Guided Proximal Policy Optimization with Structured Action Graph

Anonymous Authors[1]

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

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

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

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 $(\boldsymbol{S}, \boldsymbol{\mathcal{A}}, r, \boldsymbol{P}, \gamma)$. The set $\boldsymbol{S}$ represents environment states, and the set $\boldsymbol{\mathcal{A}}$ contains the actions that an agent can perform in the environment. $r : \boldsymbol{S} \times \boldsymbol{\mathcal{A}} \to \mathbb{R}$ is the reward function set. $\boldsymbol{P} : \boldsymbol{S} \times \boldsymbol{\mathcal{A}} \times \boldsymbol{S} \to [0, 1]$ is the set of transitions. $\gamma \in [0, 1)$ is the discount factor. Formalized by reinforcement learning, the goal of the agent policy π is to maximize the cumulative discounted reward $J(\pi) = \mathbb{E}_{\boldsymbol{\tau}\sim\pi}\big[\sum_{j=0}^{\infty} \gamma^j r(s_j, \boldsymbol{a}_j)\big]$, where $\boldsymbol{\tau} = (s_0, \boldsymbol{a}_0, s_1, \boldsymbol{a}_1, \cdots)$ is the trajectory induced by π.

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:

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

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:

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

where $clip(\rho, 1, \epsilon)$ means bounding the value of ρ 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:

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

where $\rho^{1:i-1} = \frac{\pi^{1:i-1}_{\theta^{1:i-1}}(\boldsymbol{a}_t^{1:i-1}|s_t)}{\pi^{1:i-1}_{\theta^{1:i-1}_{old}}(\boldsymbol{a}_t^{1:i-1}|s_t)}$ represents the effect of all agents updated before agent i and $\rho^i = \frac{\pi^i_{\theta^i}(a_t^i|s_t)}{\pi^i_{\theta^i_{old}}(a_t^i|s_t)}$. 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 π_b and the target policy π_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 π_b, target policy π_t, and future policy $\hat{\pi}_t$, we refer to it as follows:

$$\begin{aligned} J(\hat{\pi}_t) - J(\pi_t) \ge \frac{1}{1-\gamma}\mathbb{E}_{\boldsymbol{\tau}\sim\pi_b}\Big[\frac{\hat{\pi}_t(\boldsymbol{a}|s)}{\pi_b(\boldsymbol{a}|s)}\hat{A}_{\pi_t}(s,\boldsymbol{a})\Big] \\ -C^{\hat{\pi}_t,\pi_t}D_{TV}(\hat{\pi}_t, \pi_b) \end{aligned} \tag{4}$$

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

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 $\boldsymbol{\mathcal{A}}$ can be represented as a sequence of sub-action sets: $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{A}}^1 \times \cdots \boldsymbol{\mathcal{A}}^n$. 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 $\boldsymbol{\mathcal{P}} =< v^{init}, v^k, \cdots, v^i, v^{end} >$. In this framework, for a given state $s \in \boldsymbol{S}$ and the joint action of ancestor vertices $\boldsymbol{a}^{an(j)} \in \boldsymbol{\mathcal{A}}^{an(j)}$ of vertex $v^j \in \mathcal{V}$, an action a^j can be selected from the sub-policy $\pi^j(a^j|s, \boldsymbol{a}^{an(j)})$ associated with vertex v^j. The joint policy $\pi(\boldsymbol{a}|s) = \prod_{j=k}^{i} \pi^j(a^j|s, \boldsymbol{a}^{an(j)})$, where $an(j) \in \boldsymbol{\mathcal{P}}$. Similar to (Foerster et al., 2018), in the MDPs-SAG setting, we can define the state-action value

function of the vertex v^j:

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

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

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

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

We show in Fig. 1 the SAG for SC2LE with partitions from AlphaStar (Vinyals et al., 2019). 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: $< v^{init}, v^1, v^2, v^3, v^4, v^5, v^{end} >$, and the action "attack target position" by a path: $< v^{init}, v^1, v^2, v^3, v^4, v^6, v^{end} >$.

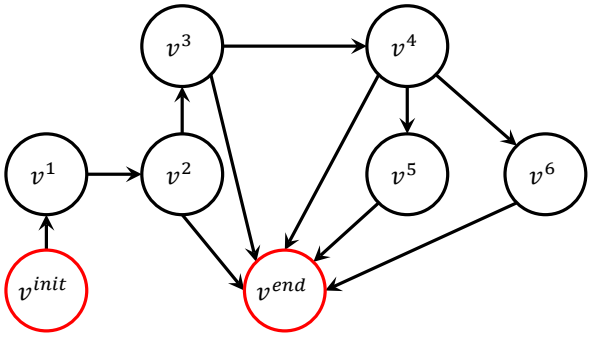


Figure 1. **SAG for SC2LE.** v^{init} and v^{end} represent the start vertex and the end vertex, respectively. $v^1, ..., v^6$ represents "action type," "delay," "queue," "select units," "target units," and "target location" respectively, in the partition of AlphaStar (Vinyals et al., 2019).

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

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

For proof, see Appendix B.1. Lemma 1 states that in MDPs-SAG, the state-action advantage of the path containing the initial vertex can be represented by the sum of the state-action advantages of each vertex in the path. Different from "Multi-Agent Advantage Decomposition" (Kuba et al., 2021)(Kuba et al., 2022), each vertex in Lemma 1 is constrained by graph $\mathcal{G}$, which means there is no practical meaning for a single vertex when some ancestors are missing. Therefore, the arbitrariness of vertex selection required by the "Multi-Agent Advantage Decomposition" does not 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 $\boldsymbol{\pi}_b$, guiding policy $\boldsymbol{\pi}_g$, target policy $\boldsymbol{\pi}_t$, and future policy $\hat{\boldsymbol{\pi}}_t$, we have*

$$J(\hat{\boldsymbol{\pi}}_t) - J(\boldsymbol{\pi}_t) \geq \frac{1}{1-\gamma}\mathbb{E}_{\boldsymbol{\tau}\sim\boldsymbol{\pi}_b}\left[\frac{\hat{\boldsymbol{\pi}}_t(\boldsymbol{a}^{\mathcal{P}}|s)}{\boldsymbol{\pi}_b(\boldsymbol{a}^{\mathcal{P}}|s)}\hat{A}^{\mathcal{P}}_{\boldsymbol{\pi}_t}(s,\boldsymbol{a}^{\mathcal{P}})\right] - C\big(D_{TV}(\hat{\boldsymbol{\pi}}_t,\boldsymbol{\pi}_g) + D_{TV}(\boldsymbol{\pi}_b,\boldsymbol{\pi}_g)\big) \tag{8}$$

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

For proof, see Appendix B.2. In the context of our discussion, the behavior policy π_b represents the policy that interacts with the environment and collects data, while the guiding policy π_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 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 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 $\boldsymbol{\pi}_b$, 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 $\mathcal{P} =< v^{init}, v^k, \cdots, v^i, v^{end} >$ in $\mathcal{G}$

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

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

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

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

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

where $clip(\rho_1, \rho_2, \epsilon)$ means bounding ρ_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 $clip(\rho^{an(j)}, \rho^{an(j)}_g, \kappa\epsilon)$ 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. It thus 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. 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 $\rho^{an(j)}_g = 1, \rho^j_g = 1$ 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?

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

To answer these questions, we compare the performance of GPPO-SAG with AlphaStar (we use DI-star (star Contributors, 2021), an open-source implementation of AlphaStar, as our code base), and PPO (Schulman et al., 2017) 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 IMPALA (Espeholt et al., 2018b). The main objective function of AlphaStar is the penalized policy gradient as follows:

$$\sum_{j=k}^{i} \log \pi_t^j \cdot \hat{A}^{\mathscr{P}}_{\boldsymbol{\pi}_t}(s, \boldsymbol{a}^{\mathscr{P}}) - D_{KL}(\pi_g^j, \pi_t^j) \tag{11}$$

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

$$\sum_{j=k}^{i} \min\Big[\frac{\pi_t^j}{\pi_b^j}\hat{A}^{\mathscr{P}}_{\boldsymbol{\pi}_t}(s, \boldsymbol{a}^{\mathscr{P}}), \\ clip\Big(\frac{\pi_t^j}{\pi_b^j}, 1, \epsilon\Big)\hat{A}^{\mathscr{P}}_{\boldsymbol{\pi}_t}(s, \boldsymbol{a}^{\mathscr{P}})\Big] - D_{KL}(\pi_g^j, \pi_t^j) \tag{12}$$

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

We implement experiments on a platform with 4 NVIDIA-A100 GPUs and 256 AMD EPYC 7742 CPU cores[1].

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 9000 zerg-vs-zerg human replays to train a policy network with 71000 epochs using 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 4000 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 E.2

For a fair comparison, all methods adopt AlphaStar's (Vinyals et al., 2019) 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 LSTM to 32. More hyperparameters for reinforcement learning are listed in Appendix E.3

[1]The source code will be released after we finish organizing it.

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

Hearthstone Hearthstone is a trading card game that allows players to build their 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.

Figure 2(a) 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.

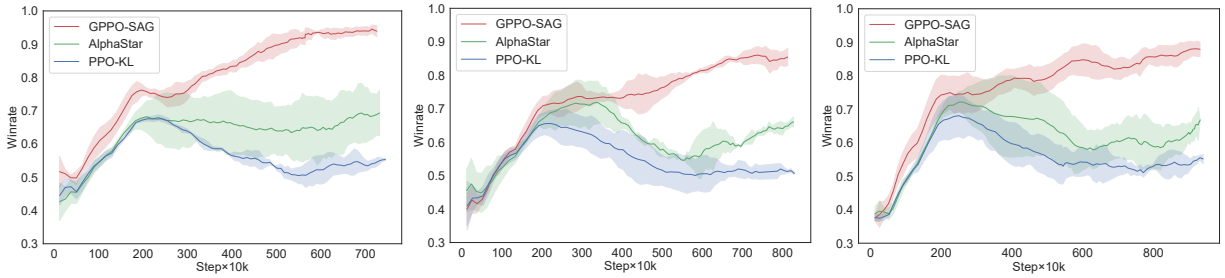
We represent the Hearthstone action space as a structured action graph, as shown in Figure 2(b). In our experiments, we use one Warlock deck and optimize the policy by self-play. We first use 5000 replays to train the guiding policy and adopt an autoregressive policy network structure. We set the learning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

5.2. Experiments for question 1

Figure 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 the PPO algorithm with a KL constraint. During the early training steps, the win rate of each method steadily increases, but at around 2×10^6 game steps, AlphaStar and PPO-KL hit a bottleneck, and their performance stops

(a) (b)

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.



(a) (b) (c)

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 7×10^6 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 π_t and the guiding policy π_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 π_t and the guiding policy π_g to be stable within a certain range in almost all action heads. In contrast, the distribution distance between π_t and π_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 π_t and π_g. This means π_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.

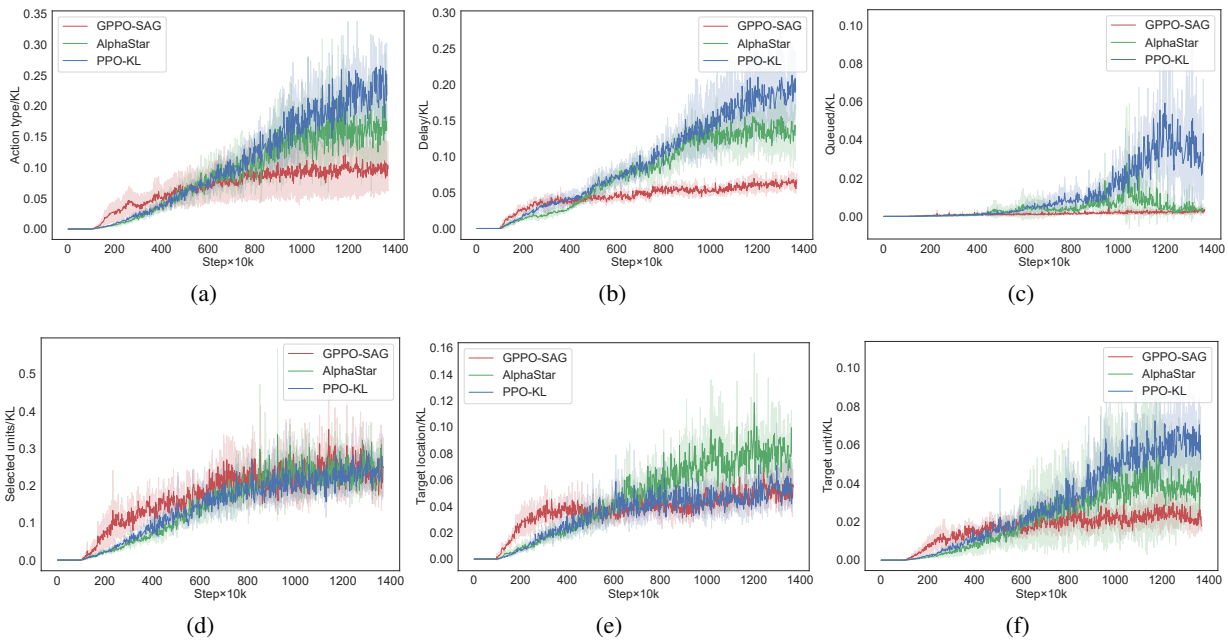


Figure 4. **The KL divergence curve between the target policy π_t and the guiding policy π_g during training for each method on the StarCraft II's KairosJunction map.** (a) Action type vertex; (b) Delay vertex; (c) Queued vertex; (d) Selected units vertex; (e) Target location vertex; (f) Target unit vertex.

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 Figure 3 and Figure 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.

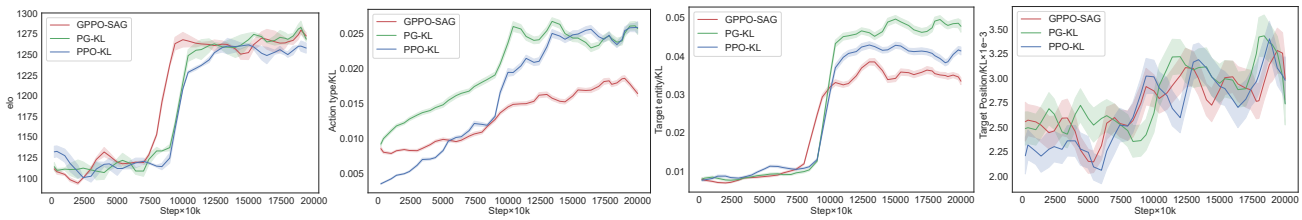
In summary, our proposed GPPO-SAG can more effectively constrain the target policy in the flow space near the bootstrap policy, providing a more consistent policy improvement for bright body learning in complex decision spaces.

5.3. Experiments for question 2

We 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 π_t and the guiding policy π_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



(a) (b) (c) (d)

Figure 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.

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
$0.25\epsilon_{def}$	$59.2\% \pm 1.5\%$
$0.5\epsilon_{def}$	$68.4\% \pm 2.1\%$
ϵ_{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 κ affected the performance of GPPO-SAG. The algorithm performs optimally at $\kappa = \frac{1}{2}$, while performance is impacted by κ values that are either too little or too large. When κ is too small, clipping continues to activate as training proceeds,

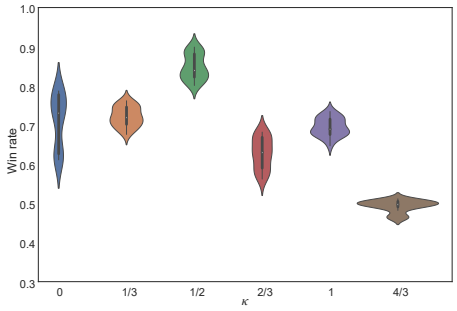


Figure 6. **Effect of the relative magnitude of inner and outer clipping ratio on performance.** The horizontal axis represents the different values of κ. The vertical axis represents the win rate. Each set was obtained by averaging the win rates of the last 500 games during training in three seeds.

and the ancestral importance sampling ratio of π_t has a limited impact on the current sub-action vertex. When κ 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

Agarwal, A., Henaff, M., Kakade, S., and Sun, W. Pc-pg: Policy cover directed exploration for provable policy gradient learning. *Advances in neural information processing systems*, 33:13399–13412, 2020.

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Bush, K. and Pineau, J. Manifold embeddings for model-based reinforcement learning under partial observability. *Advances in neural information processing systems*, 22, 2009.

Eisenach, C., Yang, H., Liu, J., and Liu, H. Marginal policy gradients: A unified family of estimators for bounded action spaces with applications. In *International Conference on Learning Representations*, 2018.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1407–1416. PMLR, 10–15 Jul 2018a.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pp. 1407–1416. PMLR, 2018b.

Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Jiang, N., Krishnamurthy, A., Agarwal, A., Langford, J., and Schapire, R. E. Contextual decision processes with low bellman rank are pac-learnable. In *International Conference on Machine Learning*, pp. 1704–1713. PMLR, 2017.

Jothimurugan, K., Bansal, S., Bastani, O., and Alur, R. Compositional reinforcement learning from logical specifications. *Advances in Neural Information Processing Systems*, 34:10026–10039, 2021.

Kuba, J. G., Wen, M., Meng, L., Zhang, H., Mguni, D., Wang, J., Yang, Y., et al. Settling the variance of multi-agent policy gradients. *Advances in Neural Information Processing Systems*, 34:13458–13470, 2021.

Kuba, J. G., Chen, R., Wen, M., Wen, Y., Sun, F., Wang, J., and Yang, Y. Trust region policy optimisation in multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2022.

Levine, S. Understanding the world through action. In *Conference on Robot Learning*, pp. 1752–1757, 2022.

Lyu, D., Yang, F., Liu, B., and Gustafson, S. Sdrl: Interpretable and data-efficient deep reinforcement learning leveraging symbolic planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 2970–2977, 2019.

Ma, Z., Zhuang, Y., Weng, P., Zhuo, H. H., Li, D., Liu, W., and Hao, J. Learning symbolic rules for interpretable deep reinforcement learning. *arXiv preprint arXiv:2103.08228*, 2021.

Metz, L., Ibarz, J., Jaitly, N., and Davidson, J. Discrete sequential prediction of continuous actions for deep rl. *arXiv preprint arXiv:1705.05035*, 2017.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518: 529–533, 2015.

Pang, Z.-J., Liu, R.-Z., Meng, Z.-Y., Zhang, Y., Yu, Y., and Lu, T. On reinforcement learning for full-length game of starcraft. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 4691–4698, 2019.

Pirotta, M., Parisi, S., and Restelli, M. Multi-objective reinforcement learning with continuous pareto frontier approximation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

Queeney, J., Paschalidis, Y., and Cassandras, C. G. Generalized proximal policy optimization with sample reuse. *Advances in Neural Information Processing Systems*, 34: 11909–11919, 2021.

Schmitt, S., Hessel, M., and Simonyan, K. Off-policy actor-critic with shared experience replay. In *International Conference on Machine Learning*, pp. 8545–8554. PMLR, 2020.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897, 2015.

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

Srivastava, S., Li, C., Lingelbach, M., Martín-Martín, R., Xia, F., Vainio, K. E., Lian, Z., Gokmen, C., Buch, S., Liu, K., et al. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Conference on Robot Learning*, pp. 477–490, 2022.

star Contributors, D. Di-star: An open-sourse reinforcement learning framework for starcraftii. https://github.com/opendilab/DI-star, 2021.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Team, O. E. L., Stooke, A., Mahajan, A., Barros, C., Deck, C., Bauer, J., Sygnowski, J., Trebacz, M., Jaderberg, M., Mathieu, M., et al. Open-ended learning leads to generally capable agents. *arXiv preprint arXiv:2107.12808*, 2021.

Tessler, C., Givony, S., Zahavy, T., Mankowitz, D., and Mannor, S. A deep hierarchical approach to lifelong learning in minecraft. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Tunyasuvunakool, S., Muldal, A., Doron, Y., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T., Heess, N., and Tassa, Y. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

Uchendu, I., Xiao, T., Lu, Y., Zhu, B., Yan, M., Simon, J., Bennice, M., Fu, C., Ma, C., Jiao, J., et al. Jump-start reinforcement learning. *arXiv preprint arXiv:2204.02372*, 2022.

Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J., et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

Wu, B. Hierarchical macro strategy model for moba game ai. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 1206–1213, 2019.

Wu, Z., Yu, C., Ye, D., Zhang, J., Zhuo, H. H., et al. Coordinated 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 learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 6672–6679, 2020.

A. Related Works

The action space for complex decision tasks usually has high-dimensional hybrid action spaces, which pose difficulties in 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, so random policy exploration is usually not feasible. 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 $\mathcal{P} =< v^{init}, v^k, \cdots, v^i >$ in graph $\mathcal{G}$.*

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

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

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

Then we use multi-agent advantage decomposition (Kuba et al., 2021) 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

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

□

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

Lemma 2. *Given behavior policy $\boldsymbol{\pi}_b$, guiding policy $\boldsymbol{\pi}_g$, target policy $\boldsymbol{\pi}_t$, and future policy $\hat{\boldsymbol{\pi}}_t$, we have*

$$J(\hat{\boldsymbol{\pi}}_t) - J(\boldsymbol{\pi}_t) \geq \frac{1}{1-\gamma}\mathbb{E}_{\boldsymbol{\tau}\sim\boldsymbol{\pi}_b}\Big[\frac{\hat{\boldsymbol{\pi}}_t(\boldsymbol{a}^{\mathscr{P}}|s)}{\boldsymbol{\pi}_b(\boldsymbol{a}^{\mathscr{P}}|s)}\hat{A}^{\mathscr{P}}_{\boldsymbol{\pi}_t}(s,\boldsymbol{a}^{\mathscr{P}})\Big] - C\big(D_{TV}(\hat{\boldsymbol{\pi}}_t,\boldsymbol{\pi}_g) + D_{TV}(\boldsymbol{\pi}_b,\boldsymbol{\pi}_g)\big)$$

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

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

$$J(\hat{\boldsymbol{\pi}}_t) - J(\boldsymbol{\pi}_t) \geq \frac{1}{1-\gamma}\mathbb{E}_{\boldsymbol{\tau}\sim\boldsymbol{\pi}_b}\Big[\frac{\hat{\boldsymbol{\pi}}_t(\boldsymbol{a}|s)}{\boldsymbol{\pi}_b(\boldsymbol{a}|s)}\hat{A}_{\boldsymbol{\pi}_t}(s,\boldsymbol{a})\Big] - CD_{TV}(\hat{\boldsymbol{\pi}}_t,\boldsymbol{\pi}_b)$$

where $C = \frac{2\gamma\max_s\left|\mathbb{E}_{\boldsymbol{a}\sim\hat{\boldsymbol{\pi}}_t(\cdot,s)}\hat{A}_{\boldsymbol{\pi}_t}(s,\boldsymbol{a})\right|}{(1-\gamma)^2}$.
Since D_{TV} as a metric satisfies the triangle inequality.
So $D_{TV}(\hat{\boldsymbol{\pi}}_t,\boldsymbol{\pi}_g) + D_{TV}(\boldsymbol{\pi}_b,\boldsymbol{\pi}_g) \geq D_{TV}(\hat{\boldsymbol{\pi}}_t,\boldsymbol{\pi}_b)$
Then we have

$$\begin{aligned}
J(\hat{\boldsymbol{\pi}}_t) - J(\boldsymbol{\pi}_t) \geq& \frac{1}{1-\gamma}\mathbb{E}_{\boldsymbol{\tau}\sim\boldsymbol{\pi}_b}\Big[\frac{\hat{\boldsymbol{\pi}}_t(\boldsymbol{a}|s)}{\boldsymbol{\pi}_b(\boldsymbol{a}|s)}\hat{A}_{\boldsymbol{\pi}_t}(s,\boldsymbol{a})\Big] - CD_{TV}(\hat{\boldsymbol{\pi}}_t,\boldsymbol{\pi}_b)\\
\geq& \frac{1}{1-\gamma}\mathbb{E}_{\boldsymbol{\tau}\sim\boldsymbol{\pi}_b}\Big[\frac{\hat{\boldsymbol{\pi}}_t(\boldsymbol{a}^{\mathscr{P}}|s)}{\boldsymbol{\pi}_b(\boldsymbol{a}^{\mathscr{P}}|s)}\hat{A}^{\mathscr{P}}_{\boldsymbol{\pi}_t}(s,\boldsymbol{a}^{\mathscr{P}})\Big] - C\big(D_{TV}(\hat{\boldsymbol{\pi}}_t,\boldsymbol{\pi}_g) + D_{TV}(\boldsymbol{\pi}_b,\boldsymbol{\pi}_g)\big)
\end{aligned}$$

□

B.3. Proof for Theorem 1

Theorem 1. *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 could form a path graph $\mathscr{P} =< \mathfrak{v}^{init}, \mathfrak{v}^k, \cdots, \mathfrak{v}^i, \mathfrak{v}^{end} >$ in $\mathcal{G}$*

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

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

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

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\tau}\sim\pi_b}\big[\frac{\hat{\boldsymbol{\pi}}_t(\boldsymbol{a}^{\mathscr{P}}|s)}{\boldsymbol{\pi}_b(\boldsymbol{a}^{\mathscr{P}}|s)}\hat{A}^{\mathscr{P}}_{\boldsymbol{\pi}_t}(s,\boldsymbol{a}^{\mathscr{P}})\big] &= \mathbb{E}_{\boldsymbol{\tau}\sim\pi_b}\Big[\sum_{j=k}^{i}\frac{\hat{\boldsymbol{\pi}}_t(\boldsymbol{a}^{\mathscr{P}}|s)}{\boldsymbol{\pi}_b(\boldsymbol{a}^{\mathscr{P}}|s)}\hat{A}^j_{\boldsymbol{\pi}_t}(s,\boldsymbol{a}^{j,an(j)})\Big]\\
&= \mathbb{E}_{\boldsymbol{\tau}\sim\pi_b}\Big[\sum_{j=k}^{i}\frac{\hat{\pi}^j_t(\boldsymbol{a}^{j,an(j)}|s)}{\pi^j_b(\boldsymbol{a}^{j,an(j)}|s)}\cdot\frac{\hat{\pi}^{an(j)}_t(\boldsymbol{a}^{an(j)}|s)}{\pi^{an(j)}_b(\boldsymbol{a}^{an(j)}|s)}\hat{A}^j_{\boldsymbol{\pi}_t}(s,\boldsymbol{a}^{j,an(j)})\Big]\\
&= \sum_{j=k}^{i}\mathbb{E}_{\boldsymbol{\tau}\sim\pi_b}\Big[\frac{\hat{\pi}^j_t(\boldsymbol{a}^{j,an(j)}|s)}{\pi^j_b(\boldsymbol{a}^{j,an(j)}|s)}\cdot\frac{\hat{\pi}^{an(j)}_t(\boldsymbol{a}^{an(j)}|s)}{\pi^{an(j)}_b(\boldsymbol{a}^{an(j)}|s)}\hat{A}^j_{\boldsymbol{\pi}_t}(s,\boldsymbol{a}^{j,an(j)})\Big]
\end{aligned}$$

Step 2 We need to prove

$$D_{TV}(\boldsymbol{\pi}_b,\boldsymbol{\pi}_g) \leq \sum_{j=k}^{i}D_{TV}(\pi^j_b,\pi^j_g) \tag{13}$$

That is

$$\frac{1}{2}\sum_{a\in\mathcal{A}^{\mathcal{P}}}\Big|\prod_{j=k}^{i}\pi_b^j(\boldsymbol{a}^{an(j)}|s)-\prod_{j=k}^{i}\pi_g^j(s,\boldsymbol{a}^{an(j)})\Big|\leq\frac{1}{2}\sum_{j=k}^{i}\sum_{a\in\mathcal{A}^{\mathcal{P}}}|\pi_b^j(\boldsymbol{a}^{an(j)}|s)-\pi_g^j(s,\boldsymbol{a}^{an(j)})|$$

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

$$\Big|\prod_{j=k}^{i}\pi_b^j(s,\boldsymbol{a}^{an(j)})-\prod_{j=k}^{i}\pi_g^j(s,\boldsymbol{a}^{an(j)})\Big|\leq\sum_{j=k}^{i}|\pi_b^j(s,\boldsymbol{a}^{an(j)})-\pi_g^j(s,\boldsymbol{a}^{an(j)})| \tag{14}$$

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

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

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

$$\begin{aligned}
\Big|\prod_{j=k}^{i+1}\pi_b^j(s,\boldsymbol{a}^{an(j)})-\prod_{j=k}^{i+1}\pi_g^j(s,\boldsymbol{a}^{an(j)})\Big|&=\pi_b^{i+1}(s,\boldsymbol{a}^{an(i+1)})\cdot\Big|\prod_{j=k}^{i}\pi_b^j(s,\boldsymbol{a}^{an(j)})-\prod_{j=k}^{i}\pi_g^j(s,\boldsymbol{a}^{an(j)})\Big|\\
&\quad+\prod_{j=k}^{i}\pi_g^j(s,\boldsymbol{a}^{an(j)})\cdot|\pi_b^{i+1}(s,\boldsymbol{a}^{an(i+1)})-\pi_g^{i+1}(s,\boldsymbol{a}^{an(i+1)})|\\
&\leq\sum_{j=k}^{i+1}|\pi_b^j(s,\boldsymbol{a}^{an(j)})-\pi_g^j(s,\boldsymbol{a}^{an(j)})|
\end{aligned}$$

From the discussion above:
i) We've shown that formula 14 is true for any $k\in\mathbb{N}^*$ and $i=k+1$.
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 $i+1$
Through mathematical induction, we can confirm that the formula 14 is true.
So the formula 13 is true.

The same goes for the proof of the formula

$$D_{TV}(\hat{\boldsymbol{\pi}}_t,\boldsymbol{\pi}_g)\leq\sum_{j=k}^{i}D_{TV}(\hat{\pi}_t^j,\pi_g^j).$$

From the two steps above, we have

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

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

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

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

$$\min\left[clip(\rho_t^{an(j)},\rho_g^{an(j)},\kappa\epsilon)\rho_t^j\hat{A}^{\mathcal{P}}_{\boldsymbol{\pi}_t}(s,\boldsymbol{a}^{\mathcal{P}}),clip\big(clip(\rho_t^{an(j)},\rho_g^{an(j)},\kappa\epsilon)\rho_t^j,\rho_g^j,\epsilon\big)\hat{A}^{\mathcal{P}}_{\boldsymbol{\pi}_t}(s,\boldsymbol{a}^{\mathcal{P}})\right]$$

where $clip(\rho_1, \rho_2, \epsilon)$ means bounding the value of ρ_1 between $[\rho_2 - \epsilon, \rho_2 + \epsilon]$, $\kappa \in [0, +\infty)$ is a constant scaling factor.

Within our paradigm, the behavior policies are from the target policy. So we only need to bound $D_{TV}(\hat{\boldsymbol{\pi}}_t, \boldsymbol{\pi}_g)$, where

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

So we can use $clip\left(\frac{\hat{\boldsymbol{\pi}}_t}{\boldsymbol{\pi}_b}, \frac{\boldsymbol{\pi}_g}{\boldsymbol{\pi}_b}, \epsilon\right)$ 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 arises variance, and a simple and effective way to deal with this problem is to clip the combination of importance sampling ratios (Espeholt et al., 2018a)(Schmitt et al., 2020)(Wu et al., 2021). Based on this consideration, we design the clip of the importance sampling ratio of the ancestral policies within the PPO clip.

C. Limitations

We have experimentally verified that GPPO-SAG has advantages in environments with macro action spaces, where a prior policy can be obtained.

The role of the guiding policy is to direct and constrain the optimization direction of the RL agent, thereby accelerating and stabilizing the learning of the model. If there is no prior policy available, 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 $\rho_g^{an(j)} = 1, \rho_g^j = 1$ in Equation 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 macro action spaces. In scenarios with macro 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 macro action instructions, such as *dm_control* (Tunyasuvunakool et al., 2020), the applicability of our method may be limited.

D. Pratical 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 $\boldsymbol{\pi}_{\theta_g} = (\pi^1_{\theta^1_g}, \cdots, \pi^n_{\theta^n_g})$

Initialize the behavior policy $\boldsymbol{\pi}_{\theta_b} = (\pi^1_{\theta^1_b}, \cdots, \pi^n_{\theta^n_b})$, the target policy $\boldsymbol{\pi}_{\theta_t} = (\pi^1_{\theta^1_t}, \cdots, \pi^n_{\theta^n_t})$, centralized critic V_ψ, and replay buffer $\mathcal{B}$

for $k = 0$ to $K - 1$ **do**
  Synchronize $\boldsymbol{\pi}_b$'s parameters with policy pool
  for each environment step **do**
    Collect transitions $(s, \boldsymbol{a}, s', r)$ with behavior policy $\boldsymbol{\pi}_b$ and push them into $\mathcal{B}$
  end for
  for each update step **do**
    Sample a minibatch of B transitions from $\mathcal{B}$
    Compute advantage $\hat{A}^{\mathcal{P}}_{\boldsymbol{\pi}_t}(s, \boldsymbol{a})$ and reward to go $\hat{R}_t$
    Set vertex v^i be v^{init} and query path graph $\mathcal{P}$ from $\mathcal{G}$ for each action
    while vertex v^i has child vertex v^{i+1} in $\mathcal{P}$ **do**
      $v^i = v^{i+1}$
      Update sub-policy π^i_t by maximizing
      $\frac{1}{BT}\sum_{b=1}^{B}\sum_{t=1}^{T}\min\left[clip(\rho^{an(j)}, \rho_g^{an(j)}, \kappa\epsilon)\rho^j \hat{A}^{\mathcal{P}}_{\pi_t}(s, \boldsymbol{a}^{\mathcal{P}}), clip\big(clip(\rho^{an(j)}, \rho_g^{an(j)}, \kappa\epsilon)\rho^j, \rho_g^j, \epsilon\big)\hat{A}^{\mathcal{P}}_{\pi_t}(s, \boldsymbol{a}^{\mathcal{P}})\right]$
    end while
    Update centralized critic V_ψ by minimizing
    $\frac{1}{BT}\sum_{b=1}^{B}\sum_{t=1}^{T}\left(V_\psi(s_t) - \hat{R}_t\right)^2$
  end for
  Update policy pool with $\boldsymbol{\pi}_{\theta_t}$
end for

E. Experimental supplementary materials

E.1. Ablation study with varying quality of π_g

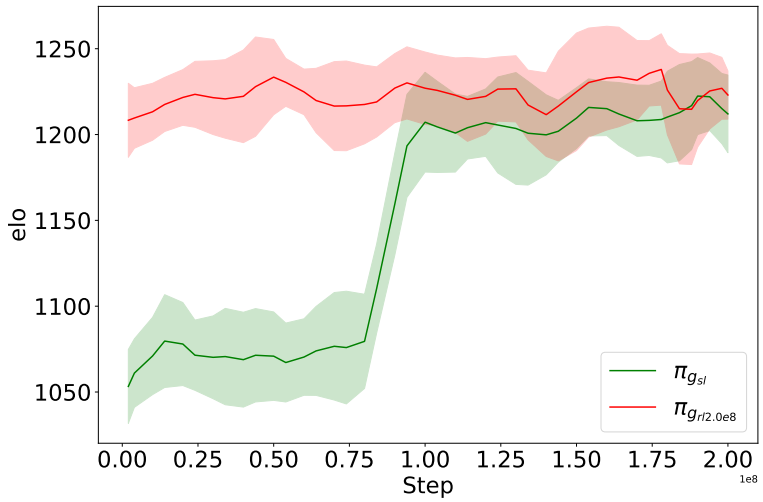


Figure 1. **Ablation of varying quality of π_g on Hearthstone**

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. 1, we selected two guiding policies for training GPPO-SAG: the supervised learning

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 for 200 million frames of self-play. Then, the saved training snapshots of each model were placed into a shared policy pool, and three rounds of 5000 games were played to calculate the Elo rating.

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

E.2. Supervised Learning in SC2

Detail hyperparameters for supervised learning in StarCraft II full game have been shown in Table 1. We show the training results in https://tensorboard.dev/experiment/IpznauMnTraAYrL8S7uEQw/.

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	2×10^4	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

E.3. Reinforcement Learning in SC2

We show the hyperparameters for reinforcement learning in StarCraft II full game in Table 2. Where the "kl_loss_weight" and "action_type_kl_loss_weight" are 0 in our method. The sub-action loss weight is all equal to 1 in RL training. In PPO-KL, we set the ϵ to 0.3 in the following experiment. We present plots of KL divergence with reinforcement learning training for each method on the other two maps in Fig 2 and Fig 3, and it can be seen that both results are equally consistent with corroborating the analysis in our main paper.

E.4. Experiment in Hearthstone

We build the Hearthstone environment on SabberStone [2], an open-source Hearthstone simulator for AI research. For modeling convenience, we skip the mulligan phase at the beginning of the game which is not relevant to our research goal.

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

Table 2. 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 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

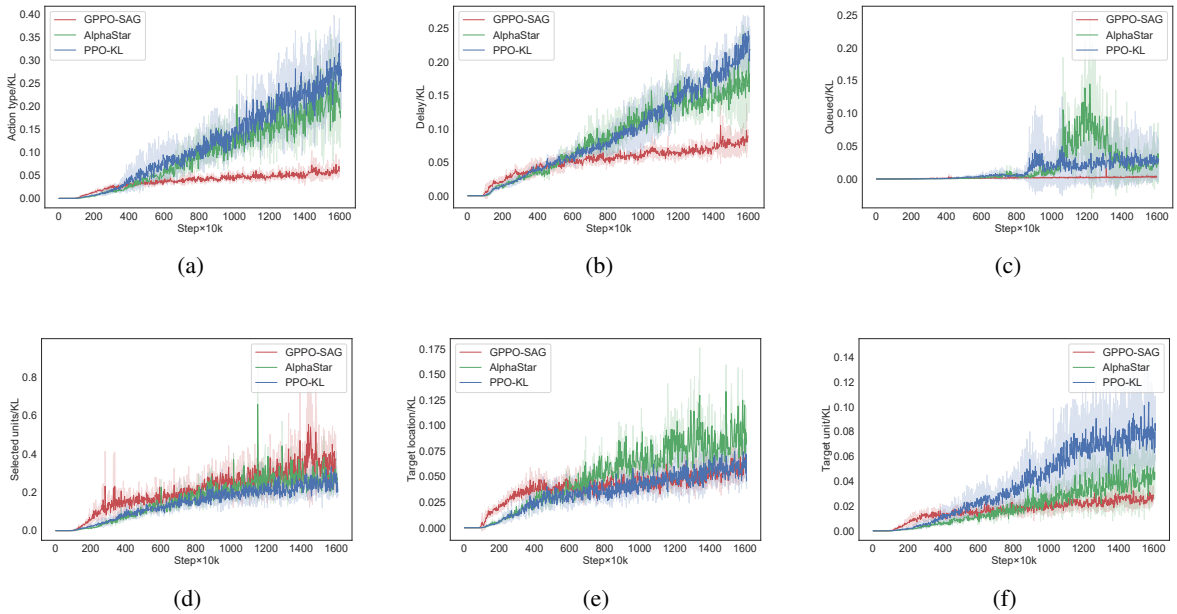


Figure 2. **The KL divergence curve between the target policy π_t and the guiding policy π_g during training for each method on the StarCraft II's KingsCove map.**(a) Action type vertex;(b) Delay vertex;(c) Queued vertex;(d) Selected units vertex;(e) Target location vertex;(f) Target unit vertex.

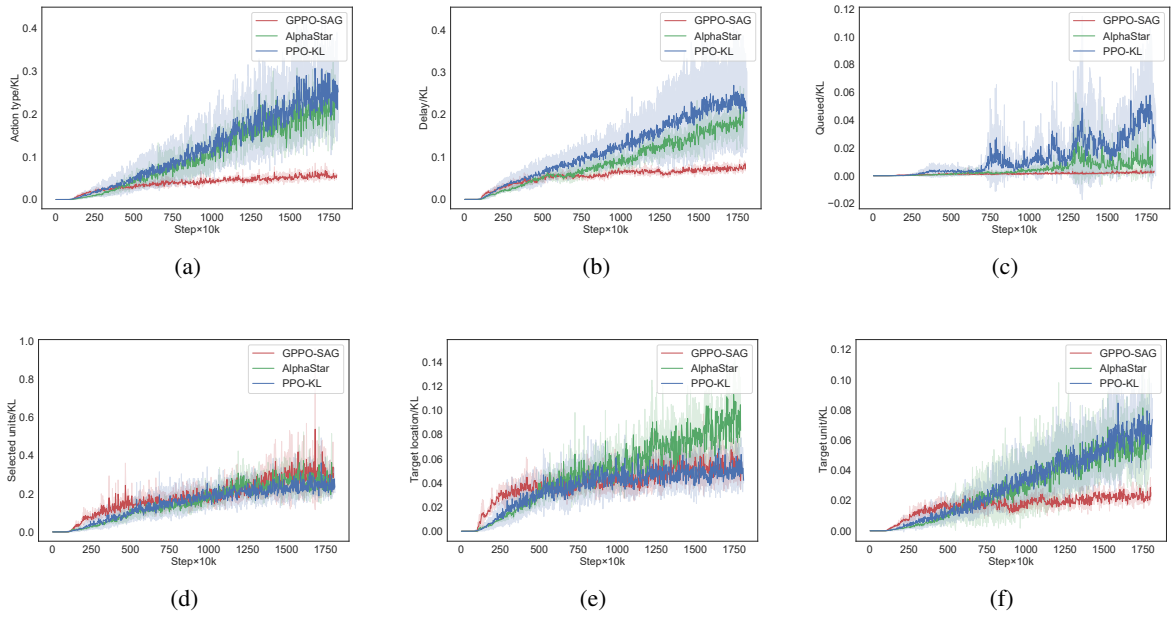


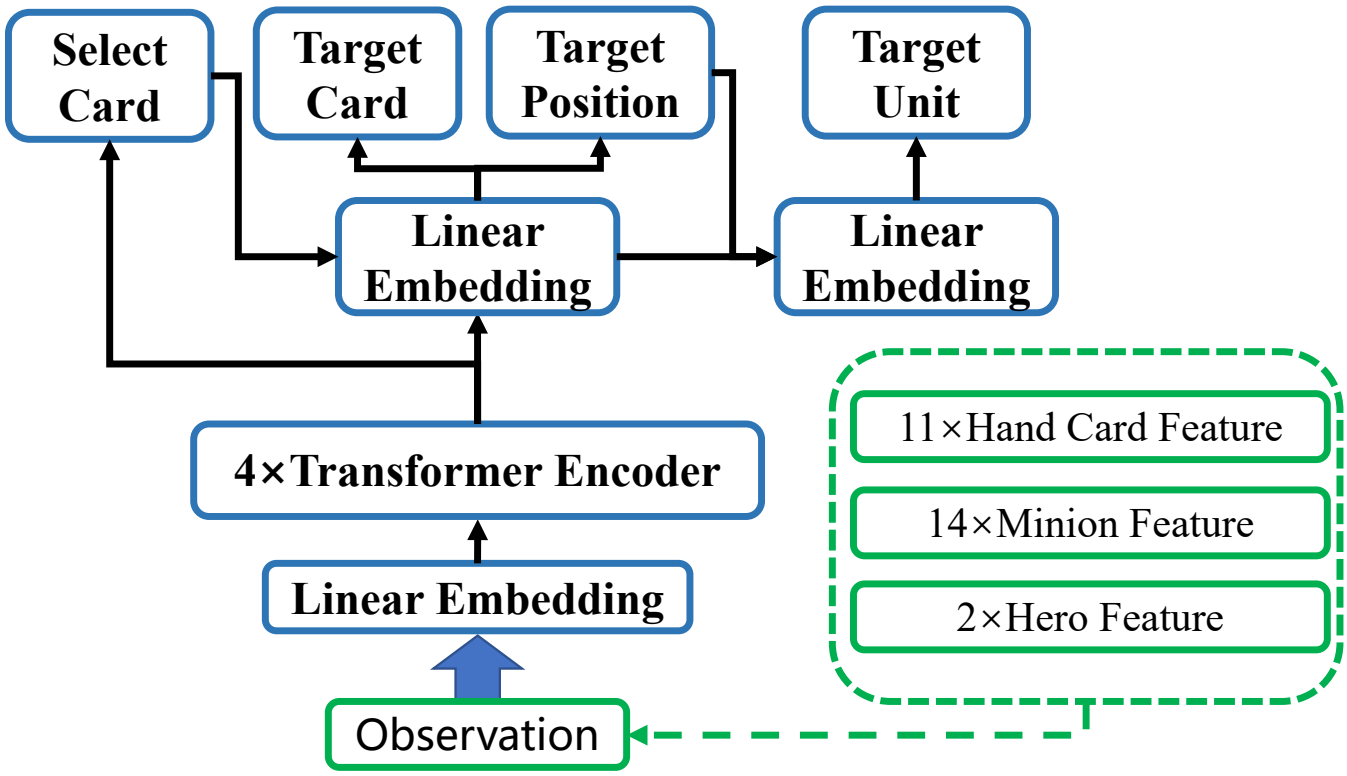
Figure 3. **The KL divergence curve between the target policy π_t and the guiding policy π_g during training for each method on the StarCraft II's NewRepugnancy map.**(a) Action type vertex;(b) Delay vertex;(c) Queued vertex;(d) Selected units vertex;(e) Target location vertex;(f) Target unit vertex.

We set 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.

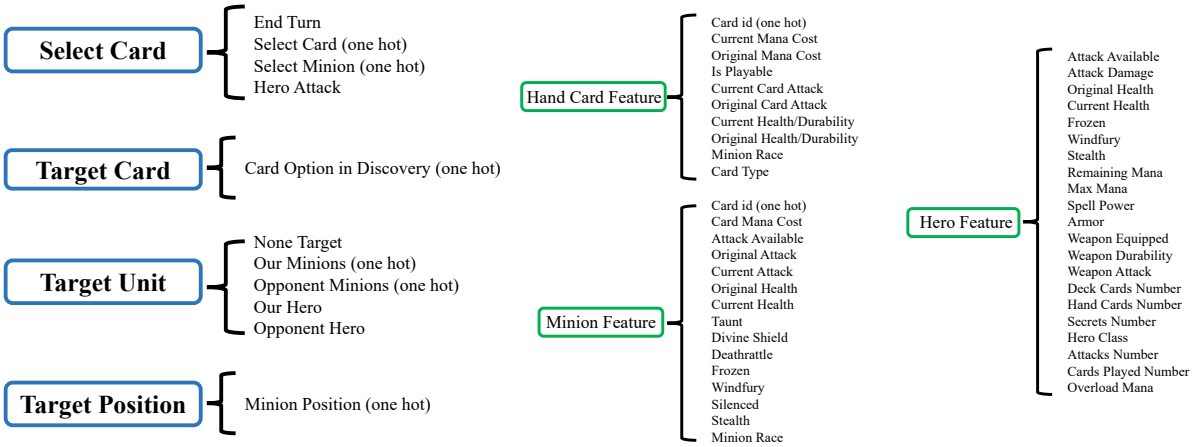
We set batch size as 24, and learning rate to 10^{-5}, and we simultaneously use 12 collectors interacting with the environment to provide data for a trainer.

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. Fig 4(b) shows the functions represented by each sub-action vertex. The state observations are encoded and compressed into a 512-dimensional vector by Transformer Encoder layers to the later iterative action selection module, in which the structure of each sub-policy network is the same as that of the "Action Head" network in AlphaStar.

Fig 4(c) shows the details of the input state encoding. At each step, we input the observed information of these 27 entities as states into the transformer encoder and then encode the current state as a 256-dimensional vector to the action selection network (state value decoder), and finally output the action (state value).



(a)



(b) (c)

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	2×10^8	Max steps for training
learning_rate	10^{-5}	
value_pretrain_iters	4×10^7	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