the future with a length of horizon H. Presumably this is because during planning we will plan into the future with the exact same length H thus the proposed updating process will help the agent to focus on predicting the parts of state more accurately which will affect the future more and help the agent achieve better cumulative return.

Another major modification we make in the CEM planning process is the **PAMDP-specific MPPI**, where we keep a separate distribution over the continuous parameters for each discrete action and update them at each iteration instead of keeping just one independent distribution for the discrete actions and one independent distribution for the continuous parameters. We investigate the influence of this change by comparing to just a version of DLPA that just uses one independent distribution for all the continuous parameters. The results are shown in Figure 5 third column,without this technique it is quite hard for DLPA to do proper planning.

	DLPA	HyAR	HPPO	PDQN	PATD3
Platform	0.92 ± 0.05	$\mathbf{0.98 \pm 0.08}$	0.82 ± 0.02	0.91 ± 0.07	0.92 ± 0.09
Goal	28.75 ± 6.91	$\mathbf{34.23 \pm 3.71}$	-6.17 ± 0.06	33.13 ± 5.68	-2.25 ± 8.11
Hard Goal	$\mathbf{28.38 \pm 2.88}$	26.41 ± 3.59	-6.16 ± 0.06	1.04 ± 10.82	2.60 ± 11.12
Catch Point	$\mathbf{7.56 \pm 4.86}$	5.20 ± 4.18	4.44 ± 3.25	6.64 ± 2.52	0.56 ± 10.40
Hard Move (4)	$\mathbf{6.29 \pm 5.74}$	6.09 ± 1.67	-31.20 ± 5.58	4.29 ± 4.86	-10.67 ± 3.57
Hard Move (6)	$\mathbf{8.48 \pm 5.45}$	6.33 ± 2.12	-32.21 ± 6.54	-15.62 ± 8.65	-35.50 ± 25.43
Hard Move (8)	$\mathbf{7.80 \pm 6.27}$	-0.88 ± 3.83	-37.11 ± 10.10	-37.90 ± 4.07	-30.56 ± 12.21
Hard Move (10)	$\mathbf{6.35 \pm 9.97}$	-7.05 ± 3.74	-39.18 ± 8.76	-39.68 ± 5.93	-43.17 ± 15.98

Table 1. Comparison of different algorithms on all the eight benchmarks at the end of training (asymptotic performance). We report the mean and standard deviation of the last ten steps before the end of training. Value in bold indicates the best result for each task.

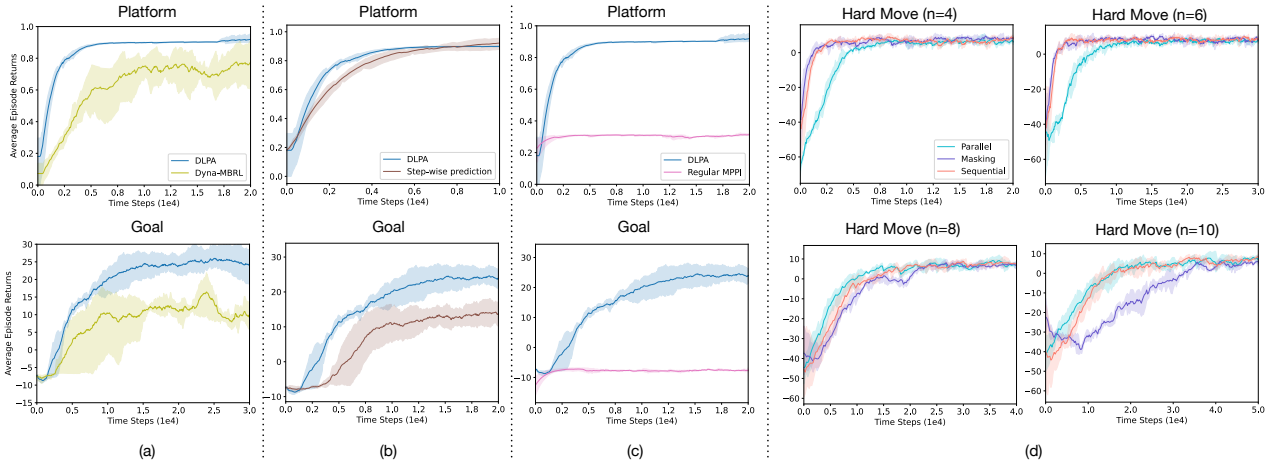

Figure 5. Ablation study on (a) the planning algorithm, (b) H-step prediction loss, (c) PAMDP-specific MPPI, (d) different inference model architectures.

	Two reward predictors	One reward predictor
Catch Point	$\mathbf{7.56 \pm 4.86}$	-17.65 ± 7.75
Hard Move (4)	$\mathbf{6.29 \pm 5.74}$	-37.38 ± 8.62
Hard Move (6)	$\mathbf{8.48 \pm 5.45}$	-39.66 ± 10.60
Hard Move (8)	$\mathbf{7.80 \pm 6.27}$	-40.17 ± 11.82
Hard Move (10)	$\mathbf{6.35 \pm 9.97}$	-35.94 ± 13.69

Table 2. Comparison between two separate reward predictors and one unified reward predictor on 5 domains after convergence.

Then we investigate the influence of **different inference model structures** we propose in Section 4. As we show in Figure 5, the **sequential** structure has the overall best performance compared to the others. This indicates that by decoupling the prediction process for the discrete and continuous component, the model is able to more precisely predict the long-term sequences specified by our H-step prediction loss.

Finally, we investigate how **separate reward predictors** influence the performance of DLPA. The comparison results are shown in Table 2. We compared to the default setting where only one reward predictor is trained regardless of the continuation signal. From the results, we find that in harder domains like catch point and hard move, without the separate reward predictors the method can hardly learn achieve any success.

6.3. Visualization of Planning Iterations

As we mentioned before, for each planning step we run our prediction and sampling algorithm for 6 iterations and then pick the parameterized action. We show in Figure 6 the visualization of the imagined trajectories with top-30 returns for each iteration at a random time step when we evaluate DLPA in the Catch Point environment. Recall that we first sample the action sequences given a set of distribution parameters and then generate the trajectories and compute cumulative returns using the learned dynamics models. We can see that at the first iteration, the generated actions are quite random and cover a large part of the space. Then as we keep updating the distribution parameters with the cumulative returns inferred by the learned models, the imagined trajectories become more and more concentrated and finally narrow down to a relatively small-entropy distribution centering at the optimal actions. This indicates that the proposed planning method (described in Section 4.2) is able to help the agent find the greedy action to execute given the learned dynamics model while also keep a mind for exploration.

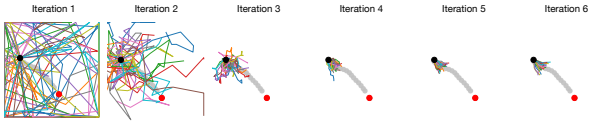

Figure 6. Visualization of DLPA's Planning iterations on the Catch Point tasks. Different color represents different imagined trajectories. The black point represents the agent's current position and the red point represents the target point. The grey trajectory is the actual trajectory taken by the agent.

7. Conclusion

We have introduced DLPA, the first model-based Reinforcement Learning algorithm for parameterized action spaces (also known as discrete-continuous hybrid action spaces). DLPA first learns a dynamics model that is conditioned on the parameterized actions with a weighted trajectory-level prediction loss. Then we propose a novel planning method for parameterized actions by keep updating and sampling from the distribution over the discrete actions and continuous parameters. DLPA outperforms the state-of-the-art PAMDP algorithms on 8 standard PAMDP benckmarks. We further empirically demonstrate the effectiveness of the different components of the proposed algorithm.

Acknowledgement

This work was conducted using computational resources and services at the Center for Computation and Visualization, Brown University. The authors would like to thank the anonymous reviewers for valuable feedbacks. This work was supported in part by NSF grant #1955361, and CAREER award #1844960 to Konidaris, and ONR grant #N00014-22-1-2592. Partial funding for this work provided by The Boston Dynamics AI Institute ("The AI Institute").

Impact Statement

We do not foresee significant societal impact resulting from our proposed method.

References

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., and Riedmiller, M. A. Maximum a posteriori policy optimisation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Asadi, K., Misra, D., and Littman, M. L. Lipschitz continuity in model-based reinforcement learning. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 264–273. PMLR, 2018.

Bester, C. J., James, S., and Konidaris, G. D. Multi-pass q-networks for deep reinforcement learning with parameterised action spaces. *ArXiv*, abs/1905.04388, 2019.

Bhardwaj, M., Handa, A., Fox, D., and Boots, B. Information theoretic model predictive q-learning. In *L4DC*, volume 120 of *Proceedings of Machine Learning Research*, pp. 840–850. PMLR, 2020.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 4759–4770, 2018.

Doshi-Velez, F. and Konidaris, G. D. Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. In Kambhampati, S. (ed.), *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pp. 1432–1440. IJCAI/AAAI Press, 2016.

Ebert, F., Finn, C., Dasari, S., Xie, A., Lee, A. X., and Levine, S. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *CoRR*, abs/1812.00568, 2018.

Fan, Z., Su, R., Zhang, W., and Yu, Y. Hybrid actor-critic reinforcement learning in parameterized action space. In *IJCAI*, 2019.

Fu, H., Tang, H., Hao, J., Lei, Z., Chen, Y., and Fan, C. Deep multi-agent reinforcement learning with discrete-continuous hybrid action spaces. In *IJCAI*, 2019.

Fu, H., Yao, J., Gottesman, O., Doshi-Velez, F., and Konidaris, G. Performance bounds for model and policy transfer in hidden-parameter mdps. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023a.

Fu, H., Yu, S., Tiwari, S., Littman, M. L., and Konidaris, G. Meta-learning parameterized skills. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10461–10481. PMLR, 2023b.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1582–1591. PMLR, 2018.

Garcia, C. E., Prett, D. M., and Morari, M. Model predictive control: Theory and practice - a survey. *Autom.*, 25:335–348, 1989.

Gelada, C., Kumar, S., Buckman, J., Nachum, O., and Bellemare, M. G. Deepmdp: Learning continuous latent space models for representation learning. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2170–2179. PMLR, 2019.

Ha, D. and Schmidhuber, J. Recurrent world models facilitate policy evolution. In *NeurIPS*, pp. 2455–2467, 2018.

Hafner, D., Lillicrap, T. P., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 2019.

Hafner, D., Lillicrap, T. P., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. In *ICLR*. OpenReview.net, 2020.

Hafner, D., Lillicrap, T. P., Norouzi, M., and Ba, J. Mastering atari with discrete world models. In *ICLR*. OpenReview.net, 2021.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. P. Mastering diverse domains through world models. *CoRR*, abs/2301.04104, 2023.

Hansen, N., Su, H., and Wang, X. Temporal difference learning for model predictive control. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., and Sabato, S. (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8387–8406. PMLR, 2022.

Hausknecht, M. and Stone, P. Half field offense: An environment for multiagent learning and ad hoc teamwork. 2016a. URL https://api.semanticscholar.org/CorpusID:501883.

Hausknecht, M. J. and Stone, P. Deep reinforcement learning in parameterized action space. In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016b.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. In *NeurIPS*, pp. 12498–12509, 2019.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., Mohiuddin, A., Sepassi, R., Tucker, G., and Michalewski, H. Model based reinforcement learning for atari. In *ICLR*. OpenReview.net, 2020.

Killian, T. W., Daulton, S., Doshi-Velez, F., and Konidaris, G. D. Robust and efficient transfer learning with hidden parameter markov decision processes. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6250–6261, 2017.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In Bengio, Y. and LeCun, Y. (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Li, B., Tang, H., Zheng, Y., Hao, J., Li, P., Wang, Z., Meng, Z., and Wang, L. Hyar: Addressing discrete-continuous action reinforcement learning via hybrid action representation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Lowrey, K., Rajeswaran, A., Kakade, S. M., Todorov, E., and Mordatch, I. Plan online, learn offline: Efficient learning and exploration via model-based control. In *ICLR (Poster)*. OpenReview.net, 2019.

Masson, W., Ranchod, P., and Konidaris, G. Reinforcement learning with parameterized actions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M. A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik,

A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015.

Nagabandi, A., Kahn, G., Fearing, R. S., and Levine, S. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *ICRA*, pp. 7559–7566. IEEE, 2018.

Neunert, M., Abdolmaleki, A., Wulfmeier, M., Lampe, T., Springenberg, J. T., Hafner, R., Romano, F., Buchli, J., Heess, N. M. O., and Riedmiller, M. A. Continuous-discrete reinforcement learning for hybrid control in robotics. *ArXiv*, abs/2001.00449, 2019.

Nguyen, T. D., Shu, R., Pham, T., Bui, H., and Ermon, S. Temporal predictive coding for model-based planning in latent space. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8130–8139. PMLR, 2021.

Okada, M. and Taniguchi, T. Variational inference MPC for bayesian model-based reinforcement learning. In Kaelbling, L. P., Kragic, D., and Sugiura, K. (eds.), *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pp. 258–272. PMLR, 2019.

Pirotta, M., Restelli, M., and Bascetta, L. Policy gradient in lipschitz markov decision processes. *Mach. Learn.*, 100 (2-3):255–283, 2015.

Pong, V., Gu, S., Dalal, M., and Levine, S. Temporal difference models: Model-free deep RL for model-based control. In *ICLR (Poster)*. OpenReview.net, 2018.

Rachelson, E. and Lagoudakis, M. G. On the locality of action domination in sequential decision making. In *International Symposium on Artificial Intelligence and Mathematics, ISAIM 2010, 2010*, 2010.

Rubinstein, R. Y. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99:89–112, 1997.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T. P., and Silver, D. Mastering atari, go, chess and shogi by planning with a learned model. *Nat.*, 588(7839):604–609, 2020.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., and Moritz, P. Trust region policy optimization. In Bach, F. R. and Blei, D. M. (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1889–1897. JMLR.org, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

Sekar, R., Rybkin, O., Daniilidis, K., Abbeel, P., Hafner, D., and Pathak, D. Planning to explore via self-supervised world models. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8583–8592. PMLR, 2020.

Sutton, R. S. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In Porter, B. W. and Mooney, R. J. (eds.), *Machine Learning, Proceedings of the Seventh International Conference on Machine Learning, Austin, Texas, USA, June 21-23, 1990*, pp. 216–224. Morgan Kaufmann, 1990.

Williams, G., Aldrich, A., and Theodorou, E. A. Model predictive path integral control using covariance variable importance sampling. *CoRR*, abs/1509.01149, 2015.

Xiong, J., Wang, Q., Yang, Z., Sun, P., Han, L., Zheng, Y., Fu, H., Zhang, T., Liu, J., and Liu, H. Parametrized deep q-networks learning: Reinforcement learning with discrete-continuous hybrid action space. *ArXiv*, abs/1810.06394, 2018.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. MOPO: model-based offline policy optimization. In *NeurIPS*, 2020.

Zhang, M., Vikram, S., Smith, L. M., Abbeel, P., Johnson, M. J., and Levine, S. SOLAR: deep structured latent representations for model-based reinforcement learning. *CoRR*, abs/1808.09105, 2018.

A. Algorithm

Algorithm 1 DLPA

Require: Initialize Dynamics models $T_\phi(\hat{s}_{t+1}|s_t, k_t, z_{k_t}), R_\phi(\hat{r}_t|s_t, k_t, z_{k_t}), p_\phi(\hat{c}_t|s_{t+1})$, planning horizon H, a set of parameters $\mathcal{C}^0$
  for Time t = 0 to TaskHorizon **do**
    for Iteration j=1 to E **do**
      Sample N action sequences with horizon H from $\mathcal{C}^j$
      Forward the dynamics model $T_\phi(\hat{s}_{t+1}|s_t, k_t, z_{k_t})$ to time step $t+H$ with input s_t and the sampled action sequences and get N trajectories $\{\tau_i\}_{1:N}$
      Compute the cumulative return for each trajectory $\mathcal{J}_\tau$ with $R_\phi(\hat{r}_t|s_t, k_t, z_{k_t}), p_\phi(\hat{c}_t|s_{t+1})$
      Select the trajectories with top-n cumulative returns
      Update C^j with Equation 5, 6, 7
    end for
    Execute the first action, $\{\hat{k}_{t_0}, \hat{z}_{\hat{k}_{t_0}}\}$, in the sampled optimal trajectory
    Receive transitions from the environment and add to replay buffer $\mathcal{B}$
    Sample trajectories $\{s_t, k_t, z_{k_t}, r_t, s_{t+1}, c_t\}_{t_0:t_0+H}$ with from the replay buffer $\mathcal{B}$
    Initialize $\mathcal{L}_{joint}$=0
    for $t = t_0 : t_0 + H$ **do**
      $\hat{s}_{t+1} \sim T_\phi(\hat{s}_{t+1}|\hat{s}_t, k_t, z_{k_t})$
      $\hat{c}_t \sim p_\phi(\hat{c}_t|\hat{s}_{t+1})$
      $\hat{r}_t \sim R_\phi(\hat{r}_t|\hat{s}_t, k_t, z_{k_t})$
      $\mathcal{L}_{joint} \leftarrow \mathcal{L}_{joint} + \beta[\lambda_1\|T_\phi(\hat{s}_{t+1}|\hat{s}_t, k_t, z_{k_t}) - s_{t+1}\|_2^2 + \lambda_2\|R_\phi(\hat{r}_{t+1}|\hat{s}_t, k_t, z_{k_t}) - r_{t+1}\|_2^2 + \lambda_3\|p_\phi(\hat{c}_t|\hat{s}_{t+1}) - c_t\|_2^2]$
    end for
    $\phi \leftarrow \phi - \frac{1}{H}\eta\nabla_\phi L_{joint}$
  end for

B. More theoretical results and proofs

Theorem B.1. *For a $(L_R^S, L_R^K, L_R^Z, L_T^S, L_T^K, L_T^Z)$-Lipschitz PAMDP and the learned DLPA ϵ_T-accurate transition model $\hat{T}$ and ϵ_R-accurate reward model $\hat{T}$, let $q(s)$ be the initial state distribution, $L_{\hat{T}}^S = \min\{L_T^S, L_{\hat{T}}^S\}$, $L_{\hat{T}}^K = \min\{L_T^K, L_{\hat{T}}^K\}$, $L_{\hat{T}}^Z = \min\{L_T^Z, L_{\hat{T}}^Z\}$ and similarly define $L_{\hat{R}}^S$, $L_{\hat{R}}^K$, $L_{\hat{R}}^Z$. Then $\forall n > 1$, starting from the same initial distribution $q(s)$, the n-step prediction error is bounded by:*

$$\begin{aligned}\Delta(n) :=& W\Big(T_n(q(s), k, \omega(\cdot|k)), \hat{T}_n(q(s), \hat{k}, \hat{\omega}(\cdot|\hat{k}))\Big) \\ \leq\ & \epsilon_T \sum_{i=0}^{n-1}(L_{\hat{T}}^S)^i + (L_{\hat{T}}^Z\Delta_{\omega,\hat{\omega}} + L_{\hat{T}}^Z\Delta_{k,\hat{k}})\sum_{i=0}^{n-1}(L_{\hat{T}}^S)^i + L_{\hat{T}}^K\sum_{i=0}^{n-1}(L_{\hat{T}}^S)^i\mathbb{1}(k_{n-i} \neq \hat{k_{n-i}})\end{aligned} \tag{10}$$

Proof. Firstly, for one-step prediction given the initial distribution $q(s)$ and the same k as well as ω:

$$\begin{aligned}\Delta(1) :=& W\Big(T(q(s), k, \omega(\cdot|k)), \hat{T}(q(s), k, \omega(\cdot|k))\Big) \\ =& \sup_f \int\int (\hat{T}(s'|s,k,\omega) - T(s'|s,k,\omega))f(s')q(s)dsds' \textbf{ Duality for Wasserstein Metric} \\ \leq& \int \sup_f \int (\hat{T}(s'|s,k,\omega) - T(s'|s,k,\omega))f(s')ds'q(s)ds \textbf{ Jensen's inequality} \\ =& \int W(\hat{T}(s'|s,k,\omega), T(s'|s,k,\omega))q(s)ds \leq \int \epsilon_T q(s)ds = \epsilon_T\end{aligned} \tag{11}$$

Then for the n-step prediction error with different discrete action and continuous parameter distributions:

$$\begin{aligned}\Delta(n) :=& W\Big(T_n(q(s),k,\omega(\cdot|k)),\hat{T}_n(q(s),\hat{k},\hat{\omega}(\cdot|\hat{k}))\Big)\\ \leq& W\Big(T_n(q(s),k,\omega(\cdot|k)),T_n(q(s),\hat{k},\hat{\omega}(\cdot|\hat{k}))\Big)+W\Big(T_n(q(s),\hat{k},\hat{\omega}(\cdot|\hat{k})),\hat{T}_n(q(s),\hat{k},\hat{\omega}(\cdot|\hat{k}))\Big)\ \textbf{Triangle inequality}\end{aligned} \tag{12}$$

For the second term in 12:

$$\begin{aligned}W\Big(T_n(q(s),\hat{k},\hat{\omega}(\cdot|\hat{k})),&\hat{T}_n(q(s),\hat{k},\hat{\omega}(\cdot|\hat{k}))\Big)=W\Big(T(T_{n-1}(q(s),\hat{k},\hat{\omega}(\cdot|\hat{k})),\hat{k}_n,\hat{\omega}_n),\hat{T}(\hat{T}_{n-1}(q(s),\hat{k},\hat{\omega}(\cdot|\hat{k})),\hat{k}_n,\hat{\omega}_n)\Big)\\ &\leq \epsilon_T+L_T^S W\Big(T_{n-1}(q(s),\hat{k},\hat{\omega}(\cdot|\hat{k})),\hat{T}_{n-1}(q(s),\hat{k},\hat{\omega}(\cdot|\hat{k}))\Big)\ \textbf{Composition Lemma (Asadi et al., 2018)}\\ &\cdots\\ &\leq \epsilon_T\sum_{i=0}^{n-1}(L_T^S)^i\end{aligned} \tag{13}$$

For the first term in 12

$$\begin{aligned}&W\Big(T_n(q(s),k,\omega(\cdot|k)),T_n(q(s),\hat{k},\hat{\omega}(\cdot|\hat{k}))\Big)\leq\\ &W\Big(T_n(q(s),k,\omega(\cdot|k)),T_n(q(s),k,\hat{\omega}(\cdot|\hat{k}))\Big)+W\Big(T_n(q(s),k,\hat{\omega}(\cdot|\hat{k})),T_n(q(s),\hat{k},\hat{\omega}(\cdot|\hat{k}))\Big)\end{aligned} \tag{14}$$

For the second term in 14:

$$\begin{aligned}&W\Big(T_n(q(s),k,\hat{\omega}(\cdot|\hat{k})),T_n(q(s),\hat{k},\hat{\omega}(\cdot|\hat{k}))\Big)\\ &=W\Big(T(T_{n-1}(q(s),k,\hat{\omega}(\cdot|\hat{k})),k_n,\hat{\omega_n}),T(T_{n-1}(q(s),\hat{k},\hat{\omega}(\cdot|\hat{k})),\hat{k_n},\hat{\omega_n})\Big)\\ &\leq L_T^K d(k_n,\hat{k_n})+L_T^S W\Big(T_{n-1}(q(s),k,\hat{\omega}(\cdot|\hat{k})),T_{n-1}(q(s),\hat{k},\hat{\omega}(\cdot|\hat{k}))\Big)\\ &=L_T^K\mathbb{1}(k_n\neq\hat{k_n})+L_T^S W\Big(T_{n-1}(q(s),k,\hat{\omega}(\cdot|\hat{k})),T_{n-1}(q(s),\hat{k},\hat{\omega}(\cdot|\hat{k}))\Big)\\ &\leq L_T^K\mathbb{1}(k_n\neq\hat{k_n})+L_T^S L_T^K\mathbb{1}(k_{n-1}\neq\hat{k_{n-1}})+(L_T^S)^2 W\Big(T_{n-2}(q(s),k,\hat{\omega}(\cdot|\hat{k})),T_{n-2}(q(s),\hat{k},\hat{\omega}(\cdot|\hat{k}))\Big)\\ &\cdots\\ &\leq L_T^K\sum_{i=0}^{n-1}(L_T^S)^i\mathbb{1}(k_{n-i}\neq\hat{k_{n-i}})\end{aligned} \tag{15}$$

For the first term in 14:

$$\begin{aligned}&W\Big(T_n(q(s),k,\omega(\cdot|k)),T_n(q(s),k,\hat{\omega}(\cdot|\hat{k}))\Big)\leq\\ &W\Big(T_n(q(s),k,\omega(\cdot|k)),T_n(q(s),k,\hat{\omega}(\cdot|k))\Big)+W\Big(T_n(q(s),k,\hat{\omega}(\cdot|k)),T_n(q(s),k,\hat{\omega}(\cdot|\hat{k}))\Big)\end{aligned} \tag{16}$$

For the first term in 16:

$$\begin{aligned}&W\Big(T_n(q(s),k,\omega(\cdot|k)),T_n(q(s),k,\hat{\omega}(\cdot|k))\Big)\\ &=W\Big(T(T_{n-1}(q(s),k,\omega(\cdot|k)),k_n,\omega_n),T(T_{n-1}(q(s),k,\hat{\omega}(\cdot|k)),k_n,\hat{\omega_n})\Big)\\ &\leq L_T^Z\Delta_{\omega,\hat{\omega}}+L_T^S W\Big(T_{n-1}(q(s),k,\omega(\cdot|k)),T_{n-1}(q(s),k,\hat{\omega}(\cdot|k))\Big)\\ &\leq L_T^Z\Delta_{\omega,\hat{\omega}}+L_T^S L_T^Z\Delta_{\omega,\hat{\omega}}+(L_T^S)^2 W\Big(T_{n-2}(q(s),k,\omega(\cdot|k)),T_{n-2}(q(s),k,\hat{\omega}(\cdot|k))\Big)\\ &\cdots\\ &\leq L_T^Z\Delta_{\omega,\hat{\omega}}\sum_{i=0}^{n-1}(L_T^S)^i\end{aligned} \tag{17}$$

Similarly, for the second term in 16:

$$\begin{aligned} &W\Big(T_n(q(s),k,\hat{\omega}(\cdot|k)),T_n(q(s),k,\hat{\omega}(\cdot|\hat{k}))\Big) \\ &\leq L_T^Z\Delta_{k,\hat{k}}\sum_{i=0}^{n-1}(L_T^S)^i\mathbb{1}(k_{n-i}\neq \hat{k_{n-i}}) \end{aligned} \tag{18}$$

Combining all the results above and continue 12, we have:

$$\begin{aligned} \Delta(n) \leq &\epsilon_T\sum_{i=0}^{n-1}(L_T^S)^i + L_T^K\sum_{i=0}^{n-1}(L_T^S)^i + L_T^Z\Delta_{\omega,\hat{\omega}}\sum_{i=0}^{n-1}(L_T^S)^i + L_T^Z\Delta_{k,\hat{k}}\sum_{i=0}^{n-1}(L_T^S)^i \\ = &\epsilon_T\sum_{i=0}^{n-1}(L_T^S)^i + \sum_{i=0}^{n-1}(L_T^S)^i(L_T^Z\Delta_{\omega,\hat{\omega}} + L_T^Z\Delta_{k,\hat{k}}\mathbb{1}(k_{n-i}\neq \hat{k_{n-i}})) + L_T^K\sum_{i=0}^{n-1}(L_T^S)^i\mathbb{1}(k_{n-i}\neq \hat{k_{n-i}}) \end{aligned} \tag{19}$$

Now if replace $T_n(q(s),\hat{k},\hat{\omega}(\cdot|\hat{k}))$ with $\hat{T}_n(q(s),k,\omega(\cdot|k))$ in the triangle inequality 12 and do all the derivation again, we have:

$$\Delta(n) \leq \epsilon_T\sum_{i=0}^{n-1}(L_T^S)^i + \sum_{i=0}^{n-1}(L_T^S)^i(L_{\hat{T}}^Z\Delta_{\omega,\hat{\omega}} + (L_{\hat{T}}^Z\Delta_{k,\hat{k}} + L_{\hat{T}}^K)\mathbb{1}(k_{n-i}\neq \hat{k_{n-i}})) \tag{20}$$

Combining 19 and 20 concludes the proof. □

Theorem 5.2. For a $(L_R^S, L_R^K, L_R^Z, L_T^S, L_T^K, L_T^Z)$-Lipschitz PAMDP and the learned DLPA ϵ_T-accurate transition model $\hat{T}$ and ϵ_R-accurate reward model $\hat{T}$, let $L_{\bar{T}}^S = \min\{L_T^S, L_{\hat{T}}^S\}$, $L_{\bar{T}}^K = \min\{L_T^K, L_{\hat{T}}^K\}$, $L_{\bar{T}}^Z = \min\{L_T^Z, L_{\hat{T}}^Z\}$ and similarly define $L_{\bar{R}}^S$, $L_{\bar{R}}^K$, $L_{\bar{R}}^Z$. If $L_{\bar{T}}^S < 1$, then the regret of the rollout trajectory $\hat{\tau} = \{\hat{s_1}, \hat{k_1}, \hat{\omega_1}(\cdot|\hat{k_1}), \hat{s_2}, \hat{k_2}, \hat{\omega_2}(\cdot|\hat{k_2}), \cdots\}$ from DLPA is bounded by:

$$\begin{aligned} |\mathcal{J}_{\tau^*} - \mathcal{J}_{\hat{\tau}}| :=& \sum_{t=1}^{H}\gamma^{t-1}|R(s_t,k_t,\omega_t(\cdot|k_t)) - \hat{R}(\hat{s_t},\hat{k_t},\hat{\omega_t}(\cdot|\hat{k_t}))| \\ &\leq \mathcal{O}\Big((L_{\bar{R}}^K + L_{\bar{R}}^S L_{\bar{T}}^K)m + H(\epsilon_R + L_{\bar{R}}^S\epsilon_T + (L_{\bar{R}}^Z + L_{\bar{R}}^S L_{\bar{T}}^Z)(\frac{m}{H}\Delta_{k,\hat{k}} + \Delta_{\omega,\hat{\omega}}))\Big), \end{aligned} \tag{21}$$

where $m = \sum_{t=1}^{H}\mathbb{1}(k_t \neq \hat{k_t})$, $\Delta_{\omega,\hat{\omega}} = W(\omega(\cdot|k),\hat{\omega}(\cdot|k))$, $\Delta_{k,\hat{k}} = W(\omega(\cdot|k),\omega(\cdot|\hat{k}))$, N denotes the number of samples and H is the planning horizon.

Proof. At timestep t:

$$\begin{aligned} &|R(s_t,k_t,\omega_t(\cdot|k_t)) - \hat{R}(\hat{s_t},\hat{k_t},\hat{\omega_t}(\cdot|\hat{k_t}))| \leq \\ &|R(s_t,k_t,\omega_t(\cdot|k_t)) - \hat{R}(\hat{s_t},k_t,\hat{\omega_t}(\cdot|\hat{k_t}))| + |\hat{R}(\hat{s_t},k_t,\hat{\omega_t}(\cdot|\hat{k_t})) - \hat{R}(\hat{s_t},\hat{k_t},\hat{\omega_t}(\cdot|\hat{k_t}))| \end{aligned} \tag{22}$$

For the second term in 22:

$$|\hat{R}(\hat{s_t},k_t,\hat{\omega_t}(\cdot|\hat{k_t})) - \hat{R}(\hat{s_t},\hat{k_t},\hat{\omega_t}(\cdot|\hat{k_t}))| \leq L_{\hat{R}}^K d(k,\hat{k}) = L_{\hat{R}}^K\mathbb{1}(k_t\neq\hat{k_t}) \tag{23}$$

For the first term in 22:

$$
\begin{aligned}
&|R(s_t,k_t,\omega_t(\cdot|k_t))-\hat{R}(\hat{s}_t,k_t,\hat{\omega}_t(\cdot|\hat{k}_t))|\leq\\
&|R(s_t,k_t,\omega_t(\cdot|k_t))-\hat{R}(\hat{s}_t,k_t,\omega_t(\cdot|k_t))|+|\hat{R}(\hat{s}_t,k_t,\omega_t(\cdot|k_t))-\hat{R}(\hat{s}_t,k_t,\hat{\omega}_t(\cdot|\hat{k}_t))|\\
&\leq\epsilon_R+L^S_{\hat{R}}\Delta(t-1)+|\hat{R}(\hat{s}_t,k_t,\omega_t(\cdot|k_t))-\hat{R}(\hat{s}_t,k_t,\hat{\omega}_t(\cdot|\hat{k}_t))|\ \textbf{By the definition of}\ \Delta(n)\ \textbf{and Composition Lemma}\\
&\leq\epsilon_R+L^S_{\hat{R}}\Delta(t-1)+|\hat{R}(\hat{s}_t,k_t,\omega_t(\cdot|k_t))-\hat{R}(\hat{s}_t,k_t,\omega_t(\cdot|\hat{k}_t))|+|\hat{R}(\hat{s}_t,k_t,\omega_t(\cdot|\hat{k}_t))-\hat{R}(\hat{s}_t,k_t,\hat{\omega}_t(\cdot|\hat{k}_t))|\\
&\leq\epsilon_R+L^S_{\hat{R}}\Delta(t-1)+L^Z_{\hat{R}}\Delta_{k,k'}+|\hat{R}(\hat{s}_t,k_t,\omega_t(\cdot|\hat{k}_t))-\hat{R}(\hat{s}_t,k_t,\hat{\omega}_t(\cdot|\hat{k}_t))|\\
&\leq\epsilon_R+L^S_{\hat{R}}\Delta(t-1)+L^Z_{\hat{R}}(\Delta_{k,\hat{k}}+\Delta_{\omega,\hat{\omega}})\\
&\leq\epsilon_R+L^S_{\hat{R}}\epsilon_T\sum_{i=0}^{t-2}(L^S_{\hat{T}})^i+L^S_{\hat{R}}\sum_{i=0}^{t-2}(L^S_{\hat{T}})^i(L^Z_{\hat{T}}\Delta_{\omega,\hat{\omega}}+(L^Z_{\hat{T}}\Delta_{k,\hat{k}}+L^K_{\hat{T}})\mathbb{1}(k_{t-1-i}\neq\hat{k_{t-1-i}}))\\
&+L^Z_{\hat{R}}(\Delta_{k,\hat{k}}+\Delta_{\omega,\hat{\omega}})\ \textbf{According to Theorem B.1}
\end{aligned}
\tag{24}
$$

Now we go back to 22:

$$
\begin{aligned}
&|R(s_t,k_t,\omega_t(\cdot|k_t))-\hat{R}(\hat{s}_t,\hat{k}_t,\hat{\omega}_t(\cdot|\hat{k}_t))|\leq\\
&|R(s_t,k_t,\omega_t(\cdot|k_t))-\hat{R}(\hat{s}_t,k_t,\hat{\omega}_t(\cdot|\hat{k}_t))|+|\hat{R}(\hat{s}_t,k_t,\hat{\omega}_t(\cdot|\hat{k}_t))-\hat{R}(\hat{s}_t,\hat{k}_t,\hat{\omega}_t(\cdot|\hat{k}_t))|\\
&\leq\epsilon_R+L^K_{\hat{R}}\mathbb{1}(k_t\neq\hat{k}_t)+L^S_{\hat{R}}\sum_{i=0}^{t-2}(L^S_{\hat{T}})^i(\epsilon_T+L^Z_{\hat{T}}\Delta_{\omega,\hat{\omega}}+(L^Z_{\hat{T}}\Delta_{k,\hat{k}}+L^K_{\hat{T}})\mathbb{1}(k_{t-1-i}\neq\hat{k_{t-1-i}}))\\
&+L^Z_{\hat{R}}(\Delta_{k,\hat{k}}+\Delta_{\omega,\hat{\omega}})
\end{aligned}
\tag{25}
$$

Then we can compute the regret:

$$
\begin{aligned}
|\mathcal{J}_{\tau^*}-\mathcal{J}_{\hat{\tau}}|&:=\sum_{t=1}^{H}\gamma^{t-1}|R(s_t,k_t,\omega_t(\cdot|k_t))-\hat{R}(\hat{s}_t,\hat{k}_t,\hat{\omega}_t(\cdot|\hat{k}_t))|\\
&\leq\sum_{t=1}^{H}\gamma^{t-1}[\epsilon_R+L^K_{\hat{R}}\mathbb{1}(k_t\neq\hat{k}_t)+\\
&L^S_{\hat{R}}\sum_{i=0}^{t-2}(L^S_{\hat{T}})^i(\epsilon_T+L^Z_{\hat{T}}\Delta_{\omega,\hat{\omega}}+(L^Z_{\hat{T}}\Delta_{k,\hat{k}}+L^K_{\hat{T}})\mathbb{1}(k_{t-1-i}\neq\hat{k_{t-1-i}}))+L^Z_{\hat{R}}(\Delta_{k,\hat{k}}+\Delta_{\omega,\hat{\omega}})]\\
&\leq\ \mathcal{O}\Big((L^K_{\hat{R}}+L^S_{\hat{R}}L^K_{\hat{T}})m+H(\epsilon_R+L^S_{\hat{R}}\epsilon_T+(L^Z_{\hat{R}}+L^S_{\hat{R}}L^Z_{\hat{T}})(\frac{m}{H}\Delta_{k,\hat{k}}+\Delta_{\omega,\hat{\omega}}))\Big)\ \textbf{Assuming}\ \gamma=1
\end{aligned}
\tag{26}
$$

Similar to the proof of theorem B.1, if we change the middle term in the triangle inequalities we will have:

$$
|\mathcal{J}_{\tau^*}-\mathcal{J}_{\hat{\tau}}|\leq\ \mathcal{O}\Big((L^K_{\hat{R}}+L^S_{\hat{R}}L^K_{\hat{T}})m+H(\epsilon_R+L^S_{\hat{R}}\epsilon_T+(L^Z_{\hat{R}}+L^S_{\hat{R}}L^Z_{\hat{T}})(\frac{m}{H}\Delta_{k,\hat{k}}+\Delta_{\omega,\hat{\omega}}))\Big)
\tag{27}
$$

Combine 26 and 27 concludes the proof. □

Lemma B.2. *Let $|Z|$ denote the cardinality of the continuous parameters space and N be the number of samples. Then, with probability at least $1-\delta$:*

$$
\frac{m}{H}\Delta_{k,\hat{k}}+\Delta_{\omega,\hat{\omega}}\leq\frac{2m}{H}\sqrt{|Z|}+\frac{2}{N}\ln\frac{2|Z|}{\delta}
\tag{28}
$$

Proof. By definition:

$$
\Delta_{k,\hat{k}}:=W(\omega(\cdot|k),\omega(\cdot|\hat{k}))
\tag{29}
$$

Recall that in our algorithm, we assume $\omega(\cdot|k)$ is a Gaussian distribution $\mathcal{N}(\mu_k,\Sigma_k)$ and $z\sim\omega(\cdot)$ takes value in the range $[-1,1]$.

Now we use the definition of **2nd Wasserstein distance** W_2 between multivariate Gaussian distributions:

$$W_2(\omega(\cdot|k),\omega(\cdot|\hat{k})) = \|\mu_k - \mu_{\hat{k}}\|_2^2 + Tr(\Sigma + \hat{\Sigma} - 2(\Sigma^{1/2}\hat{\Sigma}\Sigma^{1/2})^{1/2}) \tag{30}$$

Ignore the covariance term, we have:

$$W(\omega(\cdot|k),\omega(\cdot|\hat{k})) \leq 2\sqrt{|Z|} \tag{31}$$

By definition:

$$\Delta_{\omega,\hat{\omega}} := W(\omega(\cdot|k),\hat{\omega}(\cdot|k)) \tag{32}$$

Similarly, we have:

$$W_2(\omega(\cdot|k),\hat{\omega}(\cdot|k)) = \|\mu_k - \hat{\mu_k}\|_2^2 + Tr(\Sigma + \hat{\Sigma} - 2(\Sigma^{1/2}\hat{\Sigma}\Sigma^{1/2})^{1/2}) \tag{33}$$

Ignore the covariance term, by **Hoeffding's inequality**, with probability $1-\delta$ we have for each dimension i of Z:

$$\|\mu_{k,i} - \hat{\mu_{k,i}}\|_2^2 \leq \frac{(z_i^{max} - z_i^{min})^2}{2N} \ln \frac{2}{\delta} \tag{34}$$

By the union bound and the range of z:

$$\|\mu_k - \hat{\mu_k}\|_2^2 \leq \frac{2}{N} \ln \frac{2|Z|}{\delta} \tag{35}$$

Combining the results, we have:

$$\frac{m}{H}\Delta_{k,\hat{k}} + \Delta_{\omega,\hat{\omega}} = \frac{m}{H} W_2(\omega(\cdot|k),\omega(\cdot|\hat{k})) + W_2(\omega(\cdot|k),\hat{\omega}(\cdot|k)) \leq \frac{2m}{H}\sqrt{|Z|} + \frac{2}{N}\ln\frac{2|Z|}{\delta} \tag{36}$$

□

C. Environment description

- Platform: The agent is expected to reach the final goal while avoiding an enemy, or leaping over a gap. There are three parameterized actions (run, hop and leap) and each discrete action has one continuous parameter.
- Goal: The agent needs to find a way to avoid the goal keeper and shoot the ball into the gate. There are three parameterized actions: kick-to(x,y), shoot-goal-left(h), shoot-goal-right(h).
- Hard Goal: A more challenging version of the Goal environment where there are ten parameterized actions.
- Catch Point: The agent is expected to catch a goal point within limited trials. There are two parameterized actions: Move(d), catch(d).
- Hard Move (n= 4, 6, 8, 10): This is a set of environments where the agent needs to control n actuators to reach a goal point. The number of parameterized actions is 2^n.

D. Network Structure

We use the official code provided by HyAR[3] to implement all the baselines on the 8 benchmarks. The dynamics models in our proposed DLPA consists of three components: transition predictor T_ϕ, continue predictor p_ϕ and reward predictor R_ϕ, whose structures are shown in Table G and the hyperparameters are shown in Table 4.

It is worth noting that we train two reward predictors each time in Hard Move and Catch Point environments. Conditioned on whether the prediction for termination is True or False, we train one reward prediction network to only predict the reward when the trajectory has not terminated, and one network for the case when the prediction from the continue predictor is True.

E. Complete Learning Curves

We provide the full timescale plot of the training performance comparison on the 8 PAMDP benchmarks in Fig. 7. In general, the proposed method DLPA achieves significantly better sample efficiency and asymptotic performance than all the state-of-the-art PAMDP algorithms in most scenarios.

[3] https://github.com/TJU-DRL-LAB/self-supervised-rl/tree/main/RL_with_Action_Representation/HyAR

Layer	Transition Predictor	Continue Predictor	Reward Predictor
Fully Connected	$(inp\_dim, 64)$	$(inp\_dim, 64)$	$(inp\_dim, 64)$
Activation	ReLU	ReLU	ReLU
Fully Connected	$(64, 64)$	$(64, 64)$	$(64, 64)$
Activation	ReLU	ReLU	ReLU
Fully Connected	$(64, state\_dim)$	$(64, 2)$	$(64, 1)$
Fully Connected	$(64, state\_dim)$	$(64, 2)$	$(64, 1)$

Table 3. Network structures for all three predictors, inp_dim is the size of state space, discrete action space and continuous parameter space. Instead of outputting a deterministic value, our networks output parameters of a Gaussian distribution, which are mean and log standard deviation.

Hyperparameter	Value
Discount factor(γ)	0.99
Horizon	10, 8, 8, 5, 5, 5, 5, 5
Replay buffer size	10^6
Population size	1000
Elite size	400
CEM iteration	6
Temperature	0.5
Learning rate	3e-4
Transition loss coefficient	1
Reward loss coefficient	0.5
Termination loss coefficient	1
Batch size	128
Steps per update	1

Table 4. DLPA hyperparameters. We list the most important hyperparameters during both training and evaluating. If there's only one value in the list, it means all environments use the same value, otherwise, it's in the order of Platform, Goal, Hard Goal, Catch Point, Hard Move (4), Hard Move (6), Hard Move (8), and Hard Move (10).

F. Additional ablation study

Next we conduct an ablation study on the planning algorithm. In Section 4.2, we design a special sampling and updating algorithm for parameterized action spaces, here we compare it with a method that just randomly samples from a fixed distribution and picks the best action at every time step (also known as "random shooting"). The results are shown in Figure F. The proposed method DLPA significantly outperforms the version of the algorithm that uses random shooting to do sampling and get the optimal actions. Parameterized action space in general is a much larger sampling space, comparing to just discrete or continuous action space. This is because each time the agent need to first sample the discrete actions and each discrete action has a independent continuous parameter space. The problem becomes more severe when the number of discrete actions is extremely large. Thus it is hard for a random-shooting method to consistently find the optimal action distributions while also augment exploration.

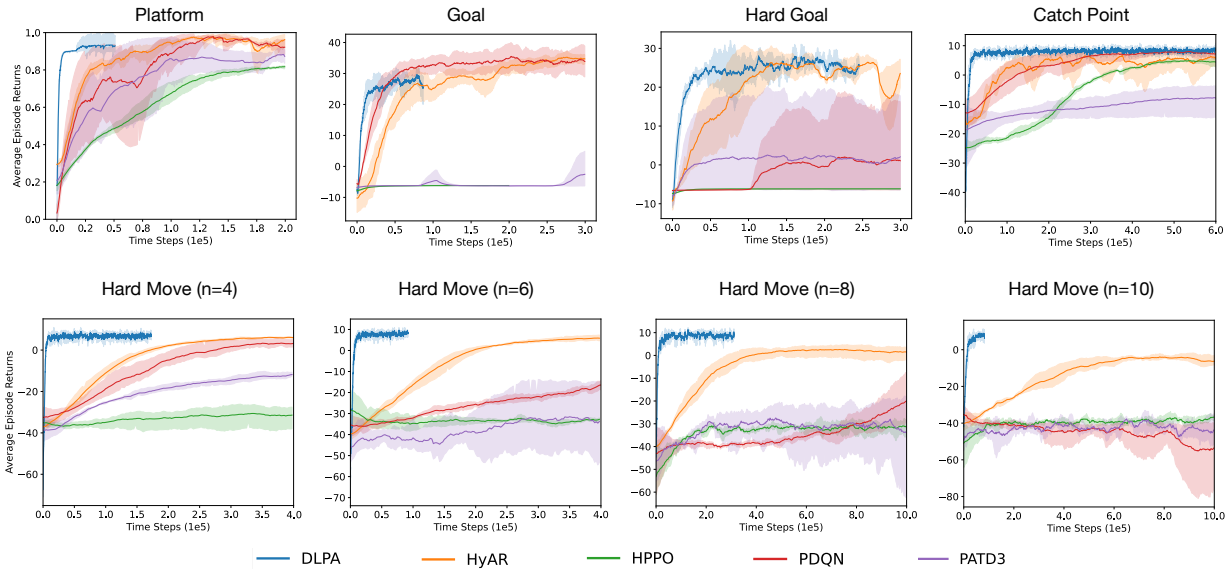

Figure 7. Comparison of different algorithms across the 8 PAMDP benchmarks, when the model-free methods converges. Our algorithm DLPA stops early in the experiments because it already converges.

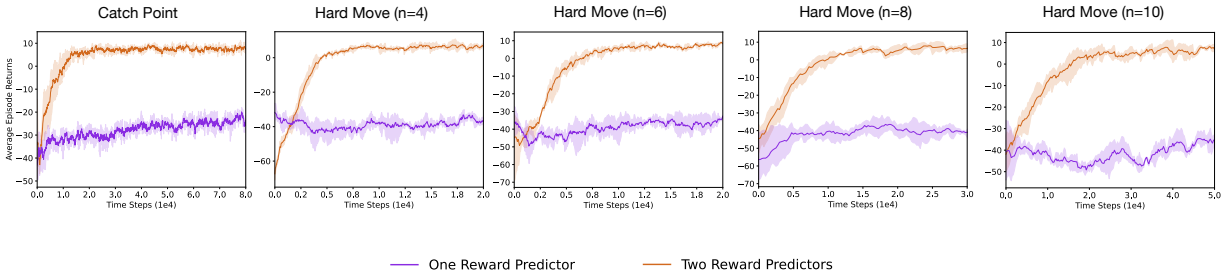

Figure 8. Comparison between using two separate reward predictors and one unified reward predictor across the 5 PAMDP domains.

G. Computational complexity

We also compare the clock time of planning and number of timesteps needed to converge as the action space expands. We tested this on the Hard Move domain, where the number of discrete actions changes from 2^4 to 2^{10} (one continuous parameter for each of them). As shown in table G, while the number of samples increases as the action space expands, it's still within an acceptable range even when it's extremely large. The results are also consistent with our theoretical analysis. Besides, table 6 and table 7 show the computational cost with respect to different inference structures as we increase the dimensionality of the parameterized action space.

# Discrete actions	Planning clock time /s	Training clock time /s	# Timesteps to converge
2^4	1.71e-1	6.99e-3	6,000
2^6	3.05e-1	7.18e-3	8,500
2^8	8.12e-1	7.53e-3	15,000
2^{10}	2.85	7.81e-3	23,000

Table 5. Computational complexity study. We evaluate the number of timesteps needed to converge as the action space expands on the Hard Move domain.

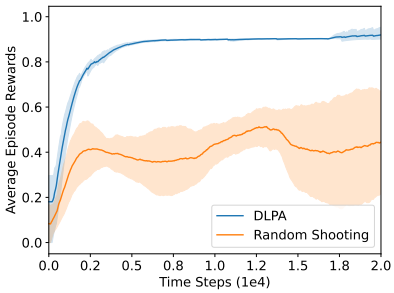

Figure 9. Ablation study on the planning algorithm (Platform).

	$K = 2^4$	$K = 2^6$	$K = 2^8$	$K = 2^{10}$
PADDPG	$2.54e-2$	$2.62e-2$	$2.71e-2$	$3.40e-2$
DLPA - Sequential	$2.18e-1$	$4.47e-1$	1.01	2.18
DLPA - Masking	$5.37e-1$	$6.04e-1$	1.03	2.95
DLPA - Parallel	$1.71e-1$	$3.05e-1$	$8.12e-1$	2.85

Table 6. Planning clock time (seconds per step) of PADDPG and three different structure of DLPA methods in Hard Move domain, where the number of discrete actions K changes from 2^4 to 2^{10} (one continuous parameter for each of them).

	$K = 2^4$	$K = 2^6$	$K = 2^8$	$K = 2^{10}$
PADDPG	$1.13e-3$	$1.28e-3$	$1.39e-3$	$1.57e-3$
DLPA - Sequential	$6.39e-3$	$6.32e-3$	$7.18e-3$	$7.30e-3$
DLPA - Masking	$7.30e-3$	$7.92e-3$	$8.12e-3$	$8.53e-3$
DLPA - Parallel	$6.99e-3$	$7.18e-3$	$7.53e-3$	$7.81e-3$

Table 7. Training clock time (seconds per step) of PADDPG and three different structure of DLPA methods in Hard Move domain, where the number of discrete actions K changes from 2^4 to 2^{10} (one continuous parameter for each of them).

H. Sensitivity of Planning Horizon and Weights in H-step Prediction Loss

Figure 10 explores how the hyperparameters (i.e. Horizon and the weights of the H-step prediction loss) affect DLPA's performance on "Platform" and "Goal" tasks.

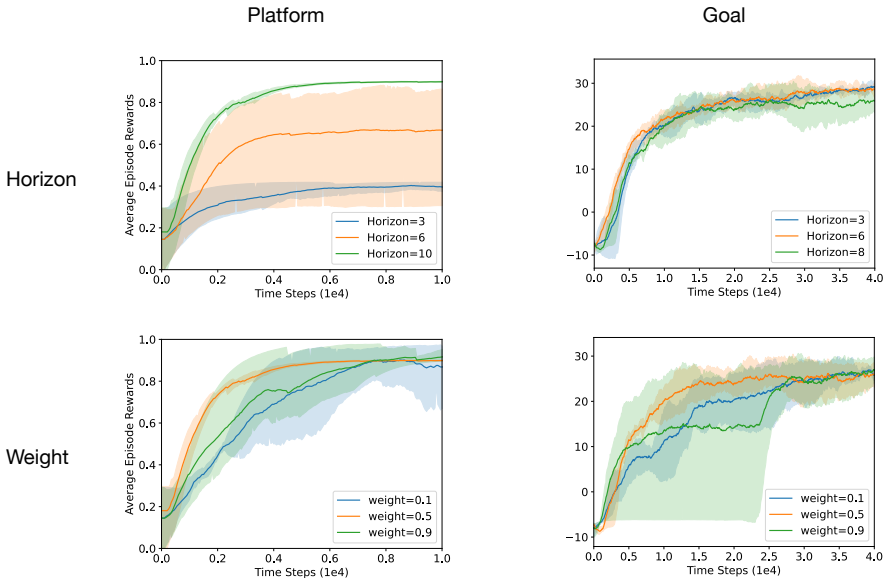

Figure 10. Sensitivity of Planning Horizon and Weight of H-step Prediction Loss

I. Additional Experiments on HFO

We further test our method on a much more complex domain—Half-Field-Offense (HFO) (Hausknecht & Stone, 2016a), where both the state space and action space are much larger than the 8 benchmarks. Besides, the task horizon is 10× longer and there is more randomness existing in the environment. HFO is originally a subtask in RoboCup simulated soccer[4]. As shown in Figure 11, DLPA is still able to reach a better performance than all the model-free PAMDP RL baselines in this higher dimensional domain.

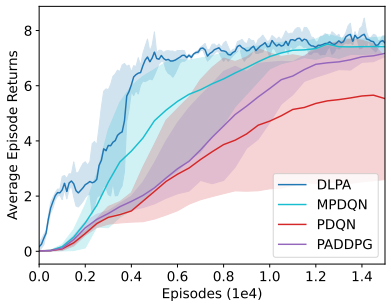

Figure 11. Additional experimental results on HFO

[4] https://www.robocup.org/leagues/24