Information Sciences 686 (2025) 121361

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Understanding world models through multi-step pruning policy via reinforcement learning

Zhiqiang He [a], Wen Qiu [b,*], Wei Zhao [c], Xun Shao [d], Zhi Liu [e]

[a] College of Information Science and Engineering, Northeastern University, China
[b] Doctoral Course in Manufacturing Engineering, Kitami Institute of Technology, Japan
[c] School of Computer Science and Technology, Anhui University of Technology, China
[d] Department of Electrical and Electronic Information Engineering, Toyohashi University of Technology, Japan
[e] The University of Electro-Communications, Tokyo, Japan

ARTICLE INFO

Keywords:
Model based reinforcement learning
Pruning policy
Multi-step
World model
Planning
Trajectory imagination

ABSTRACT

In model-based reinforcement learning, the conventional approach to addressing world model bias is to use gradient optimization methods. However, using a singular policy from gradient optimization methods in response to a world model bias inevitably results in an inherently biased policy. This is because of constraints on the imperfect and dynamic data of state-action pairs. The gap between the world model and the real environment can never be completely eliminated. This article introduces a novel approach that explores a variety of policies instead of focusing on either world model bias or singular policy bias. Specifically, we introduce the Multi-Step Pruning Policy (MSPP), which aims to reduce redundant actions and compress the action and state spaces. This approach encourages a different perspective within the same world model. To achieve this, we use multiple pruning policies in parallel and integrate their outputs using the cross-entropy method. Additionally, we provide a convergence analysis of the pruning policy theory in tabular form and an updated parameter theoretical framework. In the experimental section, the newly proposed MSPP method demonstrates a comprehensive understanding of the world model and outperforms existing state-of-the-art model-based reinforcement learning baseline techniques.

1. Introduction

Sequence decision-making problems are highly challenging tasks in artificial intelligence and are often modeled as Markov Decision Processes (MDPs) [1]. There are two main methods commonly used to address these problems: Planning and Reinforcement Learning [2]. Reinforcement Learning involves gaining an understanding of the control object through interaction data and then using that information to output control policies. However, when the sampling policy of the data is updated, the underlying data distribution changes, resulting in a high sample complexity. Model-based Reinforcement Learning (MBRL) explicitly learns a world model and then enhances the policy through interaction with this model, reducing the need for interaction with the actual environment. This approach reduces the complexity of the sample to a certain extent but still results in an effective strategy [3]. Therefore, MBRL is a promising research avenue for addressing practical challenges in the domains of control and robotics [4–8].

* Corresponding author.
E-mail addresses: tinyzqh@gmail.com (Z. He), clorisqiu1@gmail.com (W. Qiu), zhaowei@ahut.edu.cn (W. Zhao), x-shao@ieee.org (X. Shao), liuzhi@uec.ac.jp (Z. Liu).

https://doi.org/10.1016/j.ins.2024.121361
Received 5 April 2024; Received in revised form 17 July 2024; Accepted 15 August 2024

Available online 22 August 2024

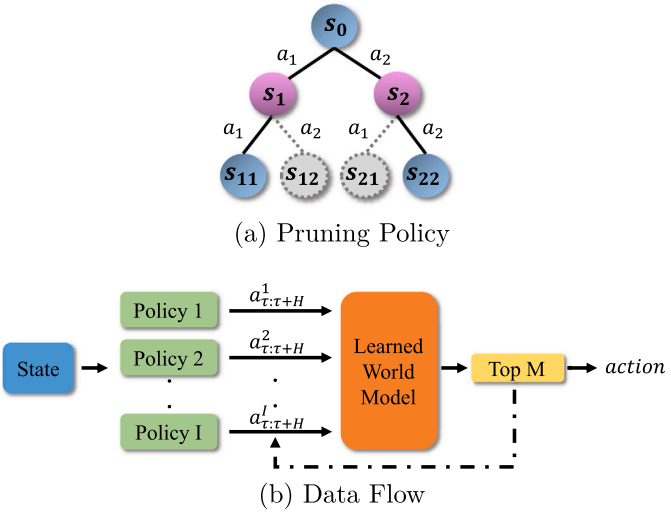

Fig. 1. Parallel Multi-Pruning Strategy. (a) Illustration of policy pruning through repeating selection process. (b) The same environment model learned from experience is used for knowledge extraction in multiple policies.

In a real-world environment, there are two important functions that play a pivotal role: the state transition dynamics function and the reward function [2]. These functions allow agents to explore without incurring any costs. By collecting data from interactions between the agent and the environment, we can train a world model using supervised learning techniques that can approximate these functions. Ideally, if the model is highly accurate, it can be used to derive an effective policy. As a result, current research places significant emphasis on the model learning process in order to minimize model inaccuracies and ensure better alignment with policy. However, it is important to recognize that any learned model will inherently have a degree of imprecision, which can lead to compounded errors during simulation. In the two-stage learning problem, where the first step is acquiring an environmental model and the second is refining policies based on this model, having a model with a smaller environment error does not necessarily result in a better policy [2]. Therefore, this paper focuses on the utilization of the model rather than its bias. Specifically, we explore multiple policies for learning the same biased black-box world model.

A biased world model is likely to propagate bias to the policy as well [9]. To mitigate this issue, we propose a pruning policy that skips fake state inputs and repeats the previous action. This approach is designed to improve the consistency of neural networks in deep model-based reinforcement learning, leading to more effective training in various scenarios. For instance, in autonomous driving, a vehicle must select a braking policy when a pedestrian is detected. This requires the policy to continuously apply thousands of steps within a millisecond-long decision cycle. Our pruning policy allows for multiple repetitions of an action at a single decision point, which can be challenging to achieve with variable inputs. By pruning non-optimal options, our approach promotes policy diversification. Specifically, if there are two action repetition cycles, as illustrated in Fig. 1 (a). Our approach prunes the policies entering s_{12} and s_{21}, and only retains the decisions to perform actions $\{a_1, a_1\}$ and $\{a_2, a_2\}$. This paper presents a diverse pruning policy method, Parallel Multi-Step Pruning Policy (MSPP), to address problems under the same environment model. This policy considers multiple perspectives and then integrates the results using the cross-entropy method to find the optimal solution. The data flow architecture is illustrated in Fig. 1 (b). More visual explanations of MSPP are provided in Appendix B and Appendix C.

To enhance the efficiency of our planning process and facilitate long-term strategic planning, we utilize the recurrent state-space model (RSSM), in a latent space that learns the environmental model [10,11]. This approach allows for the encoding of high-dimensional input observations into a low-dimensional latent state space for planning. The learned world model can then be effectively utilized by multiple pruned agents to optimize policies that aim to maximize cumulative rewards. Each parallel pruning policy is trained independently using the same latent world model. Once the training phase is completed, these policies provide their respective optimal policy distributions for the current state. Finally, the top-M actions of the pruning policies within this latent world model are selected to produce the final policy outputs, as illustrated in Fig. 2. In experiments conducted on the DeepMind Control Suite reinforcement learning benchmark platform, the results indicate that the MSPP framework enhances the final performance through diversifying strategy distribution, surpassing the state-of-the-art Dreamer algorithm.

The remainder of this paper is organized as follows. Section 2 provides an overview of related research work. In Section 3.1, we define the basic symbols, and in Section 3.2, we derive the loss function for updating the world model. In Section 3.3, we first analyze the convergence and sampling methods of the pruning strategy in tabular form and then provide the loss functions for updating the policy and value function. In Section 3.4, we detail how to merge from the policy distribution based on the trained pruning policy to obtain the final action for interacting with the environment. Section 4 presents the experimental results and in Section 5, we summarize the paper.

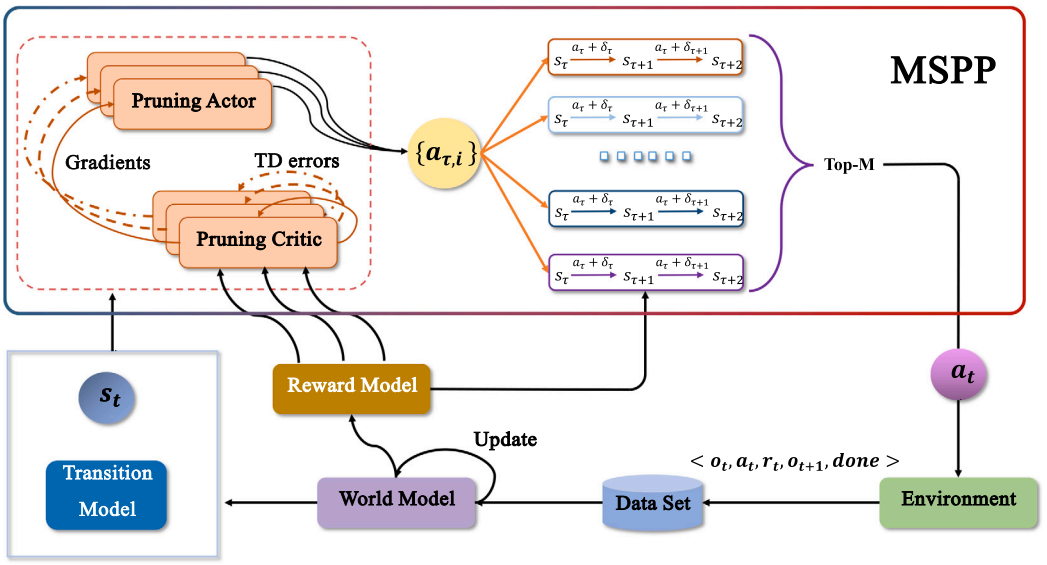

Fig. 2. Diagram of the MSPP. Once the current decision state is inputed into the MSPP, it first performs parallel inference training through multiple pruning policies and then selects the top-M ranked results to determine the final outcome.

2. Related work

We modularize our work into four components: the world model component, the multi-step component, research on multiple models, and work related to cross-entropy fusion.

The world model can be saved in a table by recording the counting of the state-action-next-state counts [12,13]. With the prosperity of neural networks, it has been widely employed for fitting one-step transitions [14]. The optimizer object can be designed to minimize different objective functions, such as the mean squared prediction error of the world model on the next state [15], the multi-step L2-norm [16], or to minimize the KL divergence between the world model and the real environment [11,17]. Theoretical analysis suggests that the value loss increases quadratically as the horizon grows, which is why researchers only use short rollouts [14,18]. However, incorporating appropriate noise in the state is more relevant to the real environment [19]. To address this, researchers have adopted truncated imaginary model rollout length to avoid unreliable fake samples in policy optimization [20,21], but this sacrifices the ability to plan for the long term. This paper aims to enhance the final policy performance by considering diverse perspectives, such as the model bias that is inevitable to eliminate. To achieve this, a partially stochastic sequential latent variable model, RSSM [11], is applied for multiple time steps [9,22].

Repetition of actions in an environment can be seen as an abstraction of actions. In Hierarchical Reinforcement Learning, designing different abstract actions is typically modeled as a Semi-Markov process [23]. However, Hierarchical Reinforcement Learning breaks down complex problems by dividing them into several sub-problems and solving each one individually using a divide-and-conquer approach [24]. In contrast, our diverse pruning policies aim to address the same problem of finding the optimal policy. In Multi-time Scale Markov Decision Processes [25], time-scale MDPs are divided into fast time-scale and slow time-scale, with each representing different task requirements. The method proposed in this paper aims to reduce the redundant action space, enhance the diversity of policies, and improve long-term planning and exploration capabilities, rather than focusing solely on abstract action or task decomposition as adopted in Hierarchical Reinforcement Learning. The n-hop Multiple Markovian Environments are quite similar to our work [26]; however, their focus is on using averaged Jensen-Shannon divergence to aggregate different Q-functions in order to improve the estimation accuracy of value functions in Q-Learning. Unlike our approach, these methods are all model-free reinforcement learning algorithms.

The concept of multi-sequential actions has been introduced in other model-based approaches, such as the multi-step dynamics model. This model is capable of directly outputting the execution sequence of actions [27], allowing for predicting outcomes of H steps into the future. This approach helps prevent fake inputs to the policy and dynamic world model, reducing the accumulation of errors [27]. MPPVE uses multi-step plans instead of multi-step actions. This method aims to directly predict the future K-step action from the current state. However, the presence of biases and significant uncertainties in the world model poses challenges in acquiring such long-term planning capabilities. In contrast, our approach addresses the fake state problem by repeating actions, which is a more realistic approach than attempting to predict future actions. In our world model, transition dynamics are applied at each step, rather than relying on anticipation of future actions. This means that the gradients at the repeat step do not affect the decision, as they are determined by the policy at the decision step. Our method also differs from the MPPVE approach [28] in that it employs a leapfrog approach for calculating gradients, rather than minimizing gradient computation on fake states by predicting K steps ahead. Additionally, our method is not simply an improvement of a single pruning policy, but rather an integration of a combination of different pruning policies.

Ensemble Reinforcement Learning is a method that aims to increase diversity by utilizing multiple base learners. While traditional ensemble strategies focus on Q-function, reward, and loss function [29], these are all vectors or scalars, similar to ensemble learning in supervised learning. However, our MSPP ensemble focuses on the distribution of policies. In order to effectively implement ensemble learning methods, it is crucial to generate diverse base learners and integrate them. This can be achieved through various methods such as using different training algorithms [29], varying weight initialization [30], and utilizing different metrics to measure the distance between policies within the population [31–33]. These techniques promote diversity within the same state-action space. In contrast, our approach generates diversity by designing different spaces through restricting the action space. When it comes to integrating base learners, there are several common methods such as voting, optimal combination, binning, aggregation, weighted aggregation, stacking, and Boltzmann multiplication [29]. However, we have developed a new cross-entropy optimization method that can effectively integrate both discrete and continuous action spaces.

Model Predictive Control (MPC) [34] is a widely adopted technique for leveraging a model to determine an optimal sequence of actions. This method involves generating and evaluating multiple action sequences, with the optimal sequence being selected based on the reward provided by the world model. Despite the presence of a well-performed world model, there's no guarantee of deriving an effective policy [3]. Therefore, research on value-aware [35] and policy-aware [36] approaches focus on loss functions [3]. They are tailored to synchronize the training of dynamic models with policy development. This paper focuses on the potential for diverse policies to improve the performance of the final policy, rather than concentrating on model bias or policy alignment. MSPP employs the learned dynamics model to generate an ensemble of diverse pruning plans, distinguishing itself from ensemble learning approaches that utilize multiple world models [17,37]. Our approach involves the integration of multiple policies for controlling a single world model. In Appendix D, we have included a comparison diagram of the system architecture. Unlike methods that utilize policy sampling across the entire forecasted trajectory [38,39], our method involves sampling exclusively from the initial time-step distributions specified by each pruning policy after their training. It is achieved without the implementation of policy rollout over multiple steps.

3. Methodology

The architecture of the MSPP is depicted in Fig. 2, consisting of two main components: the world model and policy learning. The policy learning is further divided into two parts: a group of multi-pruning agents for a comprehensive understanding of the world model, and the cross-entropy fusion method for aggregating diverse information.

3.1. Preliminaries

MDPs can be defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, providing a mathematical framework for our algorithm. The state space $\mathcal{S}$ is a set of states s, while the action space $\mathcal{A}$ is a set of action a. The state transition function is defined as $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, where $\Delta(\mathcal{S})$ represents the space of probability distribution over $\mathcal{S}$. The reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \left[0, R_{\max}\right]$ assigns a reward $r(s,a)$ to an action a in a state s. The discount factor $\gamma \in [0,1)$ defines a horizon for the problem. The objective of reinforcement learning is to maximize the total expected cumulative reward. In practical problems, we typically only have access to observations $o \in \mathcal{O}$ rather than the full state. These types of problems are usually modeled as partially observable Markov decision processes (POMDPs) [40].

The process for Fig. 2 is as follows: When an observation o_t is received from the environment, the world model first encodes it into a low-dimensional state s_t. The pruning agent then plans and learns from this state s_t using the world model's state transitions and rewards function. Once the agent has updated its parameters, it provides a set of action distribution $\{\pi(a_\tau \mid s_\tau)\}$ for the state s_t. The cross-entropy method then adds noise to this distribution to create an action sequence, which is then used to interact with the world model. Based on the rewards given by the world model, the final action is selected.

3.2. World model

For effective planning, it is crucial to evaluate thousands of action sequences at each time step for the agent. We employ a RSSM, enabling forward predictions entirely within the low-dimension latent state space. Model-based reinforcement learning involves the agent collecting data through its interaction with the environment, and using this data to learn a world model. This world model consists of two components: the transition function model $s_{t+1} \sim f_p(s_t, a_t)$ and the reward function model $r_{t+1} \sim f_r(s_t, a_t)$, as shown in Fig. 2. In RSSM, the transition function model $s_{t+1} \sim f_p(s_t, a_t)$ is divided into two parts: a deterministic part $h_{t+1} = f\left(h_t, s_t, a_t\right)$ and a stochastic part $s_t \sim p\left(s_t \mid h_t\right)$, where h_t represents the deterministic state.

Agents sample data from the real environment to form the dataset $\mathcal{D} = \{o_t, a_t, r_t\}_{t=1}^{T}$, serving as the basis for next round of world model learning. The collected data consists of image observations. RSSM uses an encoder network to map the data from the observation space to a low-dimensional state space $\mathcal{O} \to \mathcal{S}$. Subsequently, in the state space, a pruning agent performs parallel imagination. We denote the time steps of interacting with the real environment as t, and the time steps of interacting with the world model for imagination as τ. The transition probability in world model denoted as $\mathcal{P}\left(s_{\tau+1} \mid s_\tau, a_\tau\right)$. The state transition is updated by a recurrent process that takes the previous latent state and hypothetical subsequent actions, updating the state within a low-dimensional latent space. This process establishes a mapping that associates each latent state with its corresponding reward.

Utilizing the encoder $q\left(s_t | o_{\leq t}, a_{<t}\right)$ and reward model $p(r_t \mid s_t)$, this approach establishes a bound on the data log-likelihood. As a result, the predictive losses for observations $\mathcal{L}(o_{1:T} | a_{1:T})$ in Eq. (1) are optimized through gradient ascent.

$$\mathcal{L}\left(o_{1:T} \mid a_{1:T}\right) := \ln p\left(o_{1:T} \mid a_{1:T}\right) := \ln \prod_{t=1}^{T} p\left(o_t \mid s_t\right) p\left(s_t \mid s_{t-1}, a_t\right) \tag{1}$$

$$= \ln \mathbb{E}_{q\left(s_{1:T} \mid o_{1:T}, a_{1:T}\right)}\left[\prod_{t=1}^{T} \frac{p\left(o_t \mid s_t\right) p\left(s_t \mid s_{t-1}, a_t\right)}{q\left(s_t \mid o_{\leq t}, a < t\right)}\right]$$

$$\geq \mathbb{E}_{q\left(s_{1:T} \mid o_{1:T}, a_{1:T}\right)}\left[\sum_{t=1}^{T} \ln p\left(o_t \mid s_t\right) + \ln p\left(s_t \mid s_{t-1}, a_t\right) - \ln q\left(s_t \mid o_{\leq t, a<t}\right)\right]$$

$$= \sum_{t=1}^{T}\left(\mathbb{E}_{q\left(s_t \mid o_{\leq t}, a_{<t}\right)}\left[\log p\left(o_t \mid s_t\right)\right] - D_{KL}\left(q\left(s_t \mid o_{\leq t}, a_{<t}\right) || p\left(s_t \mid s_{t-1}, a_{t-1}\right)\right)\right).$$

The state transition model is based on a Gaussian normal distribution, with the mean and variance determined by a feedforward neural network parameterized by θ. The function $h_t = f_\theta\left(h_{t-1}, s_{t-1}, a_{t-1}\right)$ is efficiently implemented using a Gated Recurrent Unit (GRU). This ensures that all observational data is comprehensively processed through the encoder's sampling step, preventing any deterministic shortcuts in the reconstruction process. Similarly, the observation model $o_t \sim p_\theta\left(o_t \mid h_t, s_t\right)$ is modeled as a normal distribution, with its parameters defined by a de-convolution network. The reward model $r_t \sim p_\theta\left(r_t \mid h_t, s_t\right)$ is represented as a scalar within a Gaussian framework, with a feedforward neural network specifying its mean and variance parameters.

3.3. Pruning agents

The multi-pruning planning agents utilize the encoded latent state s_t and the transition model to simulate trajectories $\{s_\tau, a_\tau, r_\tau\}_\tau^{\tau+H}$, and employ the reward model for these trajectories, thereby updating the multi-pruning agents themselves. At any given state, the agent can generate a sequence of actions to be executed in the subsequent fixed steps. This sequence of actions, referred to as the repeated pruning policy, is denoted by π^i. Consequently, given a latent state s_τ, the pruning plan $\mathbf{a}_\tau^i = \left(a_\tau^0, a_\tau^1, \cdots, a_\tau^{i-1}\right)$, follows a distribution described in Eq. (2), where i represents the number of repetition times.

$$\mathbf{a}_\tau^i \sim \pi^i\left(\cdot \mid s_\tau\right). \tag{2}$$

For each z within the range 0 to $i-1$, the action anticipated by the latent world model at time step τ is defined from Eq. (3) to Eq. (5).

$$a_{\tau+z} \sim \pi\left(a_\tau \mid s_\tau\right), \tag{3}$$

$$s_{\tau+z+1} \sim \mathcal{P}\left(\cdot \mid s_{\tau+z}, a_\tau\right), \tag{4}$$

$$\pi^i\left(\mathbf{a}_\tau^i \mid s_\tau\right) = \prod_{z=0}^{i-1} \pi\left(a_\tau \mid s_{\tau+z}\right) = \pi(a_\tau \mid s_\tau). \tag{5}$$

For any finite state and action spaces, there is always an optimal policy that is both deterministic and stationary [26,41]. The pruning policy $\pi^i\left(\mathbf{a}_\tau^i \mid s_\tau\right)$ reduces both the action space and the state space of the original MDPs. The optimal pruning policy is the optimal policy within its sub-space. Next, we will introduce how to obtain the optimal policy through the pruning policy iteration theorem by using tabular pruning policy evaluation and pruning policy improvement. Finally, we will present the network approximation pruning policy update equation and theorem.

3.3.1. Pruning policy evaluation

For any pruning policy $\pi^i \in \Pi^i$, the value function over i steps, Q^{π^i} for pruning is defined in Eq. (6).

$$Q^{\pi^i}\left(s_\tau, \mathbf{a}_\tau^i\right) = Q^{\pi^i}\left(s_\tau, a_\tau^0, a_\tau^1, \cdots, a_\tau^{i-1}\right)$$

$$= \mathbb{E}_{p,r,\pi^i}\left[\sum_{z=0}^{i-1} \gamma^z r_{\tau+z} + \gamma^i \sum_{z=0}^{\infty} \gamma^z r_{t+i+z}\right]$$

$$= \mathbb{E}_{p,r}\left[\sum_{z=0}^{i-1} \gamma^z r_{\tau+z} + \gamma^i \mathbb{E}_{\mathbf{a}_{\tau+i}^i \sim \pi^i}\left[Q^{\pi^i}\left(s_{\tau+i}, \mathbf{a}_{\tau+i}^i\right)\right]\right]$$

$$= \mathbb{E}_{p,r}\left[\sum_{z=0}^{i-1} \gamma^z r_{\tau+z}\right] + \gamma^i \mathbb{E}_p\left[V\left(s_{\tau+i}\right)\right]. \tag{6}$$

Thus, the extended Bellman backup operator, denoted as $\mathcal{T}^{\pi^i}$, can be expressed in Eq. (7).

$$\mathcal{T}^{\pi^i} Q\left(s_\tau, \mathbf{a}_\tau^i\right) = \mathbb{E}_{p,r}\left[\sum_{z=0}^{i-1} \gamma^z r_{\tau+z}\right] + \gamma^i \mathbb{E}_p\left[V\left(s_{\tau+i}\right)\right]. \tag{7}$$

Lemma 1 *(Pruning Policy Evaluation). Consider a pruning policy π^i with action space $\mathcal{A}^i \subset \mathcal{A}$ and state space $\mathcal{S}^i \subset \mathcal{S}$. Let there be an initial function $Q_0 : \mathcal{S}^i \times \mathcal{A}^i \to \mathbb{R}$ with a finite action space $|\mathcal{A}^i| < \infty$. By iteratively updating Q_n to Q_{n+1} using $Q_{n+1} = \mathcal{T}^{\pi^i} Q_n$ for all $n \in \mathbb{N}$, the sequence Q_n converges to the value of pruning policy π^i as $n \to \infty$.*

Proof of Lemma 1. For any iteration $n \in \mathbb{N}$, the update from Q_n to Q_{n+1} through $\mathcal{T}^{\pi^i}$ yields Eq. (8) for all $s_\tau \in \mathcal{S}^i$ and $\mathbf{a}_\tau^i \in \mathcal{A}^i$.

$$\begin{aligned} &Q_{n+1}\left(s_\tau, \mathbf{a}_\tau^i\right) = \mathcal{T}^{\pi^i} Q_n\left(s_\tau, \mathbf{a}_\tau^i\right) = \\ &\mathbb{E}_{p,r}\left[\sum_{z=0}^{i-1} \gamma^z r_{\tau+z} + \gamma^i \mathbb{E}_{\mathbf{a}_{\tau+i}^i \sim \pi^i}\left[Q^{\pi^i}\left(s_{\tau+i}, \mathbf{a}_{\tau+i}^i\right)\right]\right]. \end{aligned} \tag{8}$$

To assess convergence by Eq. (9).

$$\begin{aligned} &\left\|Q_{n+1} - Q^{\pi^i}\right\|_\infty \\ &= \max_{s_\tau \in S^i, \mathbf{a}_\tau^i \in \mathcal{A}^i} \left|Q_{n+1}\left(s_\tau, \mathbf{a}_\tau^i\right) - Q^{\pi^i}\left(s_\tau, \mathbf{a}_\tau^i\right)\right| \\ &= \max_{s_\tau \in S^i, \mathbf{a}_\tau^i \in \mathcal{A}^i} \gamma^i \left|\mathbb{E}_{p,\pi^i}\left[Q_n\left(s_{\tau+i}, \mathbf{a}_{\tau+i}^i\right) - Q^{\pi^i}\left(s_{\tau+i}, \mathbf{a}_{\tau+i}^i\right)\right]\right| \\ &\le \max_{s_\tau \in S^i, \mathbf{a}_\tau^i \in \mathcal{A}^i} \gamma^i \left\|Q_n - Q^{\pi^i}\right\|_\infty \\ &= \gamma^i \left\|Q_n - Q^{\pi^i}\right\|_\infty. \end{aligned} \tag{9}$$

The second equation eliminates the immediate rewards from step τ to step $\tau + i$. Therefore, $Q_n = Q^{\pi^i}$ becomes a fixed point under this update mechanism, guaranteeing that the sequence Q_n converges to the value function of the pruning policy π^i, within the subspace as $n \to \infty$. □

3.3.2. Pruning policy improvement

The pruning policy π^i can be updated by Eq. (10).

$$\pi_{new}^i = \arg\max_{\pi^i \in \Pi^i} \sum_{\mathbf{a}_\tau^i \in \mathcal{A}^i} \pi^i\left(\mathbf{a}_\tau^i \mid s_\tau\right) Q^{\pi_{old}^i}\left(s_\tau, \mathbf{a}_\tau^i\right). \tag{10}$$

For each state, π_{new}^i achieves greater value than π_{old}^i after applying Eq. (10). This method ensures that the policy is gradually refined to maximize the expected return, as represented by the Q value, with the pruning policy's action and state space.

Lemma 2 *(Pruning Policy Improvement). Assume the action space $\mathcal{A}^i \subset \mathcal{A}$ and the state space $\mathcal{S}^i \subset \mathcal{S}$ define the domains of the pruning policy. For any policy $\pi_{old}^i \in \Pi^i$, mapping $\mathcal{S}^i$ to $\mathcal{A}^i$ within a finite action space $|\mathcal{A}^i| < \infty$, updating π_{old} according to Eq. (10) ensures that $Q^{\pi_{new}^i}(s_\tau, \mathbf{a}_\tau^i) \ge Q^{\pi_{old}}(s_\tau, \mathbf{a}_\tau^i)$ for all $s_\tau \in \mathcal{S}^i$ and $\mathbf{a}_\tau^i \in \mathcal{A}^i$.*

Proof of Lemma 2. Starting from the definition of the state value function, we obtain Eq. (11) for any state $s_\tau \in \mathcal{S}^i$.

$$\begin{aligned} &V^{\pi_{\text{old}}^i}\left(s_\tau\right) = \sum_{\mathbf{a}_\tau^i \in \mathcal{A}^i} \pi_{\text{old}}^i\left(\mathbf{a}_\tau^i \mid s_\tau\right) Q^{\pi_{\text{old}}}\left(s_\tau, \mathbf{a}_t^i\right) \\ &\le \sum_{\mathbf{a}_\tau^i \in \mathcal{A}^i} \pi_{\text{new}}^i\left(\mathbf{a}_\tau^i \mid s_\tau\right) Q^{\pi_{\text{old}}}\left(s_\tau, \mathbf{a}_t^i\right). \end{aligned} \tag{11}$$

Expanding upon this inequality gives rise to Eq. (12) for every $s_\tau \in \mathcal{S}^i$ and $\mathbf{a}_\tau^i \in \mathcal{A}^i$.

$$\begin{aligned} &Q^{\pi_{\text{old}}^i}\left(s_\tau, \mathbf{a}_\tau^i\right) = \mathbb{E}_{p,r}\left[\sum_{z=0}^{i-1} \gamma^z r_{\tau+z} + \gamma^i V^{\pi_{\text{old}}}\left(s_{\tau+i}\right)\right] \\ \le &\mathbb{E}_{p,r}\left[\sum_{z=0}^{i-1} \gamma^z r_{\tau+z} + \gamma^i \sum_{\mathbf{a}_{\tau+i}^i \in \mathcal{A}^i} \pi_{\text{new}}^i\left(\mathbf{a}_{\tau+i}^i \mid s_{\tau+i}\right) Q^{\pi_{\text{old}}}\left(s_{\tau+i}, \mathbf{a}_{\tau+i}^i\right)\right] \\ = &\mathbb{E}_{p,r,\pi_{\text{new}}}\left[\sum_{z=0}^{2r-1} \gamma^z r_{\tau+z} + \gamma^{2r} V^{\pi_{\text{old}}}\left(s_{\tau+2r}\right)\right] \\ &\vdots \\ = &\mathbb{E}_{p,r,\pi_{\text{new}}}\left[\sum_{z=0}^{\infty} \gamma^z r_{\tau+z}\right] = Q^{\pi_{\text{new}}}\left(s_\tau, \mathbf{a}_\tau^i\right). \end{aligned} \tag{12}$$

This demonstrates that the policy update from π^i_{old} to π^i_{new} ensures an improvement or at least equality in the Q value across all states and actions within the pruning policy's operational framework. □

3.3.3. Pruning policy iteration

Pruning Policy iteration alternates between pruning policy evaluation and pruning policy improvement until the pruning policy π^i_n converges to the optimal value π^i_* within its corresponding policy space Π^i.

Theorem 1 *(Pruning Policy Iteration). Initiating with any preliminary mapping $\pi^i_0 \in \Pi^i : \mathcal{S}^i \to \mathcal{A}^i$ where $|\mathcal{A}^i| < \infty$, the corresponding $Q^{\pi^i_n}$ in the pruning policy evaluation phase can be calculated. Subsequently, update π^i_n to π^i_{n+1} in the pruning policy improvement step for all $n \in \mathbb{N}$. This process ensures convergence of π^i_n to the optimal policy π^i, satisfying $Q^{\pi^i_*}\left(s_\tau, \mathbf{a}^i_\tau\right) \geq Q^{\pi^i}\left(s_\tau, \mathbf{a}^i_\tau\right), \forall s_\tau \in \mathcal{S}^i, \mathbf{a}^i_\tau \in \mathcal{A}^i$.*

Proof. Given the monotonic increase of $\{Q^{\pi^i_n}\}$ with respect to n and the bounded nature of $\pi^i \in \Pi^i$, the sequence π^i converges to a policy π^i_*. Leveraging Lemma 2, we can see that Eq. (13) holds for each $s_\tau \in \mathcal{S}^i$ and any $\pi^i \in \Pi^i$.

$$V^{\pi^i_*}\left(s_\tau\right) \geq \sum_{\mathbf{a}^i_\tau \in \mathcal{A}^i} \pi^i\left(\mathbf{a}^i_\tau \mid s_\tau\right) Q^{\pi^i_*}\left(s_\tau, \mathbf{a}^i_\tau\right). \tag{13}$$

This results in Eq. (14).

$$\begin{aligned}
Q^{\pi^i_*}\left(s_t, \mathbf{a}^i_\tau\right) &= \mathbb{E}_{p,r}\left[\sum_{z=0}^{i-1} \gamma^z r_{\tau+z} + \gamma^i V^{\pi^i_*}\left(s_{\tau+i}\right)\right] \\
&\geq \mathbb{E}_{p,r}\left[\sum_{z=0}^{i-1} \gamma^z r_{\tau+z} + \gamma^i \sum_{\mathbf{a}^i_\tau \in \mathcal{A}^i} \pi^i\left(\mathbf{a}^i_\tau \mid s_{\tau+i}\right) Q^{\pi^i_*}\left(s_{\tau+i}, \mathbf{a}^i_\tau\right)\right] \\
&= \mathbb{E}_{p,r,\pi}\left[\sum_{z=0}^{2r-1} \gamma^z r_{\tau+z} + \gamma^{2r} V^{\pi^i_*}\left(s_{\tau+2r}\right)\right] \\
&\vdots \\
&= \mathbb{E}_{p,r,\pi}\left[\sum_{z=0}^{\infty} \gamma^z r_{\tau+z}\right] = Q^{\pi^i}\left(s_\tau, \mathbf{a}^i_\tau\right).
\end{aligned} \tag{14}$$

Conclusively prove that π^i_* represents the optimal policy in Π^i. □

3.3.4. Pruning policy with neural network

To broaden the applicability of the tabular pruning policy to a wider range of scenarios, including those with continuous or high-dimensional state-action spaces within world models, we utilize the actor-critic approximation approach outlined in [11]. This approach serves as the fundamental framework for developing planning agents, which are trained to optimize their decision-making processes by considering rewards projected along trajectories within a fixed to a horizon H and anticipating the expected rewards of states extending beyond this horizon. Within this framework, the action model in Eq. (15) is parameterized by ϕ, allowing for a more adaptable and scalable approach to policy learning in complex environments.

$$a_\tau \sim q_\phi\left(a_\tau \mid s_\tau\right). \tag{15}$$

The value model in Eq. (16), which is represented by the parameters ψ, is designed to approximate the expected rewards over a fixed horizon H from a given state.

$$v_\psi\left(s_\tau\right) \approx \mathbb{E}_{q(\cdot|s_\tau)}\left(\sum_{\tau=t}^{t+H} \gamma^{\tau-t} r_\tau\right), \tag{16}$$

where t, τ, and γ are used to represent the current moment, subsequent moments within the trajectory imagination, and a discount factor, respectively. The action model generates a policy action based on the current state, while the value model estimates the expected rewards based on those policies. To optimize the learning process, a formula is used to balance the trade-off between high variance and high bias in value estimation [11,1]. This balanced approach improves the efficiency and effectiveness of the learning mechanism in complex environments. The update target is defined in Eq. (17).

$$\mathrm{V}_\lambda\left(s_\tau\right) \doteq \lambda^{H-1}\, \mathrm{V}^H_{\mathrm{N}}\left(s_\tau\right) + (1-\lambda) \sum_{n=1}^{H-1} \lambda^{n-1}\, \mathrm{V}^n_{\mathrm{N}}\left(s_\tau\right), \tag{17}$$

where the horizon parameter H specifies the extent to which future events are considered in the policy's decision-making process. The function $\mathrm{V}^m_{\mathrm{N}}\left(s_\tau\right)$ in Eq. (18) represents the expected return from state s_τ over m steps. The term h is defined as $\min(\tau+m, \tau+H)$.

$$\mathrm{V}_{\mathrm{N}}^{m}\left(s_{\tau}\right) \doteq \mathbb{E}_{q_{\theta}, q_{\phi}}\left(\sum_{n=\tau}^{h-1} \gamma^{n-\tau} r_{n}+\gamma^{h-\tau} v_{\psi}\left(s_{h}\right)\right). \tag{18}$$

In the context of the MSPP world model, state transitions occur at a fixed timescale t_s. The execution time of a decision by a pruning planning agent, denoted by t_i, is a multiple of t_s, with I representing the set of all such multiples. The process of deriving an action $a_\tau \sim q_\phi\left(a_\tau \mid s_\tau\right)$ from the current state s_τ involves the agent's actor at t_i skipping states and mapping action $i-1$ times, then applying the chosen action i times, as defined in Eq. (19).

$$\pi^{i}(a_{\tau:\tau+H} \mid s_{\tau:\tau+H})=\begin{cases} a_{\tau} \sim q_{\phi_i}\left(a_{\tau} \mid s_{\tau}\right) & i \mid\left((\tau-t)/t_s\right) \\ a_{\tau}=a_{\tau-1} & \text{otherwise.} \end{cases} \tag{19}$$

To ensure a cohesive optimization process, the objectives of pruning agents are determined by two key formulas that aim to achieve the goals of both the actor and value components. To maintain stable policy updates for pruning planning agents, it is crucial to keep the world model constant during their learning phase. The optimization objective for the actor component of a pruning agent is defined in Eq. (20).

$$J\left(\pi_{\phi_i}\right)=\mathbb{E}_{q_{\theta}, q_{\phi_i}}\left[\sum_{\tau=t}^{t+H} \mathrm{V}_{\lambda}\left(s_{\tau}\right)\right]. \tag{20}$$

To refine the value formula, the optimization objective is set by Eq. (21).

$$J\left(\pi_{\psi_i}\right)=\mathbb{E}_{q_{\theta}, q_{\psi_i}}\left[\sum_{\tau=t}^{t+H} \frac{1}{2}\left\|v_{\psi_i}\left(s_{\tau}\right)-\mathrm{V}_{\lambda}\left(s_{\tau}\right)\right\|^{2}\right]. \tag{21}$$

The Multi Pruning Policy Gradient Theorem provides a structured approach for calculating the value function $V^{\pi^i_{\phi_i}}\left(s_\tau\right)$ gradients, allowing for the refinement of these objectives.

Theorem 2 *(Multi-Step Pruning Policy Gradient). For any MDP, the gradient of value function V relative to policy parameters is computed as the expected gradient of the action-value function Q across state and action distributions in Eq. (22).*

$$\nabla_{\phi_i} V^{\pi^i_{\phi_i}}\left(s_{\tau}\right)=\mathbb{E}_{s_{\tau} \sim \mu_{ri}, a_{\tau} \sim \pi^i_{\phi_i}}\left[Q^{\pi^i_{\phi_i}}(s_r, a_r) \nabla_{\phi_i} \ln \pi^i_{\phi_i}\left(a_r \mid s_r\right)\right]. \tag{22}$$

The expression $\nabla_{\phi_i} V^{\pi^i_{\phi_i}}\left(s_\tau\right)$ represents the gradient of the value function $V^{\pi^i_{\phi}}$ for state s_τ, with respect to the pruning policy parameters ϕ_i in the ith step, where s_r and a_r refer to the state and action of the multi-pruning sub-space. $s \sim u_{ri}$ represents the pruning discount state distribution of the sub-space. The policy π^i determines the probability of taking an action given a state and its parameters. The term $\mathbb{E}_{s \sim \mu_{ri}, a_\tau \sim \pi^i_{\phi_i}}$ represents the expected value over the states and actions sampled from the distribution μ_{ri} and the policy $\pi^i_{\phi_i}$.

Proof. See Appendix A. □

3.4. Cross entropy fusion

Following Theorem 2, the pruning policy $\pi^i_{\phi_i}$ can be updated by independently adjusting the network parameters ϕ_i separately for each $i \in I$. Once the pruning policies have been trained, they can be used to infer different perspectives within the same world model. At time step t, given the latent state s_τ is inputted into pruning policies $\{\pi^i_{\phi_i}\}$, and the output of the pruning policy network is the distribution set $\{\pi^i_{\phi_i}\left(a_\tau \mid s_\tau\right)\}$. To verify whether different pruning policies have a distinct understanding of the given state s_τ, it is necessary to limit the merge to only the action distribution $\{\pi^i_{\phi_i}\left(a_\tau \mid s_\tau\right)\}$ for the given state s_τ, rather than the entire policy $\{\pi^i_{\phi_i}\}$ for rollout, which would introduce other latent states.

We denote the noise sequence at time-step τ with horizon H as $\boldsymbol{\delta}_\tau=\left\{\delta_\tau, \delta_{\tau+1}, \ldots, \delta_{\tau+H}\right\}$, initializing the noise distribution as a standard normal distribution. Similarly, we define $a^{i,j}_{\tau:\tau+H}=\left\{a^{i,j}_\tau+\delta^{i,j}_\tau, a^{i,j}_\tau+\delta^{i,j}_{\tau+1}, \ldots, a^{i,j}_\tau+\delta^{i,j}_{\tau+H}\right\}$. The optimization process for pruning agents involves generating and utilizing trajectories to inform the decision-making of cross-entropy planning agents, as shown in Eq. (23) and Eq. (24).

$$a^{i,j}_{\tau:\tau+H}, s^{i,j}_{\tau:\tau+H} \sim q_{\phi_i,\theta}\left(a_{\tau:\tau+H}, s_{\tau:\tau+H} \mid s_\tau\right). \tag{23}$$

This equation describes the generation of action and state sequences over a horizon H for each pruning agent i and each sampling instance j, based on the current state s_τ and governed by the pruning policy parameter ϕ_i and the world model parameter θ.

$$tr^{i,j} \doteq\left\{a^{i,j}_\tau, s^{i,j}_\tau\right\}^{t+H}_{\tau=t}, \tag{24}$$

where $tr^{i,j}$ represents the trajectory of actions and states from time t to $t+H$ for the i-th pruning agent on the i-th sampling. The decision-making process of cross-entropy planning agents leverages these trajectories, denoted as $\left\{tr^{i,j} \mid i \in I, j \in J\right\}$, which

are sampled J times from the current state s_τ by all pruning agents $\left\{ q_{\phi_i}\left(a_\tau \mid s_\tau\right) \mid i \in I \right\}$. This approach allows the cross-entropy planning method to navigate within the constraints set by the step scale of the world model t_s and the capabilities of the diverse pruning agents $\{t_i\}$. The timing for actions executed by the cross-entropy planning agent is thus specified in Eq. (25).

$$t_* \in \left\{ t_s, \min_i \left(t_i\right) \right\} \bigcup \left\{ k \cdot i \cdot t_s \mid k, i \in \mathbb{N}^+ \right\}. \tag{25}$$

In this planning scheme, the expected reward within a given noise $\boldsymbol{\delta}_\tau = \left\{ \delta_\tau, \delta_{\tau+1}, \ldots, \delta_{\tau+H} \right\}$ is expressed in Eq (26).

$$R_\theta(s_\tau, \boldsymbol{\delta}_\tau) = \mathbb{E}\left[\sum_{z=\tau}^{\tau+H} \gamma^z r\left(s_z, a_\tau + \delta_z\right) \right], \tag{26}$$

where $s_{z+1} \sim p_\theta(s_{z+1} \mid s_z, a_\tau + \delta_z)$. The policy of the cross-entropy method iteratively refines the search for the optimal action sequence at each timestep, τ, as delineated in Equation (27).

$$a^* = \arg\max_{a_{t:t+H}} \mathbb{E}\left[R_\theta\left(s_\tau\right) \right]. \tag{27}$$

According to the theory proposed by [42], the Gaussian distribution for action distribution can be represented as natural exponential families, with the form $f_{w_j}(\delta^j) = \exp\left(w^T \Gamma(\delta) - K(w)\right)$. This distribution is parametrized by $w_j \in W$, and is initialized by $\{\pi^i_{\phi_i}\left(a_\tau \mid s_\tau\right)\}$. The function $K(w)$ is defined as $\ln \int_\Delta \exp\left(w^T \Gamma(\delta)\right) \nu(d\delta)$, where ν is a discrete measure on Δ. Note that $\nabla K(w) = E_w[\Gamma(\delta)]$. We can utilize importance sampling $a_{j+1}(\delta^j) = S(R_\theta(\delta^j)) f_{w_j}(\delta^j) / E_{a_j}[S(R_\theta(\delta^j))]$ to form the process of choosing Top-M. Here $S(\cdot)$ is a positive increasing (possibly iteration-varying) function. When $j \to \infty$, the action $a_j \to a^*$, therefore, the objective is to minimize the Kullback-Leibler (KL) divergence between a_{j+1} and f_{w_j}, as detailed in Eq. (28).

$$w_{j+1} = \arg\min_{w \in W} \mathbb{D}_{KL}\left(a_{j+1}, f_{w_j}\right), \tag{28}$$

where the KL divergence $\mathbb{D}\left(a_{j+1}, f_w\right)$ is defined as $\int_\Delta \ln\left(a_{j+1}(\delta)/f_w(\delta)\right) a_{j+1}(\delta) \nu(dx)$. In practice, we utilize Eq. (29) to update f_w.

$$\tilde{a}_{j+1}(\delta) = \alpha_j a_{j+1}(\delta) + \left(1 - \alpha_j\right) f_{w_j}(\delta), \tag{29}$$

where $\alpha \in (0, 1]$ is the smoothing parameter. The mean vector function is defined as $m(w) := \mathbb{E}(\Gamma(\delta))$. Thus, $m\left(w_{j+1}\right) = E_{w_{j+1}}[\Gamma(\delta)] = E_{\tilde{g}_{j+1}}[\Gamma(\delta)]$, as described in Eq. (30).

$$m\left(w_{j+1}\right) = \alpha_j \frac{E_{a_j}[S(R_\theta(\delta))\Gamma(\delta)]}{E_{a_j}[S(R_\theta(\delta))]} + \left(1 - \alpha_j\right) m\left(w_j\right). \tag{30}$$

The difference between $m_{w_{j+1}}$ and m_{w_j} is expressed in Eq. (31).

$$\begin{aligned} m\left(w_{j+1}\right) - m\left(w_j\right) &= \alpha_j \frac{E_{a_k}\left[S(R(\delta))\left(\Gamma(\delta) - m\left(w_j\right)\right)\right]}{E_{a_k}[S(R(\delta))]} \\ &= -\alpha_k \nabla_w \mathbb{D}\left(a_{k+1}, f_w\right)\Big|_{w=w_k}. \end{aligned} \tag{31}$$

Therefore, the direction of update of the mean vectors at each step is in the direction of mining KL between a_{j+1} and f_{w_j}. This process is shown in Algorithm 1. The overall process of iteration with the environment is shown in Algorithm 2.

4. Experiments

In this section, we focus on three primary questions: 1) Can our proposed pruning strategy effectively capture a diverse understanding of the environment? 2) How does the performance of our approach compare to existing state-of-the-art (SOTA) reinforcement learning methods for understanding the world model? 3) What is the impact of varying the values of i in the pruning policy on our approach?

4.1. Experiment settings

We evaluated the effectiveness of the MSPP bi-level planning algorithm, which utilizes parallel multi-step pruning agents. Our initial focus is on a comprehensive assessment of a single multi-step pruning agent, examining its impact on performance to clarify the pruned agent's understanding of a consistent world model. We then investigate the influence of each pruning agent on overall performance and illustrate our findings within the context of the DeepMind Control Suite [43,44]. Our analysis also compares our approach to established benchmarks in MBRL and various sampling policies. Our main interest is in deciphering the latent state space encapsulated by the world model. To this end, we draw comparisons with three distinct methods: a Cross-Entropy Method (CEM), which operates independently of a world model; the Planet method, which leverages a world model to refine policy; and a technique specifically optimized for reinforcement learning that incorporates a world model. The differences between the various comparative

Algorithm 1 Cross Entropy Method via Pruning Agents.

World Model:		**Hyper-parameters**:	
Transition	$q_\theta\left(s_t \mid s_{t-1}, a_{t-1}\right)$	Optimization times	U
Reward	$q_\theta\left(r_t \mid s_t\right)$	Time scales	I
Action Pool	$\left\{q_{\phi_i}\left(a_t \mid s_t\right) \mid i \in I\right\}$	Imagination horizon	H
Value Pool	$\left\{v_{\psi_i}\left(s_t\right) \mid i \in I\right\}$	Candidates per iteration	J
		Top candidates to fit	K

Initialize s_0 with current state s_t
for *optimization iteration* $u = 0 \cdots U$ do
  for *action sequence* $j = 1...J$
    if $u = 0$ **then**
      for *time scale* i *in* I
        $a_{\le H}^{i,j}, s_{\le H+1}^{i,j} \sim q_{\phi_i,\theta}\left(a_{\le H}, s_{\le H+1} \mid s_0\right)$
        $R^{i,j} = \sum_{\tau=t+1}^{t+H+1} \mathbb{E}\left[p_\theta\left(r_\tau \mid s_\tau^{i,j}\right)\right]$
      end for
      $R^j \leftarrow \left\{R^{i,j} \mid i \in I\right\}$
    else
      $a_{\le H}^j \sim q\left(a_{t:t+H}\right)$
      $s_{\le H}^j \sim q_\theta\left(s_{\le H} \mid a_{\le H}, s_0\right)$
      $R^j = \sum_{\tau=t+1}^{t+H+1} \mathbb{E}\left[p_\theta\left(r_\tau \mid s_\tau^j\right)\right]$
    end if
  end for
  $\mathcal{K} \leftarrow \operatorname{argsort}_{1:K}\left\{R^j\right\}_{j=1}^J$
  $\mu_{t:t+H} = \frac{1}{K}\sum_{k\in\mathcal{K}} a_{t:t+H}^k$
  $\sigma_{t:t+H} = \frac{1}{K-1}\sum_{k\in\mathcal{K}}\left|a_{t:t+H}^k - \mu_{t:t+H}\right|$
  $q\left(a_{t:t+H}\right) \leftarrow \operatorname{Normal}\left(\mu_{t:t+H}, \sigma_{t:t+H}^2 \mathbb{I}\right)$
end for
$a_t \leftarrow \mu_t$
return a_t

Algorithm 2 Multi-Step Pruning Policy (MSPP).

World Models:		**Hyper-parameters**:	
Representation	$p_\theta\left(s_t \mid s_{t-1}, a_{t-1}, o_t\right)$	Seed episodes	S
Transition	$q_\theta\left(s_t \mid s_{t-1}, a_{t-1}\right)$	Time scales	I
Reward	$q_\theta\left(r_t \mid s_t\right)$	Collect interval	C
Action Pool	$\left\{q_{\phi_i}\left(a_{t,i} \mid s_t\right) \mid i \in I\right\}$	Batch size	B
Value Pool	$\left\{v_{\psi,i}\left(s_t\right) \mid i \in I\right\}_i$	Sequence length	L
Action	$q_\omega\left(a_t \mid s_t, \left\{a_{t,i} \mid i \in I\right\}\right)$	Imagination horizon	H
Value	$v_\kappa\left(s_t\right)$	Training repeat	G

Initialize dataset D with random seed episodes.
Initialize parameters $\theta, \left\{\phi_i\right\}, \left\{\psi_i\right\}$ of neural networks randomly.
while *not converged* **do**
  for *update step* $c = 1..C$
    Draw B data sequences $\left\{\left(a_t, o_t, r_t\right)\right\}_{t=k}^{k+L} \sim D$
    Compute model states $s_t \sim p_\theta\left(s_t \mid s_{t-1}, a_{t-1}, o_t\right)$
    Update θ using representation learning.
    for *update step* $g = 1..G$
      for *time scale* i *in* I
        Imagine trajectories $\left\{\left(s_\tau, a_{\tau,i}\right)\right\}_{\tau=t}^{t+H}$
        Predict rewards $\mathbb{E}\left(q_\theta\left(r_\tau \mid s_\tau\right)\right)$
        Predict values $v_{\psi_i}\left(s_\tau\right)$
        Update ϕ_i and ψ_i
      end for
    end for
  end for
  $o_1 \leftarrow env.reset()$
  for *time step* $t = 1..T$
    Compute $s_t \sim p_\theta\left(s_t \mid s_{t-1}, a_{t-1}, o_t\right)$
    Compute a_t from Algorithm 1
    Add experience noise to action.
    $r_t, o_{t+1} \leftarrow env.step(a_t)$
    Extend dataset $D \leftarrow D \bigcup \left\{(o_t, a_t, r_t)_{t=1}^T\right\}$
  end for
end while

experiments are as follows: the cross-entropy method and Planet sample actions from a random distribution, the Dreamer method samples actions from the learned policy distribution of the agent, and MSPP samples actions from the diversity policy distribution.

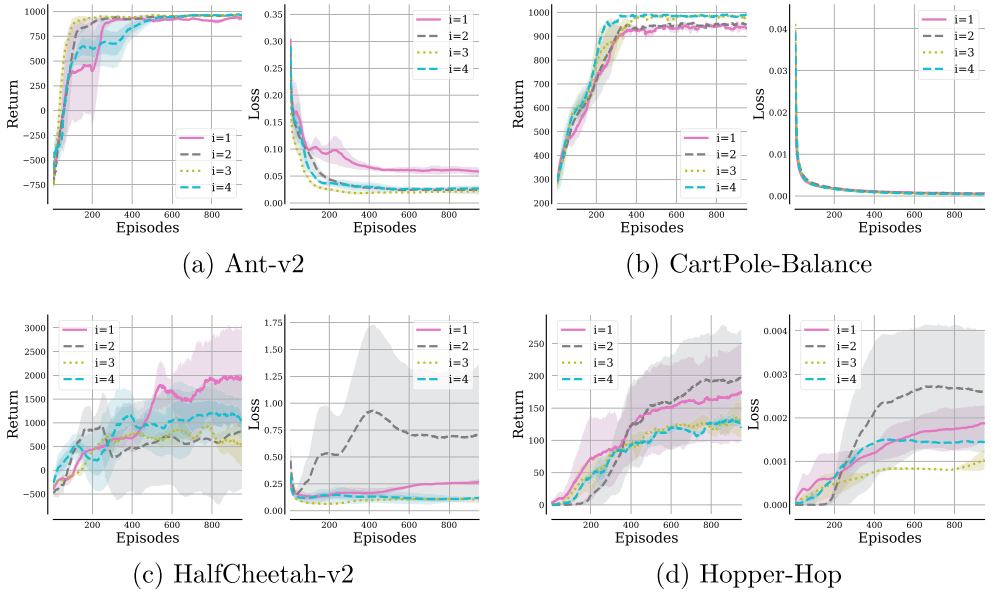

Fig. 3. The results conducted across varying action pruning scales, denoted as i, yielded insightful outcomes. Specifically, $i = 1$ corresponds to the original Dreamer algorithm, while other values of i represent the application of different pruning actions on their respective pruning scales. "Loss" in this context refers to the reward loss during the learning of the world model.

In all experiments, the form of input consists of third-person camera images with dimensions of $64 \times 64 \times 3$ pixels. Uniform hyperparameters are applied across all tasks. The parameter settings for MSPP are the same as those for Dreamer, except for any noted differences. To speed up the training process, an initial dataset comprising $S = 5$ episodes is generated through the execution of 500 steps with random actions, all in parallel. Additionally, the Recurrent State-Space Model (RSSM) is trained in parallel. The size of the replay buffer is set to 1,000,000, while the batch size utilized for sampling is 51. The code is available as open source at https://github.com/tinyzqh/MSPP, ensuring that all performance evaluations are carried out using a standardized set of parameters, with the only modification of the names of the environments as needed.

4.2. Analyzing multi-step pruning agents

To validate whether pruning strategies can lead to a diverse understanding of the same environment model, we independently execute each pruning strategy. This means that we run each pruning strategy separately to see if it can produce different results. Intuitively, the variation in the reward loss curve within the environment model reflects the agent's differing perceptions of the environment. Pruning agents are trained to maximize cumulative rewards based on latent imagination samples derived from the learned world model. The architecture includes a hidden layer with 200 units, and the imagination horizon is set at $H = 15$ for each distinct perspective pruning actor. The learning rates for both actor and value models are uniformly set at 8×10^{-5}. The results from Dreamer show that final performance is greatly affected by the control frequency hyperparameters, with a setting of 2 being the most effective for various tasks. Therefore, we use a control frequency of 2 in our approach.

Fig. 3 presents the results of the multi-perspective actor experiment, showing the episode rewards obtained during the training at different action pruning scales. The shaded areas around the lines represent the standard deviation calculated from two different seeds. Our findings indicate that the reward losses under different pruning policies exhibit a trend of diverse divergence, demonstrating varied understandings of the environment. Notably, the final control performance does not show a monotonic relationship with the quality of reward loss learning. Specifically, as illustrated in Fig. 3 (d), despite a significant reward loss at $R = 2$, the highest score is still achieved. This demonstrates that multi-pruning analysis, a feature of the pruning agents, is capable of providing additional insights to avoid local optima without incurring significant performance degradation, or in some cases achieving superior returns.

4.3. Assessing final performance

After obtaining multiple pruning policies, we verify whether there exists a policy factor within the probability distribution of multiple policies that enhances the final performance. Therefore, we neither adopt any gradient-based optimization method nor introduce additional state information. Instead, we sample data from the given state distribution and then sort and reorganize it to arrive at the final policy output. The iterative pruning actions, carried out by four distinct pruning agents, are designated as $I = \{1, 2, 3, 4\}$. Each pruning agent is trained individually using a multi-process asynchronous technique. Additionally, the training repetition parameter for the sub-actors denoted as G, is set to 5.

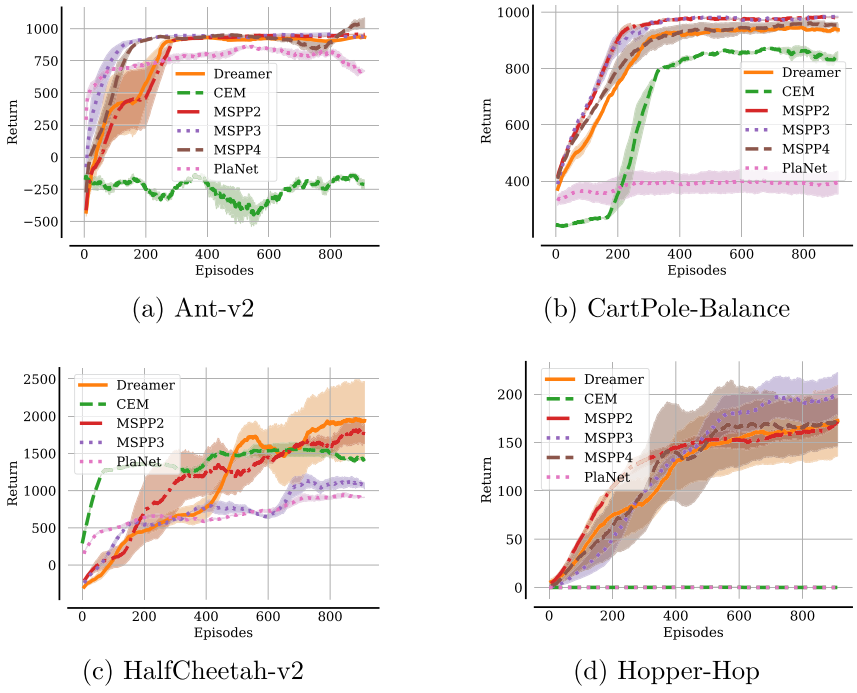

Fig. 4. The comparison of the training performance of different agent designs, including MSPP with its multi-pruning agents, and benchmarks such as Dreamer and PlaNet. The comparison is made across several episodes, with performance metrics plotted at each environment step to demonstrate the effectiveness and efficiency of the different approaches in learning and decision-making within specified environments.

Fig. 4 illustrates the training process of the Dreamer, PlaNet, and MSPP models, where they achieve episode returns after the same number of environment steps. Notably, the MSPP model outperforms the others on three randomly selected tasks, showcasing its superior performance. This result highlights the effectiveness of our MSPP model when multi-pruning agents are optimally configured in training scenarios.

It demonstrates that diverse pruning strategies can provide different perspectives on the same world model, and some of these perspectives can significantly improve the performance of the final task. Two points worth mentioning: (1) The quantity of pruning agents does not exhibit a direct positive correlation with the ultimate performance. This is attributed to interference caused by excessive noise within the distribution. (2) The training repetition hyperparameter G exhibits minimal influence on the ultimate performance, suggesting that the key to enhancing performance lies in the strategic deployment of pruning policies rather than the repetition frequency of training.

5. Conclusions and future work

This paper introduces a novel approach to understanding world models MSPP. We provide a theoretical analysis of this algorithm in tabular form, demonstrating that it can converge from any given initial policy to the optimal policy within a subspace. Subsequently, we extend this approach to neural network approximation methods and present the updated theorem of the MSPP. Finally, through experiments conducted under the same world model, we verify that the MSPP can identify multiple policy factors that enhance performance. Although this article focuses on model-based reinforcement learning algorithms, the concept can be extended to any two-stage optimization method, such as MPC. The optimization in the first stage can take a diversity-oriented approach, while the second stage can involve integration.

CRediT authorship contribution statement

Zhiqiang He: Writing – review & editing, Writing – original draft. **Wen Qiu:** Writing – review & editing, Supervision. **Wei Zhao:** Supervision. **Xun Shao:** Supervision. **Zhi Liu:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Appendix A. Proof of MSPP gradient theorem

Theorem 2 proposes that for any given MDP that has a discount reward structure, the gradient of the value function $V^{\pi^i_{\phi_i}}$ with respect to the policy parameters can be calculated using the expected value of the gradient of the action-value function $Q^{\pi^i_{\phi_i}}$ under the state and action distributions at multiple steps, expressed in Eq. (A.1).

$$\nabla_{\phi_i} V^{\pi^i_{\phi_i}}\left(s_\tau\right) = \mathbb{E}_{s_\tau \sim \mu_{ri}, a_\tau \sim \pi^i_{\phi_i}}\left[Q^{\pi^i_{\phi_i}}(s_r, a_r) \nabla_{\phi_i} \ln \pi^i_{\phi_i}\left(a_r \mid s_r\right)\right]. \tag{A.1}$$

The notion $\nabla_{\phi_i} V^{\pi^i_{\phi_i}}\left(s_\tau\right)$ represents the gradient of the value function $V^{\pi^i_\phi}$ for state s_τ, with respect to the i step pruning policy parameters ϕ_i. where s_r, a_r is state, action space of sub multi-pruning. $s \sim u_{ri}$ represents sub space pruning state distribution. The policy π^i determines the probability of taking an action given a state and its parameters. The term $\mathbb{E}_{s\sim\mu_{ri}, a_\tau \sim \pi^i_{\phi_i}}$ represents the expected value over the states and actions sampled from the multi-pruning state distribution μ_{ri} and the policy $\pi^i_{\phi_i}$.

Proof of Theorem 2. The proof begins by defining how actions are chosen in a multi-pruning context. At each time step t, the imagined action of latent env model $a_{\tau:\tau+H}$ follows the policy $\pi^i_{\phi_i}\left(a_{\tau:\tau+H} | s_{\tau:\tau+H}\right)$ from Eq. (A.2).

$$\pi^i_{\phi_i}\left(a_{\tau:\tau+H} | s_{\tau:\tau+H}\right) = \begin{cases} a_\tau \sim q_{\phi_i}\left(a_\tau \mid s_\tau\right) & i \mid \left((\tau - t)/t_s\right) \\ a_\tau = a_{\tau-1} & \text{otherwise,} \end{cases} \tag{A.2}$$

where i is repeat times. The policy $\pi^i_{\phi_i}$ at a future time $\tau + H$ given the current time τ and state s_τ is defined in Eq. (A.2). Here, a_τ denotes the action at the resolution of the imagination trajectory.

The action-value function $Q^{\pi^i_{\phi_i}}$ is then expended recursively in terms of rewards and future action-value functions, which includes terms for immediate reward r, the action-value function at the next step $Q^{\pi^i_{\phi_i}}\left(s_{\tau+1}, a_\tau\right)$ (here suppose the action at time step $\tau+1$ choose the action of last time step τ), and subsequent action-value functions. At the repeat time step, the state-action is determined as the action is the same as last time step. For given s_τ and a_τ, under pruning policy π^i, we obtain Eq. (A.3).

$$\begin{aligned}
Q^{\pi^i_{\phi_i}}(s_\tau, a_\tau) &= r_{s_\tau, a_\tau} + \gamma \sum_{s_{\tau+1}} \mathcal{P}\left(s_{\tau+1} \mid s_\tau, a_\tau\right) Q^{\pi^i_{\phi_i}}\left(s_{\tau+1}, a_\tau\right) \\
&= r_{s_\tau, a_\tau} + \gamma \sum_{s_{\tau+1}} \mathcal{P}\left(s_{\tau+1} \mid s_\tau, a_\tau\right)\left(r_{s_{\tau+1}, a_\tau} + \gamma \sum_{s_{\tau+2}} \mathcal{P}\left(s_{\tau+2} \mid s_{\tau+1}, a_\tau\right) Q^{\pi^i_{\phi_i}}(s_{\tau+2}, a_\tau)\right) \\
&= r_{s_\tau, a_\tau} + \gamma \sum_{s_{\tau+1}} \mathcal{P}\left(s_{\tau+1} \mid s_\tau, a_\tau\right) r_{s_{\tau+1}, a_\tau} \\
&\quad + \gamma^2 \sum_{s_{\tau+1}} \mathcal{P}\left(s_{\tau+1} \mid s_\tau, a_\tau\right) \sum_{s_{\tau+2}} \mathcal{P}\left(s_{\tau+2} \mid s_{\tau+1}, a_\tau\right) Q^{\pi^i_{\phi_i}}(s_{\tau+2}, a_\tau) \\
&= r_{s_\tau, a_\tau} + \sum_{z=0}^{i-3} \gamma^{z+1} \prod_{z=0}^{i-3} \sum_{s_{\tau+z+1}} \mathcal{P}\left(s_{\tau+z+1} \mid s_{\tau+z}, a_\tau\right) r_{s_{\tau+z+1}, a_\tau} \\
&\quad + \gamma^{i-1} \prod_{z=0}^{i-2} \sum_{s_{\tau+z+1}} \mathcal{P}(s_{\tau+z+1} \mid s_{\tau+z}, a_\tau) Q^{\pi^i_{\phi_i}}\left(s_{\tau+i-1}, a_\tau\right),
\end{aligned} \tag{A.3}$$

where $i \geq 1$. The derivation of the state-action value function, denoted as $\nabla_{\phi_i} Q^{\pi^i_{\phi_i}}\left(s_\tau, a_\tau\right)$, results in Eq. (A.4).

$$\begin{aligned}
\nabla_{\phi_i} Q^{\pi^i_\phi}\left(s_\tau, a_\tau\right) &= \gamma^{i-1} \prod_{z=0}^{i-2} \sum_{s_{\tau+z+1}} \mathcal{P}\left(s_{\tau+z+1} \mid s_{\tau+z}, a_\tau\right) \nabla_{\phi_i} Q^{\pi^i_{\phi_i}}\left(s_{\tau+i-1}, a_\tau\right) \\
&= \gamma^{i-1} \prod_{z=0}^{i-2} \sum_{s_{\tau+z+1}} \mathcal{P}\left(s_{\tau+z+1} \mid s_{\tau+z}, a_\tau\right) \nabla_{\phi_i}\left(\sum_{s_{\tau+i}, r} \mathcal{P}\left(s_{\tau+i} \mid s_{\tau+i-1}, a_\tau\right)\left(r_{s_{\tau+i-1}, a_\tau} + \gamma V^{\pi^i_{\phi_i}}\left(s_{\tau+i}\right)\right)\right) \\
&= \gamma^i \prod_{z=0}^{i-1} \sum_{s_{\tau+z+1}} \mathcal{P}\left(s_{\tau+z+1} \mid s_{\tau+z}, a_\tau\right) \nabla_{\phi_i} V^{\pi^i_{\phi_i}}\left(s_{\tau+i}\right).
\end{aligned} \tag{A.4}$$

The objective of sub-space multi-pruning agents is to maximize the excepted reward of the state value at planning step τ, as delineated in Eq. (A.5).

$$\begin{aligned}
&\nabla_{\phi_i} V^{\pi^i_\phi}\left(s_\tau\right) \\
&= \nabla_{\phi_i}\left(\sum_{a_\tau \in A_r} Q^{\pi^i_{\phi_i}}\left(s_\tau, a_\tau\right) \pi^i_{\phi_i}\left(a_\tau \mid s_\tau\right)\right) \\
&= \sum_{a_\tau \in A_r}\left(Q^{\pi^i_{\phi_i}}\left(s_\tau, a_\tau\right) \nabla_{\phi_i} \pi^i_{\phi_i}\left(a_\tau \mid s_\tau\right) + \pi^i_{\phi_i}\left(a_\tau \mid s_\tau\right) \nabla_{\phi_i} Q^{\pi^i_{\phi_i}}\left(s_\tau, a_\tau\right)\right) \\
&= \sum_{a_\tau \in A_r} Q^{\pi^i_{\phi_i}}\left(s_\tau, a_\tau\right) \nabla_{\phi_i} \pi^i_{\phi_i}\left(a_\tau \mid s_\tau\right) \\
&\qquad + \sum_{a_\tau \in A_r} \pi^i_{\phi_i}\left(a_\tau \mid s_\tau\right) \gamma^i \prod_{z=0}^{i-1} \sum_{s_{\tau+z+1}} \mathcal{P}\left(s_{\tau+z+1} \mid s_{\tau+z}, a_\tau\right) \nabla_{\phi_i} V^{\pi^i_\phi}\left(s_{\tau+i}\right),
\end{aligned} \tag{A.5}$$

where z represents the repeat action time step. Define the function $\xi\left(s_\tau\right) = \sum_{a_\tau \in A_r} Q^{\pi^i_{\phi_i}}\left(s_\tau, a_\tau\right) \nabla_{\phi_i} \pi^i_{\phi_i}(a_\tau \mid s_\tau)$, therefore, we derive Eq. (A.6).

$$\begin{aligned}
\nabla_{\phi_i} V^{\pi^i_{\phi_i}}\left(s_\tau\right) =& \xi(s_\tau) + \sum_{a_\tau \in A_r} \pi^i_{\phi_i}\left(a_\tau \mid s_\tau\right) \gamma^i \prod_{z=0}^{i-1} \sum_{s_{\tau+z+1}} \mathcal{P}\left(s_{\tau+z+1} \mid s_{\tau+z}, a_\tau\right) \nabla_{\phi_i} V^{\pi^i_{\phi_i}}\left(s_{\tau+i}\right) \\
=& \xi(s_\tau) + \gamma^i \prod_{z=0}^{i-1} \sum_{a_\tau \in A} \pi^i_{\phi_i}\left(a_\tau \mid s_\tau\right) \sum_{s_{\tau+z+1}} \mathcal{P}\left(s_{\tau+z+1} \mid s_{\tau+z}, a_\tau\right) \nabla_{\phi_i} V^{\pi^i_{\phi_i}}\left(s_{\tau+i}\right) \\
=& \xi(s_\tau) + \gamma^i \prod_{z=0}^{i-1} \sum_{s_{\tau+z+1}} \mathcal{P}^{\pi^i_{\phi_i}}\left(s_{\tau+z+1} \mid s_{\tau+z}\right) \nabla_{\phi_i} V^{\pi^i_\phi}\left(s_{\tau+i}\right).
\end{aligned} \tag{A.6}$$

As illustrated in Eq. (A.6), the derivative of state value at time step τ can be recursively derived from the next decision time step $\tau + i$. Furthermore, we define the k times discount state transition probability $\gamma^{i*k} \prod_{z=0}^{i*k-1} \sum_{s_{\tau+z+1}} \mathcal{P}^{\pi^i_{\phi_i}}\left(s_{\tau+z+1} \mid s_{\tau+z}, k\right)$ as described in Eq. (A.7).

$$\begin{aligned}
&\gamma^{i*k} \prod_{z=0}^{i*k-1} \sum_{s_{\tau+z+1}} \mathcal{P}^{\pi^i_{\phi_i}}\left(s_{\tau+z+1} \mid s_{\tau+z}, k\right) \\
&:= \gamma^{i*(k-1)} \prod_{z=0}^{i*(k-1)-1} \sum_{s_{\tau+z+1}} \mathcal{P}^{\pi^i_{\phi_i}}\left(s_{\tau+z+1} \mid s_{\tau+z}, k-1\right) \gamma^i \prod_{z=i*(k-1)}^{i*k-1} \sum_{s_{\tau+z}} \mathcal{P}^{\pi^i_{\phi_i}}\left(s_{\tau+z+1} \mid s_{\tau+z}\right).
\end{aligned} \tag{A.7}$$

According to Eqs. (A.6) and (A.7):

$$\begin{aligned}
&\nabla_{\phi_i} V^{\pi^i_\phi}\left(s_\tau\right) \\
&\quad = \xi(s_\tau) + \gamma^i \prod_{z=0}^{i-1} \sum_{s_{\tau+z+1}} \mathcal{P}^{\pi^i_{\phi_i}}\left(s_{\tau+z+1} \mid s_{\tau+z}\right) \nabla_{\phi_i^i} V^{\pi^i_\phi}\left(s_{\tau+i}\right) \\
&\quad = \xi(s_\tau) + \gamma^i \prod_{z=0}^{i-1} \sum_{s_{\tau+z+1}} \mathcal{P}^{\pi^i_{\phi_i}}\left(s_{\tau+z+1} \mid s_{\tau+z}, 1\right) \xi(s_{\tau+i}) \\
&\qquad\quad + \gamma^{i*2} \prod_{z=0}^{i*2-1} \sum_{s_{\tau+z}} \mathcal{P}^{\pi^i_{\phi_i}}\left(s_{\tau+z+1} \mid s_{\tau+z}, 2\right) \nabla_\theta V^{\pi^i_{\phi_i}}\left(s_{\tau+i*2}\right) \\
&\quad = \sum_{s \in S_r} \sum_{k=0}^{\infty} \gamma^{k*i} \prod_{z=0}^{i*k-1} \sum_{s_{\tau+z+1}} \mathcal{P}^{\pi^i_{\phi_i}}\left(s_{\tau+z+1} \mid s_{\tau+z}, k\right) \xi(s) \\
&\quad = \sum_{s \in S_r} d_r^{\pi^i_{\phi_i}}(s) \sum_{a_{\tau+i*k} \in A_r} Q^{\pi^i_{\phi_i}}\left(s_{\tau+i*k}, a_{\tau+i*k}\right) \nabla_{\phi_i} \pi^i_{\phi_i}(a_{\tau+i*k} \mid s_{\tau+i*k}).
\end{aligned} \tag{A.8}$$

The term $d_r^{\pi^i_{\phi_i}}(s) = \sum_{k=0}^{\infty} \gamma^{k*i} \prod_{z=0}^{i*k-1} \sum_{s_{\tau+z+1}} \mathcal{P}^{\pi^i_{\phi_i}}\left(s_{\tau+z+1} \mid s_{\tau+z}, k\right)$ denotes a multi-pruning transition probability function, which is used to describe the likelihood of transitioning from one state to another at different scales of the MDP. This distribution is characterized by summing over all possible states, indexed by k, and within each state, considering the product of the probabilities

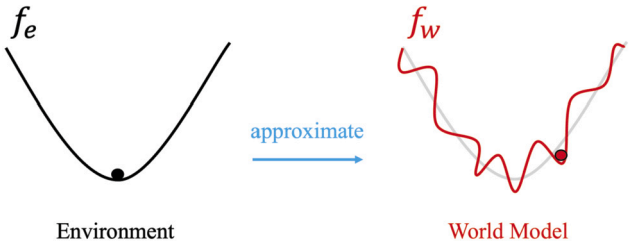

Fig. B.5. An example of model-bias causing local optimality.

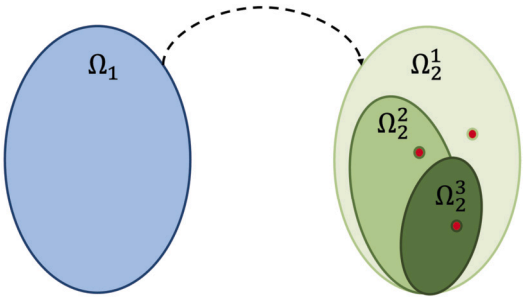

Fig. C.6. Space Mapping from environment model to world model.

of transitioning from one state to the next, as dictated by the policy $\pi^i_{\phi_i}$. Each transition probability $\mathcal{P}^{\pi^i_{\phi_i}}\left(s_{\tau+z+1} \mid s_{\tau+z}, k\right)$ represents the likelihood of moving to state $s_{\tau+z+1}$ from state $s_{\tau+z}$ repeated i times, given the step k in the process. We define $a_r = a_{\tau+i*k} \in A_r$, $s_r = s_{\tau+i*k} \in S_r$. Therefore,

$$\begin{aligned}
\nabla_{\phi_i} V^{\pi^i_{\phi_i}}\left(s_\tau\right) &= \sum_{s_r \in S_r} d_r^{\pi^i_{\phi_i}}(s) \sum_{a_r \in A_r} Q^{\pi^i_{\phi_i}}\left(s_r, a_r\right) \nabla_{\phi_i^i} \pi^i_{\phi_i}(a_r \mid s_r) \\
&\propto \sum_{s_r \in S_r} \mu_{ri}(s) \sum_{a_r \in A_r} \left(\pi^i_{\phi_i}\left(a_r, s_r\right) Q^{\pi^i_{\phi_i}}\left(s_r, a_r\right) \frac{\nabla_{\phi_i} \pi^i_{\phi_i}\left(a_r \mid s_r\right)}{\pi^i_{\phi_i}\left(a_r \mid s_r\right)} \right) \\
&= \mathbb{E}_{s_r \sim \mu_{ri}, a_r \sim \pi^i_{\phi_i}} \left[Q^{\pi^i_{\phi_i}}(s_r, a_r) \nabla_{\phi_i} \ln \pi^i_{\phi_i}\left(a_r \mid s_r\right) \right].
\end{aligned} \tag{A.9}$$

It is noteworthy that the gradient exists solely at decision nodes, effectively resulting in a leapfrogging derivative calculation. □

Appendix B. An example of local optima caused by model bias

Here, we provide an example illustrating how a policy can become trapped in a local optimum due to approximation errors in the world model within model-based reinforcement learning algorithms.

Assuming that the function f_e represents the real environment, it can be observed that f_e is a smooth convex function in Fig. B.5. This makes it relatively easy to find the optimal solution, which is the position of the small black ball at the bottom, when using optimization methods. However, in model-based reinforcement learning, a world model is approximated in order to reduce interactions with the environment. The expectation is that the policy will interact with the world model to find the optimal policy, thus avoiding expensive interactions with the real environment As the world model is approximated from the real environment, there will inevitably be approximation errors. The function learned from the world model is denoted as f_w. However, due to the presence of these approximation errors, f_w has many local optima, making it easy for the policy to become trapped in them and resulting in suboptimal outcomes.

Appendix C. An example to explain MSPP

Here, we present an example of a visual explanation of MSPP in model-based reinforcement learning. Let us consider the real environment space as Ω_1. In model-based reinforcement learning, the problem is translated from Ω_1 to Ω_2 due to the high cost of collecting samples in Ω_1. This is often the case with real equipment, which has a limited lifespan. MSPP further divides the space Ω_2 based on the number of repetitions of actions. Specifically, Ω_2 is split into three subspaces: Ω_2^1, Ω_2^2, and Ω_2^3, as illustrated in Fig. C.6.

Each pruning policy involves searching and optimizing within divided spaces in order to obtain the optimal policy. However, it appears that searching in Ω_2^2 and Ω_2^3 may have a negative impact on global convergence when compared to searching within the

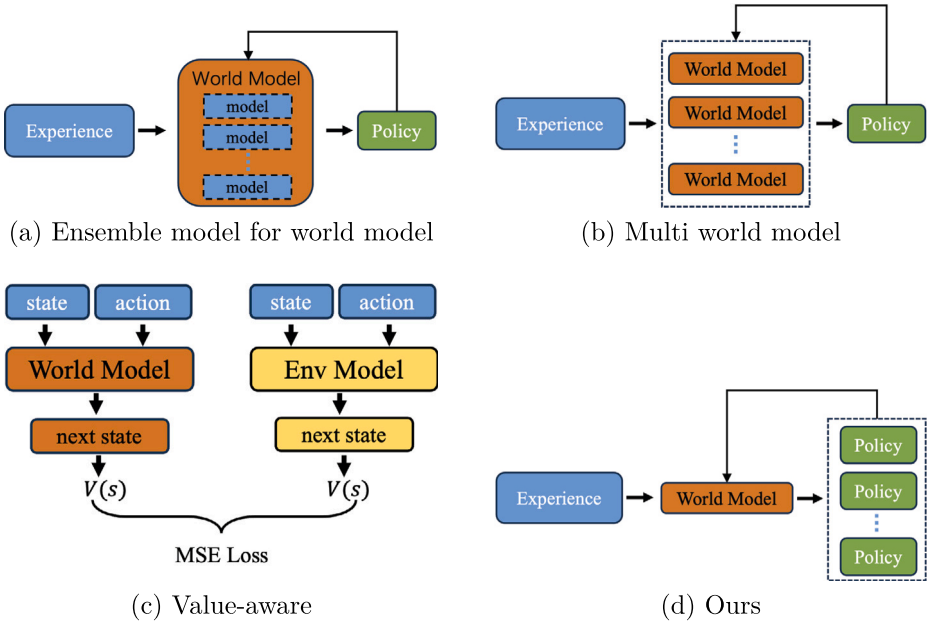

Fig. D.7. The comparison of system framework.

larger space of Ω_2^1. The optimal policy found in Ω_2^1 is likely to be better than those found in Ω_2^2 and Ω_2^3. For this issue, we offer the following explanations.

1. Ω_2^1 is a high-dimensional space, and thus, neural networks are necessary to effectively handle this problem. However, the optimization of neural networks is often hindered by the learning rate, which can easily become trapped in local optima. As a result, when using a fixed learning rate and other hyperparameters, it is not guaranteed that the global optimum can be reached in the Ω_2^1 space through neural network approximations.
2. There may be multiple optimal solutions in the Ω_2^1 space for different tasks. For instance, in Fig. C.6, the red dots represent three optimal solutions. In such cases, it is easier to find the optimal solution in a smaller space, such as Ω_2^3, due to the reduced search space. We also provided an example of autonomous driving in the introduction of our paper to illustrate the benefits of this pruning strategy. In the "4.2. Analyzing Multi-Step Pruning Agents" section of our experimental paper, we observed that different pruning policies with varying repetition times have their own unique insights and advantages in finding the optimal solution for different tasks.
3. Even if Explanation 1 and Explanation 2 do not apply to a specific task, indicating the presence of a global optimum in the Ω_2^1 space rather than the Ω_2^2 and Ω_2^3 spaces, the neural network optimization may still discover this optimal solution. In such cases, this optimal policy will be selected as a top-ranking solution in the subsequent cross-entropy method. Thus, while the pruning policy may have minimal impact on overall convergence for Ω_2^1 space, it offers numerous benefits.

Appendix D. System framework comparison

Here, we will explain the comparison of system frameworks.

In model-based reinforcement learning, it is common to use multiple neural network models to approximate a world model, or to learn multiple world models for different settings [45,46], or to reduce the impact of the world model and the real environment model on the final policy or value function [47]. These approaches are illustrated in Fig. D.7 (a), (b), and (c), respectively. However, we propose a new perspective on this problem: is it necessary to be concerned about minor model bias? Even if a world model has some bias, we can still examine it from multiple policy perspectives, in the hopes that the policy can truly understand the model and derive a better strategy.

References

[1] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[2] F.-M. Luo, T. Xu, H. Lai, X.-H. Chen, W. Zhang, Y. Yu, A survey on model-based reinforcement learning, Sci. China Inf. Sci. 67 (2024) 121101.
[3] R. Wei, N. Lambert, A.D. McDonald, A. Garcia, R. Calandra, A unified view on solving objective mismatch in model-based reinforcement learning, Trans. Mach. Learn. Res. (2024), https://openreview.net/forum?id=tQVZgvXhZb, survey Certification.
[4] M. Mohammadi, A.Z. Kouzani, M. Bodaghi, J. Long, S.Y. Khoo, Y. Xiang, A. Zolfagharian, Sustainable robotic joints 4d printing with variable stiffness using reinforcement learning, Robot. Comput.-Integr. Manuf. 85 (2024) 102636.

[5] H.R. Walke, J.H. Yang, A. Yu, A. Kumar, J. Orbik, A. Singh, S. Levine, Don't start from scratch: leveraging prior data to automate robotic reinforcement learning, in: Conference on Robot Learning, PMLR, 2023, pp. 1652–1662.
[6] M.H. Cohen, C. Belta, Safe exploration in model-based reinforcement learning using control barrier functions, Automatica 147 (2023) 110684.
[7] C. Della Santina, C. Duriez, D. Rus, Model-based control of soft robots: a survey of the state of the art and open challenges, IEEE Control Syst. Mag. 43 (2023) 30–65.
[8] W. Zhao, K. Shi, Z. Liu, X. Wu, X. Zheng, L. Wei, K. Nei, Drl connects Lyapunov in delay and stability optimization for offloading proactive sensing tasks of rsus, IEEE Trans. Mob. Comput. (2023).
[9] N.A. Hansen, H. Su, X. Wang, Temporal difference learning for model predictive control, in: K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, S. Sabato (Eds.), Proceedings of the 39th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 162, PMLR, 2022, pp. 8387–8406, https://proceedings.mlr.press/v162/hansen22a.html.
[10] D. Hafner, J. Pasukonis, J. Ba, T.P. Lillicrap, Mastering diverse domains through world models, CoRR, arXiv:2301.04104 [abs], 2023, https://doi.org/10.48550/arXiv.2301.04104.
[11] D. Hafner, T. Lillicrap, J. Ba, M. Norouzi, Dream to control: learning behaviors by latent imagination, in: International Conference on Learning Representations, 2020, https://openreview.net/forum?id=S1lOTC4tDS.
[12] R.S. Sutton, Integrated architectures for learning, planning, and reacting based on approximating dynamic programming, in: Machine Learning Proceedings 1990, Elsevier, 1990, pp. 216–224.
[13] R.I. Brafman, M. Tennenholtz, R-max-a general polynomial time algorithm for near-optimal reinforcement learning, J. Mach. Learn. Res. 3 (2002) 213–231.
[14] X. Wang, W. Wongkamjan, R. Jia, F. Huang, Live in the moment: learning dynamics model adapted to evolving policy, in: International Conference on Machine Learning, PMLR, 2023, pp. 36470–36493.
[15] A. Nagabandi, G. Kahn, R.S. Fearing, S. Levine, Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2018, pp. 7559–7566.
[16] Y. Luo, H. Xu, Y. Li, Y. Tian, T. Darrell, T. Ma, Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees, in: International Conference on Learning Representations, 2019, https://openreview.net/forum?id=BJe1E2R5KX.
[17] T. Ji, Y. Luo, F. Sun, M. Jing, F. He, W. Huang, When to update your model: constrained model-based reinforcement learning, Adv. Neural Inf. Process. Syst. 35 (2022) 23150–23163.
[18] T. Xu, Z. Li, Y. Yu, Error bounds of imitating policies and environments for reinforcement learning, IEEE Trans. Pattern Anal. Mach. Intell. 44 (2021) 6968–6980.
[19] K. Sun, Y. Zhao, S. Jui, L. Kong, Exploring the training robustness of distributional reinforcement learning against noisy state observations, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2023, pp. 36–51.
[20] H. Zhang, H. Yu, J. Zhao, D. Zhang, H. Zhou, C. Huang, C. Ye, et al., How to fine-tune the model: unified model shift and model bias policy optimization, Adv. Neural Inf. Process. Syst. 36 (2024).
[21] H. Lai, J. Shen, W. Zhang, Y. Yu, Bidirectional model-based policy optimization, in: International Conference on Machine Learning, PMLR, 2020, pp. 5618–5627.
[22] N. Hansen, H. Su, X. Wang, TD-MPC2: scalable, robust world models for continuous control, in: The Twelfth International Conference on Learning Representations, 2024, https://openreview.net/forum?id=Oxh5CstDJU.
[23] R.S. Sutton, D. Precup, S. Singh, Between mdps and semi-mdps: a framework for temporal abstraction in reinforcement learning, Artif. Intell. 112 (1999) 181–211.
[24] S. Pateria, B. Subagdja, A.-h. Tan, C. Quek, Hierarchical reinforcement learning: a comprehensive survey, ACM Comput. Surv. 54 (2021) 1–35.
[25] H.S. Chang, P.J. Fard, S.I. Marcus, M. Shayman, Multitime scale Markov decision processes, IEEE Trans. Autom. Control 48 (2003) 976–987.
[26] T. Bozkus, U. Mitra, Multi-timescale ensemble q-learning for Markov decision process policy optimization, arXiv preprint, arXiv:2402.05476, 2024.
[27] K. Asadi, D. Misra, S. Kim, M.L. Littman, Combating the compounding-error problem with a multi-step model, arXiv preprint, arXiv:1905.13320, 2019.
[28] H. Lin, Y. Sun, J. Zhang, Y. Yu, Model-based reinforcement learning with multi-step plan value estimation, in: Proceedings of the 26th European Conference on Artificial Intelligence (ECAI'23), Kraków, Poland, 2023.
[29] Y. Song, P.N. Suganthan, W. Pedrycz, J. Ou, Y. He, Y. Chen, Y. Wu, Ensemble reinforcement learning: a survey, Appl. Soft Comput. (2023) 110975.
[30] S. Faußer, F. Schwenker, Neural network ensembles in reinforcement learning, Neural Process. Lett. 41 (2015) 55–69.
[31] J. Parker-Holder, A. Pacchiano, K.M. Choromanski, S.J. Roberts, Effective diversity in population based reinforcement learning, Adv. Neural Inf. Process. Syst. 33 (2020) 18050–18062.
[32] S. Wu, J. Yao, H. Fu, Y. Tian, C. Qian, Y. Yang, Q. Fu, Y. Wei, Quality-similar diversity via population based reinforcement learning, in: The Eleventh International Conference on Learning Representations, 2023.
[33] H. Sheikh, M. Phielipp, L. Boloni, Maximizing ensemble diversity in deep reinforcement learning, in: International Conference on Learning Representations, 2022.
[34] E.F. Camacho, C. Bordons, E.F. Camacho, C. Bordons, Constrained Model Predictive Control, Springer, 2007.
[35] A.-M. Farahmand, A. Barreto, D. Nikovski, Value-aware loss function for model-based reinforcement learning, in: A. Singh, J. Zhu (Eds.), Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, in: Proceedings of Machine Learning Research, vol. 54, PMLR, 2017, pp. 1486–1494, https://proceedings.mlr.press/v54/farahmand17a.html.
[36] R. Abachi, Policy-Aware Model Learning for Policy Gradient Methods, Ph.D. thesis, University of Toronto, 2020.
[37] W. Zhang, Z. Yang, J. Shen, M. Liu, Y. Huang, X. Zhang, R. Tang, Z. Li, Learning to build high-fidelity and robust environment models, in: Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part I 21, Springer, 2021, pp. 104–121.
[38] T. Wang, J. Ba, Exploring model-based planning with policy networks, in: International Conference on Learning Representations, 2020, https://openreview.net/forum?id=H1exf64KwH.
[39] Z. Zhang, J. Jin, M. Jagersand, J. Luo, D. Schuurmans, A simple decentralized cross-entropy method, Adv. Neural Inf. Process. Syst. 35 (2022) 36495–36506.
[40] M.T. Spaan, Partially observable Markov decision processes, in: Reinforcement Learning: State-of-the-Art, Springer, 2012, pp. 387–414.
[41] D. Bertsekas, Reinforcement Learning and Optimal Control, vol. 1, Athena Scientific, 2019.
[42] J. Hu, P. Hu, H.S. Chang, A stochastic approximation framework for a class of randomized optimization algorithms, IEEE Trans. Autom. Control 57 (2011) 165–178.
[43] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D.d.L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al., Deepmind control suite, arXiv preprint, arXiv:1801.00690, 2018.
[44] E. Todorov, T. Erez, Y. Tassa, Mujoco: a physics engine for model-based control, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2012, pp. 5026–5033.
[45] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, P. Abbeel, Model-ensemble trust-region policy optimization, in: ICLR (Poster), OpenReview.net, 2018, http://dblp.uni-trier.de/db/conf/iclr/iclr2018.html#KurutachCDTA18.
[46] Y. Yao, L. Xiao, Z. An, W. Zhang, D. Luo, Sample efficient reinforcement learning via model-ensemble exploration and exploitation, in: 2021 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2021, pp. 4202–4208.
[47] C. Voelcker, V. Liao, A. Garg, A. massoud Farahmand, Value gradient weighted model-based reinforcement learning, in: ICLR, OpenReview.net, 2022, http://dblp.uni-trier.de/db/conf/iclr/iclr2022.html#VoelckerLGF22.