

---

# Guided Proximal Policy Optimization with Structured Action Graph

---

Anonymous Authors<sup>1</sup>

## Abstract

Modeling and optimizing in the macroscopic action space is imperative when solving complex decision-making tasks with deep reinforcement learning. In this work, we first model the problem as a Markov decision process with a structured action graph. Then, using the structured action graph and combining the guiding policy with the trust region constraints, we derive and prove a lower bound for monotonic policy improvement. Based on the above theories, we propose Guided Proximal Policy Optimization with a Structured Action Graph (GPPO-SAG), a reinforcement learning algorithm with theoretical guarantees for challenging tasks involving high-dimensional hybrid action spaces. Experiment results on the full StarCraft II game and Hearthstone demonstrate that our approach can effectively guide policy exploration and improve performance during reinforcement learning.

## 1. Introduction

The human-like ability of agents to reason and make decisions in complex scenes has attracted the attention of an increasing number of researchers recently (Team et al., 2021)(Ahn et al., 2022)(Srivastava et al., 2022). A potential way to achieve this goal is deep reinforcement learning (DRL) (Levine, 2022), in which the policy gradient method plays a vital role. Compared with the value-based approach, it is better suited for processing in large and continuous action spaces. However, the vanilla policy gradient method (Mnih et al., 2015) is less robust, so researchers have proposed many improvements (Schulman et al., 2015)(Schulman et al., 2017). Among them, the Proximal Policy Optimization (PPO) (Schulman et al., 2017) method has emerged as one of the most popular RL algorithms due to its superior and robust performance, primarily as a result of the solid

theoretical foundation provided by trust region theory. On the other hand, policy learning in high-dimensional hybrid action spaces is a common challenge for RL agents in complex tasks. A good example is real-time strategy (RTS) games, where each step requires the agent to select an action type from a discrete action space and then the parameters of that action. This action parameter may be in the continuous action space. For example, if you choose a soldier to move, then the action type is move, 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 mainstream method decomposes the original action space through autoregressive action selection and transforms the complex decision-making problem into several smaller decision-making problems in the appropriate action spaces (Vinyals et al., 2019)(Berner et al., 2019)(Ye et al., 2020). In these works, each decomposed sub-policy is regarded as an independent model for optimization. But some researchers have proved that simply imposing generic-purpose RL algorithms on such problems may introduce unwanted bias (Eisenach et al., 2018)(Kuba et al., 2022). We claim that the hybrid action space of these strategy games, despite being large and diverse, usually has some internal structures. And the rational use of these internal structures can simplify the complexity of policy modeling and facilitate the learning of the RL agent.

In this paper, we first propose the structured action graph for modeling complex decision problems with macroscopic action spaces. Second, we provide the guiding-based lower bound for policy monotonic improvement, which provides a theoretical guarantee for RL learning with priori policy. Finally, We propose Guided Proximal Policy Optimization with Structured Action Graph (GPPO-SAG) by integrating the guidance-based trust region optimization method into the Markov decision processes with structured action graph. GPPO-SAG performs better both theoretically and practically in complex decision-making tasks. Through experiments on the full StarCraft II game and Hearthstone, we demonstrate that GPPO-SAG outperforms state-of-the-art approaches across the board in terms of performance and sample efficiency. We summarize our contributions as follows:

- We introduce Guided Proximal Policy Optimization

<sup>1</sup>Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

with Structured Action Graph, a practical algorithm based on PPO method. Our algorithm can effectively guide policy exploration and improve performance in challenging tasks with macroscopic action space.

- We formulate autoregressive action selection as the structured action graph, which helps mathematically and quantitatively probe action space decomposition and optimization in RL.
- We propose and prove a monotonic improvement guarantee for the stochastic policy with guiding policy and structured action graph. Our principal theoretical result offers a fresh viewpoint on RL policy optimization.

## 2. Preliminaries and Background

**Markov Decision Processes** A Markov Decision Process (MDP) is defined as a tuple  $(S, \mathcal{A}, r, P, \gamma)$ . The set  $S$  represents environment states, and the set  $\mathcal{A}$  contains the actions that an agent can perform in the environment.  $r : S \times \mathcal{A} \rightarrow \mathbb{R}$  is the reward function set.  $P : S \times \mathcal{A} \times S \rightarrow [0, 1]$  is the set of transitions.  $\gamma \in [0, 1)$  is the discount factor. Formalized by reinforcement learning, the goal of the agent policy  $\pi$  is to maximize the cumulative discounted reward  $J(\pi) = \mathbb{E}_{\tau \sim \pi} [\sum_{j=0}^{\infty} \gamma^j r(s_j, a_j)]$ , where  $\tau = (s_0, a_0, s_1, a_1, \dots)$  is the trajectory induced by  $\pi$ .

**Trust Region Policy Gradients** Since the vanilla policy gradient (Sutton et al., 1999) method is suffered from determining the update step, TRPO (Schulman et al., 2015) is proposed to optimize the policy by solving a constrained optimization problem:

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

where  $\hat{A}_t$  is the estimate of the advantage function at timestep  $t$ , and  $D_{TV}(\cdot)$  means total variation distance. TRPO greatly improves learning stability. Based on that, PPO (Schulman et al., 2017) is proposed to simplify the policy update by optimizing the surrogate objective function:

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

where  $clip(\rho, 1, \epsilon)$  means bounding the value of  $\rho$  between  $[1 - \epsilon, 1 + \epsilon]$ . Furthermore, some researchers extended PPO to heterogeneous multi-agent reinforcement learning, known as HAPPO (Kuba et al., 2022), in which they optimize each agent iteratively by:

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

where  $\rho^{1:i-1} = \frac{\pi_{\theta_{old}}^{1:i-1}(a_t^{1:i-1} | s_t)}{\pi_{\theta_{old}}^{1:i-1}(a_t^{1:i-1} | s_t)}$  represents the effect of all

agents updated before agent  $i$  and  $\rho^i = \frac{\pi_{\theta_{old}}^i(a_t^i | s_t)}{\pi_{\theta_{old}}^i(a_t^i | s_t)}$ . 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  $\pi_b$  and the target policy  $\pi_t$ . According to the “Generalized Policy Improvement Lower Bound” theorem (Queeney et al., 2021), 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  $\pi_b$ , target policy  $\pi_t$ , and future policy  $\hat{\pi}_t$ , we refer to it as follows:

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

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

## 3. Decomposition and optimization for macroscopic action space

### 3.1. MDPs With Structured Action Graph

In RTS games, the action space is usually vast and diverse, which makes it difficult for the agent to choose actions directly. For instance, in SC2LE (Vinyals et al., 2017), an agent must select an action from approximately  $10^8$  valid actions per timestep. To solve this problem, some researchers have proposed to decompose the action space by action structure (Vinyals et al., 2019). Here we model it as MDPs with structured action graph (MDPs-SAG). In MDPs-SAG, the structured action  $\mathcal{A}$  can be represented as a sequence of sub-action sets:  $\mathcal{A} = \mathcal{A}^1 \times \dots \times \mathcal{A}^n$ . 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 joint action at each timestep is a path on  $\mathcal{G}$  from the initial vertex to the end vertex, which can be expressed as a path graph  $\mathcal{P} = < v^{init}, v^k, \dots, v^i, v^{end} >$ . In this framework, for a given state  $s \in S$  and the joint action of ancestor vertices  $a^{an(j)} \in \mathcal{A}^{an(j)}$  of vertex  $v^j \in \mathcal{V}$ , an action  $a^j$  can be selected from the sub-policy  $\pi^j(a^j | s, a^{an(j)})$  associated with vertex  $v^j$ . The joint policy  $\pi(a | s) = \prod_{j=k}^i \pi^j(a^j | s, a^{an(j)})$ , where  $an(j) \in \mathcal{P}$ . Similar to (Foerster et al., 2018), in the MDPs-SAG setting, we can define the state-action value

110 function of the vertex  $v^j$ :

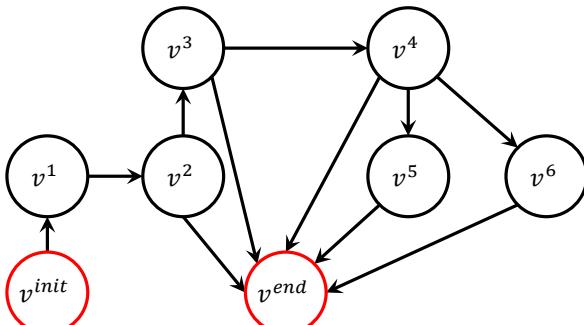
$$Q^{j,an(j)}(s, \mathbf{a}^{j,an(j)}) = \mathbb{E}_{\mathbf{a}^{an(j)} \sim \pi^{an(j)}} [Q^{j,an(j)}(s, a^j, \mathbf{a}^{an(j)})] \quad (5)$$

111 where  $Q(s, a^j, \mathbf{a}^{an(j)})$  represents the state-action value of  
 112 action  $a^j$  when taken with the joint action  $\mathbf{a}^{an(j)} \in \mathcal{A}^{an(j)}$   
 113 of ancestor vertices  $v^{an(j)} \in \mathcal{V}$ . And  $Q(s, \mathbf{a}^{j,an(j)})$  represents the state-action value of the joint action  $\mathbf{a}^{j,an(j)}$ .

114  
 115 Equation 5 means the action value of  $a^j$  when vertex  $v^j$ 's ancestor vertex actions are fixed to  
 116  $\mathbf{a}^{an(j)}$ . Note that  $Q^{init}(s, a^{init}) = V(s)$ , and  
 117  $Q^{end,an(end)}(s, \mathbf{a}^{end,an(end)}) = Q(s, \mathbf{a}^{1,\dots,n})$ . Then,  
 118 the advantage function of vertex  $v^j$  can be defined:

$$A^j(s, \mathbf{a}^{an(j)}, a^j) = Q^{j,an(j)}(s, \mathbf{a}^{j,an(j)}) - Q^{an(j)}(s, \mathbf{a}^{an(j)}) \quad (6)$$

119 We show in Fig. 1 the SAG for SC2LE with partitions from  
 120 AlphaStar (Vinyals et al., 2019). It is important to note that  
 121 different action types may have different paths from  $v^{init}$  to  
 122  $v^{end}$ , and joint actions do not have to traverse all vertices  
 123 on the SAG. For instance, the action “attack units” can be  
 124 represented by a path:  $< v^{init}, v^1, v^2, v^3, v^4, v^5, v^{end} >$ ,  
 125 and the action “attack target position” by a path:  
 126  $< v^{init}, v^1, v^2, v^3, v^4, v^6, v^{end} >$ .



127  
 128  
 129 Figure 1. SAG for SC2LE.  $v^{init}$  and  $v^{end}$  represent the start vertex  
 130 and the end vertex, respectively.  $v^1, \dots, v^6$  represents “action type,”  
 131 “delay,” “queue,” “select units,” “target units,” and “target location”  
 132 respectively, in the partition of AlphaStar (Vinyals et al., 2019).

133  
 134 **Lemma 1.** Given a structured action graph  $\mathcal{G}$ , the following  
 135 equation holds if all sub-actions form a path graph  $\mathcal{P} = < v^{init}, v^k, \dots, v^i >$  in  $\mathcal{G}$

$$A^{\mathcal{P}}(s, \mathbf{a}^{\mathcal{P}}) = \sum_{j=k}^i A^j(s, \mathbf{a}^{j,an(j)}), \text{ where } an(j) \sim \mathcal{P} \quad (7)$$

136 For proof, see Appendix B.1. Lemma 1 states that in MDPs-SAG,  
 137 the state-action advantage of the path containing the  
 138

139 initial vertex can be represented by the sum of the state-  
 140 action advantages of each vertex in the path. Different  
 141 from “Multi-Agent Advantage Decomposition” (Kuba et al.,  
 142 2021)(Kuba et al., 2022), each vertex in Lemma 1 is con-  
 143 strained by graph  $\mathcal{G}$ , which means there is no practical  
 144 meaning for a single vertex when some ancestors are miss-  
 145 ing. Therefore, the arbitrariness of vertex selection required  
 146 by the “Multi-Agent Advantage Decomposition” does not  
 147 hold in our problem.

### 3.2. Guided Proximal Policy Optimization with 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. The guiding policy can come in various forms, such as neural networks trained from human player data or behavior trees. We have low performance requirements for the guiding policy, which serves as a prior manifold (Bush & Pineau, 2009; Pirotta et al., 2015) for agents to explore and learn in the high-dimensional policy space.

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

**Lemma 2.** Given behavior policy  $\pi_b$ , guiding policy  $\pi_g$ , target policy  $\pi_t$ , and future policy  $\hat{\pi}_t$ , we have

$$J(\hat{\pi}_t) - J(\pi_t) \geq \frac{1}{1-\gamma} \mathbb{E}_{\tau \sim \pi_b} \left[ \frac{\hat{\pi}_t(\mathbf{a}^{\mathcal{P}}|s)}{\pi_b(\mathbf{a}^{\mathcal{P}}|s)} \hat{A}_{\pi_t}^{\mathcal{P}}(s, \mathbf{a}^{\mathcal{P}}) \right] - C(D_{TV}(\hat{\pi}_t, \pi_g) + D_{TV}(\pi_b, \pi_g)) \quad (8)$$

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

For proof, see Appendix B.2. In the context of our discussion, the behavior policy  $\pi_b$  represents the policy that interacts with the environment and collects data, while the guiding policy  $\pi_g$  serves as a prior policy that makes decisions concurrently with the behavior policy, but does not interact with the environment. The guiding policy is used to provide guidance during training. The target policy  $\pi_t$  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 shows that we can indirectly keep the policy updated in the trust region by constraining the total variation distance between the behavioral policy and the guiding policy and the total variation distance between the target policy and the guiding policy, respectively. Combining with Lemma 1, we propose a lower bound with SAG on the improvement of the guiding policy.

**Theorem 1.** Given a structured action graph  $\mathcal{G}$ , behavior policy  $\pi_b$ , target policy  $\pi_t$ , guiding policy  $\pi_g$ , and

future policy  $\hat{\pi}_t$ , the following policy improvement lower bound holds if all sub-actions form a path graph  $\mathcal{P} = \langle v^{init}, v^k, \dots, v^i, v^{end} \rangle$  in  $\mathcal{G}$

$$J(\hat{\pi}_t) - J(\pi_t) \geq \sum_{j=k}^i \left[ \frac{1}{1-\gamma} \mathbb{E}_{\tau \sim \pi_b} [\rho^{j,an(j)} \hat{A}_{\pi_t}^\varphi(s, a^\varphi)] - C(D_{TV}(\hat{\pi}_t^j, \pi_g^j) + D_{TV}(\pi_b^j, \pi_g^j)) \right] \quad (9)$$

where  $C = \frac{2\gamma \max_s |\mathbb{E}_{a^j \sim \hat{\pi}_t^j(\cdot, s), a^{an(j)} \sim \hat{\pi}_t^{an(j)}(\cdot, s)} \hat{A}_{\pi_t}^j(s, a^{j,an(j)})|}{(1-\gamma)^2}$ , and  $\rho^{j,an(j)} = \frac{\hat{\pi}_t^j(a^{j,an(j)}|s)}{\pi_b^j(a^{j,an(j)}|s)} \cdot \frac{\hat{\pi}_t^{an(j)}(a^{an(j)}|s)}{\pi_b^{an(j)}(a^{an(j)}|s)}$ .

For proof, see Appendix B.3. In short, we constrain the total variation distance between the behavior policy and the target policy in Lemma 2 via the triangle inequality and the guiding policy. Then Lemma 1 is used to associate the joint policy promotion with the sub-policies of the vertices in the SAG. The constraint on the total variation distance of two joint policies is also scaled down to the sum of the total variation distances between the policies at each vertex on the path graph.

We can conclude the following characteristics of trust region learning with SAG:

- The target policy  $\pi_t$ 's promotion is related to the sub-policies of each vertex  $\pi_t^j$  on the path graph  $\mathcal{P}$  and independent of any other vertex outside  $\mathcal{P}$ .
- The advantage function of each sub-policy  $\pi_t^j$  has a relationship with all of its ancestor vertices  $\pi^{an(j)}$  in addition to itself.
- With the guiding policy  $\pi_g$  as the medium, we can bound the distribution distance between the target policy  $\pi_t$  and the behavior policy  $\pi_b$ .

## 4. Algorithm

Theorem 1 shows how to optimize the joint policy based on SAG, optimizing the sub-policy for each vertex on the path while constraining its distribution distance from the guiding policy. For our proposed surrogate objective function to be solvable by commonly used optimizers, we need a clipped version similar to Equations 2 and 3 for the distributionally constrained problem. However, the problem we face is far more challenging. First, the successive multiplication of importance sampling will lead to variance accumulation, undermining the stability of agent training. Second, not only do we need to clip between the current target policy estimation and the guiding policy, but we also need to restrict the influence of the ancestor policies just like the current policy. To this end, we propose a double-clip version as the final surrogate objective function.

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

$$\sum_{j=k}^i \min \left[ \text{clip}(\rho_t^{an(j)}, \rho_g^{an(j)}, \kappa\epsilon) \rho_t^j \hat{A}_{\pi_t}^\varphi(s, a^\varphi), \text{clip}(\text{clip}(\rho_t^{an(j)}, \rho_g^{an(j)}, \kappa\epsilon) \rho_t^j, \rho_g^j, \epsilon) \hat{A}_{\pi_t}^\varphi(s, a^\varphi) \right] \quad (10)$$

where  $\text{clip}(\rho_1, \rho_2, \epsilon)$  means bounding  $\rho_1$  between  $[\rho_2 - \epsilon, \rho_2 + \epsilon]$  and  $\kappa \in [0, +\infty)$  is a constant scaling factor.

For the derivation, see Appendix B.4. Since the successive multiplication of importance sampling will increase the variance of advantage estimates (Wu et al., 2021)(Kuba et al., 2021), we need the function  $\text{clip}(\rho^{an(j)}, \rho_g^{an(j)}, \kappa\epsilon)$  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. It thus approximately limits the total variation distance between the guiding policy  $\pi_g$  and the target policy estimates  $\hat{\pi}_t$ . 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. Note that in our algorithm, the form of guiding policy is not limited to a neural network as long as it can provide guidance synchronously with the target policy's learning process.

We refer to Equation 10 as Guided Proximal Policy Optimization with Structured Action Graph (GPPO-SAG) and elaborate on it in Algorithm 1 in Appendix D.

It should be noted that if there is no prior policy available, the problem is reduced to “Proximal Policy Optimization with Structured Action Graph”. For that, we can assume that the guiding policy is the behavior policy itself, namely  $\rho_g^{an(j)} = 1, \rho_g^j = 1$  in Equation 10, and the algorithm can also 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 with macroscopic action space? Where does the GPPO-SAG advantage come from?
2. Does the complexity of the macroscopic action space affect the advantage of GPPO-SAG over other methods?

- 220 3. How sensitive is GPPO-SAG to the clipping ratio?  
 221 How does the constant scaling factor  $\kappa$  affect the per-  
 222 formance of the algorithm?

223  
 224 To answer these questions, we compare the performance of  
 225 GPPO-SAG with AlphaStar (we use DI-star ([star Contributors, 2021](#)), an open-source implementation of AlphaStar, as  
 226 our code base), and PPO ([Schulman et al., 2017](#)) with additional  
 227 KL penalty between the target policy and the guiding  
 228 policy in the StarCraft II full game and Hearthstone. All  
 229 methods in our experiments are implemented based on IM-  
 230 PALA ([Espeholt et al., 2018b](#)). The main objective function  
 231 of AlphaStar is the penalized policy gradient as follows:

$$234 \sum_{j=k}^i \log \pi_t^j \cdot \hat{A}_{\pi_t}^{\varphi}(s, a^{\varphi}) - D_{KL}(\pi_g^j, \pi_t^j) \quad (11)$$

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

$$237 \sum_{j=k}^i \min \left[ \frac{\pi_t^j}{\pi_b^j} \hat{A}_{\pi_t}^{\varphi}(s, a^{\varphi}), \right. \\ \left. \text{clip} \left( \frac{\pi_t^j}{\pi_b^j}, 1, \epsilon \right) \hat{A}_{\pi_t}^{\varphi}(s, a^{\varphi}) \right] - D_{KL}(\pi_g^j, \pi_t^j) \quad (12)$$

238 where all symbolic markers have the same meaning as in  
 239 Proposition 10.

240 We implement experiments on a platform with 4 NVIDIA-  
 241 A100 GPUs and 256 AMD EPYC 7742 CPU cores<sup>1</sup>.

### 242 5.1. Environment description and model configuration

243 **StarCraft II** We only consider the zerg civil war in our  
 244 experiments to save computer resources. We first use 9000  
 245 zerg-vs-zerg human replays to train a policy network with  
 246 71000 epochs using supervised learning paradigm. Then the  
 247 supervised model is used for initializing the agent's policy  
 248 model for RL with the level-3 built-in bot. During RL  
 249 training, the first 4000 steps are used for value pre-training,  
 250 after which the actor networks' gradients are enabled. The  
 251 clip ratio for each head is listed in table 1. In the following  
 252 experiments, unless otherwise noted, the default values are  
 253 used. More hyperparameters for supervised learning are  
 254 listed in Appendix E.2

255 For a fair comparison, all methods adopt AlphaStar's  
 256 ([Vinyals et al., 2019](#)) network structure, UPGO loss, and  
 257 entropy loss. We set the learning rate to  $10^{-5}$ , the batch size  
 258 to 8, and the trajectory length for the LSTM to 32. More  
 259 hyperparameters for reinforcement learning are listed in  
 260 Appendix E.3

261 <sup>1</sup>The source code will be released after we finish organizing it.

262 *Table 1.* Clip ratio with GPPO-SAG

Action head	$\epsilon_{def}$	$\kappa_{def}$
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

263 **Hearthstone** Hearthstone is a trading card game that allows  
 264 players to build their deck of 30 cards before a match and  
 265 defeat their opponents in two-player battles. Hearthstone's  
 266 cards are more diverse than traditional board games, and  
 267 almost every card has its unique effect.

268 Figure 2(a) shows two example cards, “Abusive Sergeant”  
 269 and Fireball. “Mana spent” represents the resource con-  
 270 sumption of this card, and the player’s resource per turn is  
 271 limited. If the card is a minion card, it has attack and health,  
 272 and when played, it will be placed on our field. “Special  
 273 effect” represents the special effect that this card has. Ac-  
 274 cording to different effects, some cards are non-directional,  
 275 some are directional, and some are selective. “Abusive  
 276 Sergeant” is a minion card with a directionality battle cry,  
 277 which means we need three steps to play this card: select  
 278 this card, then select a target position to place, and finally  
 279 select a target unit to make the battle cry take effect. The  
 280 card “Fireball” is a directional spell card, which means we  
 281 need to use it by step: select this card and then a target  
 282 unit. Therefore, similar to StarCraft II, the action space of  
 283 Hearthstone’s agent is also well suited to be modeled using  
 284 the structured action graph.

285 We represent the Hearthstone action space as a structured  
 286 action graph, as shown in Figure 2(b). In our experiments,  
 287 we use one Warlock deck and optimize the policy by self-  
 288 play. We first use 5000 replays to train the guiding policy  
 289 and adopt an autoregressive policy network structure. We  
 290 set the learning rate to  $10^{-5}$ , the batch size to 32, and the  
 291 clipping ratio to 1.2. The details about the deck, network  
 292 structure, and hyperparameters can be seen in Appendix E.4

### 293 5.2. Experiments for question 1

294 Figure 3 shows the training curves of GPPO-SAG, AlphaStar,  
 295 and PPO-KL against the built-in Level 3 AI on three  
 296 standard matchmaking maps, where the red curve repre-  
 297 sents GPPO-SAG, the green curve represents the AlphaStar  
 298 with constrained policy gradients, and the blue curve repre-  
 299 sents the PPO algorithm with a KL constraint. During the  
 300 early training steps, the win rate of each method steadily  
 301 increases, but at around  $2 \times 10^6$  game steps, AlphaStar  
 302 and PPO-KL hit a bottleneck, and their performance stops

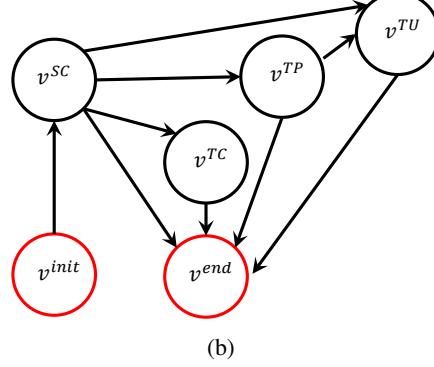


Figure 2. (a) Example cards in Hearthstone, “Abusive Sergeant” is a minion card, and “Fireball” is a spell card; (b) The structured action graph of Hearthstone.  $v^{init}$  means the initial vertex,  $v^{SC}$  means select card vertex,  $v^{TC}$  means select a target card vertex,  $v^{TP}$  means select a target position vertex,  $v^{TU}$  means select a target unit vertex, and  $v^{end}$  means the end vertex.

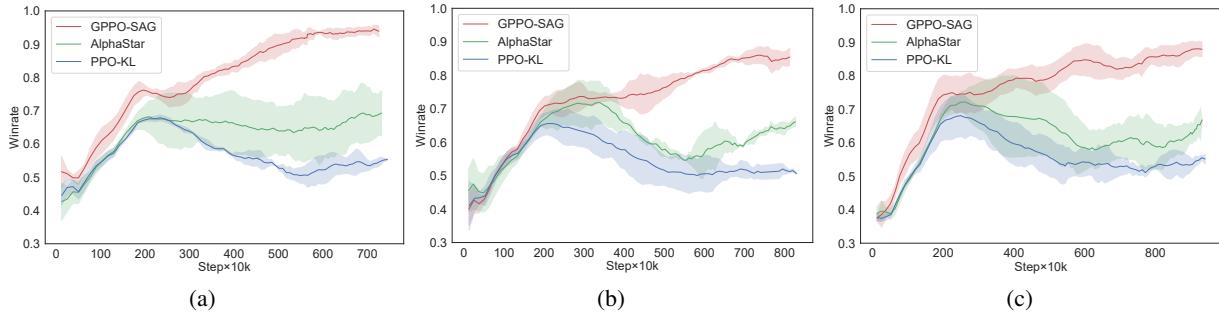


Figure 3. The win rate curve for each method against level 3 AI on StarCraft II full game. (a) Win rate curve on map KairosJunction; (b) Win rate curve on map KingsCove; (c) Win rate curve on map NewRepugnancy.

improving. Our method is unaffected, and its win rate continues to rise steadily, eventually reaching about 90% at around  $7 \times 10^6$  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. The 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  $\pi_t$  and the guiding policy  $\pi_g$  during training on the KairosJunction map. The results shown in Figure 4 demonstrate that our method can constrain the distribution distance between the target policy  $\pi_t$  and the guiding policy  $\pi_g$  to be stable within a certain range in almost all action heads. In contrast, the distribution distance between  $\pi_t$  and  $\pi_g$  is not stably bounded and grows as training progresses.

Our method constrains the distribution of vertices for each sub-action equally. But 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  $\mathcal{P}$ . As shown in Figs. 4(a), 4(b) and 4(c), these vertices are near  $v^{init}$  in the path graph. Among these vertices, our method’s  $D_{KL}(\pi_t, \pi_g)$  is about 3-4 times smaller than other methods after convergence. But as shown in Figures 4(d), 4(e) and 4(f), the gap between AlphaStar, PPO-KL, and our method shrinks to 1-3 times. We claim that it is because these vertices are usually located at the end of the path graph and are constrained more tightly in other methods. This 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  $\pi_t$  and  $\pi_g$ . This means  $\pi_g$  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.

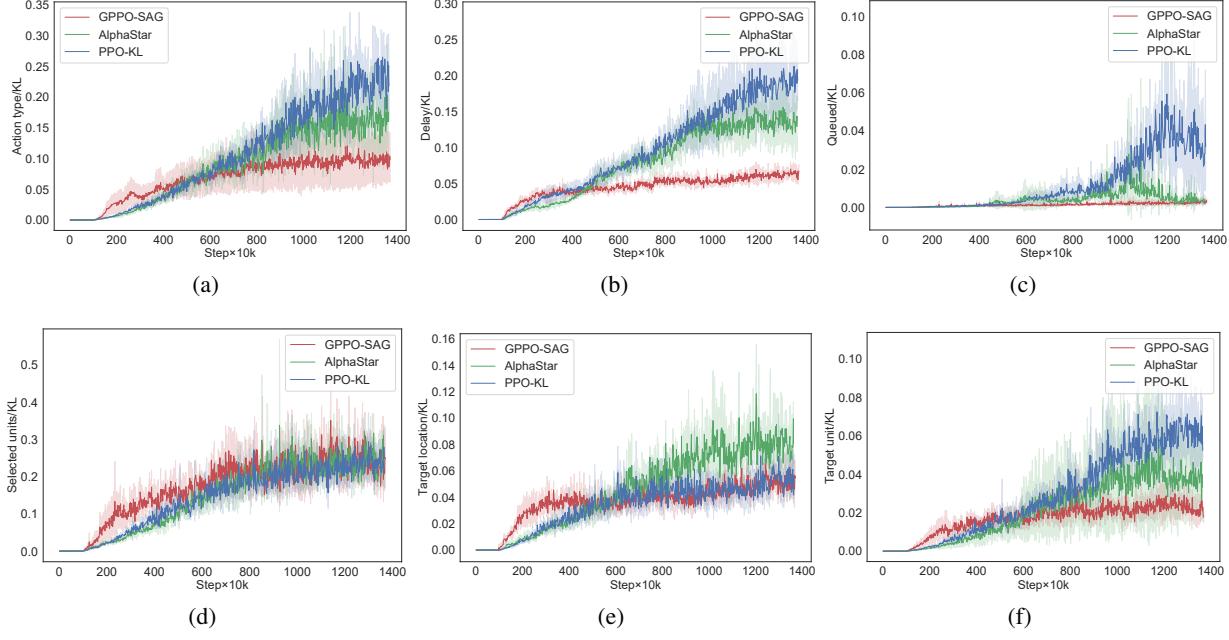


Figure 4. The KL divergence curve between the target policy  $\pi_t$  and the guiding policy  $\pi_g$  during training for each method on the StarCraft II’s KairouJunction map. (a) Action type vertex; (b) Delay vertex; (c) Queued vertex; (d) Selected units vertex; (e) Target location vertex; (f) Target unit vertex.

The distribution between the target policy  $\pi_t$  and the guiding policy  $\pi_g$  is not constrained by our method before they activate the clipping mechanism, which allows GPOO-SAG to achieve more accessible exploration in the vicinity of the guiding policy space. As a result, combining Figure 3 and Figure 4, we can see that during the early training steps, the agent’s win rate of GPOO-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 GPOO-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 discovered through the experiments in 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 (Figure 1), Hearthstone’s (Figure 2(b)) 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. To this end, we used an autoregressive action selection 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  $\pi_t$  and the guiding policy  $\pi_g$  during training. Three random seeds were used to average each of the curves displayed. We replace the Starcraft II comparison method AlphaStar with PG-KL because we skipped 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 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

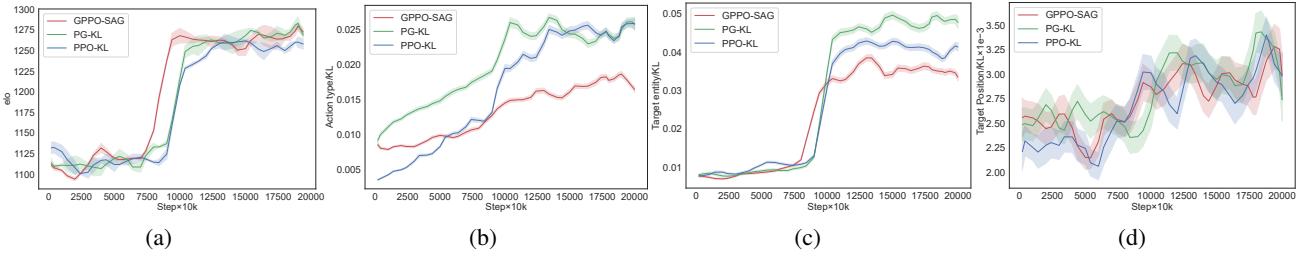


Figure 5. The training and KL divergence curve between the target policy  $\pi_t$  and the guiding policy  $\pi_g$  in Hearthstone. (a) The Elo rating scores for different training stages in Hearthstone; (b) Action type vertex; (c) Target entity vertex; (d) Target position vertex.

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  $\epsilon$  and  $\kappa$  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
$0.25\epsilon_{def}$	$59.2\% \pm 1.5\%$
$0.5\epsilon_{def}$	$68.4\% \pm 2.1\%$
$\epsilon_{def}$	$84.2\% \pm 3.1\%$
$2\epsilon_{def}$	$83.1\% \pm 1.5\%$
$4\epsilon_{def}$	$71.0\% \pm 3.9\%$

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.

As shown in Figure 6, we tested how different  $\kappa$  affected the performance of GPPO-SAG. The algorithm performs optimally at  $\kappa = \frac{1}{2}$ , while performance is impacted by  $\kappa$  values that are either too little or too large. When  $\kappa$  is too small, clipping continues to activate as training proceeds,

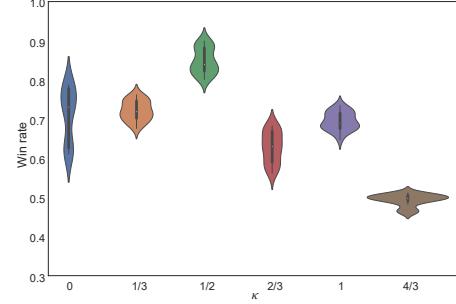


Figure 6. Effect of the relative magnitude of inner and outer clipping ratio on performance. The horizontal axis represents the different values of  $\kappa$ . The vertical axis represents the win rate. Each set was obtained by averaging the win rates of the last 500 games during training in three seeds.

and the ancestral importance sampling ratio of  $\pi_t$  has a limited impact on the current sub-action vertex. When  $\kappa$  is too large, the ancestral importance sampling ratio will overly influence the current sub-action vertex, preventing proper prioritization and impairing performance.

## 6. Conclusions

In this paper, we propose a guided proximal policy optimization algorithm with structured action graph to solve the reinforcement learning problems in the macroscopic action space. By formulating the environment with macroscopic action space as MDPs-SAG, and proposing and proving the policy improvement lower bound with guiding policy, we demonstrate the theoretical basis of GPPO-SAG. Experiments on StarCraft II and Hearthstone validate the superior performance of our approach. We further corroborated 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.

---

**References**

- 440  
441 Agarwal, A., Henaff, M., Kakade, S., and Sun, W. Pcp-  
442 pg: Policy cover directed exploration for provable policy  
443 gradient learning. *Advances in neural information pro-*  
444 *cessing systems*, 33:13399–13412, 2020.  
445
- 446 Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O.,  
447 David, B., Finn, C., Gopalakrishnan, K., Hausman, K.,  
448 Herzog, A., et al. Do as i can, not as i say: Grounding  
449 language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.  
450
- 451 Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P.,  
452 Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse,  
453 C., et al. Dota 2 with large scale deep reinforcement  
454 learning. *arXiv preprint arXiv:1912.06680*, 2019.  
455
- 456 Bush, K. and Pineau, J. Manifold embeddings for model-  
457 based reinforcement learning under partial observability.  
458 *Advances in neural information processing systems*, 22,  
459 2009.
- 460 Eisenach, C., Yang, H., Liu, J., and Liu, H. Marginal policy  
461 gradients: A unified family of estimators for bounded  
462 action spaces with applications. In *International Confer-*  
463 *ence on Learning Representations*, 2018.
- 464 Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V.,  
465 Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I.,  
466 Legg, S., and Kavukcuoglu, K. IMPALA: Scalable dis-  
467 tributed deep-RL with importance weighted actor-learner  
468 architectures. In Dy, J. and Krause, A. (eds.), *Proceed-  
469 ings of the 35th International Conference on Machine  
470 Learning*, volume 80 of *Proceedings of Machine Learn-  
471 ing Research*, pp. 1407–1416. PMLR, 10–15 Jul 2018a.  
472
- 473 Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih,  
474 V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning,  
475 I., et al. Impala: Scalable distributed deep-rl with im-  
476 portance weighted actor-learner architectures. In *Inter-  
477 national conference on machine learning*, pp. 1407–1416.  
478 PMLR, 2018b.
- 479 Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and  
480 Whiteson, S. Counterfactual multi-agent policy gradi-  
481 ents. In *Proceedings of the AAAI conference on artificial  
482 intelligence*, volume 32, 2018.
- 483 Jiang, N., Krishnamurthy, A., Agarwal, A., Langford, J.,  
484 and Schapire, R. E. Contextual decision processes with  
485 low bellman rank are pac-learnable. In *International Con-  
486 ference on Machine Learning*, pp. 1704–1713. PMLR,  
487 2017.
- 488 Jothimurugan, K., Bansal, S., Bastani, O., and Alur, R.  
489 Compositional reinforcement learning from logical spec-  
490 ifications. *Advances in Neural Information Processing Systems*, 34:10026–10039, 2021.  
491
- 492 Kuba, J. G., Wen, M., Meng, L., Zhang, H., Mguni, D.,  
493 Wang, J., Yang, Y., et al. Settling the variance of multi-  
494 agent policy gradients. *Advances in Neural Information  
495 Processing Systems*, 34:13458–13470, 2021.  
496
- 497 Kuba, J. G., Chen, R., Wen, M., Wen, Y., Sun, F., Wang, J.,  
498 and Yang, Y. Trust region policy optimisation in multi-  
499 agent reinforcement learning. In *International Confer-  
500 ence on Learning Representations*, 2022.
- 501 Levine, S. Understanding the world through action. In  
502 *Conference on Robot Learning*, pp. 1752–1757, 2022.
- 503 Lyu, D., Yang, F., Liu, B., and Gustafson, S. Sdrl: Inter-  
504 pretative and data-efficient deep reinforcement learning  
505 leveraging symbolic planning. In *Proceedings of the  
506 AAAI Conference on Artificial Intelligence*, volume 33,  
507 pp. 2970–2977, 2019.
- 508 Ma, Z., Zhuang, Y., Weng, P., Zhuo, H. H., Li, D., Liu, W.,  
509 and Hao, J. Learning symbolic rules for interpretable deep  
510 reinforcement learning. *arXiv preprint arXiv:2103.08228*,  
511 2021.
- 512 Metz, L., Ibarz, J., Jaitly, N., and Davidson, J. Discrete  
513 sequential prediction of continuous actions for deep rl.  
514 *arXiv preprint arXiv:1705.05035*, 2017.
- 515 Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Ve-  
516 ness, J., Bellemare, M. G., Graves, A., Riedmiller, M.,  
517 Fidjeland, A. K., Ostrovski, G., et al. Human-level con-  
518 trol through deep reinforcement learning. *nature*, 518:  
519 529–533, 2015.
- 520 Pang, Z.-J., Liu, R.-Z., Meng, Z.-Y., Zhang, Y., Yu, Y., and  
521 Lu, T. On reinforcement learning for full-length game  
522 of starcraft. In *Proceedings of the AAAI conference on  
523 artificial intelligence*, volume 33, pp. 4691–4698, 2019.
- 524 Pirotta, M., Parisi, S., and Restelli, M. Multi-objective  
525 reinforcement learning with continuous pareto frontier  
526 approximation. In *Proceedings of the AAAI conference  
527 on artificial intelligence*, volume 29, 2015.
- 528 Queeney, J., Paschalidis, Y., and Cassandras, C. G. Gen-  
529 eralized proximal policy optimization with sample reuse.  
530 *Advances in Neural Information Processing Systems*, 34:  
531 11909–11919, 2021.
- 532 Schmitt, S., Hessel, M., and Simonyan, K. Off-policy actor-  
533 critic with shared experience replay. In *International Con-  
534 ference on Machine Learning*, pp. 8545–8554. PMLR,  
535 2020.
- 536 Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz,  
537 P. Trust region policy optimization. In *International  
538 conference on machine learning*, pp. 1889–1897, 2015.

- 495 Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and  
 496 Klimov, O. Proximal policy optimization algorithms.  
 497 *arXiv preprint arXiv:1707.06347*, 2017.
- 498
- 499 Srivastava, S., Li, C., Lingelbach, M., Martín-Martín, R.,  
 500 Xia, F., Vainio, K. E., Lian, Z., Gokmen, C., Buch, S., Liu,  
 501 K., et al. Behavior: Benchmark for everyday household  
 502 activities in virtual, interactive, and ecological environ-  
 503 ments. In *Conference on Robot Learning*, pp. 477–490,  
 504 2022.
- 505
- 506 star Contributors, D. Di-star: An open-source reinforcement  
 507 learning framework for starcraftii. <https://github.com/opendilab/DI-star>, 2021.
- 508
- 509 Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y.  
 510 Policy gradient methods for reinforcement learning with  
 511 function approximation. *Advances in neural information*  
 512 *processing systems*, 12, 1999.
- 513
- 514 Team, O. E. L., Stooke, A., Mahajan, A., Barros, C., Deck,  
 515 C., Bauer, J., Sygnowski, J., Trebacz, M., Jaderberg, M.,  
 516 Mathieu, M., et al. Open-ended learning leads to gener-  
 517 ally capable agents. *arXiv preprint arXiv:2107.12808*,  
 518 2021.
- 519
- 520 Tessler, C., Givony, S., Zahavy, T., Mankowitz, D., and Man-  
 521 nor, S. A deep hierarchical approach to lifelong learning  
 522 in minecraft. In *Proceedings of the AAAI Conference on*  
 523 *Artificial Intelligence*, volume 31, 2017.
- 524
- 525 Tunyasuvunakool, S., Muldal, A., Doron, Y., Liu, S., Bohez,  
 526 S., Merel, J., Erez, T., Lillicrap, T., Heess, N., and Tassa,  
 527 Y. dm\_control: Software and tasks for continuous control.  
 528 *Software Impacts*, 6:100022, 2020.
- 529
- 530 Uchendu, I., Xiao, T., Lu, Y., Zhu, B., Yan, M., Simon, J.,  
 531 Bennice, M., Fu, C., Ma, C., Jiao, J., et al. Jump-start  
 532 reinforcement learning. *arXiv preprint arXiv:2204.02372*,  
 533 2022.
- 534
- 535 Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhn-  
 536 evets, A. S., Yeo, M., Makhzani, A., Küttler, H., Agapiou,  
 537 J., Schrittwieser, J., et al. Starcraft ii: A new challenge for  
 538 reinforcement learning. *arXiv preprint arXiv:1708.04782*,  
 539 2017.
- 540
- 541 Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M.,  
 542 Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds,  
 543 T., Georgiev, P., et al. Grandmaster level in starcraft ii  
 544 using multi-agent reinforcement learning. *Nature*, 575  
 545 (7782):350–354, 2019.
- 546
- 547 Wu, B. Hierarchical macro strategy model for moba game  
 548 ai. In *Proceedings of the AAAI Conference on Artificial*  
 549 *Intelligence*, volume 33, pp. 1206–1213, 2019.
- Wu, Z., Yu, C., Ye, D., Zhang, J., Zhuo, H. H., et al. Coordin-  
 ated proximal policy optimization. *Advances in Neural*  
*Information Processing Systems*, 34:26437–26448, 2021.
- Yang, Y., Inala, J. P., Bastani, O., Pu, Y., Solar-Lezama, A.,  
 and Rinard, M. Program synthesis guided reinforcement  
 learning for partially observed environments. *Advances*  
*in Neural Information Processing Systems*, 34:29669–  
 29683, 2021.
- Ye, D., Liu, Z., Sun, M., Shi, B., Zhao, P., Wu, H., Yu, H.,  
 Yang, S., Wu, X., Guo, Q., et al. Mastering complex  
 control in moba games with deep reinforcement learn-  
 ing. In *Proceedings of the AAAI Conference on Artificial*  
*Intelligence*, volume 34, pp. 6672–6679, 2020.

## A. Related Works

Complex decision-making tasks often involve high-dimensional hybrid action spaces, which pose challenges for action selection and policy exploration for RL agents.

For the action selection problem, some researchers propose to compile a commonly used macro-action space from human data to replace the original action space, and then use HRL or graph to learn the policy schedule (Pang et al., 2019)(Jothimurugan et al., 2021). However, while this approach simplifies the action space, it also limits the possibility for the agent to discover new macro-action combinations. Some work attempts to learn macro actions in a hierarchical way (Tessler et al., 2017)(Wu, 2019). 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 (Metz et al., 2017) works better when dealing with high-dimensional mixed 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 (Vinyals et al., 2019)(Berner et al., 2019)(Ye et al., 2020). 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.

In complex decision-making problems, the policy space is also tremendous, making random policy exploration infeasible. To simplify policy exploration, some researchers construct a simpler abstract policy space by artificially predefining entities, predicates, and causal relations between them in the environment (Lyu et al., 2019)(Ma et al., 2021)(Yang et al., 2021). 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 through a priori policy (Jiang et al., 2017)(Vinyals et al., 2019)(Agarwal et al., 2020)(Uchendu et al., 2022). However, these approaches either lack theoretical assurance or have limitations on the form of the guidance policy. Therefore, there is a lack of a guiding policy exploration method general to complex decision problems with macro-action spaces.

## B. Mathematical Proofs

### B.1. Proof for Lemma 1

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

$$A^{\mathcal{P}}(s, \mathbf{a}^{\mathcal{P}}) = \sum_{j=k}^i A^j(s, \mathbf{a}^{j, an(j)}), \text{ where } an(j) \sim \mathcal{P}$$

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

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

Then we use multi-agent advantage decomposition (Kuba et al., 2021) to the right-hand side.

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

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

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

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

□

**B.2. Proof for Lemma 2**

Given behavior policy  $\pi_b$ , guiding policy  $\pi_g$ , target policy  $\pi_t$ , and future policy  $\hat{\pi}_t$ , we have

$$J(\hat{\pi}_t) - J(\pi_t) \geq \frac{1}{1-\gamma} \mathbb{E}_{\tau \sim \pi_b} \left[ \frac{\hat{\pi}_t(\mathbf{a}^\varphi | s)}{\pi_b(\mathbf{a}^\varphi | s)} \hat{A}_{\pi_t}^\varphi(s, \mathbf{a}^\varphi) \right] - C(D_{TV}(\hat{\pi}_t, \pi_g) + D_{TV}(\pi_b, \pi_g))$$

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

*Proof.* From paper (Queeney et al., 2021), we have

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

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

Since  $D_{TV}$  as a metric satisfies the triangle inequality.

$$\text{So } D_{TV}(\hat{\pi}_t, \pi_g) + D_{TV}(\pi_b, \pi_g) \geq D_{TV}(\hat{\pi}_t, \pi_b)$$

Then we have

$$\begin{aligned} J(\hat{\pi}_t) - J(\pi_t) &\geq \frac{1}{1-\gamma} \mathbb{E}_{\tau \sim \pi_b} \left[ \frac{\hat{\pi}_t(\mathbf{a} | s)}{\pi_b(\mathbf{a} | s)} \hat{A}_{\pi_t}(s, \mathbf{a}) \right] - C D_{TV}(\hat{\pi}_t, \pi_b) \\ &\geq \frac{1}{1-\gamma} \mathbb{E}_{\tau \sim \pi_b} \left[ \frac{\hat{\pi}_t(\mathbf{a}^\varphi | s)}{\pi_b(\mathbf{a}^\varphi | s)} \hat{A}_{\pi_t}^\varphi(s, \mathbf{a}^\varphi) \right] - C(D_{TV}(\hat{\pi}_t, \pi_g) + D_{TV}(\pi_b, \pi_g)) \end{aligned}$$

□

**B.3. Proof for Theorem 1**

**Theorem 1.** Given a structured action graph  $\mathcal{Q}$ , behavior policy  $\pi_b$ , target policy  $\pi_t$ , guiding policy  $\pi_g$ , and future policy  $\hat{\pi}_t$ . The following policy improvement lower bound holds if all sub-actions could form a path graph  $\mathcal{P} = \langle v^{init}, v^k, \dots, v^i, v^{end} \rangle$  in  $\mathcal{Q}$

$$J(\hat{\pi}_t) - J(\pi_t) \geq \sum_{j=k}^i \left[ \frac{1}{1-\gamma} \mathbb{E}_{\tau \sim \pi_b} [\rho^{j,an(j)} \hat{A}_{\pi_t}^\varphi(s, \mathbf{a}^\varphi)] - C(D_{TV}(\hat{\pi}_t^j, \pi_g^j) + D_{TV}(\pi_b^j, \pi_g^j)) \right]$$

$$\text{where } C = \frac{2\gamma \max_s |\mathbb{E}_{a^j \sim \hat{\pi}_t^j(\cdot, s), a^{an(j)} \sim \pi_t^{an(j)}(\cdot, s)} \hat{A}_{\pi_t}^j(s, \mathbf{a}^{j,an(j)})|}{(1-\gamma)^2}, \text{ and } \rho^{j,an(j)} = \frac{\hat{\pi}_t^j(\mathbf{a}^{j,an(j)} | s)}{\pi_b^j(\mathbf{a}^{j,an(j)} | s)} \cdot \frac{\hat{\pi}_t^{an(j)}(\mathbf{a}^{an(j)} | s)}{\pi_b^{an(j)}(\mathbf{a}^{an(j)} | s)}.$$

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

From Lemma 1, we have

$$\begin{aligned} \mathbb{E}_{\tau \sim \pi_b} \left[ \frac{\hat{\pi}_t(\mathbf{a}^\varphi | s)}{\pi_b(\mathbf{a}^\varphi | s)} \hat{A}_{\pi_t}^\varphi(s, \mathbf{a}^\varphi) \right] &= \mathbb{E}_{\tau \sim \pi_b} \left[ \sum_{j=k}^i \frac{\hat{\pi}_t(\mathbf{a}^\varphi | s)}{\pi_b(\mathbf{a}^\varphi | s)} \hat{A}_{\pi_t}^j(s, \mathbf{a}^{j,an(j)}) \right] \\ &= \mathbb{E}_{\tau \sim \pi_b} \left[ \sum_{j=k}^i \frac{\hat{\pi}_t^j(\mathbf{a}^{j,an(j)} | s)}{\pi_b^j(\mathbf{a}^{j,an(j)} | s)} \cdot \frac{\hat{\pi}_t^{an(j)}(\mathbf{a}^{an(j)} | s)}{\pi_b^{an(j)}(\mathbf{a}^{an(j)} | s)} \hat{A}_{\pi_t}^j(s, \mathbf{a}^{j,an(j)}) \right] \\ &= \sum_{j=k}^i \mathbb{E}_{\tau \sim \pi_b} \left[ \frac{\hat{\pi}_t^j(\mathbf{a}^{j,an(j)} | s)}{\pi_b^j(\mathbf{a}^{j,an(j)} | s)} \cdot \frac{\hat{\pi}_t^{an(j)}(\mathbf{a}^{an(j)} | s)}{\pi_b^{an(j)}(\mathbf{a}^{an(j)} | s)} \hat{A}_{\pi_t}^j(s, \mathbf{a}^{j,an(j)}) \right] \end{aligned}$$

**Step 2** We need to prove

$$D_{TV}(\pi_b, \pi_g) \leq \sum_{j=k}^i D_{TV}(\pi_b^j, \pi_g^j) \tag{13}$$

660 That is

$$661 \quad 662 \quad 663 \quad \frac{1}{2} \sum_{a \in \mathcal{A}^{\rho}} \left| \prod_{j=k}^i \pi_b^j(a^{an(j)}|s) - \prod_{j=k}^i \pi_g^j(s, a^{an(j)}) \right| \leq \frac{1}{2} \sum_{j=k}^i \sum_{a \in \mathcal{A}^{\rho}} |\pi_b^j(a^{an(j)}|s) - \pi_g^j(s, a^{an(j)})|$$

664 Next, we prove a stronger version of the above inequation, which is

$$665 \quad 666 \quad 667 \quad 668 \quad \left| \prod_{j=k}^i \pi_b^j(s, a^{an(j)}) - \prod_{j=k}^i \pi_g^j(s, a^{an(j)}) \right| \leq \sum_{j=k}^i |\pi_b^j(s, a^{an(j)}) - \pi_g^j(s, a^{an(j)})| \quad (14)$$

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

$$670 \quad 671 \quad 672 \quad 673 \quad 674 \quad \begin{aligned} & |\pi_b^k(s, a^{an(k)}) \cdot \pi_b^{k+1}(s, a^{an(k)}) - \pi_g^k(s, a^{an(k)}) \cdot \pi_g^{k+1}(s, a^{an(k)})| \\ &= \pi_b^k(s, a^{an(k)}) \cdot |\pi_b^{k+1}(s, a^{an(k)}) - \pi_g^{k+1}(s, a^{an(k)})| + \pi_g^{k+1}(s, a^{an(k)}) \cdot |\pi_b^k(s, a^{an(k)}) - \pi_g^k(s, a^{an(k)})| \\ &\leq |\pi_b^k(s, a^{an(k)}) - \pi_g^k(s, a^{an(k)})| + |\pi_b^{k+1}(s, a^{an(k)}) - \pi_g^{k+1}(s, a^{an(k)})| \end{aligned}$$

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

$$676 \quad 677 \quad 678 \quad 679 \quad \begin{aligned} & \left| \prod_{j=k}^{i+1} \pi_b^j(s, a^{an(j)}) - \prod_{j=k}^{i+1} \pi_g^j(s, a^{an(j)}) \right| = \pi_b^{i+1}(s, a^{an(i+1)}) \cdot \left| \prod_{j=k}^i \pi_b^j(s, a^{an(j)}) - \prod_{j=k}^i \pi_g^j(s, a^{an(j)}) \right| \\ & \quad + \prod_{j=k}^i \pi_g^j(s, a^{an(j)}) \cdot |\pi_b^{i+1}(s, a^{an(i+1)}) - \pi_g^{i+1}(s, a^{an(i+1)})| \\ &\leq \sum_{j=k}^{i+1} |\pi_b^j(s, a^{an(j)}) - \pi_g^j(s, a^{an(j)})| \end{aligned}$$

680 From the discussion above:

- 681 i) We've shown that formula 14 is true for any  $k \in \mathbb{N}^*$  and  $i = k + 1$ .
  - 682 ii) Given that the formula 14 applies to any  $i \in \mathbb{N}^*$  and  $i > k$ , then we've shown that the formula also applies to  $i + 1$ .
- 683 Through mathematical induction, we can confirm that the formula 14 is true.

684 So the formula 13 is true.

685

686 The same goes for the proof of the formula

$$687 \quad 688 \quad 689 \quad 690 \quad 691 \quad D_{TV}(\hat{\pi}_t, \pi_g) \leq \sum_{j=k}^i D_{TV}(\hat{\pi}_t^j, \pi_g^j).$$

692 From the two steps above, we have

$$693 \quad 694 \quad 695 \quad 696 \quad 697 \quad J(\hat{\pi}_t) - J(\pi_t) \geq \sum_{j=k}^i \left[ \frac{1}{1-\gamma} \mathbb{E}_{\tau \sim \pi_b} [\rho^{j,an(j)} \hat{A}_{\pi_t}^{\rho}(s, a^{\rho})] - C(D_{TV}(\hat{\pi}_t^j, \pi_g^j) + D_{TV}(\pi_b^j, \pi_g^j)) \right]$$

$$698 \quad 699 \quad 700 \quad 701 \quad 702 \quad 703 \quad \text{where } C = \frac{2\gamma \max_s |\mathbb{E}_{a^j \sim \hat{\pi}_t^j(\cdot, s), a^{an(j)} \sim \hat{\pi}_t^{an(j)}(\cdot, s)} \hat{A}_{\pi_t}^j(s, a^{j,an(j)})|}{(1-\gamma)^2}, \text{ and } \rho^{j,an(j)} = \frac{\hat{\pi}_t^j(a^{j,an(j)}|s)}{\pi_b^j(a^{j,an(j)}|s)} \cdot \frac{\hat{\pi}_t^{an(j)}(a^{an(j)}|s)}{\pi_b^{an(j)}(a^{an(j)}|s)}. \quad \square$$

#### B.4. The relation between Theorem 1 and the surrogate objective function (Equation 10)

704 If we specify  $\rho_t^{an(j)} = \frac{\hat{\pi}_t^{an(j)}(a^{an(j)}|s)}{\pi_b^{an(j)}(a^{an(j)}|s)}$ ,  $\rho_g^{an(j)} = \frac{\pi_g^{an(j)}(a^{an(j)}|s)}{\pi_b^{an(j)}(a^{an(j)}|s)}$ ,  $\rho_t^j = \frac{\hat{\pi}_t^j(a^{j,an(j)}|s)}{\pi_b^j(a^{j,an(j)}|s)}$ , and  $\rho_g^j = \frac{\pi_g^j(a^{j,an(j)}|s)}{\pi_b^j(a^{j,an(j)}|s)}$ , then the 705 surrogate objective function to optimize as follows:

$$706 \quad 707 \quad 708 \quad 709 \quad 710 \quad 711 \quad \min \left[ \text{clip}(\rho_t^{an(j)}, \rho_g^{an(j)}, \kappa\epsilon) \rho_t^j \hat{A}_{\pi_t}^{\rho}(s, a^{\rho}), \text{clip}(\text{clip}(\rho_t^{an(j)}, \rho_g^{an(j)}, \kappa\epsilon) \rho_t^j, \rho_g^j, \epsilon) \hat{A}_{\pi_t}^{\rho}(s, a^{\rho}) \right]$$

715 where  $\text{clip}(\rho_1, \rho_2, \epsilon)$  means bounding the value of  $\rho_1$  between  $[\rho_2 - \epsilon, \rho_2 + \epsilon]$ ,  $\kappa \in [0, +\infty)$  is a constant scaling factor.  
 716 Within our paradigm, the behavior policies are from the target policy. So we only need to bound  $D_{TV}(\hat{\pi}_t, \pi_g)$ , where  
 717  
 718  
 719  
 720  
 721  
 722  
 723  
 724  
 725

$$\begin{aligned}
 D_{TV}(\hat{\pi}_t, \pi_g) &\leq \frac{1}{2} \sum_{j=k}^i D_{TV}(\hat{\pi}_t^j, \pi_g^j) \\
 &\leq \frac{1}{2} \sum_{a \in \mathcal{A}^{\rho}} \left| \prod_{j=k}^i \hat{\pi}_t^j(a^{an(j)}|s) - \prod_{j=k}^i \pi_g^j(a^{an(j)}|s) \right| \\
 &\leq \frac{1}{2} \sum_{a \in \mathcal{A}^{\rho}} \prod_{j=k}^i \pi_b^j(a^{an(j)}|s) \cdot \left| \frac{\prod_{j=k}^i \hat{\pi}_t^j(a^{an(j)}|s)}{\prod_{j=k}^i \pi_b^j(a^{an(j)}|s)} - \frac{\prod_{j=k}^i \pi_g^j(a^{an(j)}|s)}{\prod_{j=k}^i \pi_b^j(a^{an(j)}|s)} \right| \\
 &\leq \frac{1}{2} \left| \frac{\prod_{j=k}^i \hat{\pi}_t^j(a^{an(j)}|s)}{\prod_{j=k}^i \pi_b^j(a^{an(j)}|s)} - \frac{\prod_{j=k}^i \pi_g^j(a^{an(j)}|s)}{\prod_{j=k}^i \pi_b^j(a^{an(j)}|s)} \right|
 \end{aligned}$$

726  
 727  
 728  
 729  
 730  
 731  
 732  
 733  
 734  
 735  
 736  
 737  
 738  
 739  
 740  
 741  
 742  
 743  
 744  
 745  
 746  
 747  
 748  
 749  
 750 So we can use  $\text{clip}\left(\frac{\hat{\pi}_t}{\pi_b}, \frac{\pi_g}{\pi_b}, \epsilon\right)$  to bound  $D_{TV}(\hat{\pi}_t, \pi_g)$  between  $\frac{1}{2}\epsilon$ .  
 751 There have been lots of works showing that the continuous multiplication of importance sampling can arises variance, and a  
 752 simple and effective way to deal with this problem is to clip the combination of importance sampling ratios (Espeholt et al.,  
 753 2018a)(Schmitt et al., 2020)(Wu et al., 2021). Based on this consideration, we design the clip of the importance sampling  
 754 ratio of the ancestral policies within the PPO clip.  
 755

756  
 757 **C. Limitations**  
 758 We have experimentally verified that GPPO-SAG has advantages in environments with macro action spaces, where a prior  
 759 policy can be obtained.  
 760  
 761 The role of the guiding policy is to direct and constrain the optimization direction of the RL agent, thereby accelerating  
 762 and stabilizing the learning of the model. If there is no prior policy available, we can simplify the problem to Proximal  
 763 Policy Optimization with Structured Action Graph. In this case, we can assume that the behavior policy itself is the guiding  
 764 policy, which is expressed as  $\rho_g^{an(j)} = 1, \rho_q^j = 1$  in Equation 10, enabling the algorithm to run without any prior guidance.  
 765 Structured action graphs are mainly used for modeling action spaces in RL problems that involve macro action spaces.  
 766 In scenarios with macro instructions, a simple partitioning of the original action space suffices, as is the case in strategy  
 767 games like StarCraft II and Hearthstone. However, in scenarios that lack macro action instructions, such as *dm\_control*  
 768 (Tunyasuvunakool et al., 2020), the applicability of our method may be limited.  
 769

---

**D. Practical algorithm**

770  
 771  
 772 **Algorithm 1** Guided Proximal Policy Optimization with Structured Action Graph  
 773  
 774 **Input:** the Structured Action Graph  $\mathcal{G}$  with  $n$  vertices, batch size  $B$ , episodes  $K$ , steps per episode  $T$ , the guiding policy  
 775  $\pi_{\theta_g} = (\pi_{\theta_g}^1, \dots, \pi_{\theta_g}^n)$   
 776 Initialize the behavior policy  $\pi_{\theta_b} = (\pi_{\theta_b}^1, \dots, \pi_{\theta_b}^n)$ , the target policy  $\pi_{\theta_t} = (\pi_{\theta_t}^1, \dots, \pi_{\theta_t}^n)$ , centralized critic  $V_\psi$ , and  
 777 replay buffer  $\mathcal{B}$   
 778 1: **for**  $k = 0$  to  $K - 1$  **do**  
 779 2:   Synchronize  $\pi_b$ 's parameters with policy pool  
 780 3:   **for** each environment step **do**  
 781 4:     Collect transitions  $(s, a, s', r)$  with behavior policy  $\pi_b$  and push them into  $\mathcal{B}$   
 782 5:   **end for**  
 783 6:   **for** each update step **do**  
 784 7:     Sample a minibatch of  $B$  transitions from  $\mathcal{B}$   
 785 8:     Compute advantage  $\hat{A}_{\pi_t}^\varphi(s, a)$  and reward to go  $\hat{R}_t$   
 786 9:     Set vertex  $v^i$  be  $v^{init}$  and query path graph  $\mathcal{P}$  from  $\mathcal{G}$  for each action  
 787 10:   **while** vertex  $v^i$  has child vertex  $v^{i+1}$  in  $\mathcal{P}$  **do**  
 788 11:      $v^i = v^{i+1}$   
 789 12:     Update sub-policy  $\pi_t^i$  by maximizing  
 790 13:        $\frac{1}{BT} \sum_{b=1}^B \sum_{t=1}^T \min \left[ \text{clip}(\rho^{an(j)}, \rho_g^{an(j)}, \kappa\epsilon) \rho^j \hat{A}_{\pi_t}^\varphi(s, a^\varphi), \text{clip}(\text{clip}(\rho^{an(j)}, \rho_g^{an(j)}, \kappa\epsilon) \rho^j, \rho_g^j, \epsilon) \hat{A}_{\pi_t}^\varphi(s, a^\varphi) \right]$   
 791 14:   **end while**  
 792 15:     Update centralized critic  $V_\psi$  by minimizing  
 793 16:        $\frac{1}{BT} \sum_{b=1}^B \sum_{t=1}^T (V_\psi(s_t) - \hat{R}_t)^2$   
 794 17:   **end for**  
 795 18:     Update policy pool with  $\pi_{\theta_t}$   
 796 19: **end for**

---

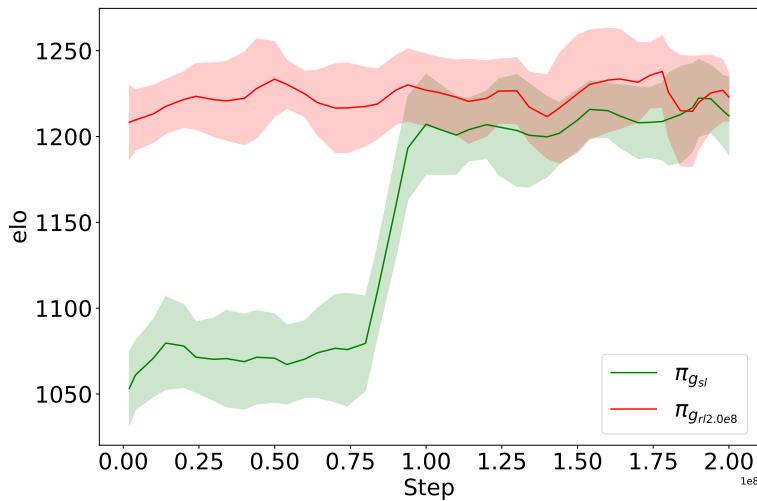
**E. Experimental supplementary materials**
**E.1. Ablation study with varying quality of  $\pi_g$** 


Figure 1. **Ablation of varying quality of  $\pi_g$  on Hearthstone**

803  
 804  
 805 In this experiment, we aim to investigate the impact of the quality of the guiding policy on the learning performance of  
 806 GPPO-SAG. As shown in Fig. 1, we selected two guiding policies for training GPPO-SAG: the supervised learning  
 807 of  $\pi_g$  and the random policy  $\pi_{gr2.0e8}$ . The results show that the performance of GPPO-SAG is significantly  
 808 improved when using a high-quality guiding policy.  
 809  
 810  
 811  
 812  
 813  
 814  
 815  
 816  
 817  
 818  
 819  
 820  
 821  
 822

825 policy  $\pi_{g_{sl}}$ , and the target policy snapshot  $\pi_{g_{rl2.0e8}}$  trained up to 200 million frames. Each policy was used to train the agent  
 826 for 200 million frames of self-play. Then, the saved training snapshots of each model were placed into a shared policy pool,  
 827 and three rounds of 5000 games were played to calculate the Elo rating.

828 It can be seen that the guiding policy with poorer quality (the green line) does not significantly affect the final performance  
 829 of the model. Based on this, we believe that GPPO-SAG has a certain degree of robustness to the quality of the guiding  
 830 policy.  
 831

## 832 E.2. Supervised Learning in SC2

833 Detail hyperparameters for supervised learning in StarCraft II full game have been shown in Table 1. We show the training  
 834 results in <https://tensorboard.dev/experiment/IpznaMnTraAYrL8S7uEQw/>.

837 *Table 1. 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	$2 \times 10^4$	Max warm-up step
steps	$10^5$	Max training step
epochs	10	Repeat 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

## 866 E.3. Reinforcement Learning in SC2

867 Table 2 shows the hyperparameters used for reinforcement learning in the full StarCraft II game. In our method, the  
 868 “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  
 869 PPO-KL, we set  $\epsilon$  to 0.3 in the following experiment. We also present plots of KL divergence with reinforcement learning  
 870 training for each method on the other two maps in Figure 2 and Figure 3, which are consistent with the analysis in our main  
 871 paper.  
 872

## 873 E.4. Experiment in Hearthstone

874 We implemented the Hearthstone environment using SaberStone<sup>2</sup>, an open-source Hearthstone simulator for AI research.  
 875 To simplify the modeling process, we skipped the mulligan phase at the beginning of the game, which is not relevant to our  
 876

877 <sup>2</sup><https://github.com/HearthSim/SabreStone>

880  
 881  
 882  
 883  
 884  
 885  
 886  
 887  
 888  
 889  
 890  
 891  
 892  
 893  
 894  
 895  
 896  
 897  
 898  
 899  
 900  
 901  
 902  
 903  
 904  
 905  
 906  
 907  
 908  
 909  
 910  
 911  
 912  
 913  
 914  
 915  
 916  
 917  
 918  
 919  
 920  
 921  
 922  
 923  
 924  
 925  
 926  
 927  
 928  
 929  
 930  
 931  
 932  
 933  
 934

Table 2. RL hyperparameters in SC2

Name	Value	Description
steps	$10^7$	Max steps for training
learning_rate	$10^{-5}$	
value_pretrain_iters	$4 \times 10^3$	Max step for value pre train
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

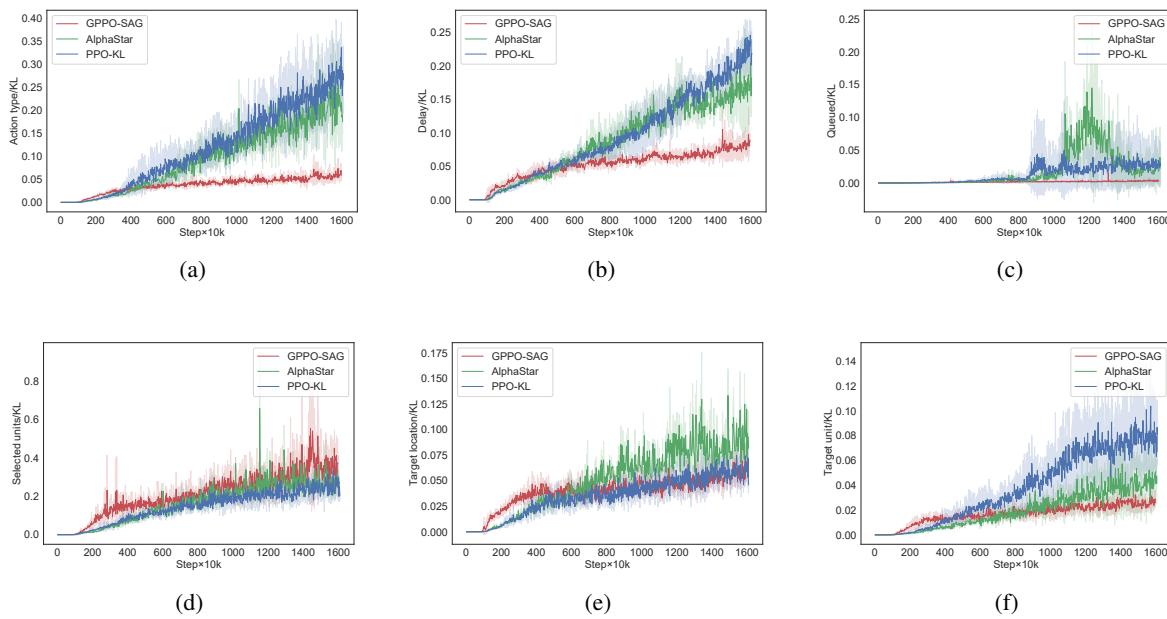


Figure 2. The KL divergence curve between the target policy  $\pi_t$  and the guiding policy  $\pi_g$  during training for each method on the StarCraft II's KingsCove map.(a) Action type vertex;(b) Delay vertex;(c) Queued vertex;(d) Selected units vertex;(e) Target location vertex;(f) Target unit vertex.

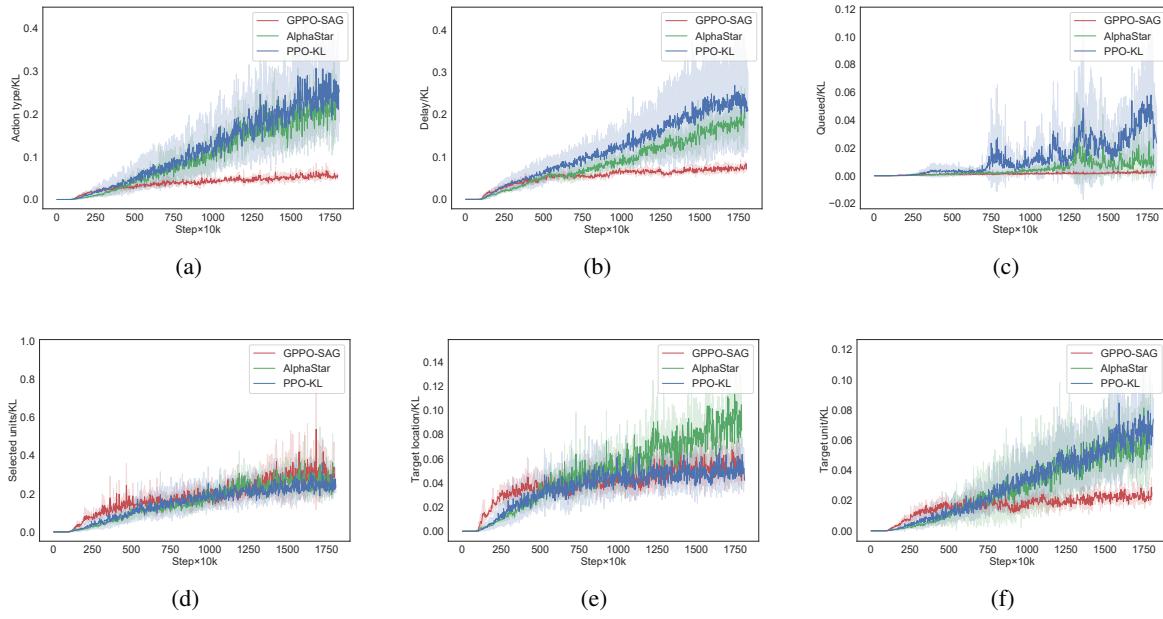


Figure 3. The KL divergence curve between the target policy  $\pi_t$  and the guiding policy  $\pi_g$  during training for each method on the StarCraft II’s NewRepugnancy map.(a) Action type vertex;(b) Delay vertex;(c) Queued vertex;(d) Selected units vertex;(e) Target location vertex;(f) Target unit vertex.

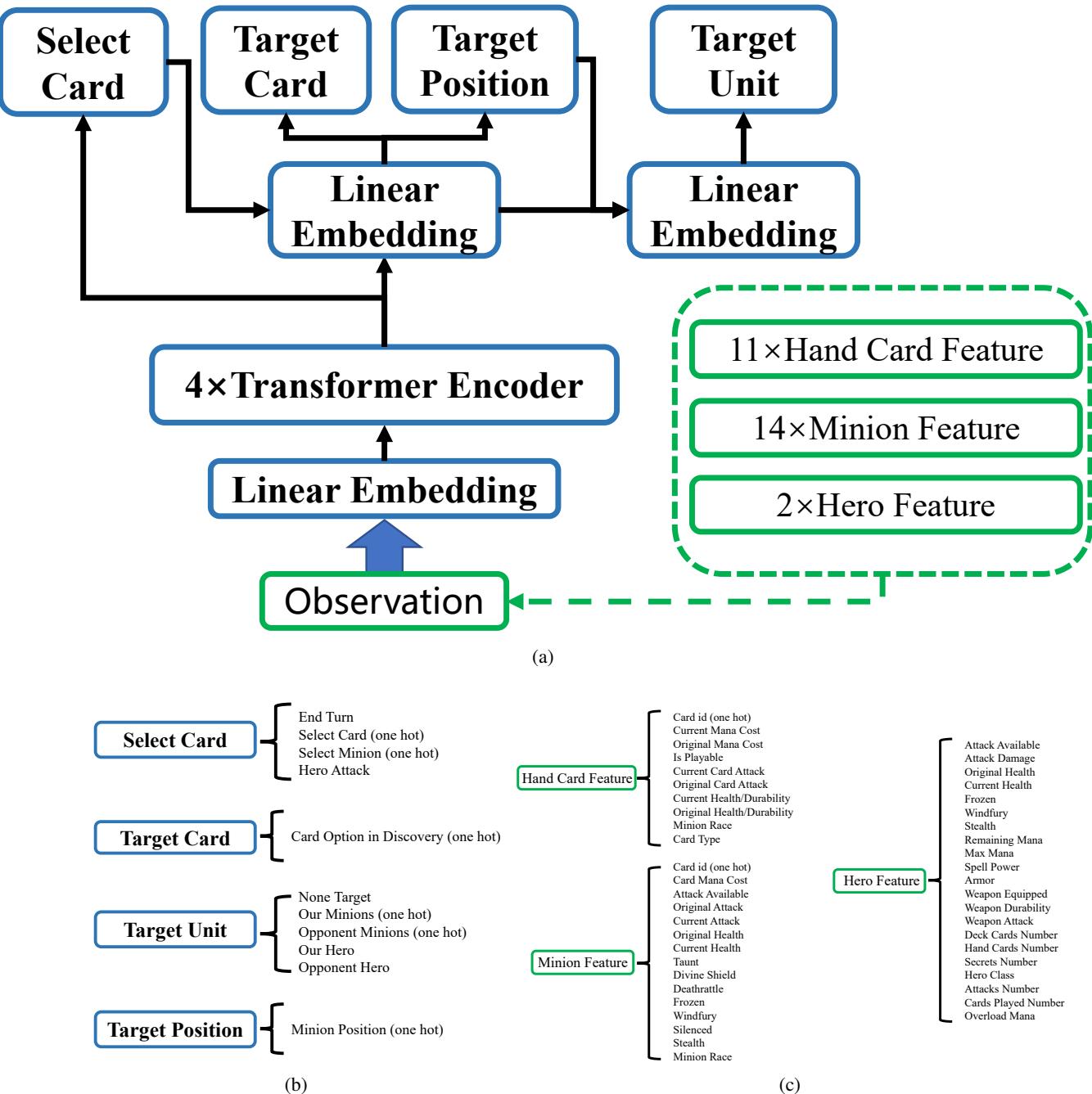
research goals. We used a Warlock deck for all games, and the composition of the card set is shown in Table 3. Each card appears twice in the deck.

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.

Figure 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.

Figure 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.



**Figure 4.** (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.

Table 3. 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

Table 4. RL hyperparameters in HS

Name	Value	Description
steps	$2 \times 10^8$	Max steps for training
learning_rate	$10^{-5}$	
value_pretrain_iters	$4 \times 10^7$	Max step for value pre train
batch_size	24	
unroll_length	100	The maximum time step of a game
value_loss_weight	1.0	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	0.1	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	0.01	Entropy loss weight