Machine Intelligence Research
www.mi-research.net
DOI: 10.1007/s11633-024-1503-7

Guided Proximal Policy Optimization with Structured Action Graph for Complex Decision-making

Yiming Yang[1,2] Dengpeng Xing[1,2] Wannian Xia[1,2] Peng Wang[1,2,3]

[1] Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

[2] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China

[3] Centre for Artificial Intelligence and Robotics, Hong Kong Institute of Science and Innovation, Chinese Academy of Sciences, Hong Kong 999077, China

Abstract: Reinforcement learning encounters formidable challenges when tasked with intricate decision-making scenarios, primarily due to the expansive parameterized action spaces and the vastness of the corresponding policy landscapes. To surmount these difficulties, we devise a practical structured action graph model augmented by guiding policies that integrate trust region constraints. Based on this, we propose guided proximal policy optimization with structured action graph (GPPO-SAG), which has demonstrated pronounced efficacy in refining policy learning and enhancing performance across sophisticated tasks characterized by parameterized action spaces. Rigorous empirical evaluations of our model have been performed on comprehensive gaming platforms, including the entire suite of StarCraft II and Hearthstone, yielding exceptionally favorable outcomes. Our source code is at https://github.com/sachiel321/GPPO-SAG.

Keywords: Reinforcement learning, trust region policy optimization, complex decision-making, policy guiding, structured action graph.

Citation: Y. Yang, D. Xing, W. Xia, P. Wang. Guided proximal policy optimization with structured action graph for complex decision-making. *Machine Intelligence Research*. http://doi.org/10.1007/s11633-024-1503-7

1 Introduction

Recently, the human-like ability of agents to reason and make decisions in complex scenes has been attracted by an increasing number of researchers[1–4]. One potential approach to achieve this goal is deep reinforcement learning (DRL)[5–6], where the policy gradient[7] method plays a crucial role. Compared to the value-based approach, the policy gradient method is more efficient for tasks with large and continuous action spaces. However, the vanilla policy gradient method[8] is less robust, and many advanced algorithms improve the policy gradient method[9–10]. The proximal policy optimization (PPO)[10] method has emerged as one of the most popular RL algorithms, primarily due to its superior and robust performance, which stems from the theoretical foundation provided by trust region theory[9].

Policy learning in parameterized action space is a notorious challenge for RL agents in complex tasks. The real-time strategy (RTS) game[11] is a typical example, where each step requires the agent to select an action type from a discrete action space and then the parameters of that action. These action parameters may be in the continuous action space. For example, if you choose a soldier to move, then the action type is moving, and you need to select a soldier and its moving target location, which is a point in a continuous action space. The combination of the aforementioned factors typically results in a vast space for action. The popular method decomposes the original action space through autoregressive action selection and transforms the complex decision-making problem into several simpler and easier decision-making problems in the appropriate action spaces[11].

The autoregressive action selection method has proven to be a highly effective solution for tackling complex problems with large parameterized action spaces. Its versatility is evident in its application to a range of challenges, including SCC[12], DI-star[13], OpenAI-Five[14], and JueWu AI for Honor of Kings[15]. With the emergence of

Research Article
Manuscript received on November 3, 2023; accepted on March 19, 2024
Recommended by Associate Editor Wei He
Colored figures are available in the online version at https://link.springer.com/journal/11633

offline reinforcement learning, DeepMind has introduced AlphaStar Unplugged[16], which is inherited from the autoregressive action selection framework. As a result, autoregressive action selection has become the standard method for addressing complex problems with large parameterized action spaces.

Although these heuristics have also achieved good results, surprisingly no researcher has conducted exhaustive mathematical modeling and quantitative analysis of guided reinforcement learning via autoregressive action selection. In all existing research on RL based on autoregressive action selection, each decomposed sub-policy is modeled as an independent solution of the policy optimization problem. However, some researchers have shown that simply imposing generic-purpose RL algorithms on a policy consisting of several sub-policies may introduce unwanted bias[17, 18]. Our study is dedicated to bridging this gap. We claim that although the parameterized action space of these problems is large and diverse, it usually has some internal structures. We can simplify the complexity of policy modeling and facilitate the learning of the RL agent if we consider these internal structures. Hence we first formalize the RL with autoregressive action selection as a Markov decision process with structured action graph, give a lower bound on the improvement of policy monotonicity based on guiding policy, and then based on this provide a theoretical guarantee for the optimization of on-policy RL algorithms guided by a priori policy. Finally, based on the above theoretical foundation, we propose the guided proximal policy optimization with structured action graph (GPPO-SAG). Compared to the existing methods[6, 11, 12], GPPO-SAG performs theoretically and practically better in complex decision-making tasks.

GPPO-SAG outperforms state-of-the-art approaches across the board in terms of performance and sample efficiency on the full StarCraft II game and Hearthstone. We summarize our contributions as follows:

1) We introduce guided proximal policy optimization with structured action graph, which effectively guides policy learning and improves performance in complex decision-making tasks with parameterized action space.

2) We formulate autoregressive action selection as the structured action graph, which helps mathematically and quantitatively probe action decomposition and optimization in RL.

3) We propose and prove a monotonic improvement guarantee for stochastic policy with a guiding policy and structured action graph, which provides a fresh viewpoint on RL policy optimization.

2 Related works

Tasks requiring complex decision-making often involve action spaces that are parametrized, which presents challenges for RL agents in both action selection and policy learning.

For the action selection problem, some researchers propose compiling a commonly used parameterized-action space from human data to replace the original action space, and then using HRL or graph to learn the policy schedule[19–20]. However, while this approach simplifies the action space, it also limits the possibility for the agent to discover new parameterized-action combinations. Some studies have attempted to learn parameterized actions in a hierarchical way[21–22]. While this approach can handle hybrid action spaces, it is typically limited to two levels of action selection, which makes it difficult to model more complex action spaces such as the StarCraft II full game. Compared with the above methods, the autoregressive action selection method[23] works better when dealing with parameterized action spaces. It decomposes the original action space into several smaller action spaces and composes these smaller actions autoregressively to choose an action. Many works oriented to complex decision-making problems have verified the effectiveness of this method[11, 14–15]. With the rise of the large language model (LLM), some recent researchers combine LLM with autoregressive action selection to realize complex decision-making and long-time planning. These include solving StarCraft II via LLM[24], and completing complex tasks in open-world environments such as Minecraft using LLM and RL[25, 26]. However, the problem of cooperative learning of different subaction spaces in autoregressive action selection has not received much attention from researchers. We claim that solving this problem will further improve the efficiency of solving complex decision-making problems.

Another significant challenge that reinforcement learning faces when dealing with complex decision-making problems is the exploration and learning problem caused by the enormous policy space. Many researchers have found that if the RL agent is allowed to explore and learn freely in complex problems without constraints, the agent policy will be difficult to improve effectively, and may even collapse[19, 11, 6].

Some researchers construct a simpler abstract policy space by artificially predefining entities, predicates, and causal relations between them in the environment[27–29] to simplify the problem. But it is tedious and time-consuming to construct the abstract policy space when facing complex environments, and the effectiveness of these methods depends on the researcher′s understanding of the environment. In the second kind of approach, many works focus on guiding policy exploration and learning through a priori policy or heuristic rules[11, 30–32]. Nonetheless, these approaches either lack theoretical assurance or have limitations on the form of the guidance rule. For complex decision-making problems, some researchers have employed the method of indirectly constraining policy learning by including a penalty term based on the Kullback-Leibler (KL) divergence between the target policy

and the guiding policy during policy updates[4, 6, 11]. However, our experiments suggest that this method may not effectively constrain the target policy distribution. Therefore, there is currently a lack of a guided policy learning method that is both theoretically guaranteed and truly effective for complex decision problems.

3 Preliminaries and background

Markov decision processes. A Markov decision process (MDP) is defined as a tuple $(\boldsymbol{S}, \mathcal{A}, r, \boldsymbol{P}, \gamma)$. Set $\boldsymbol{S}$ denotes the state space and set $\mathcal{A}$ represents the action space. $r : \boldsymbol{S} \times \mathcal{A} \to \mathbf{R}$ is the reward function. $\boldsymbol{P} : \boldsymbol{S} \times \mathcal{A} \times \boldsymbol{S} \to [0, 1]$ denotes the probability of transitions. $\gamma \in [0, 1)$ is the discount factor. The policy $\pi : \boldsymbol{S} \times \mathcal{A} \to [0, 1]$ and $\pi(\boldsymbol{a}|s)$ denotes the probability of playing $\boldsymbol{a}$ in state s. The goal of the agent policy π is to maximize the cumulative discounted reward $J(\pi) = \mathbf{E}_{\boldsymbol{\tau}\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \boldsymbol{a}_t)\right]$, where $\boldsymbol{\tau} = (s_0, \boldsymbol{a}_0, s_1, \boldsymbol{a}_1, \cdots)$ is the trajectory induced by π. Besides, in the field of reinforcement learning, we commonly refer to the following definitions for the state-action value function Q_π, the value function V_π:

$$Q_\pi(s_t, \boldsymbol{a}_t) = \mathbf{E}_{s_{t+1}, \boldsymbol{a}_{t+1}, \cdots}\left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l})\right],$$

$$V_\pi(s_t) = \mathbf{E}_{\boldsymbol{a}_t, s_{t+1}, \cdots}\left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l})\right]$$

where $\boldsymbol{a}_t \sim \pi(\boldsymbol{a}_t|s_t)$, $s_{t+1} \sim \boldsymbol{P}(s_{t+1}|s_t, \boldsymbol{a}_t)$ for $t \geq 0$.

Distribution difference measurement between policies. Given two policies π_θ and π_ψ on $\boldsymbol{S} \times \mathcal{A}$, we assume that π_θ and π_ψ have full support and smooth densities. For a state $s \in S$, the KL divergence between $\pi_\theta(\cdot|s)$ and $\pi_\psi(\cdot|s)$ is defined as

$$D_{KL}(\pi_\theta(\cdot|s), \pi_\psi(\cdot|s)) = \int_{\boldsymbol{a}\in\mathcal{A}} \pi_\theta(\boldsymbol{a}|s) \log \frac{\pi_\theta(\boldsymbol{a}|s)}{\pi_\psi(\boldsymbol{a}|s)} \mathrm{d}\boldsymbol{a}.$$

Moreover, the total variation distance between $\pi_\theta(\cdot|s)$ and $\pi_\psi(\cdot|s)$ is given by

$$D_{TV}(\pi_\theta(\cdot|s), \pi_\psi(\cdot|s)) = \sup_{\boldsymbol{a}\in\mathcal{A}} |\pi_\theta(\boldsymbol{a}|s) - \pi_\psi(\boldsymbol{a}|s)|.$$

Trust region policy gradients. Since the vanilla policy gradient[7] method suffers from determining the update step, trust region policy optimization (TRPO)[9] is proposed to optimize the policy by solving a constrained optimization problem:

$$\max_\theta \mathbf{E}_{(s_t,a_t)\sim\pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t\right]$$
$$\text{s.t. } \mathbf{E}_{(s_t,a_t)\sim\pi_{\theta_{\text{old}}}}\left[D_{TV}(\pi_{\theta_{\text{old}}}(\cdot|s_t), \pi_\theta(\cdot|s_t))\right] \leq \delta \quad (1)$$

where $\hat{A}_t$ is the estimate of the advantage function $A_\pi(s, \boldsymbol{a}) = Q_\pi(s, \boldsymbol{a}) - V_\pi(s)$ at timestep t. TRPO greatly improves learning stability. Based on that, PPO[10] is proposed to simplify the policy update by optimizing the surrogate objective function:

$$\max_\theta \mathbf{E}_{(s_t,a_t)\sim\pi_{\theta_{\text{old}}}}\left[\min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t, \right.\right.$$
$$\left.\left.\text{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1, \epsilon\right)\hat{A}_t\right)\right] \quad (2)$$

where $\text{clip}(\rho, 1, \epsilon)$ means bounding the value of ρ between $[1 - \epsilon, 1 + \epsilon]$. Furthermore, heterogeneous-agent proximal policy optimization (HAPPO)[18] extends PPO to heterogeneous multi-agent reinforcement learning as follows:

$$\max_\theta \mathbf{E}_{(s_t,a_t)\sim\pi_{\theta_{\text{old}}}}\left[\min\left(\rho^i \times \rho^{1:i-1}\hat{A}_t, \right.\right.$$
$$\left.\left.\text{clip}(\rho^i, 1, \epsilon)\rho^{1:i-1}\hat{A}_t\right)\right] \quad (3)$$

where $\rho^{1:i-1} = \frac{\boldsymbol{\pi}_\theta^{1:i-1}(\boldsymbol{a}_t^{1:i-1}|s_t)}{\boldsymbol{\pi}_{\theta_{\text{old}}}^{1:i-1}(\boldsymbol{a}_t^{1:i-1}|s_t)}$ represents the effect of all agents updated before agent i and $\rho^i = \frac{\pi_{\theta^i}^i(a_t^i|s_t)}{\pi_{\theta_{\text{old}}^i}^i(a_t^i|s_t)}$. For ease of distinction, we use normal font a and π for single action and policy and bold font $\boldsymbol{a}$ and $\boldsymbol{\pi}$ for joint action and joint policy. Note that the order of updates among agents is randomly assigned. The restriction that derives PPO from on-policy to off-policy is the distribution difference between the behavior policy π_b and the target policy π_t. According to the "generalized policy improvement lower bound" theorem[33], off-policy sampling can also be applied to trust region learning if the distribution difference between the behavior policy and the target policy is trivial. Given behavior policy π_b, target policy π_t, and future policy $\hat{\pi}_t$, we refer to it as follows:

$$J(\hat{\pi}_t) - J(\pi_t) \geq \frac{1}{1-\gamma}\mathbf{E}_{(s,\boldsymbol{a})\sim\pi_b}\left[\frac{\hat{\pi}_t(\boldsymbol{a}|s)}{\pi_b(\boldsymbol{a}|s)}\hat{A}_{\pi_t}(s, \boldsymbol{a})\right] -$$
$$C^{\hat{\pi}_t,\pi_t}\mathbf{E}_{s\sim\pi_b}\left[D_{TV}(\hat{\pi}_t(\cdot|s), \pi_b(\cdot|s))\right] \quad (4)$$

where $C = \frac{2\gamma \max_s \left|\mathbf{E}_{\boldsymbol{a}\sim\hat{\pi}_t(\cdot,s)}\hat{A}_{\pi_t}(s,\boldsymbol{a})\right|}{(1-\gamma)^2}$.

4 Modeling and analysis for complex decision-making problems

In this section, we analyze the policy optimization lower bound for complex decision-making problems with parameterized action space and propose a practical RL algorithm. Firstly, we formalize the problem as MDPs with structured action graph in Section 4.1. Then, we follow by a thorough analysis of the policy optimization lower bound with the guiding policy in Section 4.2. Finally, in Section 4.3, we derive a surrogate objective and propose a practical algorithm: guided proximal policy optimization

with structured action graph.

4.1 MDPs with structured action graph

In complex decision-making tasks, the action space $\mathcal{A}$ is often diverse, large and parametric, which makes it difficult for the agent to play actions directly. For instance, in SC2LE[34], the agent must select an action from approximately 10^8 valid actions per timestep. To address this issue, we propose MDPs with structured action graph (MDPs-SAG) to model and decompose the parametric action space. In MDPs-SAG, we decompose the action space $\mathcal{A}$ as a sequence of sub-action sets: $\mathcal{A} = \mathcal{A}^1 \times \cdots \times \mathcal{A}^n$. Hierarchy order is given to the sequence by a directed acyclic graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, whose vertex set $\mathcal{V}$ contains each sub-actions set, an initial vertex v^{init} (the only root in $\mathcal{G}$), and an end vertex v^{end} (the only leaf in $\mathcal{G}$). The edge set $\mathcal{E}$ represents the hierarchical dependencies between these vertices. The parameterized action is defined as the joint action in a path on $\mathcal{G}$ from v^{init} to v^{end}: $\mathscr{P} = \langle v^{\text{init}}, v^k, \cdots, v^i, v^{\text{end}} \rangle$. Thus, if $\text{an}(j)$ is used to represent all the ancestor vertices of j vertices, for a given state $s \in \boldsymbol{S}$ and the joint action of ancestor vertices $\boldsymbol{a}^{\text{an}(j)} \in \mathcal{A}^{\text{an}(j)}$ of vertex $v^j \in \mathcal{V}$, an action a^j can be selected from the sub-policy $\pi^j(a^j|s, \boldsymbol{a}^{\text{an}(j)})$ associated with vertex v^j. Then the joint policy $\boldsymbol{\pi}(\boldsymbol{a}|s) = \prod_{j=k}^{i} \pi^j(a^j|s, \boldsymbol{a}^{\text{an}(j)})$, where $\text{an}(j) \in \mathscr{P}$.

Similar to [35], in MDPs-SAG, we can define the state-action value function of the vertex v^j:

$$Q^{j,\text{an}(j)}(s, \boldsymbol{a}^{j,\text{an}(j)}) = \mathbf{E}_{\boldsymbol{a}^{\text{an}(j)} \sim \boldsymbol{\pi}^{\text{an}(j)}} \left[Q^{j,\text{an}(j)}(s, a^j, \boldsymbol{a}^{\text{an}(j)}) \right] \tag{5}$$

where $Q(s, a^j, \boldsymbol{a}^{\text{an}(j)})$ represents the state-action value of action a^j when taken with the joint action $\boldsymbol{a}^{\text{an}(j)} \in \mathcal{A}^{\text{an}(j)}$ of ancestor vertices $v^{\text{an}(j)} \in \mathcal{V}$. And $Q(s, \boldsymbol{a}^{j,\text{an}(j)})$ denotes the state-action value of the joint action $\boldsymbol{a}^{j,\text{an}(j)}$. Equation (5) denotes the expected accumulative reward when vertex v^j's ancestor vertex actions are fixed to $\boldsymbol{a}^{\text{an}(j)}$ and the current vertex action is a^j. Note that $Q^{\text{init}}(s, a^{\text{init}}) = V(s)$, and $Q^{\text{end},\text{an}(\text{end})}(s, \boldsymbol{a}^{\text{end},\text{an}(\text{end})}) = Q(s, \boldsymbol{a}^{k,\cdots,i})$. Then, the advantage function of vertex v^j is

$$A^j(s, \boldsymbol{a}^{\text{an}(j)}, a^j) = Q^{j,\text{an}(j)}(s, \boldsymbol{a}^{j,\text{an}(j)}) - Q^{\text{an}(j)}(s, \boldsymbol{a}^{\text{an}(j)}). \tag{6}$$

We show the SAG for SC2LE in Fig. 1, where the partitions are adaptive to Alphastar[11]. It is important to note that different action types may have different paths from v^{init} to v^{end}, and joint actions do not have to traverse all vertices on the SAG. For instance, the action "attack units" can be represented by a path: $\langle v^{\text{init}}, v^1, v^2, v^3, v^4, v^5, v^{\text{end}} \rangle$, and the action "attack target position" by a path: $\langle v^{\text{init}}, v^1, v^2, v^3, v^4, v^6, v^{\text{end}} \rangle$.

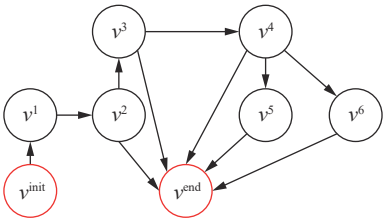


Fig. 1 SAG for SC2LE. v^{init} and v^{end} represent the initial vertex and the end vertex, respectively. $v^1, \cdots, v^6$ represent "action type", "delay", "queue", "select units", "target units", and "target location" respectively, in the partition of AlphaStar[11]. (Colored figures are available in the online version at https://link.springer.com/journal/11633)

Lemma 1. Given a structured action graph $\mathcal{G}$, the following equation holds if all sub-actions form a path graph $\mathscr{P} = \langle v^{\text{init}}, v^k, \cdots, v^i \rangle \in \mathcal{G}$:

$$A^{\mathscr{P}}(s, \boldsymbol{a}^{\mathscr{P}}) = \sum_{j=k}^{i} A^j(s, \boldsymbol{a}^{j,\text{an}(j)}), \text{ where } \text{an}(j) \sim \mathscr{P}. \tag{7}$$

For proof, see Appendix A.1. Lemma 1 shows that in MDPs-SAG, the state-action advantage of the path containing the initial vertex can be decomposed by the sum of the state-action advantages of each vertex in the path. Although Lemma 1 shares a decomposition similar to "multi-agent advantage decomposition"[18, 36], their results do not hold in our problem. In fact, each vertex in Lemma 1 is constrained by graph $\mathcal{G}$, which means that there is no practical meaning for a single vertex when some ancestors are missing, while the arbitrariness of vertex selection is required in [18, 36].

4.2 Policy improvement with guiding policy and SAG

In strategy games, we usually expect the agent to learn human-like policies to discover new tactics or assist in decision-making. Besides, it is difficult for an agent to learn an optimal policy through random exploration in complex tasks due to the high dimensionality of states and actions. Therefore, it is necessary to introduce a guiding policy to guide exploration and constrain the learning direction.

Based on the above, we propose Lemma 2, which introduces the guiding policy into the trust region policy gradient.

Lemma 2. Given behavior policy $\boldsymbol{\pi}_b$, guiding policy $\boldsymbol{\pi}_g$, target policy $\boldsymbol{\pi}_t$, and future policy $\hat{\boldsymbol{\pi}}_t$, the following policy improvement lower bound holds:

$$J(\hat{\boldsymbol{\pi}}_t) - J(\boldsymbol{\pi}_t) \geq C_1 \times \mathbf{E}_{\boldsymbol{\tau} \sim \boldsymbol{\pi}_b} \left[\frac{\hat{\boldsymbol{\pi}}_t(\boldsymbol{a}^{\mathscr{P}}|s)}{\boldsymbol{\pi}_b(\boldsymbol{a}^{\mathscr{P}}|s)} \hat{A}^{\mathscr{P}}_{\boldsymbol{\pi}_t}(s, \boldsymbol{a}^{\mathscr{P}}) \right] - C_2 \times \mathbf{E}_{s \sim \pi_b} [D_{TV}(\hat{\boldsymbol{\pi}}_t(\cdot|s), \boldsymbol{\pi}_g(\cdot|s)) + D_{TV}(\boldsymbol{\pi}_b(\cdot|s), \boldsymbol{\pi}_g(\cdot|s))] \tag{8}$$

where $C_1 = \frac{1}{1-\gamma}$, $C_2 = \frac{2\gamma \max_s \left|\mathbf{E}_{\boldsymbol{a}^{\mathscr{P}} \sim \hat{\boldsymbol{\pi}}_t(\cdot,s)} \hat{A}^{\mathscr{P}}_{\boldsymbol{\pi}_t}(s,\boldsymbol{a}^{\mathscr{P}})\right|}{(1-\gamma)^2}$.

For proof, see Appendix A.2. In our setting, the behavior policy π_b interacts with the environment and collects data. The guiding policy π_g is a prior policy that makes decisions concurrently with the π_b, which provides guidance during training, but it does not interact with the environment. The target policy π_t is the policy being updated during training, and the future policy $\hat{\pi}_t$ is an estimate of the future target policy updated after the current training step. Lemma 2 demonstrates a way to update the policy within the trust region indirectly. This is achieved by placing limits on the total variation distance between the behavioral policy and the guiding policy, as well as between the target policy and the guiding policy. These constraints ensure that the updated policy stays within the trust region. Combining with Lemma 1, we propose a lower bound with SAG on the improvement of the guiding policy.

Theorem 1. Assume S and $\mathcal{A}^{\mathscr{P}}$ are both finite. Given a structured action graph $\mathcal{G}$, behavior policy $\boldsymbol{\pi}_b$, target policy $\boldsymbol{\pi}_t$, guiding policy $\boldsymbol{\pi}_g$, and future policy $\hat{\boldsymbol{\pi}}_t$, the following policy improvement lower bound holds if all sub-actions form a path graph $\mathscr{P} = \langle \mathfrak{v}^{\text{init}}, \mathfrak{v}^k, \cdots, \mathfrak{v}^i, \mathfrak{v}^{\text{end}} \rangle \in \mathcal{G}$:

$$J(\hat{\boldsymbol{\pi}}_t) - J(\boldsymbol{\pi}_t) \geq \sum_{j=k}^{i} \Bigg[\frac{1}{1-\gamma} \mathbf{E}_{\boldsymbol{\tau} \sim \pi_b} \left[\rho^{j,\text{an}(j)} \hat{A}^{\mathscr{P}}_{\pi_t}(s, \boldsymbol{a}^{\mathscr{P}}) \right] - C \times \mathbf{E}_{s \sim \pi_b} \left[D_{TV}(\hat{\pi}^j_t(\cdot|s), \pi^j_g(\cdot|s)) + D_{TV}(\pi^j_b(\cdot|s), \pi^j_g(\cdot|s)) \right] \Bigg] \tag{9}$$

where $C = \frac{2\gamma \max_s \left|\mathbf{E}_{a^j \sim \hat{\pi}^j_t(\cdot,s), \boldsymbol{a}^{\text{an}(j)} \sim \hat{\boldsymbol{\pi}}^{\text{an}(j)}_t(\cdot,s)} \hat{A}^j_{\pi_t}(s, \boldsymbol{a}^{j,\text{an}(j)})\right|}{(1-\gamma)^2}$, and $\rho^{j,\text{an}(j)} = \frac{\hat{\pi}^j_t(a^j|s,\boldsymbol{a}^{\text{an}(j)})}{\pi^j_b(a^j|s,\boldsymbol{a}^{\text{an}(j)})} \times \frac{\hat{\boldsymbol{\pi}}^{\text{an}(j)}_t(\boldsymbol{a}^{\text{an}(j)}|s)}{\boldsymbol{\pi}^{\text{an}(j)}_b(\boldsymbol{a}^{\text{an}(j)}|s)}$.

For proof, see Appendix A.3. Theorem 1 implies the following characteristics of trust region learning with SAG:

1) The target policy $\boldsymbol{\pi}_t$'s promotion is dependent on the sub-policies of each vertex π^j_t on the path graph $\mathscr{P}$ and independent of any other vertex outside $\mathscr{P}$.

2) The advantage function of each sub-policy π^j_t is based on all of its ancestor vertices $\boldsymbol{\pi}^{\text{an}(j)}$ in addition to itself.

3) With the guiding policy $\boldsymbol{\pi}_g$ as the medium, (9) implies that we can bound the distribution distance between the target policy $\boldsymbol{\pi}_t$ and the behavior policy $\boldsymbol{\pi}_b$.

4.3 Guided proximal policy optimization with SAG

Theorem 1 implies how to learn the joint policy based on SAG, maximizing the advantage for each vertex on the path while constraining the future policy distribution distance from the guiding policy. Concretely, according to Theorem 1, we use the lower bound in (9) as the surrogate objective function to be objective. For our proposed surrogate objective function to be solvable by commonly used optimizers, we need a clipped version similar to (2) and (3) for the distributionally constrained problem. However, it is still very challenging for our problem. First, the successive multiplication of importance sampling results in variance accumulation, reducing the stability of training. Second, we not only need to clip between the current target policy estimation and the guiding policy but also need to control the influence of the ancestor policies, similar to the current policy. Thus, we propose a double-clip version as the surrogate objective function.

Let $\rho^{\text{an}(j)}_t = \frac{\hat{\boldsymbol{\pi}}^{\text{an}(j)}_t(\boldsymbol{a}^{\text{an}(j)}|s)}{\boldsymbol{\pi}^{\text{an}(j)}_b(\boldsymbol{a}^{\text{an}(j)}|s)}$, $\rho^{\text{an}(j)}_g = \frac{\boldsymbol{\pi}^{\text{an}(j)}_g(\boldsymbol{a}^{\text{an}(j)}|s)}{\boldsymbol{\pi}^{\text{an}(j)}_b(\boldsymbol{a}^{\text{an}(j)}|s)}$, $\rho^j_t = \frac{\hat{\pi}^j_t(a^j|s,\boldsymbol{a}^{\text{an}(j)})}{\pi^j_b(a^j|s,\boldsymbol{a}^{\text{an}(j)})}$, and $\rho^j_g = \frac{\pi^j_g(a^j|s,\boldsymbol{a}^{\text{an}(j)})}{\pi^j_b(a^j|s,\boldsymbol{a}^{\text{an}(j)})}$, then the surrogate objective function to optimize is as follows:

$$\sum_{j=k}^{i} \min \Bigg[\text{clip}(\rho^{\text{an}(j)}_t, \rho^{\text{an}(j)}_g, \kappa\epsilon) \rho^j_t \hat{A}^{\mathscr{P}}_{\boldsymbol{\pi}_t}(s, \boldsymbol{a}^{\mathscr{P}}), \; \text{clip}\big(\text{clip}(\rho^{\text{an}(j)}_t, \rho^{\text{an}(j)}_g, \kappa\epsilon) \rho^j_t, \rho^j_g, \epsilon\big) \hat{A}^{\mathscr{P}}_{\boldsymbol{\pi}_t}(s, \boldsymbol{a}^{\mathscr{P}}) \Bigg] \tag{10}$$

where $\text{clip}(\rho_1, \rho_2, \epsilon)$ puts ρ_1 into the interval $[\rho_2 - \epsilon, \rho_2 + \epsilon]$ and $\kappa \in [0, +\infty)$ is a constant scaling factor. For the derivation, see Appendix A.4. Since the successive multiplication of importance sampling will increase the variance of advantage estimates[36–37], we need the function $\text{clip}(\rho^{\text{an}(j)}, \rho^{\text{an}(j)}_g, \kappa\epsilon)$ to bound the ancestral importance sampling ratio to the corresponding vicinity of the guiding ratio. The outer clipping function constrains the joint importance sampling ratio to reduce the variance caused by the ancestor vertices. It approximately limits the total variation distance between the guiding policy $\boldsymbol{\pi}_g$ and the target policy estimates $\hat{\boldsymbol{\pi}}_t$. The behavior policy's parameters come from the current target policy periodically. So limiting the total variation distance is equivalent to limiting the distance between the behavior policy and the guiding policy. We refer to (10) as guided proximal policy optimization with structured action graph and elaborate on it in Algorithm 1.

Guiding policy. Usually, we obtain the guiding policy through supervised learning or offline reinforcement learning via pre-collected data. It is important to note that the algorithm imposes certain standards on the quality of the guiding policy. If the quality of the guiding policy is too low, it is difficult to provide effective guidance for the advancement of the target policy. To illustrate this issue, we conduct ablation experiments, which are detailed in Appendix B.1. In cases where there is a shortage of prior guiding policy, our method still operates by reducing to "proximal policy optimization with structured action graph." In such cases, we can assume

that the guiding policy is the behavior policy itself, i.e., $\rho_g^{\mathrm{an}(j)}=1, \rho_g^j=1$ in (10), and the algorithm can run without any prior guidance.

5 Experiments

In this section, we design experiments to explore the following questions:

1) Does GPPO-SAG have an advantage over other methods for complex decision-making tasks? Where does the GPPO-SAG advantage come from?

2) Does the size of the action space affect the algorithm advantages?

3) How sensitive is GPPO-SAG with respect to the clipping ratio? How does the constant scaling factor κ affect the performance of the algorithm?

Algorithm 1. Guided proximal policy optimization with structured action graph

Input: The structured action graph $\mathcal{G}$ with n vertices, batch size B, episodes K, steps per episode T, the guiding policy $\boldsymbol{\pi}_{\theta_g}=(\pi_{\theta_g^1}^1,\cdots,\pi_{\theta_g^n}^n)$

1) **Initialize** the behavior policy $\boldsymbol{\pi}_{\theta_b}=(\pi_{\theta_b^1}^1,\cdots,\pi_{\theta_b^n}^n)$, the target policy $\boldsymbol{\pi}_{\theta_t}=(\pi_{\theta_t^1}^1,\cdots,\pi_{\theta_t^n}^n)$, critic network V_ψ, and replay buffer $\mathcal{B}$
2) **for** $k=0$ to $K-1$ **do**
3) Synchronize $\boldsymbol{\pi}_b$'s parameters with policy pool
4) **for** each environment step **do**
5) Collect transitions $(s,\boldsymbol{a},s',r)$ with behavior policy $\boldsymbol{\pi}_b$ and push them into $\mathcal{B}$
6) **end for**
7) **for** each update step **do**
8) Sample a minibatch of B transitions from $\mathcal{B}$
9) Compute advantage $\hat{A}_{\boldsymbol{\pi}_t}^{\mathcal{P}}(s,\boldsymbol{a})$ and reward to go $\hat{R}_t$
10) Set vertex v^i be v^{init} and query path graph $\mathcal{P}$ from $\mathcal{G}$ for each action
11) **while** vertex v^i has child vertex v^{i+1} in $\mathcal{P}$ **do**
12) $v^i=v^{i+1}$
13) Update sub-policy π_t^i by maximizing $\frac{1}{BT}\sum_{b=1}^{B}\sum_{t=1}^{T}\min[\mathrm{clip}(\rho^{\mathrm{an}(j)},\rho_g^{\mathrm{an}(j)},\kappa\epsilon)\rho^j\hat{A}_{\pi_t}^{\mathcal{P}}(s,\boldsymbol{a}^{\mathcal{P}}),\mathrm{clip}\big(\mathrm{clip}(\rho^{\mathrm{an}(j)},\rho_g^{\mathrm{an}(j)},\kappa\epsilon)\rho^j,\rho_g^j,\epsilon\big)\hat{A}_{\pi_t}^{\mathcal{P}}(s,\boldsymbol{a}^{\mathcal{P}})]$
14) **end while**
15) Update centralized critic V_ψ by minimizing
16) $\frac{1}{BT}\sum_{b=1}^{B}\sum_{t=1}^{T}\big(V_\psi(s_t)-\hat{R}_t\big)^2$
17) **end for**
18) Update policy pool with $\boldsymbol{\pi}_{\theta_t}$
19) **end for**

To answer these questions, we compare the performance of GPPO-SAG with AlphaStar (we use DI-star[13], an open-source implementation of AlphaStar, as our code base), and PPO[10] with additional KL penalty between the target policy and the guiding policy in the StarCraft II full game and Hearthstone. All methods in our experiments are implemented based on importance weighted actor-learner architecture (IMPALA)[38]. With batch size B and episode steps T, the main objective function of AlphaStar is the penalized policy gradient as follows:

$$\frac{1}{BT}\left(\sum_{j=k}^{i}\log\pi_t^j\times\hat{A}_{\boldsymbol{\pi}_t}^{\mathcal{P}}(s,\boldsymbol{a}^{\mathcal{P}})-D_{KL}(\pi_g^j(\cdot|s),\pi_t^j(\cdot|s))\right) \tag{11}$$

and the surrogate objective function of PPO with additional KL penalty is

$$\frac{1}{BT}\left(\sum_{j=k}^{i}\min\left[\frac{\pi_t^j}{\pi_b^j}\hat{A}_{\boldsymbol{\pi}_t}^{\mathcal{P}}(s,\boldsymbol{a}^{\mathcal{P}}),\mathrm{clip}\left(\frac{\pi_t^j}{\pi_b^j},1,\epsilon\right)\hat{A}_{\boldsymbol{\pi}_t}^{\mathcal{P}}(s,\boldsymbol{a}^{\mathcal{P}})\right]-D_{KL}(\pi_g^j(\cdot|s),\pi_t^j(\cdot|s))\right). \tag{12}$$

For the experiments, we use 4 NVIDIA-A100 GPUs and 256 AMD EPYC 7 742 CPU cores.

5.1 Environment description and model configuration

StarCraft II. We only consider the Zerg civil war in our experiments to save computer resources. We first use 9 000 zerg-vs-zerg human replays to train a policy network with 71 000 epochs using a supervised learning paradigm. Then the supervised model is used for initializing the agent's policy model for RL with the level-3 built-in bot. During RL training, the first 4 000 steps are used for value pre-training, after which the actor networks' gradients are enabled. The clip ratio for each head is listed in Table 1. In the following experiments, unless otherwise noted, the default values are used. More hyperparameters for supervised learning are listed in Appendix B.2.

Table 1 Clip ratio with GPPO-SAG

Action head	ϵ_{def}	κ_{def}
Action type	0.4	0.5
Delay	0.6	0.5
Queued	0.6	0.5
Target unit	0.6	0.5
Selected units	1.2	0.5
Target location	0.8	0.5

For a fair comparison, all methods adopt AlphaStar's[11] network structure, UPGO loss, and entropy loss. We set the learning rate to 10^{-5}, the batch size to 8, and the trajectory length for the long short-term memory (LSTM) to 32. More hyperparameters for RL are listed in Appendix B.3. Except for the hyperparameters specific to

our method, all other hyperparameters in StarCraft II are consistent with the settings in DI-star[13].

Hearthstone. Hearthstone is a trading card game that allows players to build a deck of 30 cards before a match and defeat their opponents in two-player battles. Hearthstone's cards are more diverse than traditional board games, and almost every card has its unique effect. The RL environment for Hearthstone used in our experiments is inherited from Cardsformer by Xia et al.[39]

We represent the Hearthstone action space as a structured action graph, as shown in Fig. 2. In our experiments, we use one warlock deck and optimize the policy by self-play. We first use 5 000 replays to train the guiding policy and adopt an autoregressive policy network structure. We set the learning rate to 10^{-5}, the batch size to 32, and the clipping ratio to 1.2. The details about the environment, network structure, and hyperparameters can be found in Appendix B.4. Except for the hyperparameters specific to our method, all other hyperparameters in Hearthstone are consistent with the settings in Cardsformer[39].

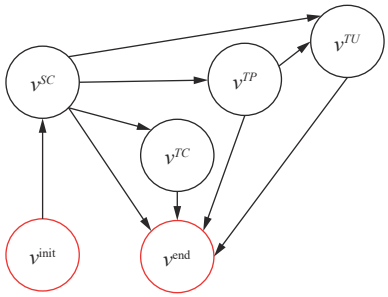


Fig. 2 The structured action graph of Hearthstone. v^{init} means the initial vertex, v^{SC} means selecting card vertex, v^{TC} means selecting a target card vertex, v^{TP} means selecting a target position vertex, v^{TU} means selecting a target unit vertex, and v^{end} means the end vertex. (Colored figure is available in the online version at https://link.springer.com/journal/11633)

5.2 Experiments for question 1

Fig. 3 shows the training curves of GPPO-SAG, AlphaStar, and PPO-KL against the built-in level 3 AI on three standard matchmaking maps, where the red curve represents GPPO-SAG, the green curve represents the AlphaStar with constrained policy gradients, and the blue curve represents PPO algorithm with a KL constraint.

During the early training steps, the win rate of each method steadily increases, but at approximately 2×10^6 game steps, AlphaStar and PPO-KL hit a bottleneck, and their performance stops improving. Our method is unaffected, and its win rate continues to rise steadily, eventually reaching about 90% at around 7×10^6 game steps. Compared with AlphaStar, the final performance of GPPO-SAG after 10 000 games against the built-in level 3 AI is about 30% higher than AlphaStar on average. PPO with a KL constraint is slightly less effective than AlphaStar.

We believe that GPPO-SAG achieves better results than AlphaStar and PPO-KL because GPPO-SAG can better constrain the target policy near the guiding policy, thus improving exploration efficiency. Therefore, we recorded the KL divergence between the target policy $\boldsymbol{\pi}_t$ and the guiding policy $\boldsymbol{\pi}_g$ during training on the KairosJunction map. The results shown in Fig. 4 demonstrate that our method can constrain the distribution distance between the target policy $\boldsymbol{\pi}_t$ and the guiding policy $\boldsymbol{\pi}_g$ to be stable within a certain range in almost all action heads. In contrast, the distribution distance between $\boldsymbol{\pi}_t$ and $\boldsymbol{\pi}_g$ is not stably bounded and grows as training progresses.

Our method constrains the distribution of vertices for each sub-action equally. However, the constraints applied by other methods on the distribution of each sub-action node are positively related to the depth of the corresponding sub-action vertex in the path graph $\mathscr{P}$. As shown in Figs. 4(a)–4(c), these vertices are near v^{init} in the path graph. Among these vertices, our method's $D_{KL}(\boldsymbol{\pi}_t, \boldsymbol{\pi}_g)$ is about 3–4 times smaller than other methods after convergence. But as shown in Figs. 4(d)–4(f), the gap between AlphaStar, PPO-KL, and our method shrinks to 1–3 times. We claim that this is because these vertices are usually located at the end of the path graph and are constrained more tightly in other methods. This

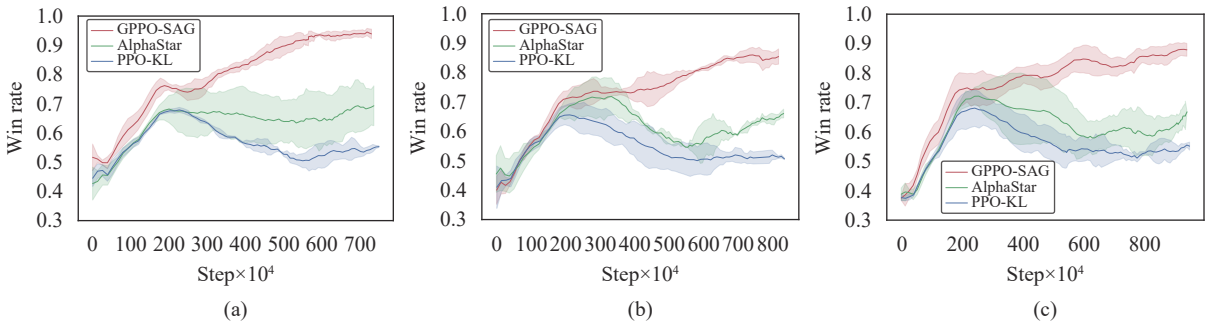


Fig. 3 The win rate curve for each method against level 3 AI on StarCraft II full game. (a) Win rate curve on map KairosJunction; (b) Win rate curve on map KingsCove; (c) Win rate curve on map NewRepugnancy. (Colored figures are available in the online version at https://link.springer.com/journal/11633)

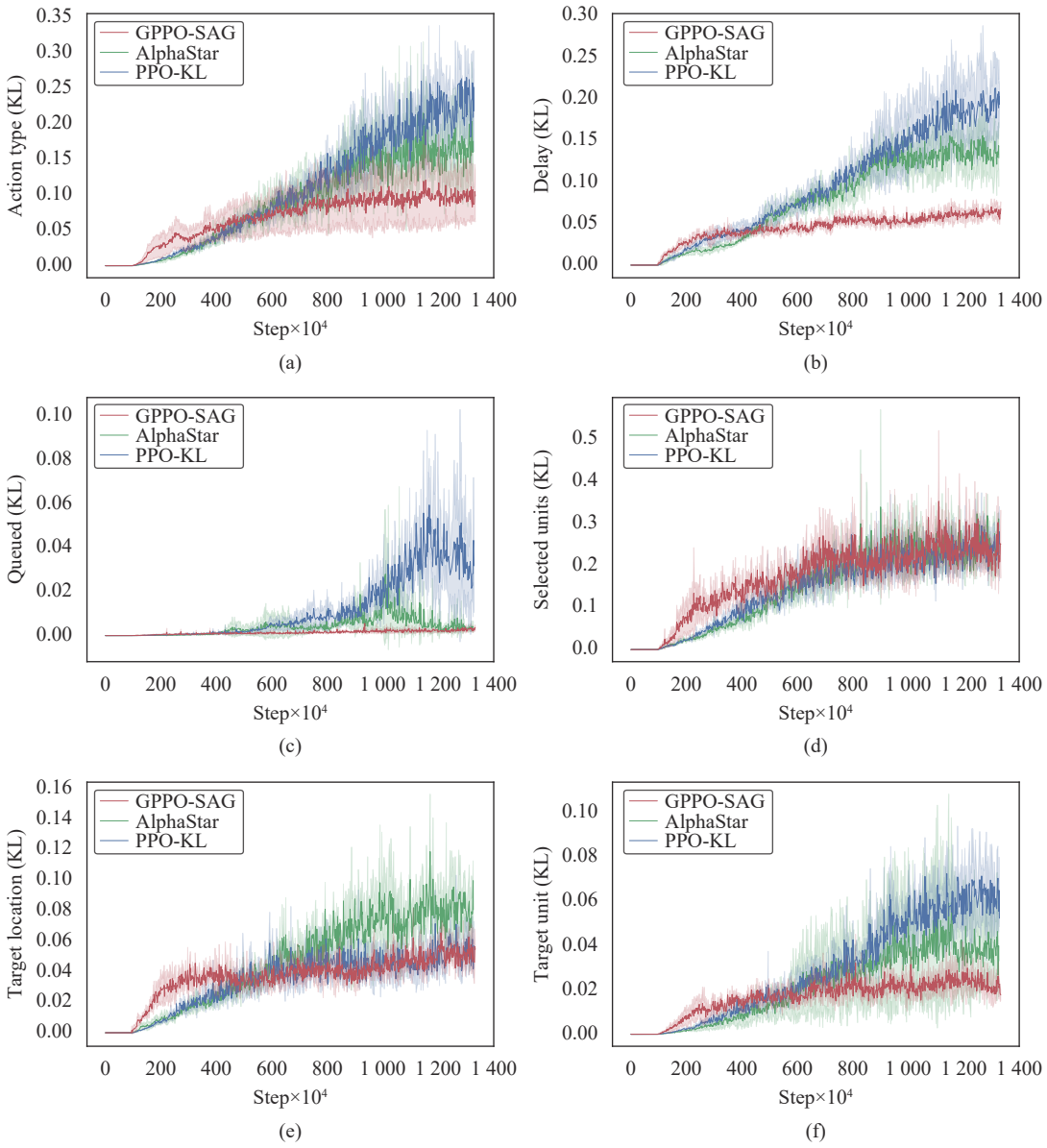


Fig. 4 The KL divergence curve between the target policy $\boldsymbol{\pi}_t$ and the guiding policy $\boldsymbol{\pi}_g$ during training for each method on the StarCraft II's KairosJunction map. (a) Action type vertex; (b) Delay vertex; (c) Queued vertex; (d) Selected units vertex; (e) Target location vertex; (f) Target unit vertex. (Colored figures are available in the online version at https://link.springer.com/journal/11633)

phenomenon indicates that the "uniformity" of constraint strength ensures that our approach can better balance the optimization progress among the sub-action vertices, resulting in more stable performance improvement during training. On the other hand, both PPO-KL and AlphaStar are unable to effectively constrain the distribution distance between $\boldsymbol{\pi}_t$ and $\boldsymbol{\pi}_g$. This means $\boldsymbol{\pi}_g$ cannot effectively constrain exploration during rollout. The policy space for StarCraft II is very large, and if it is not properly constrained during exploration, the agent will have difficulty getting high-quality training data, which will hinder performance improvement during policy optimization.

With 64 environments collecting data in parallel, each algorithm in StarCraft II requires approximately seven days to complete 8×10^7 training steps on a pair of NVIDIA A100 GPUs, with one allocated for training and the other for inference and data collection. The temporal divergence between our proposed algorithm and the comparative benchmarks is negligible. Such a minimal discrepancy is attributed to the reduced efficiency of the agent's interactions and data acquisition from the environment. While the network update latency in our method is marginally higher than that of the baseline algorithm – owing to the necessity of navigating through sub-action vertices in a graph-like structure during training – the increment is substantially less than the latency associated with data gathering. Notably, our training

paradigm ensures synchronization between data collection and training phases, rendering the inference time a decisive factor in the overall training timeframe.

The distribution between the target policy π_t and the guiding policy π_g is not constrained by our method before they activate the clipping mechanism, which allows GPPO-SAG to achieve more accessible exploration in the vicinity of the guiding policy space. As a result, combining Figs. 2 and 4, we can see that during the early training steps, the agent's win rate of GPPO-SAG rises rapidly. Contrarily, the agent policy exploration is permanently constrained by the KL constraint in AlphaStar and PPO-KL, which causes a slower policy improvement in the early phase.

In summary, our proposed GPPO-SAG can more effectively constrain the target policy in the flow space near the bootstrap policy, providing a more consistent policy improvement for bright body learning in complex decision spaces.

5.3 Experiments for question 2

We discover through the experiments in Section 5.2 that one advantage of our approach is that it can handle the optimization progress between different sub-action vertices in a more balanced manner. As a result, we consider testing how well our method works in environments with simpler action spaces. Compared to the structured action graph in StarCraft II (Fig. 1), Hearthstone's (Fig. 3) longest path graph only contains three vertices from the start node to the end node, while the longest path graph in StarCraft II has five vertices. To this end, we used an autoregressive model to model the Hearthstone action space and migrated our approach to Hearthstone.

In the Hearthstone environment, all methods are trained through self-play. We save model checkpoints periodically, add all checkpoints to the battle pool after training, randomly draw two players, and record the Elo scores for each model based on wins and losses. Finally, 50 000 games are played to provide the Elo score curve for each method with the training process (Fig. 5(a)). Figs. 5(b)–5(d) represent different sub-action vertex's KL divergence between the target policy π_t and the guiding policy π_g during training. Three random seeds are used to average each of the curves displayed. We replace the Starcraft II comparison method AlphaStar with PG-KL because we skip the UPGO loss that AlphaStar employed in the Starcraft II trial.

Our method has a faster convergence speed than the others, as demonstrated in Fig. 5, although all methods produce similar results in the end. Since Hearthstone is considerably simpler than StarCraft II, agents without the constraints of a guiding policy can also quickly find

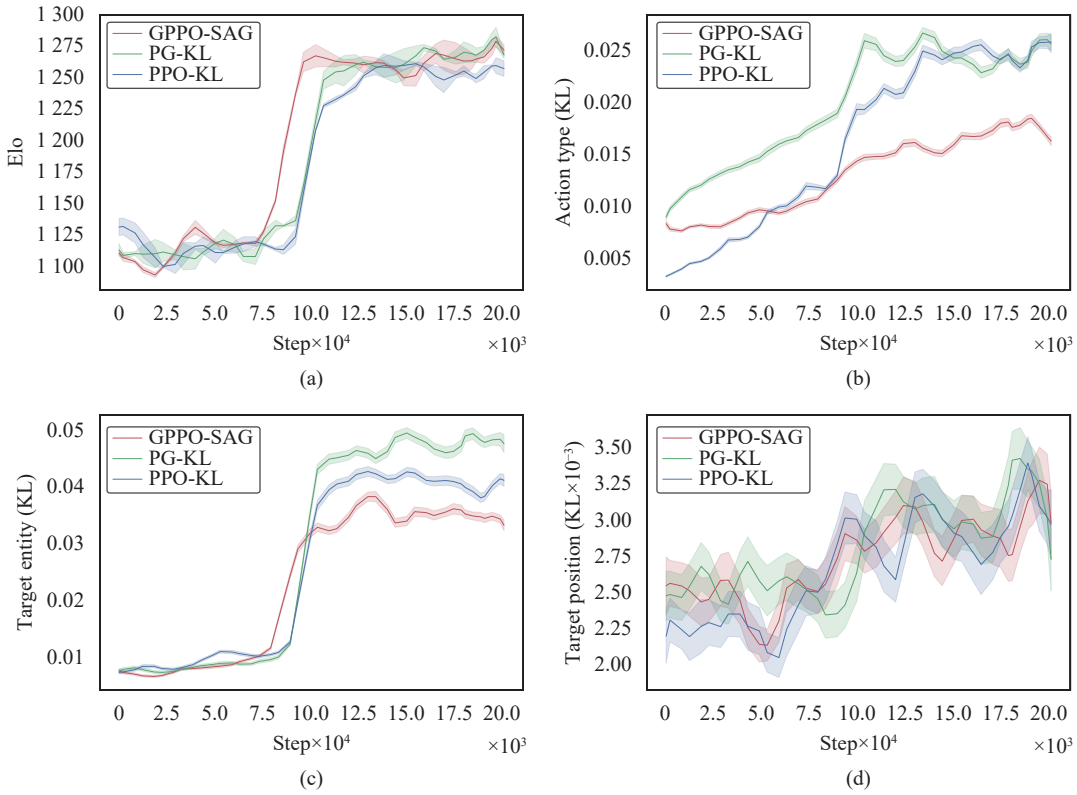


Fig. 5 The training and KL divergence curve between the target policy π_t and the guiding policy π_g in Hearthstone. (a) The Elo rating scores for different training stages in Hearthstone; (b) Action type vertex; (c) Target entity vertex; (d) Target position vertex. (Colored figures are available in the online version at https://link.springer.com/journal/11633)

optimization directions in the policy space. Therefore, all methods have shown good performance. The KL divergence of each sub-action vertex at different training stages shows that our method can constrain the exploration range of the target policy more effectively, even in the relatively straightforward Hearthstone. Compared with StarCraft II, Hearthstone has a much smaller policy space. Therefore, even if the guiding policy constraints of PPO-KL and PG-KL are not effective, they can still explore effectively and achieve relatively robust policy improvements. So their performance on Hearthstone does not drop later in training.

5.4 Experiments for question 3

We demonstrate the effectiveness of the clipping rate on the performance of GPPO-SAG through experiments on the full game of StarCraft II, and the results are shown in Table 2, where ϵ and κ represent the clipping ratio and constant scaling factor taken from Table 1, respectively. We can conclude that both too-small and too-large clipping rates affect learning performance. This is due to the fact that an excessively low clipping rate severely constrains the policy space and acts as a barrier to further policy improvement, while an excessively high clipping rate hinders the ability of the guiding policy to effectively guide the policy exploration during the optimization step and lowers learning efficiency.

Table 2 Different clip ratio in GPPO-SAG

Clip ratio	Win rate
$0.25\epsilon_{\text{def}}$	$59.2\% \pm 1.5\%$
$0.5\epsilon_{\text{def}}$	$68.4\% \pm 2.1\%$
ϵ_{def}	$84.2\% \pm 3.1\%$
$2\epsilon_{\text{def}}$	$83.1\% \pm 1.5\%$
$4\epsilon_{\text{def}}$	$71.0\% \pm 3.9\%$

Unlike vanilla PPO, GPPO-SAG no longer limits the step of each update compared with the previous one by using a clipping rate. Still, it limits the distance between the target policy and the guiding policy throughout the training process. Therefore, the clipping rate in GPPO-SAG is relatively higher. Combining the experimental results in Table 2, we believe that a steady and effective result can be achieved with a clipping rate of 0.4–1.2.

In our methodology, the hyperparameter κ is introduced to modulate the influence of ancestral sub-policy constraints within the trust region upon the current sub-policy updates. A diminutive κ value corresponds to a more restrictive trust region, thereby diminishing the ancestral sub-policies impact on the current sub-policy update, and conversely, a larger κ value enlarges the trust region, increasing this influence. To rigorously assess the ramifications of κ on the performance of our algorithm, a comprehensive investigation of its various settings was conducted.

Fig. 6 delineates the performance trajectory of our GPPO-SAG across an array of κ configurations. At $\kappa = 0$, ancestral sub-policies exert no influence on the current sub-policy decisions, rendering each sub-policy update conditionally independent and devoid of information from ancestral sub-policies. Conversely, at $\kappa = 1$, the ancestral sub-policies fully inform the current sub-policy decisions, but this complete reliance leads to an escalation in variance, which is caused by the multiplication of importance sampling. Ultimately deteriorating the algorithm's performance. For $\kappa > 1$, the ancestral sub-policies influence on the current sub-policy update is magnified by a factor of κ, exacerbating the variance issue exponentially, which further degrades the algorithm's effectiveness. In the regime where $0 < \kappa < 1$, the contribution of the ancestor sub-policies to the current sub-policy update will be reduced by k times, but the high variance problem will be reduced by an exponential multiple of κ. Therefore, there is an optimal value of k at this time, which makes the algorithm performance optimal.

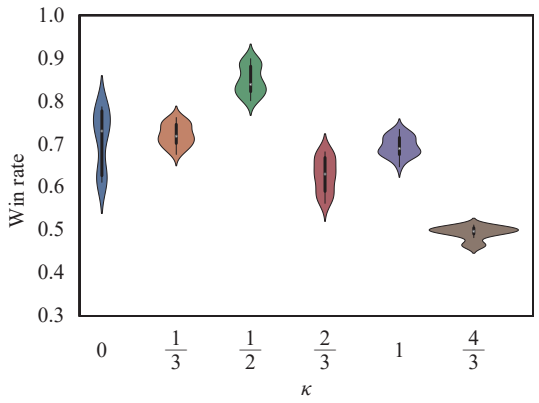


Fig. 6 Effect of the relative magnitude of inner and outer clipping ratio on performance in KairosJunction. The horizontal axis represents the different values of κ. The vertical axis represents the win rate. Each set was obtained by averaging the win rates of the last 500 games during training in three seeds. (Colored figures are available in the online version at https://link.springer.com/journal/11633)

Within this interval, an optimal κ value is discerned, at which the algorithm's performance is maximized. We ascertained that the algorithm attains peak performance at $\kappa = \frac{1}{2}$. This equilibrium facilitates a harmonious balance between leveraging the guidance of ancestral policies and preserving the stability of policy updates. At this juncture, the algorithm capitalizes on the ancestral policy constraints while attenuating the destabilizing effects of high variance introduced by the compounded importance sampling of ancestral policies, thus fostering consistent and efficacious learning. The empirical insights gleaned from this analysis informed our selection of κ for the ensuing experiments. By eschewing extreme κ values and

opting for the interval $0 < \kappa < 1$, we fortified the GPPO-SAG algorithm′s robustness and learning efficiency. This meticulous hyperparameter tuning accentuates the crucial role such parameters play in reinforcement learning algorithms, particularly their influence on successful outcomes in intricate decision-making tasks.

5.5 Limitations

The role of the guiding policy is to guide and constrain the learning direction, thereby accelerating and stabilizing the learning of the model. Our approach requires that the guiding policy be an effective one, meaning that during the process of interacting with the environment, it must be capable of collecting some high-value data. If there is no prior policy available or the quality of the prior policy cannot meet the requirements, we can simplify the problem to proximal policy optimization with structured action graph. In this case, we can assume that the behavior policy itself is the guiding policy, which is expressed as $\rho_g^{\mathrm{an}(j)} = 1, \rho_g^j = 1$ in (10), enabling the algorithm to run without any prior guidance. Structured action graphs are mainly used for modeling action spaces in RL problems that involve parameterized action spaces. In scenarios with parametrized instructions, a simple partitioning of the original action space suffices, as is the case in strategy games like StarCraft II and Hearthstone. However, in scenarios that lack parameterized action instructions, such as dm_control[40], the applicability of our method may be limited.

6 Conclusions

In this paper, we propose GPPO-SAG to solve the reinforcement learning challenges in complex decision-making problems. By formulating the environment as MDPs-SAG, and proposing and proving the policy improvement lower bound with guiding policy, we demonstrate the theoretical basis of GPPO-SAG. Furthermore, we provide a practical algorithm for GPPO-SAG and validate its superior performance in the StarCraft II and Hearthstone environments. We further corroborate our theoretical analysis by observing the change in KL divergence between the guiding policy and the target policy during training. We also give a multidimensional analysis of the sensitivity of our method to the clipping ratio.

In future work, we will introduce a guiding policy judgment mechanism to enable the agent to identify the shortcomings in the guiding policy and adopt it with trade-offs.

Acknowledgements

This work was supported by National Nature Science Foundation of China (Nos. 62073324, 6200629, 61771471 and 91748131), and in part by the InnoHK Project, China.

Declarations of conflict of interest

The authors declared that they have no conflicts of interest to this work.

Appendix A. Mathematical proofs

A.1 Proof for lemma 1

Proof. For simplicity, we use $\mathrm{an}(\cdot)$ pronouns $\mathrm{an}(\cdot) \sim \mathcal{P}$ and $\mathrm{an}'(\cdot)$ pronouns $\mathrm{an}(\cdot) \sim \mathcal{G}$. We have

$$A^{\mathcal{P}}(s, \boldsymbol{a}^{\mathcal{P}}) = A^{i,\mathrm{an}(i)(s,\boldsymbol{a}^{\mathrm{an}'(k)},\boldsymbol{a}^{i,\mathrm{an}(i)})}.$$

Then we use multi-agent advantage decomposition[36] to the right-hand side.

$$A^{\mathcal{P}}(s, \boldsymbol{a}^{\mathcal{P}}) = \sum_{j=k}^{i} A^j(s, \boldsymbol{a}^{-j}, a^j).$$

Remove the irrelevant vertices with a^j in a^{-j}, and we have

$$A^{\mathcal{P}}(s, \boldsymbol{a}^{\mathcal{P}}) = \sum_{j=k}^{i} A^j(s, \boldsymbol{a}^{j,\mathrm{an}(j)}).$$

□

Note that when removing v^{init} from $\mathcal{P}$, this lemma does not hold. Because in SAG, only v^{init} has no ancestor.

A.2 Proof for lemma 2

Proof. From "generalized policy improvement lower bound" theorem[33], we have

$$J(\hat{\boldsymbol{\pi}}_t) - J(\boldsymbol{\pi}_t) \geq \frac{1}{1-\gamma}\mathbf{E}_{\boldsymbol{\tau}\sim\boldsymbol{\pi}_b}\left[\frac{\hat{\boldsymbol{\pi}}_t(\boldsymbol{a}|s)}{\boldsymbol{\pi}_b(\boldsymbol{a}|s)}\hat{A}_{\boldsymbol{\pi}_t}(s,\boldsymbol{a})\right] - C\ \mathbf{E}_{s\sim\pi_b}[D_{TV}(\hat{\boldsymbol{\pi}}_t(\cdot|s), \boldsymbol{\pi}_b(\cdot|s))]$$

where $C = \frac{2\gamma \max_s \left|\mathbf{E}_{\boldsymbol{a}\sim\hat{\boldsymbol{\pi}}_t(\cdot,s)}\hat{A}_{\boldsymbol{\pi}_t}(s,\boldsymbol{a})\right|}{(1-\gamma)^2}$.

Since D_{TV} as a metric satisfies the triangle inequality, we have

$$D_{TV}(\hat{\boldsymbol{\pi}}_t, \boldsymbol{\pi}_g) + D_{TV}(\boldsymbol{\pi}_b, \boldsymbol{\pi}_g) \geq D_{TV}(\hat{\boldsymbol{\pi}}_t, \boldsymbol{\pi}_b).$$

Then we have

$$\begin{aligned} J(\hat{\boldsymbol{\pi}}_t) - J(\boldsymbol{\pi}_t) \geq\ & C_1\mathbf{E}_{\boldsymbol{\tau}\sim\boldsymbol{\pi}_b}\left[\frac{\hat{\boldsymbol{\pi}}_t(\boldsymbol{a}|s)}{\boldsymbol{\pi}_b(\boldsymbol{a}|s)}\hat{A}_{\boldsymbol{\pi}_t}(s,\boldsymbol{a})\right] - \\ & C_2\ \mathbf{E}_{s\sim\pi_b}[D_{TV}(\hat{\boldsymbol{\pi}}_t(\cdot|s), \boldsymbol{\pi}_b(\cdot|s))] \geq \\ & C_1\mathbf{E}_{\boldsymbol{\tau}\sim\boldsymbol{\pi}_b}\left[\frac{\hat{\boldsymbol{\pi}}_t(\boldsymbol{a}^{\mathcal{P}}|s)}{\boldsymbol{\pi}_b(\boldsymbol{a}^{\mathcal{P}}|s)}\hat{A}^{\mathcal{P}}_{\boldsymbol{\pi}_t}(s,\boldsymbol{a}^{\mathcal{P}})\right] - \\ & C_2\ \mathbf{E}_{s\sim\pi_b}[D_{TV}(\hat{\boldsymbol{\pi}}_t(\cdot|s), \boldsymbol{\pi}_g(\cdot|s)) + \\ & D_{TV}(\boldsymbol{\pi}_b(\cdot|s), \boldsymbol{\pi}_g(\cdot|s))]. \end{aligned}$$

□

A.3 Proof for theorem 1

Proof. Step 1. We derive the relationship between the joint policy and each vertex sub-policy under path graph $\mathscr{P}$ based on lemma 2.

From lemma 1, we have

$$\begin{aligned}&\mathbf{E}_{\boldsymbol{\tau}\sim\pi_b}\left[\frac{\hat{\boldsymbol{\pi}}_t(\boldsymbol{a}^{\mathscr{P}}|s)}{\boldsymbol{\pi}_b(\boldsymbol{a}^{\mathscr{P}}|s)}\hat{A}^{\mathscr{P}}_{\boldsymbol{\pi}_t}(s,\boldsymbol{a}^{\mathscr{P}})\right]=\\&\mathbf{E}_{\boldsymbol{\tau}\sim\pi_b}\left[\sum_{j=k}^{i}\frac{\hat{\boldsymbol{\pi}}_t(\boldsymbol{a}^{\mathscr{P}}|s)}{\boldsymbol{\pi}_b(\boldsymbol{a}^{\mathscr{P}}|s)}\hat{A}^{j}_{\boldsymbol{\pi}_t}(s,\boldsymbol{a}^{j,\mathrm{an}(j)})\right]=\\&\mathbf{E}_{\boldsymbol{\tau}\sim\pi_b}\left[\sum_{j=k}^{i}\frac{\hat{\pi}^j_t(a^j|s,\boldsymbol{a}^{\mathrm{an}(j)})}{\pi^j_b(a^j|s,\boldsymbol{a}^{\mathrm{an}(j)})}\times\right.\\&\left.\frac{\hat{\pi}^{\mathrm{an}(j)}_t(\boldsymbol{a}^{\mathrm{an}(j)}|s)}{\pi^{\mathrm{an}(j)}_b(\boldsymbol{a}^{\mathrm{an}(j)}|s)}\hat{A}^{j}_{\boldsymbol{\pi}_t}(s,\boldsymbol{a}^{j,\mathrm{an}(j)})\right]=\\&\sum_{j=k}^{i}\mathbf{E}_{\boldsymbol{\tau}\sim\pi_b}\left[\frac{\hat{\pi}^j_t(a^j|s,\boldsymbol{a}^{\mathrm{an}(j)})}{\pi^j_b(a^j|s,\boldsymbol{a}^{\mathrm{an}(j)})}\times\right.\\&\left.\frac{\hat{\pi}^{\mathrm{an}(j)}_t(\boldsymbol{a}^{\mathrm{an}(j)}|s)}{\pi^{\mathrm{an}(j)}_b(\boldsymbol{a}^{\mathrm{an}(j)}|s)}\hat{A}^{j}_{\boldsymbol{\pi}_t}(s,\boldsymbol{a}^{j,\mathrm{an}(j)})\right].\end{aligned}$$

Step 2. We need to prove

$$\begin{aligned}&\mathbf{E}_{s\sim\pi_b}\left[D_{TV}(\boldsymbol{\pi}_b(\cdot|s),\boldsymbol{\pi}_g(\cdot|s))\right]\leq\\&\sum_{j=k}^{i}\mathbf{E}_{s\sim\pi_b}\left[D_{TV}(\pi^j_b(\cdot|s),\pi^j_g(\cdot|s))\right].\end{aligned}\tag{A1}$$

According to proposition 4.2 in [41], when S and $\mathscr{A}^{\mathscr{P}}$ are both finite, then (A1) is equal to

$$\begin{aligned}&\frac{1}{2}\mathbf{E}_{s\sim\pi_b}\left[\sum_{a\in\mathscr{A}^{\mathscr{P}}}\left|\prod_{j=k}^{i}\pi^j_b(\boldsymbol{a}^{\mathrm{an}(j)}|s)-\prod_{j=k}^{i}\pi^j_g(\boldsymbol{a}^{\mathrm{an}(j)}|s)\right|\right]\leq\\&\frac{1}{2}\sum_{j=k}^{i}\mathbf{E}_{s\sim\pi_b}\left[\sum_{a\in\mathscr{A}^{\mathscr{P}}}|\pi^j_b(\boldsymbol{a}^{\mathrm{an}(j)}|s)-\pi^j_g(\boldsymbol{a}^{\mathrm{an}(j)}|s)|\right].\end{aligned}$$

And then we put the sum of the sub-policies into the expectation

$$\begin{aligned}&\frac{1}{2}\mathbf{E}_{s\sim\pi_b}\left[\sum_{a\in\mathscr{A}^{\mathscr{P}}}\left|\prod_{j=k}^{i}\pi^j_b(\boldsymbol{a}^{\mathrm{an}(j)}|s)-\prod_{j=k}^{i}\pi^j_g(\boldsymbol{a}^{\mathrm{an}(j)}|s)\right|\right]\leq\\&\frac{1}{2}\mathbf{E}_{s\sim\pi_b}\left[\sum_{j=k}^{i}\sum_{a\in\mathscr{A}^{\mathscr{P}}}|\pi^j_b(\boldsymbol{a}^{\mathrm{an}(j)}|s)-\pi^j_g(\boldsymbol{a}^{\mathrm{an}(j)}|s)|\right].\end{aligned}$$

Next, we prove a stronger version of the above inequation, that is $\forall s\in S,\forall\boldsymbol{a}\in\mathscr{A}$:

$$\begin{aligned}&\left|\prod_{j=k}^{i}\pi^j_b(\boldsymbol{a}^{\mathrm{an}(j)}|s)-\prod_{j=k}^{i}\pi^j_g(\boldsymbol{a}^{\mathrm{an}(j)}|s)\right|\leq\\&\sum_{j=k}^{i}|\pi^j_b(\boldsymbol{a}^{\mathrm{an}(j)}|s)-\pi^j_g(\boldsymbol{a}^{\mathrm{an}(j)}|s)|.\end{aligned}\tag{A2}$$

Substep 1. For any $k\in\mathbf{N}^*$, we have

$$\begin{aligned}&\left|\pi^k_b(\boldsymbol{a}^{\mathrm{an}(k)}|s)\times\pi^{k+1}_b(\boldsymbol{a}^{\mathrm{an}(k)}|s)-\pi^k_g(\boldsymbol{a}^{\mathrm{an}(k)}|s)\times\right.\\&\left.\pi^{k+1}_g(\boldsymbol{a}^{\mathrm{an}(k)}|s)\right|=\pi^k_b(\boldsymbol{a}^{\mathrm{an}(k)}|s)\times\\&\left|\pi^{k+1}_b(\boldsymbol{a}^{\mathrm{an}(k)}|s)-\pi^{k+1}_g(\boldsymbol{a}^{\mathrm{an}(k)}|s)\right|+\\&\pi^{k+1}_g(\boldsymbol{a}^{\mathrm{an}(k)}|s)\times\left|\pi^k_b(\boldsymbol{a}^{\mathrm{an}(k)}|s)-\pi^k_g(\boldsymbol{a}^{\mathrm{an}(k)}|s)\right|\leq\\&\left|\pi^k_b(\boldsymbol{a}^{\mathrm{an}(k)}|s)-\pi^k_g(\boldsymbol{a}^{\mathrm{an}(k)}|s)\right|+\\&\left|\pi^{k+1}_b(\boldsymbol{a}^{\mathrm{an}(k)}|s)-\pi^{k+1}_g(\boldsymbol{a}^{\mathrm{an}(k)}|s)\right|.\end{aligned}$$

Substep 2. For any $i\in\mathbf{N}^*$ and $i>k$, we have

$$\begin{aligned}&\left|\prod_{j=k}^{i+1}\pi^j_b(\boldsymbol{a}^{\mathrm{an}(j)}|s)-\prod_{j=k}^{i+1}\pi^j_g(\boldsymbol{a}^{\mathrm{an}(j)}|s)\right|=\\&\pi^{i+1}_b(\boldsymbol{a}^{\mathrm{an}(i+1)}|s)\times\left|\prod_{j=k}^{i}\pi^j_b(\boldsymbol{a}^{\mathrm{an}(j)}|s)-\prod_{j=k}^{i}\pi^j_g(\boldsymbol{a}^{\mathrm{an}(j)}|s)\right|+\\&\prod_{j=k}^{i}\pi^j_g(\boldsymbol{a}^{\mathrm{an}(j)}|s)\times|\pi^{i+1}_b(\boldsymbol{a}^{\mathrm{an}(i+1)}|s)-\pi^{i+1}_g(\boldsymbol{a}^{\mathrm{an}(i+1)}|s)|\leq\\&\sum_{j=k}^{i+1}|\pi^j_b(\boldsymbol{a}^{\mathrm{an}(j)}|s)-\pi^j_g(\boldsymbol{a}^{\mathrm{an}(j)}|s)|.\end{aligned}$$

From the discussion above:

1) We have shown that (A2) is true for any $k\in\mathbf{N}^*$ and $i=k+1$.

2) Given that (A2) applies to any $i\in\mathbf{N}^*$ and $i>k$, then we have shown that the formula also applies to $i+1$.

Through mathematical induction, we can confirm that (A2) is true.

So (A1) is true.

The same goes for the proof of the formula

$$\begin{aligned}&\mathbf{E}_{s\sim\pi_b}\left[D_{TV}(\hat{\boldsymbol{\pi}}_t(\cdot|s),\boldsymbol{\pi}_g(\cdot|s))\right]\leq\\&\sum_{j=k}^{i}\mathbf{E}_{s\sim\pi_b}[D_{TV}(\hat{\boldsymbol{\pi}}^j_t(\cdot|s),\pi^j_g(\cdot|s))].\end{aligned}$$

From the two steps above, we have

$$\begin{aligned}&J(\hat{\boldsymbol{\pi}}_t)-J(\boldsymbol{\pi}_t)\geq\\&\sum_{j=k}^{i}\left[\frac{1}{1-\gamma}\mathbf{E}_{\boldsymbol{\tau}\sim\pi_b}\left[\rho^{j,\mathrm{an}(j)}\hat{A}^{\mathscr{P}}_{\pi_t}(s,\boldsymbol{a}^{\mathscr{P}})\right]-\right.\\&C\ \mathbf{E}_{s\sim\pi_b}\left[D_{TV}(\hat{\pi}^j_t(\cdot|s),\pi^j_g(\cdot|s))+\right.\\&\left.\left.D_{TV}(\pi^j_b(\cdot|s),\pi^j_g(\cdot|s))\right]\right]\end{aligned}$$

where $C=\frac{2\gamma\max_s\left|\mathbf{E}_{a^j\sim\hat{\pi}^j_t(\cdot,s),\boldsymbol{a}^{\mathrm{an}(j)}\sim\hat{\boldsymbol{\pi}}^{\mathrm{an}(j)}_t(\cdot,s)}\hat{A}^{j}_{\pi_t}(s,\boldsymbol{a}^{j,\mathrm{an}(j)})\right|}{(1-\gamma)^2}$,

and $\rho^{j,\mathrm{an}(j)}=\frac{\hat{\pi}^j_t(a^j|s,\boldsymbol{a}^{\mathrm{an}(j)})}{\pi^j_b(a^j|s,\boldsymbol{a}^{\mathrm{an}(j)})}\times\frac{\hat{\boldsymbol{\pi}}^{\mathrm{an}(j)}_t(\boldsymbol{a}^{\mathrm{an}(j)}|s)}{\boldsymbol{\pi}^{\mathrm{an}(j)}_b(\boldsymbol{a}^{\mathrm{an}(j)}|s)}$. □

A.4 The relation between Theorem 1 and the surrogate objective function

Let $\rho_t^{\mathrm{an}(j)} = \frac{\hat{\boldsymbol{\pi}}_t^{\mathrm{an}(j)}(\boldsymbol{a}^{\mathrm{an}(j)}|s)}{\boldsymbol{\pi}_b^{\mathrm{an}(j)}(\boldsymbol{a}^{\mathrm{an}(j)}|s)}$, $\rho_g^{\mathrm{an}(j)} = \frac{\boldsymbol{\pi}_g^{\mathrm{an}(j)}(\boldsymbol{a}^{\mathrm{an}(j)}|s)}{\boldsymbol{\pi}_b^{\mathrm{an}(j)}(\boldsymbol{a}^{\mathrm{an}(j)}|s)}$, $\rho_t^j = \frac{\hat{\pi}_t^j(a^j|s,\boldsymbol{a}^{\mathrm{an}(j)})}{\pi_b^j(a^j|s,\boldsymbol{a}^{\mathrm{an}(j)})}$, and $\rho_g^j = \frac{\pi_g^j(a^j|s,\boldsymbol{a}^{\mathrm{an}(j)})}{\pi_b^j(a^j|s,\boldsymbol{a}^{\mathrm{an}(j)})}$, then the surrogate objective function to optimize is as follows:

$$\sum_{j=k}^{i} \min\Big[\mathrm{clip}(\rho_t^{\mathrm{an}(j)}, \rho_g^{\mathrm{an}(j)}, \kappa\epsilon)\rho_t^j \hat{A}_{\boldsymbol{\pi}_t}^{\mathscr{P}}(s, \boldsymbol{a}^{\mathscr{P}}), \mathrm{clip}\big(\mathrm{clip}(\rho_t^{\mathrm{an}(j)}, \rho_g^{\mathrm{an}(j)}, \kappa\epsilon)\rho_t^j, \rho_g^j, \epsilon\big)\hat{A}_{\boldsymbol{\pi}_t}^{\mathscr{P}}(s, \boldsymbol{a}^{\mathscr{P}})\Big]$$

where $\mathrm{clip}(\rho_1, \rho_2, \epsilon)$ puts ρ_1 into the interval $[\rho_2 - \epsilon, \rho_2 + \epsilon]$ and $\kappa \in [0, +\infty)$ is a constant scaling factor.

Within our paradigm, the behavior policies are from the target policy. So we only need to bound $\mathbf{E}_{s\sim\pi_b}[D_{TV}(\hat{\boldsymbol{\pi}}_t(\cdot|s), \boldsymbol{\pi}_g(\cdot|s))]$:

$$\mathbf{E}_{s\sim\pi_b}[D_{TV}(\hat{\boldsymbol{\pi}}_t(\cdot|s), \boldsymbol{\pi}_g(\cdot|s))] \leq \tag{A3}$$

$$\frac{1}{2}\sum_{j=k}^{i} \mathbf{E}_{s\sim\pi_b}[D_{TV}(\hat{\boldsymbol{\pi}}_t^j(\cdot|s), \pi_g^j(\cdot|s))] \leq \tag{A4}$$

$$\frac{1}{2}\mathbf{E}_{s\sim\pi_b}\Bigg[\sum_{a\in\mathscr{A}^{\mathscr{P}}}\Big|\prod_{j=k}^{i}\hat{\pi}_t^j(\boldsymbol{a}^{\mathrm{an}(j)}|s) - \prod_{j=k}^{i}\pi_g^j(\boldsymbol{a}^{\mathrm{an}(j)}|s)\Big|\Bigg] = \tag{A5}$$

$$\frac{1}{2}\mathbf{E}_{s\sim\pi_b}\Bigg[\sum_{a\in\mathscr{A}^{\mathscr{P}}}\prod_{j=k}^{i}\pi_b^j(\boldsymbol{a}^{\mathrm{an}(j)}|s)\times\Bigg|\frac{\prod_{j=k}^{i}\hat{\pi}_t^j(\boldsymbol{a}^{\mathrm{an}(j)}|s)}{\prod_{j=k}^{i}\pi_b^j(\boldsymbol{a}^{\mathrm{an}(j)}|s)} - \frac{\prod_{j=k}^{i}\pi_g^j(\boldsymbol{a}^{\mathrm{an}(j)}|s)}{\prod_{j=k}^{i}\pi_b^j(\boldsymbol{a}^{\mathrm{an}(j)}|s)}\Bigg|\Bigg] = \tag{A6}$$

$$\frac{1}{2}\left|\frac{\prod_{j=k}^{i}\hat{\pi}_t^j(\boldsymbol{a}^{\mathrm{an}(j)}|s)}{\prod_{j=k}^{i}\pi_b^j(\boldsymbol{a}^{\mathrm{an}(j)}|s)} - \frac{\prod_{j=k}^{i}\pi_g^j(\boldsymbol{a}^{\mathrm{an}(j)}|s)}{\prod_{j=k}^{i}\pi_b^j(\boldsymbol{a}^{\mathrm{an}(j)}|s)}\right| \tag{A7}$$

where we have proven (A4) in Appendix A.3. From (A4) through (A5) holds since we have beared out it in step 2 of Appendix A.3.

So we can use $\mathrm{clip}\left(\frac{\hat{\pi}_t}{\pi_b}, \frac{\pi_g}{\pi_b}, \epsilon\right)$ to bound $D_{TV}(\hat{\boldsymbol{\pi}}_t, \boldsymbol{\pi}_g)$ between $\frac{1}{2}\epsilon$.

There have been lots of works showing that the continuous multiplication of importance sampling can arise variance, and a simple and effective way to deal with this problem is to clip the combination of importance sampling ratios[37, 38, 42]. Based on this consideration, we design the clip of the importance sampling ratio of the ancestral sub-policies.

Appendix B. Experiments setting and additional experiments

B.1 Ablation study with varying quality of $\boldsymbol{\pi}_g$

In this experiment, we aim to investigate the impact of the quality of the guiding policy on the learning performance of GPPO-SAG. As shown in Fig. B1, we have conducted the ablation experiment on StarCraft II map KingsCove. The red curve represents the training curve of the model with an 80% win rate achieved through reinforcement learning, which is used as both the guiding policy and the initialization target policy. The blue curve represents the training curve of the model with a 20% win rate achieved through supervised learning, which is used as both the guiding policy and the initialization target policy. The green curve represents the training curve of the model with a 20% win rate achieved through supervised learning, which is used only as the initialization target policy and not as the guiding policy ($\rho_g^{\mathrm{an}(j)} = 1$, $\rho_g^j = 1$).

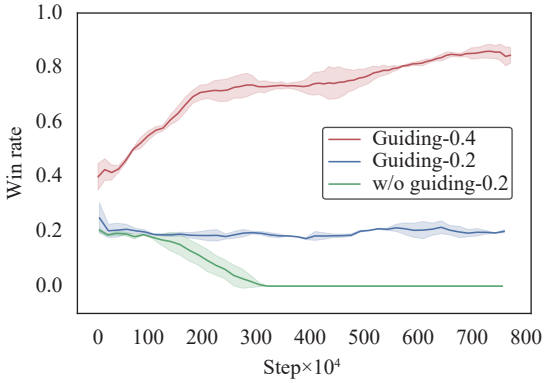


Fig. B1 Ablation of varying quality of π_g on StarCraft II map KingsCove (Colored figure is available in the online version at https://link.springer.com/journal/11633)

It is evident that the quality of the guiding policy can have a significant impact on the learning effectiveness of the intelligent agent in complex problems. In the absence of a guiding policy ("w/o guiding-0.2"), the large policy space can quickly destroy the prior flow space provided by the initialization policy, leading to a rapid decline in win rate and ineffective policy improvement. Similarly, when the quality of the guiding policy is poor (e.g., "guiding-0.2"), although the constraint of the guiding policy ensures that the win rate does not decrease as rap-

idly as the green line, the lack of high-quality trajectory samples still makes it difficult to achieve effective policy improvement for the target policy. However, when we provide a higher quality guiding policy ("guiding-0.4"), our method can achieve robust policy improvement within the prior policy flow space.

In summary, for complex decision-making problems, our method imposes certain performance requirements of the guiding policy, namely that it must be capable of collecting high-value data during the process of interacting with the environment.

B.2 Supervised learning in SC2

Detail hyperparameters for supervised learning in StarCraft II full game have been shown in Table B1. We show the training results in https://tensorboard.dev/experiment/IpznauMnTraAYrL8S7uEQw/.

B.3 Reinforcement learning in SC2

Table B2 shows the hyperparameters used for reinforcement learning in the full StarCraft II game. In our method, the "kl_loss_weight" and "action_type_kl_loss_weight" are set to 0, and the sub-action loss weight is set to 1 in RL training. In PPO-KL, we set ϵ to 0.3 in the following experiment. We also present plots of KL divergence with reinforcement learning training for each method on the other two maps in Figs. B2 and B3, which are consistent with the analysis in our main paper.

B.4 Experiment in Hearthstone

We show the experimental hyperparameters for

Table B1 SL hyperparameters in SC2

Name	Value	Description
Beginning_order_prob	0.8	Probability of using building order in SL training
Learning_rate	10^{-3}	
Weight_decay	10^{-5}	Weight decay in adam
Warm_up_steps	2×10^4	Max warm-up step
Steps	10^5	Max training step
Epochs	10	Repeating number of replay paths
Batch_size	12	
Trajectory_length	32	Context states length for training
Action_type _loss_weight	30.0	Loss weight for action type vertex
Delay_loss_weight	9.0	Loss weight for delay vertex
Queued_loss_weight	1.0	Loss weight for queued vertex
Selected_units _loss_weight	4.0	Loss weight for selected units vertex
Target_unit _loss_weight	4.0	Loss weight for target unit vertex
Target_location _loss_weight	8.0	Loss weight for target location vertex

Table B2 RL hyperparameters in SC2

Name	Value	Description
Steps	10^7	Max steps for training
Learning_rate	10^{-5}	
Value_pretrain _iters	4×10^3	Max step for value pre-training
Optimizer_warm _up_steps	100	Warm-up adam optimizer without updating model
Batch_size	8	
Trajectory_length	32	Context states length for training
Value_loss _weight	10.0	Loss weight for critic
Pg_loss_weight	1.0	Loss weight for policy
Upgo_loss_weight	1.0	Loss weight for UPGO
Kl_loss_weight	0.002	Loss weight for KL divergence with guiding policy (only enabled in the baseline)
Action_type_kl _loss_weight	0.1	Extra action type KL loss weight with guiding policy (only enabled in the baseline)
Entropy_loss_weight	10^{-4}	Entropy loss weight

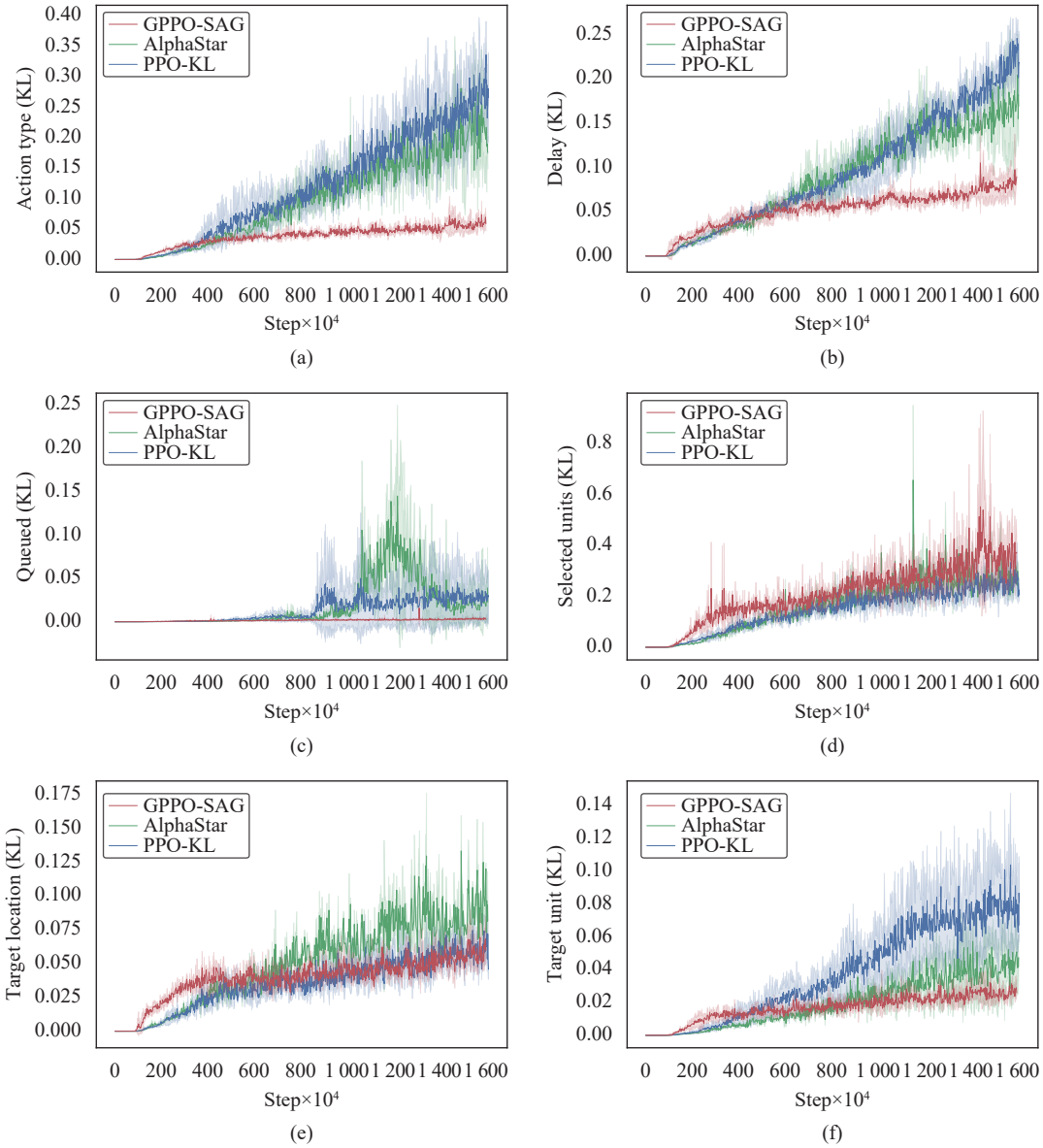
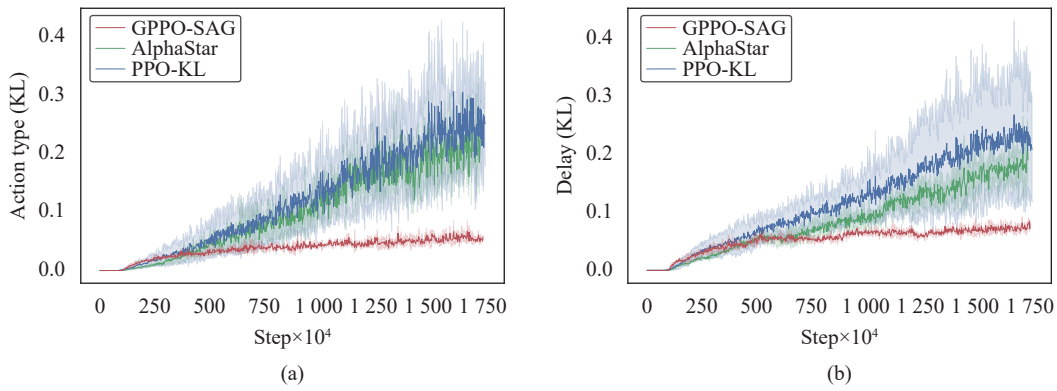


Fig. B2 The KL divergence curve between the target policy $\boldsymbol{\pi}_t$ and the guiding policy $\boldsymbol{\pi}_g$ during training for each method on the StarCraft II's KingsCove map. (a) Action type vertex; (b) Delay vertex; (c) Queued vertex; (d) Selected units vertex; (e) Target location vertex; (f) Target unit vertex. (Colored figures are available in the online version at https://link.springer.com/journal/11633)



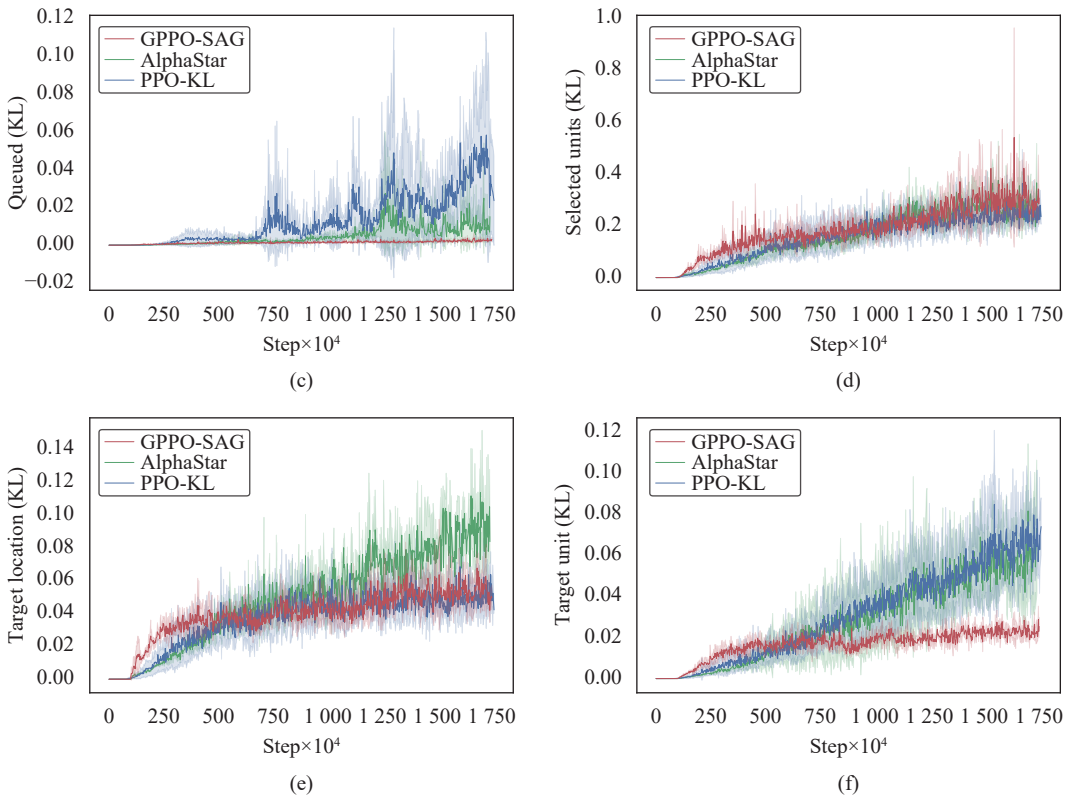


Fig. B3 The KL divergence curve between the target policy $\boldsymbol{\pi}_t$ and the guiding policy $\boldsymbol{\pi}_g$ during training for each method on the StarCraft II's NewRepugnancy map. (a) Action type vertex; (b) Delay vertex; (c) Queued vertex; (d) Selected units vertex; (e) Target location vertex; (f) Target unit vertex. (Colored figures are available in the online version at https://link.springer.com/journal/11633)

Hearthstone's environment in Table B3. Fig. B4 shows two example cards, "abusive sergeant" and fireball. "Mana spent" represents the resource consumption of this card, and the player's resource per turn is limited. If the card is a minion card, it has attack and health, and when played, it will be placed on our field. "Special effect" represents the special effect that this card has. According to different effects, some cards are non-directional, some are directional, and some are selective. "Abusive sergeant" is a minion card with a directionality battle cry, which means we need three steps to play this card: Select this card, then select a target position to place, and finally select a target unit to make the battle cry take effect. The card "fireball" is a directional spell card, which means we need to use it by step: Select this card and then a target unit. Therefore, similar to StarCraft II, the action space of Hearthstone's agent is also well suited to be modeled using the structured action graph.

We implement the Hearthstone environment using SabberStone[1], an open-source Hearthstone simulator for AI research. To simplify the modeling process, we skip

[1] https://github.com/HearthSim/SabberStone

Table B3 RL hyperparameters in HS

Name	Value	Description
Steps	2×10^8	Max steps for training
Learning_rate	10^{-5}	
Value_pretrain _iters	4×10^7	Max step for value pretrain
Batch_size	24	
Unroll_length	100	The maximum time step of a game
Value_loss _weight	1.00	Loss weight for critic
Pg_loss_weight	1.00	Loss weight for policy
Upgo_loss_weight	1.00	Loss weight for UPGO
Kl_loss_weight	0.10	Loss weight for KL divergence with guiding policy (only enabled in the baseline)
Action_type_kl _loss_weight	0.10	Extra action type KL loss weight with guiding policy (only enabled in the baseline)
Entropy_loss_weight	0.01	Entropy loss weight

Fig. B4 Example cards in Hearthstone, "abusive sergeant" is a minion card, and "fireball" is a spell card.

the mulligan phase at the beginning of the game, which is not relevant to our research goals. We use a warlock deck for all games, and the composition of the card set is shown in Table B4. Each card appears twice in the deck.

We set batch size as 24, and learning rate to 10^{-5}, and we simultaneously use 12 collectors interacting with the environment to provide data for a trainer.

The network structure used in our Hearthstone experiments is illustrated in Fig. 4(a). For simplicity, in Hearthstone, we do not use context information but only input the current moment agent's observations for the action and value networks.

Fig. 4(b) shows the functions represented by each sub-action vertex in our method. The state observations are encoded and compressed into a 512-dimensional vector by transformer encoder layers, which are then passed to the iterative action selection module. The structure of each sub-policy network is the same as that of the "action head" network in AlphaStar.

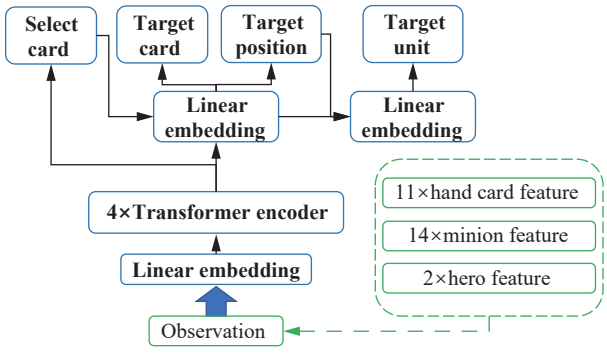
Here we explain with the help of an example how we represent parameterized actions with structured action diagrams: In Fig. B5, we can see a player taking the action of "using the first 'molten reflection' card in their left hand to target their left minion 'sorcerer's apprentice'", which results in a copy of that minion being placed onto the field. This action can be broken down into two sub-actions: First, the player "chooses the 'molten reflection' card from the left hand side", and second, they "target the card to the left minion 'sorcerer's apprentice'". Note that the currently active sub-action

Table B4 The deck for our experiment

Name
Soulfire
Abusive sergeant
Argent squire
Flame imp
Leper gnome
Mortal coil
Voidwalker
Young priestess
Dire wolf alpha
Knife juggler
Harvest golem
Shattered sun cleric
Dark iron dwarf
Defender of argus
Doomguard

header is marked on the bottom side with a bolded red circle. It is important to note that the decision made in the second sub-action is conditional on the first sub-action: The player must know which card was chosen in the first sub-action to determine the entire action of "using the first 'molten reflection' in your left hand, targeting your left attachment, 'sorcerer's apprentice'".

Fig. 4(c) shows the details of the input state encoding. At each step, we input the observed information of 27 entities as states into the transformer encoder and encode the current state as a 256-dimensional vector to the action selection network (state value decoder). Finally, the action (state value) is outputted based on this encoded state, Fig. B6 shows in detail our process of splitting actions into sub-action sequences using the structural action graph: First select a card, which is represented by the vertex v^{SC} in SAG, and then select the action target of this card, which is represented by the vertex v^{TU} in SAG. Finally, based on the above two decomposed sub-actions, the action using the "molten reflection" card is actually made.



(a)

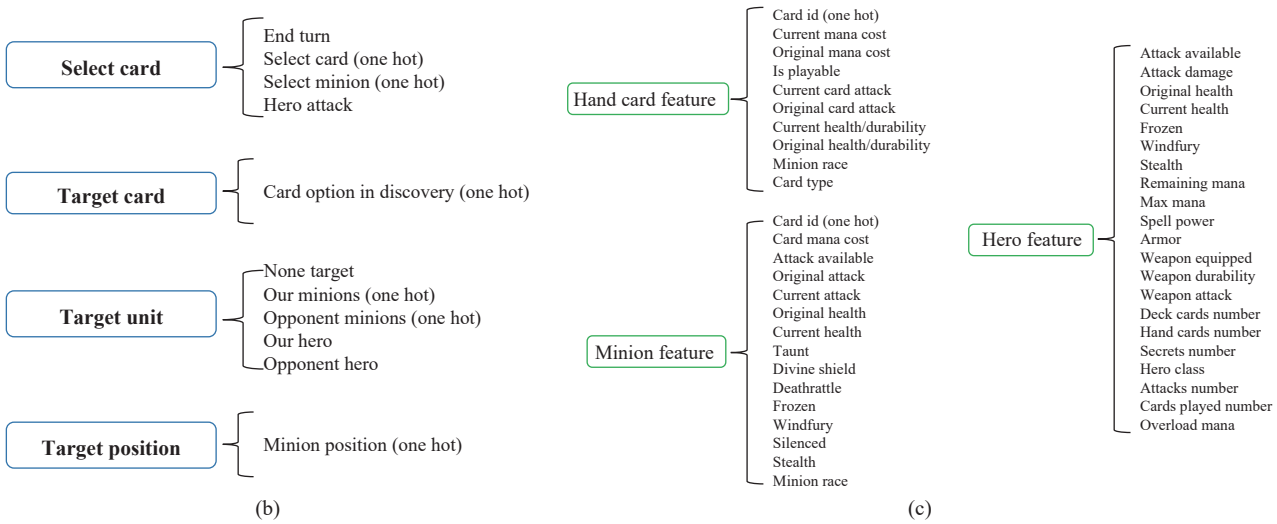


Fig. B5 Modeling of state space, action space, and strategy network in Hearthstone environment. (a) The blue part is the policy network structure in the Hearthstone agent. The green part is the composition of the observation; (b) The function of each sub-action vertex in the Hearthstone agent; (c) The detailed composition of the observation in Hearthstone. (Colored figures are available in the online version at https://link.springer.com/journal/11633)

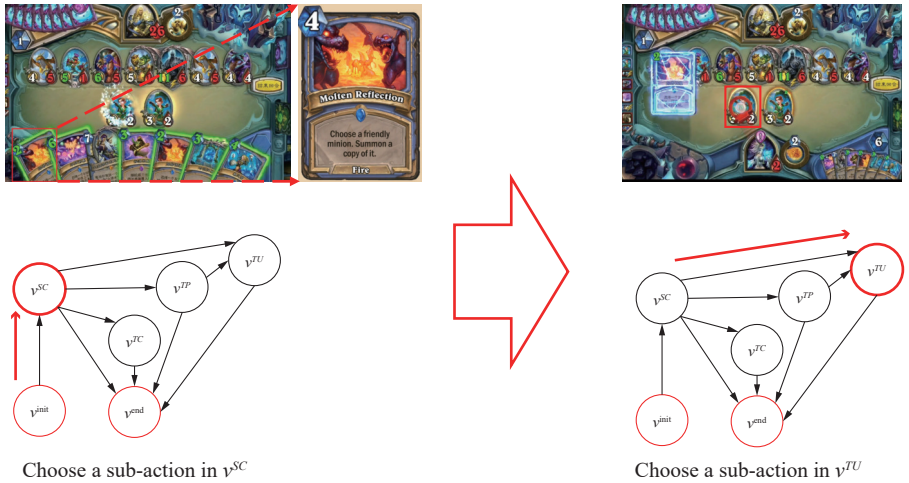


Fig. B6 Using structured action graph to represent a parameterized action (Colored figures are available in the online version at https://link.springer.com/journal/11633)

References

[1] B. Ichter, A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, D. Kalashnikov, S. Levine, Y. Lu, C. Parada, K. Rao, P. Sermanet, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, M. Y. Yan, N. Brown, M. Ahn, O. Cortes, N. Sievers, C. Tan, S. C. Xu, D. Reyes, J. Rettinghouse, J. Quiambao, P. Pastor, L. Luu, K. H. Lee, Y. H. Kuang, S. Jesmonth, N. J. Joshi, K. Jeffrey, R. J. Ruano, J. Hsu, K. Gopalakrishnan, B. David, A. Zeng, C. K. Fu. Do as I can, not as I say: Grounding language in robotic affordances. In *Proceedings of the 6th Conference on Robot Learning*, Auckland, New Zealand, pp. 287–318, 2022.

[2] S. Srivastava, C. S. Li, M. Lingelbach, R. Martín-Martín, F. Xia, K. E. Vainio, Z. Lian, C. Gokmen, S. Buch, C. K. Liu, S. Savarese, H. Gweon, J. J. Wu, F. F. Li. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Proceedings of the 5th Conference on Robot Learning*, London, UK, pp. 477–490, 2021.

[3] A. Stooke, A. Mahajan, C. Barros, C. Deck, J. Bauer, J. Sygnowski, M. Trebacz, M. Jaderberg, M. Mathieu, N. McAleese, N. Bradley-Schmieg, N. Wong, N. Porcel, R. Raileanu, S. Hughes-Fitt, V. Dalibard, W. M. Czarnecki. Open-ended learning leads to generally capable agents, [Online], Available: https://arxiv.org/abs/2107.12808, 2021.

[4] OpenAI. Gpt-4 technical report, [Online], Available: https://arxiv.org/abs/2303.08774, 2023.

[5] S. Levine. Understanding the world through action. In *Proceedings of the 5th Conference on Robot Learning*, London, UK, pp. 1752–1757, 2021.

[6] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Si-

mens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, R. Lowe. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, New Orleans, USA, pp. 27730–27744, 2022.

[7] R. S. Sutton, D. McAllester, S. Singh, Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, Denver, USA, pp. 1057–1063, 1999. DOI: 10.5555/3009657.3009806.

[8] S. Lohmann. Optimal commitment in monetary policy: Credibility versus flexibility. *The American Economic Review*, vol. 82, no. 1, pp. 273–286, 1992.

[9] J. Schulman, S. Levine, P. Moritz, M. Jordan, P. Abbeel. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, Lille, France, vol. 37, pp. 1889–1897, 2015. DOI: 10.5555/3045118.3045319.

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov. Proximal policy optimization algorithms, [Online], Available: https://arxiv.org/abs/1707.06347, 2017.

[11] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Y. Wang, T. Pfaff, Y. H. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, D. Silver. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nature*, vol. 575, no. 7782, pp. 350–354, 2019. DOI: 10.1038/s41586-019-1724-z.

[12] X. J. Wang, J. X. Song, P. H. Qi, P. Peng, Z. K. Tang, W. Zhang, W. M. Li, X. J. Pi, J. J. He, C. Gao, H. T. Long, Q. Yuan. SCC: An efficient deep reinforcement learning agent mastering the game of starcraft II. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 10905–10915, 2021.

[13] DI-starContributors: DI-star: An Open-sourse Reinforcement Learning Framework for StarCraftII. GitHub, 2021. https://github.com/opendilab/DI-star

[14] OpenAI. Dota 2 with large scale deep reinforcement learning, [Online], Available: https://arxiv.org/abs/1912.06680, 2019.

[15] D. H. Ye, Z. Liu, M. F. Sun, B. Shi, P. L. Zhao, H. Wu, H. S. Yu, S. J. Yang, X. P. Wu, Q. W. Guo, Q. B. Chen, Y. Y. T. Yin, H. Zhang, T. F. Shi, L. Wang, Q. Fu, W. Yang, L. X. Huang. Mastering complex control in moba games with deep reinforcement learning. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, New York, USA, pp. 6672–6679, 2020. DOI: 10.1609/aaai.v34i04.6144.

[16] M. Mathieu, S. Ozair, S. Srinivasan, C. Gulcehre, S. T. Zhang, R. Jiang, T. Le Paine, K. Żolna, R. Powell, J. Schrittwieser, D. Choi, P. Georgiev, D. Toyama, A. Huang, R. Ring, I. Babuschkin, T. Ewalds, M. Bordbar, S. Henderson, S. G. Colmenarejo, A. Van Den Oord, W. M. Czarnecki, N. De Freitas, O. Vinyals. Starcraft II unplugged: Large scale offline reinforcement learning. In *Proceedings of the 35th Conference on Neural Information Processing Systems*, Sydney, Australia, 2021.

[17] C. Eisenach, H. C. Yang, J. Liu, H. Liu. Marginal policy gradients: A unified family of estimators for bounded action spaces with applications. In *Proceedings of the 7th International Conference on Learning Representations*, New Orleans, USA, 2019.

[18] J. G. Kuba, R. Q. Chen, M. N. Wen, Y. Wen, F. L. Sun, J. Wang, Y. D. Yang. Trust region policy optimisation in multi-agent reinforcement learning. In *Proceedings of the 10th International Conference on Learning Representations*, pp. 1–27, 2022.

[19] Z. J. Pang, R. Z. Liu, Z. Y. Meng, Y. Zhang, Y. Yu, T. Lu. On reinforcement learning for full-length game of StarCraft. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, Honolulu, USA, pp. 4691–4698, 2019. DOI: 10.1609/aaai.v33i01.33014691.

[20] K. Jothimurugan, S. Bansal, O. Bastani, R. Alur. Compositional reinforcement learning from logical specifications. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pp. 10026–10039, 2021

[21] C. Tessler, S. Givony, T. Zahavy, D. J. Mankowitz, S. Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, San Francisco, USA, pp. 1553–1561, 2017. DOI: 10.5555/3298239.3298465.

[22] B. Wu. Hierarchical macro strategy model for MOBA game AI. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, Honolulu, USA, pp. 1206–1213, 2019. DOI: 10.1609/aaai.v33i01.33011206.

[23] L. Metz, J. Ibarz, N. Jaitly, J. Davidson. Discrete sequential prediction of continuous actions for deep RL, [Online], Available: https://arxiv.org/abs/1705.05035, 2017.

[24] W. Y. Ma, Q. R. Mi, X. Yan, Y. Q. Wu, R. J. Lin, H. F. Zhang, J. Wang. Large language models play starcraft II: Benchmarks and a chain of summarization approach, [Online], Available: https://arxiv.org/abs/2312.11865, 2023.

[25] S. F. Cai, Z. H. Wang, X. J. Ma, A. J. Liu, Y. T. Liang. Open-world multi-task control through goal-aware representation learning and adaptive horizon prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Vancouver, Canada, pp. 13734–13744, 2023. DOI: 10.1109/CVPR52729.2023.01320.

[26] H. Yuan, C. Zhang, H. Wang, F. Xie, P. Cai, H. Dong, Z. Lu. Skill reinforcement learning and planning for open-world long-horizon tasks. In *Proceedings of the Foundation Models for Decision Making Workshop*, New Orleans, USA, pp. 1–24, 2023.

[27] D. M. Lyu, F. K. Yang, B. Liu, S. Gustafson. SDRL: Interpretable and data-efficient deep reinforcement learning leveraging symbolic planning. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, Honolulu, USA, pp. 2970–2977, 2019. DOI: 10.1609/aaai.v33i01.33012970.

[28] Z. H. Ma, Y. Z. Zhuang, P. Weng, H. H. Zhuo, D. Li, W. L. Liu, J. Y. Hao. Learning symbolic rules for interpretable deep reinforcement learning, [Online], Available: https://arxiv.org/abs/2103.08228, 2021.

[29] Y. C. Yang, J. P. Inala, O. Bastani, Y. W. Pu, A. Solar-Lezama, M. C. Rinard. Program synthesis guided reinforcement learning for partially observed environments. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pp. 29669–29683, 2021.

[30] N. Jiang, A. Krishnamurthy, A. Agarwal, J. Langford, R. E. Schapire. Contextual decision processes with low bellman rank are PAC-learnable. In *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, vol. 70, pp. 1704–1713, 2017. DOI: 10.5555/3305381.3305557.

[31] A. Agarwal, M. Henaff, S. M. Kakade, W. Sun. PC-PG: Policy cover directed exploration for provable policy gradient learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pp. 13399–13412, 2020.

[32] I. Uchendu, T. Xiao, Y. Lu, B. H. Zhu, M. Y. Yan, J. Simon, M. Bennice, C. Y. Fu, C. Ma, J. T. Jiao, S. Levine, K. Hausman. Jump-start reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, Honolulu, USA, pp. 34556–34583, 2023.

[33] J. Queeney, Y. Paschalidis, C. G. Cassandras. Generalized proximal policy optimization with sample reuse. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pp. 11909–11919, 2021.

[34] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. P. Agapiou, J. Schrittwieser, J. Quan, S. Gaffney, S. Petersen, K. Simonyan, T. Schaul, H. Van Hasselt, D. Silver, T. P. Lillicrap, K. Calderone, P. Keet, A. Brunasso, D. Lawrence, A. Ekermo, J. Repp, R. Tsing. Starcraft II: A new challenge for reinforcement learning, [Online], Available: https://arxiv.org/abs/1708.04782, 2017.

[35] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, S. Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, New Orleans, USA, pp. 2974–2982, 2018. DOI: 10.1609/aaai.v32i1.11794.

[36] J. G. Kuba, M. N. Wen, L. H. Meng, S. D. Gu, H. F. Zhang, D. Mguni, J. Wang, Y. D. Yang. Settling the variance of multi-agent policy gradients. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pp. 13458–13470, 2021.

[37] Z. F. Wu, C. Yu, D. H. Ye, J. G. Zhang, H. Y. Piao, H. H. Zhuo. Coordinated proximal policy optimization. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pp. 26437–26448, 2021.

[38] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, K. Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning*, Stockholm, Sweden, pp. 1407–1416, 2018.

[39] W. N. Xia, Y. M. Yang, J. Q. Ruan, D. P. Xing, B. Xu. Cardsformer: Grounding language to learn a generalizable policy in hearthstone. In *Proceedings of the 26th European Conference on Artificial Intelligence*, Kraków, Poland, pp. 2720–2727, 2023. DOI: 10.3233/FAIA230581.

[40] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Q. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, Y. Tassa. DM_Control: Software and tasks for continuous control. *Software Impacts*, vol. 6, Article number 100022, 2020. DOI: 10.1016/j.simpa.2020.100022.

[41] D. A. Levin, Y. Peres. *Markov Chains and Mixing Times*, 2nd ed., Providence Road, USA: American Mathematical Society, pp. 1866–7414, 2017.

[42] S. Schmitt, M. Hessel, K. Simonyan. Off-policy actor-critic with shared experience replay. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 8545–8554, 2020.

Yiming Yang received the B. Sc. degree in electronic information engineering from the Beijing Institute of Technology, China in 2019. Currently, he is a Ph. D. degree candidate at the Institute of Automation, Chinese Academy of Sciences, China.

His research interests include reinforcement learning, robotics, and generative model.

E-mail: yangyiming2019@ia.ac.cn
ORCID iD: 0000-0003-1359-0364

Dengpeng Xing received the B. Sc. degree in mechanical electronics and the M. Sc. degree in mechanical manufacturing and automation from Tianjin University, China in 2002 and 2006, and the Ph. D. degree in control science and engineering from Shanghai Jiao Tong University, China in 2010. He is currently a professor at Institute of Automation, Chinese Academy of Sciences, China.

His research interests include robot control and learning, and decision-making intelligence for complex systems.

E-mail: dengpeng.xing@ia.ac.cn (Corresponding author)
ORCID iD: 0000-0002-8251-9118

Wannian Xia received the B. Sc. degree in electronic information science and technology from Nankai University, China in 2019. Currently, he is a Ph. D. degree candidate at School of Artificial Intelligence, University of Chinese Academy of Sciences, China.

His research interests include reinforcement learning, natural language processing, and game theory.

E-mail: xiawannian2020@ia.ac.cn

Peng Wang received the B. Sc. degree in electrical engineering and automation at the Harbin Institute of Technology, China in 2004, and the Ph. D. degree in control theory and control engineering at the Institute of Automation, Chinese Academy of Sciences, China in 2010. Currently, he is a professor and doctoral advisor at the Institute of Automation, Chinese Academy of Sciences, China.

His research interests include robotics, artificial intelligence, machine learning, and control theory.

E-mail: peng_wang@ia.ac.cn
ORCID iD: 0000-0002-8265-9866

Machine Intelligence Research

Citation: Y. Yang, D. Xing, W. Xia, P. Wang. Guided proximal policy optimization with structured action graph for complex decision–making. *Machine Intelligence Research*, vol.22, no.1, pp.1–20, 2025. https://doi.org/10.1007/s11633–024–1503–7