Improving Multi-agent Reinforcement Learning with Stable Prefix Policy

Yue Deng , **Zirui Wang** and **Yin Zhang** *
College of Computer Science and Technology, Zhejiang University
{devindeng, ziseoiwong, zhangyin98}@zju.edu.cn

Abstract

In multi-agent reinforcement learning (MARL), the ϵ-greedy method plays an important role in balancing exploration and exploitation during the decision-making process in value-based algorithms. However, the ϵ-greedy exploration process will introduce conservativeness when calculating the expected state value when the agents are more in need of exploitation during the approximate policy convergence, which may result in a suboptimal policy convergence. Besides, eliminating the ϵ-greedy algorithm leaves no exploration and may lead to unacceptable local optimal policies. To address this dilemma, we use the previously collected trajectories to construct a Monte-Carlo Trajectory Tree, so that an existing optimal template, a sequence of state prototypes, can be planned out. The agents start by following the planned template and act according to the policy without exploration, **Stable Prefix Policy**. The agents will adaptively dropout and begin to explore by following the ϵ-greedy method when the policy still needs exploration. We scale our approach to various value-based MARL methods and empirically verify our method in a cooperative MARL task, SMAC benchmarks. Experimental results demonstrate that our method achieves not only better performance but also faster convergence speed than baseline algorithms within early time steps.

1 Introduction

Recent research on multi-agent reinforcement learning (MARL) has a very wide range of applications in the real world such as autonomous vehicle teams [Cao *et al.*, 2012] and sensor networks [Zhang and Lesser, 2011]. A number of MARL methods have been proposed to improve either value decomposition [Sunehag *et al.*, 2017; Rashid *et al.*, 2018; Rashid *et al.*, 2020; Wang *et al.*, 2020a] or cooperative exploration [Yang *et al.*, 2020; Mahajan *et al.*, 2019; Wang *et al.*, 2020b], among which value-based MARL methods [Sunehag *et al.*, 2017; Son *et al.*, 2019; Wang *et al.*, 2019b] have shown outstanding performance on challenging tasks. e.g. StarCraft II [Samvelyan *et al.*, 2019].

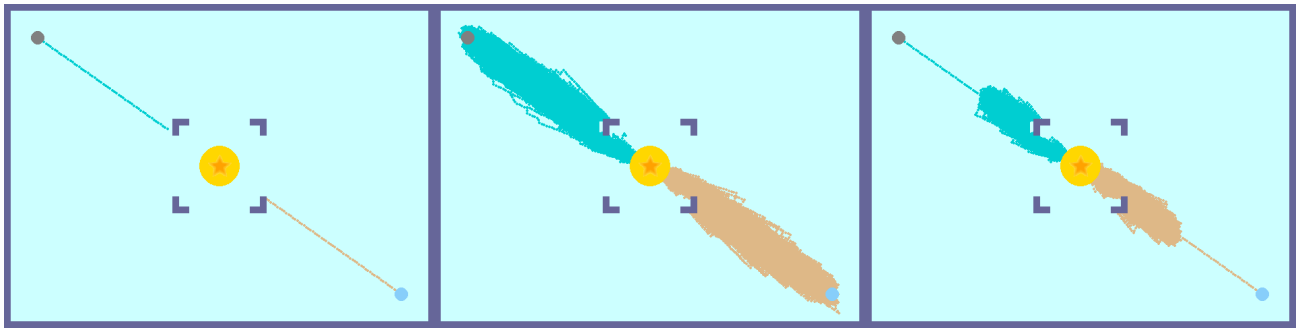
*Corresponding Author: Yin Zhang.

player 2 \ player 1	L	R
T	(1 , 1)	(1.9 , 0)
B	(0 ,1.9)	(2 , 2)

Figure 1: A matrix game showing sub-optimal solutions with exploration.

Moreover, most of the value-based MARL algorithms use ϵ-greedy [Sutton and Barto, 1998] method to balance exploration and exploitation by choosing the greedy action with a probability $1-\epsilon$ or a random choice action otherwise. To explain the sub-optimal policy selection, we show a typical matrix game as described in Figure 1 where (B, R) is the global-optimal solution. However, if player 1 applies ϵ-greedy method with $\epsilon = 0.2$, choosing action L by 0.1 and action R by 0.9, the expectation for player 2 to choose T and B are 1.81 and 1.8. Therefore, player 2 will always choose T when player 1 makes decisions with exploration, and player 2 always chooses L for the same reason. In this case, the solution will fall into (T, L), a suboptimal solution. Therefore, the calculation of Q_{tot}, mixed by each agent's Q, is inaccurate and the errors are cumulated and propagated through the transitions among a trajectory.

Figure 2 also shows the dilemma between the benefit of exploration and the sub-optimal solutions ϵ-greedy method brings. Agents are trapped in local optima when greedy selections are applied only. When using ϵ-greedy method, agents explore through the whole trajectories which makes them difficult to reach the goal. In contrast, if agents start with a stable policy for a few steps and apply ϵ-greedy method afterward, the agents achieve a higher number of successful cases. Based on this, we propose **Stable Prefix Policy** (SPP) to encourage agents to follow the existing optimal trajectory planned from previously collected trajectory data. Specifically, we implement Monte-Carlo Trajectory Tree (MCT^2) to preserve the structure of previous trajectories. The existing optimal trajectory template planned from MCT^2 is used for guiding the agents on whether to follow the template during rollouts and assemble target values in the training process.



(a) (b) (c)

Figure 2: Two agents in the opposite are asked to reach the center goal **simultaneously**. Four obstacles around the central goal stop agents from reaching the goal. The default policy is strictly moving towards the center goal. The three figures above show the rollout traces that (a) agents choose actions greedily, (b) agents choose actions following ϵ-greedy method through the rollout process, and (c) agents choose actions that start by stable prefix policy and follow ϵ-greedy method in later time steps.

When the agents dropout from the template, ϵ-greedy method is activated afterwards.

The main contributions of this work are as follows: **1)** We propose the SPP method to rebalance the exploration and exploitation process when the policy of agents is close to the optimal policy during the training process. **2)** Our proposed method can be adapted to other value-based MARL algorithms with mixing networks with minor changes to existing MARL code-bases. **3)** We validate our methods empirically by extensive experiments on SMAC benchmarks. Experimental Results indicate that existing MARL methods equipped with our method can compete with or outperform original MARL methods in terms of the winning rates or cumulated rewards respectively within 2M time steps.

2 Related Work

Multi-agent reinforcement learning. In multi-agent value-based algorithms, the centralized value function, usually a joint Q-function, is decomposed into local utility functions. Many methods have been proposed to meet the Individual-Global-Maximum (IGM) [Bu *et al.*, 2020] assumption, which indicates the consistency between the local optimal actions and the optimal global joint action. VDN [Lowe *et al.*, 2017] and QMIX [Rashid *et al.*, 2018] introduce additivity and monotonicity to Q-functions. QTRAN [Son *et al.*, 2019] transforms IGM into optimization constraints. QPLEX [Wang *et al.*, 2020a] uses duplex dueling network architecture to guarantee IGM assumption. Instead of focusing on value decomposition, multi-agent policy gradient algorithms provide a centralized value function to evaluate current joint policy and guide the update of each local utility network. Most policy-based MARL methods extend RL ideas, including MADDPG [Lowe *et al.*, 2017], MATRPO [Foerster *et al.*, 2017], MAPPO [Yu *et al.*, 2022]. FOP [Zhang *et al.*, 2021] algorithm factorizes optimal joint policy by maximum entropy and MACPF [Wang *et al.*, 2023] is the latest algorithm that mixes critic values of each agent.

Exploration in Multi-agent Reinforcement Learning. Extended from single-agent reinforcement learning, the ϵ-greedy method is widely applicable in value-based MARL algorithms. In this paper, our approach is based on the ϵ-greedy exploration method and QMIX algorithm for reward credit allocation. In policy-based algorithms, such as MAPPO and COMA [Foerster *et al.*, 2018], for exploration, multi-agent approaches rely on classical noise-based exploration in which agents explore local regions that are close to their individual actor policy. Another line of coordinated exploration algorithms has been proposed. Multi-agent variational exploration (MAVEN) [Mahajan *et al.*, 2019] introduces a latent space for hierarchical control. Agents condition their behavior on the latent variable to perform committed exploration. Influence-based exploration [Wang *et al.*, 2019a] captures the influence of one agent's behavior on others. Agents are encouraged to visit 'interaction points' that will change other agents' behavior.

3 Backgrounds

A fully cooperative multi-agent task is described as a Dec-POMDP [Oliehoek *et al.*, 2016] task which consists of a tuple $G = \langle S, A, P, r, Z, O, N, \gamma \rangle$ in which $s \in S$ is the true state of the environment and N is the number of agents. At each time step, each agent $i \in N \equiv \{1, \ldots, n\}$ chooses an action $a_i \in A$ which forms the joint action $\mathbf{a} \in \mathbf{A} \equiv A^N$. The transition on the environment is according to the state transition function that $P(\cdot|s, \mathbf{a}) : S \times \mathbf{A} \times S \to [0, 1]$. The reward, $r(s, \mathbf{a}) : S \times A \to \mathbb{R}$, is shared among all the agents, and $\gamma \in [0, 1)$ is the discount factor for future reward penalty.

Partially observable scenarios are considered in this paper that each agent draws individual observations $z \in Z$ of the environment according to the observation functions $O(s, i) : S \times N \to Z$. Meanwhile, the action-observation history, $\tau_i \in T \equiv (Z \times A)^*$, is preserved for each agent and conditions the stochastic policy $\pi_i(a_i|\tau_i) : T \times A \to [0, 1]$. The policy π for each agent is determined by a joint action-value function: $Q^\pi(s^t, \mathbf{a}^t) = \mathbb{E}_{s^{t+1:\infty}, \mathbf{a}^{t+1:\infty}}[R^t|s^t, \mathbf{a}^t]$, in which the accumulated reward is considered as a discounted return and formulated as $R^t = \sum_{i=0}^{\infty} \gamma^i r^{t+i}$. After the rollout process, the whole trajectory from the initial transition to

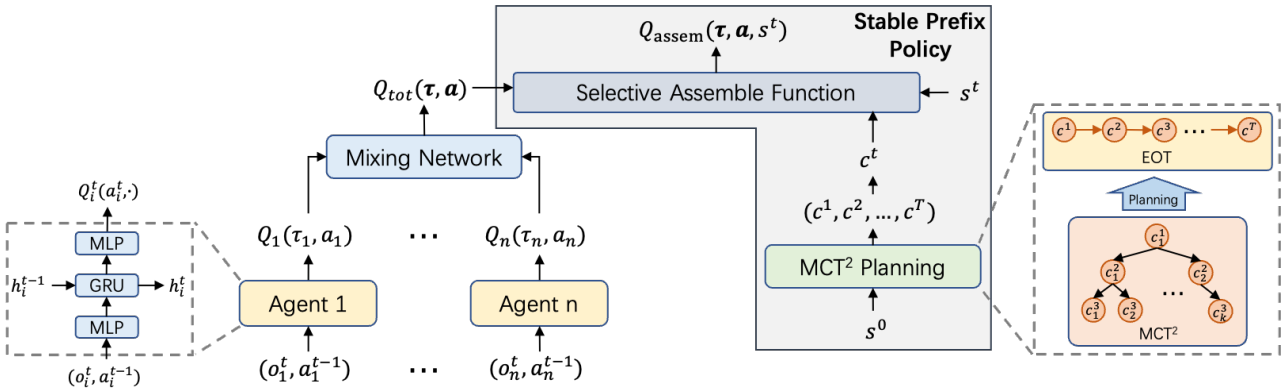


Figure 3: The utility networks and the mixing networks are from original MARL algorithms. Our method plans an existing optimal trajectory (EOT) by the trajectory tree (MCT^2). During the training, the selective assemble function assembles Q_{tot} of each sample in a batch by comparing the template cluster c^t with the input true state s_t. The Q_{assem} is used for TD update.

terminated transition are stored in the replay buffer.

Deep q-learning algorithm aims to find the optimal joint action-value function $Q^*(s, \mathbf{a}; \theta) = r(s, \mathbf{a}) + \gamma \mathbb{E}_{s'} [\max_{\mathbf{a}'} Q^*(s', \mathbf{a}'; \theta)]$. Due to partial observability, $Q(\tau, \mathbf{a}; \theta)$ is used in place of $Q(s, \mathbf{a}; \theta)$ and parameters θ are learnt by minimizing the expected TD error. Centralized training and decentralized execution (CTDE) enables agents to acquire global states during the training and only individual observations during the testing execution. In multi-agent settings, VDN learns a joint action-value function $Q_{tot}(\tau, \mathbf{a})$ as the sum of individual value functions: $Q_{tot}^{\text{VDN}}(\tau, \mathbf{a}) = \sum_{i=1}^{n} Q_i(\tau_i, a_i)$. QMIX introduces a monotonic restriction $\forall i \in \mathcal{N}, \frac{\partial Q_{tot}^{\text{QMIX}}(\tau, \mathbf{a})}{\partial Q_i(\tau_i, a_i)} > 0$ to the mixing network to meet the IGM assumption. IGM asserts the consistency between joint and local greedy action selections in the joint action-value $Q_{tot}(\tau, \mathbf{a})$ and individual action-values $[Q_i(\tau_i, a_i)]_{i=1}^n$.

4 Method

In this section, we introduce the overall architecture of our method and describe the generation of the stable prefix policy. Our method divides the decision-making of the existing MARL methods into two phases: our Stable Prefix Policy and vanilla policy. SPP balances the exploration and exploitation during the trajectory planning process with UCT, with Dirichlet noise during the planning phase. For planning, we establish a trajectory tree from data in the replay buffer in the Monte-Carlo Tree structure, which we call Monte-Carlo Trajectory Tree (MCT^2), to plan out the existing optimal trajectory. Instead of selecting one action from MCT planning, our work uses MCT to preserve trajectories across episodes to provide trajectory templates for utility network training. Additionally, we describe the rollout process, the target value assembling method, and the training pipeline in this section. The pseudo-code is provided in Appendix A.

4.1 Architecture

The training process of value-based MARL algorithms is the Q value Temporal Difference (TD) updating of each agent's utility network. In QMIX and the algorithms derived from QMIX, TD updates are applied to the mixed Q_{tot} value. The utility network is composed of multi-layer perceptron (MLP) layers and Gate Recurrent Unit (GRU) cells in which h_i^t is the historical hidden state. Similar to QMIX algorithm, the utility network at time step t of agent i takes the observation o_i^t and its chosen action a_i^t as an input and outputs the $Q_i(\tau_i, a_i)$ of each agent according to the encoded history state τ_i. Then, these Q values are fed into the mixing network which guarantees the monotonic constraints by hyper-networks and the $Q_{tot}(\tau, \mathbf{a})$ is used for TD learning.

As shown in Figure 3, our stable prefix policy is dependent on the time step t. To summarize the states into a few categories, we train a KMeans classifier $\phi(c|s)$ periodically by the data sampled from the replay buffer. To plan a potential optimal trajectory from MCT^2, the state s^0 (the initial state) is classified into a cluster c^0. Then the existing optimal trajectory is selected from the root node c^0 according to the probabilistic upper confidence bound (PUCB) value of each node and the sequence of cluster IDs is generated. At time step t, c^t is used to be compared with the true state cluster and control the Q assembling process.

Based on the trained classifier $\phi(c|s)$ and the sequence of transitions from the replay buffer during the training process, our classifier predicts the cluster ID of each state in each time step t. Whether the agents are following the trajectory template can be determined by the comparison between the predicted cluster IDs $\hat{c}^t$ (from $\phi(s^t)$) and the cluster from our stable prefix policy c^t. Once confirming the agents are following the template, the target value calculated by $Q_{tot}(\tau_{\mathbf{n}}, \mathbf{a}_{\mathbf{n}})$ is assembled with other target values with the same cluster ID to calculate TD error. According to the CTDE settings, during the testing execution phase, the actions are conditioned only on the utility networks without SPP and without ϵ-greedy exploration.

4.2 Stable Prefix Policy

As shown in Figure 4, to plan out an existing optimal trajectory as a template from previously collected interactive data,

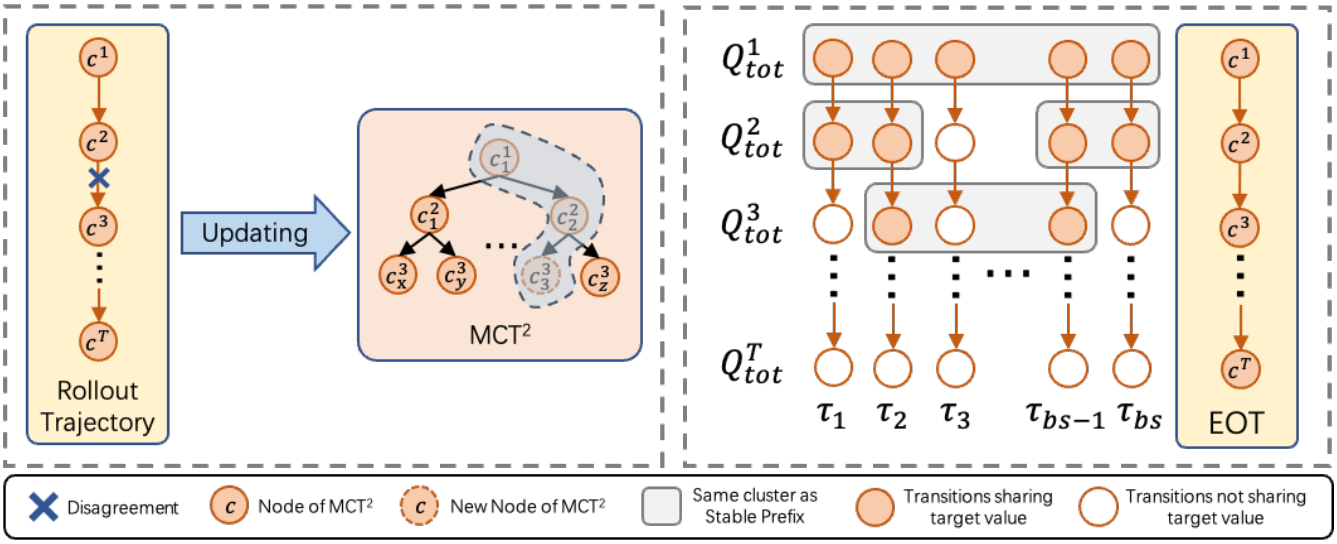


Figure 4: **Left:** Agents follow the template trajectory in the rollout process and encounter disagreement on cluster c^3. Then the transitions prior to c^3 are updated and a new node with c^3 is generated in our MCT^2. In the c^i_j of MCT^2, i is the time step of a state as well as the depth of a node and j is the cluster ID. **Right:** A batch of transition sequences are converted to cluster transitions. For each transition in each time step, Q^t_{tot} of nodes that have the same cluster IDs as the existing optimal trajectory are assembled. The transitions after the dropout process will not share the target value and the Q_{tot} will not be assembled. For example, Q^3_{tot} of τ_3 is not assembled because τ_3 has already dropped out in step 2.

our method generates trajectory trees by the data sampled from the replay buffer in Monte-Carlo Tree structure. We randomly select t_{inter} trajectories and apply clustering methods to assign states s into a cluster c such that similar states can be assigned to the same cluster. A transition $(s^t, \mathbf{a^t}, s^{t+1}, r^{t+1})$ can be regarded as a visit from a node with cluster ID $\phi(s^t)$ to its child node with $\phi(s^{t+1})$. Meanwhile, the expectation rewards from one cluster of states to its subsequent clusters are stored in the tree. Apart from the IDs of clusters, the value of a node $v(n)$ is also stored in the node and is calculated by the following formula:

$$v(n) = \lambda v(n) + (1-\lambda) \times \sum_{cn \in children(n)} \pi(cn|n) \times (R_{n \to cn} + \gamma v(cn)). \quad (1)$$

In the formula above, $v(n)$ is the value of the node n and cn are the children nodes of the node n. $\pi(cn|n)$ is the probability of visiting a child node cn from it parent node n and is usually calculated by counting numbers. $R_{n \to cn}$ is the expectation of the historical rewards from node n to node cn. γ is the discount factor and $v(cn)$ is the value of child node cn. λ is dynamically changed with the visit number of cluster Γ_n and empirically defined as $1/\Gamma_n$.

During the establishment of the MCT^2, we follow the procedure in Efficient Zero [Ye *et al.*, 2021]. A newly selected node will be expanded with the average reward and policy as its prior. Additionally, when the root node is to expand, we apply the Dirichlet noise to the policy prior to give more explorations.

$$\pi(cn|n) := (1-\rho)\pi(cn|n) + \rho \mathcal{N}_{\mathcal{D}}(\xi), \quad (2)$$

where $\mathcal{N}_{\mathcal{D}}(\xi)$ is the Dirichlet noise distribution, ρ and ξ is set to 0.25 and 0.3. However, we do not use any noise and set ρ to 0 for the non-root node or during evaluations.

Based on the MCT^2 implementation described above, we greedily select routes, a sequence of cluster IDs, from the root node to a leaf according to the PUCB values from each parent node n to its child node cn. Inside the formula below, c_{ucb} is the hyper-parameter for balancing exploration and exploitation.

$$c = \underset{cn \in children(n)}{\arg\max} \Big(v(cn) + c_{ucb} \cdot \pi(cn|n) \cdot \sqrt{\left(\frac{\log \Gamma_{cn}}{1+\Gamma_n}\right)} \Big). \quad (3)$$

After the selection, an optimal path of clusters is selected from the root node to a leaf node, $(c^0, c^1, ...c^T)$ within time steps T, which will be used for training and rollout process.

During the rollout process, agents start by following the template trajectory generated by MCT^2. When the agents are following the template, the actions are selected greedily according to their Q values for full exploitation. However, once the agents dropout from the template trajectory in a time step t ($c^t \neq \phi(s^t)$), actions are generated by ϵ-greedy for exploration in the latter rollout steps.

After the rollout processes, the trajectories from the environment interactions will be used to update the MCT^2. The states $\mathbf{s}$ are classified into clusters $\mathbf{c}$ which instead form the transition sequences $(c^0, a^0, c^1, a^1 ... c^T)$. The values of the node before the dropout time step will be updated or created in the MCT^2. It is worth noting that MCT^2 only concentrates on the cluster transitions **without** actions. In such a way, our stable prefix policy only focuses on the optimal subsequent states no matter what actions the agents take.

4.3 Training Pipeline

The existing template trajectories are also used in the training process. A mini-batch of trajectories is sampled from the replay buffer to train the utility network. Our MCT^2 generates a template for each sampled trajectory to find the time step that the agents dropout from the template. As shown in Figure 3, the $Q_{tot}(\tau, a)$ are calculated from the mixing network and the cluster ID c^t is the output of our stable prefix policy. The target values y are calculated by:

$$\begin{aligned} y^t =& r^t + \gamma[\mathbb{1}(c^{t+1} = \phi(s^{t+1})) \cdot Q^{t+1}_{assem}(s^{t+1}) \\ &+ (1 - \mathbb{1}(c^{t+1} = \phi(s^{t+1}))) \cdot Q_{tot}(\tau, a^{t+1})] \end{aligned} \tag{4}$$

While calculating the target y, we also assemble the target values of the same cluster node among the sampled batch of sequences such that the target values are close to the expectation of true discounted returns from that state.

$$Q^t_{assem}(s^t) = \frac{(\sum_{i=1}^{bs} Q_{tot}(\tau_i, a_i) \cdot \mathbb{1}(c^t = \phi(s^t))}{\sum_{i=1}^{bs} \mathbb{1}(c^t = \phi(s^t))} \tag{5}$$

Inside the formula above, bs is the batch size of sampled data, $Q_{tot}(\tau_i, a_i)$ is calculated by adding the rewards to the value of the subsequent node in the template, and the c^t is the cluster node from the trajectory. Because our method takes Q_{tot} and trajectory tree into consideration, our method can be adapted to other value-based MARL algorithms with mixing networks.

4.4 Sample Complexity Analysis

In this section, We linked our SPP method to the framework in [Koenig and Simmons, 1993], verifying that our SPP method can achieve a polynomial sample complexity. As we need to calculate the sample complexity of SPP method. Before that, since our SPP method uses the clustering method for feature extraction, we also need to give a reasonable assumption for the feature extraction module in our algorithm.

Assumption 1. *Assume that the state is parametrized by some feature mapping (clustering mapping) such that for any policy π, Q_{assem} and $\pi(s)$ depend only on $\phi(s)$, the stable prefix policy π^{spp} cover the states visited by the optimal policy:*

$$\sup_{s,t} \frac{d^{\pi^*}_t(\phi(s))}{d^{\pi^{spp}}_t(\phi(s))} \leq C$$

where π^* is the optimal policy, d^π_t is the state visit distribution under a policy π in time step t, $\phi(\cdot)$ is a feature extractor of the policy, and the constant C denotes an upper bound on the coverage ratio[Xie *et al.*, 2022] between π^{spp} and optimal policy π^*.

Assumption 1 indicates that the distributions of states being visited by each of the feature extractors corresponding to SPP π^{spp} and utility policy π should not be too different from each other. The ratio is sometimes called the distribution mismatch coefficient in the literature of policy gradient methods [Agarwal *et al.*, 2021]. We can show that given Assumption 1 our method explores the current time step without dropout of any state which gives good performance guarantees for MDP with general function approximation.

Theorem 1 ([Uchendu *et al.*, 2023] theorem 4.3)**.** *With an appropriate choice of training and evaluation process, our approach (pseudocode in Appendix A) guarantees a near-optimal bound up to a factor of $C \times poly(H)$ for MDP with general function approximation.*

At this point, we have obtained all the results we need, showing that our SPP method achieves a polynomial sample complexity, providing a reasonable assumption 1 holds. Although polynomial or near optimal-bound can be achieved by many optimism-based methods [Jin *et al.*, 2018; Ouyang *et al.*, 2017], our approach further constructs a bonus for uncertainty, which improves the empirical performance of our SPP method.

5 Experiments

We evaluate the performance of our method via the fully cooperative StarCraftII micro-management challenges by the mean winning rate in each scenario. In this environment, we mainly present 6 scenarios with 2 levels of difficulty. Meanwhile, ablation studies are also presented to show the adaptability of our approach to other algorithms and the influence of effective horizons. The details of other SMAC tasks are shown in Appendix B.

5.1 Experiment Settings

SMAC: We verify our proposed stable prefix policy methods on 6 subtasks of two difficulties, a) hard 1c3s5z, 3s_vs_5z, and 5m_vs_6m, and b) super-hard scenarios 3s5z_vs_3s6z, MMM2, and 6h_vs_8z. The difficulty is set as 7 by default. The winning rates of battles are calculated by the mean of 5 different seeds and smoothed by 0.6 for better visualization **within 2M time steps**.

Baselines: We adapt our method to QMIX and W-QMIX algorithms and compare our methods to the value-based QPLEX algorithm, popular policy-based algorithm MAPPO, and currently the latest actor-critic algorithm MACPF. In the ablation study, we also adapt our method to QPLEX algorithm. The QMIX, QPLEX, and W-QMIX in this paper are from pymarl codebase [Hu *et al.*, 2021]. MACPF is from the codebase [Zhang *et al.*, 2021; Wang *et al.*, 2023] and MAPPO is provided by [Yu *et al.*, 2022]

5.2 Experiment Results

We mainly evaluate our proposed stable prefix policy method on QMIX algorithm on 6 benchmarks of SMAC, which is composed of three hard tasks and three super-hard tasks. To demonstrate the overall performance of each algorithm, Figure 5 plots the average test winning rate across the 6 scenarios. In hard tasks, including 5m_vs_6m and 1c3s5z, our proposed method can compete with or outperform baseline algorithms. In the 3s_vs_5z scenario, our method has lower variance within 2M training steps. In the MMM2 task, our method can compete with policy-based methods, however, our proposed stable prefix policy still augments QMIX algorithm and outperforms other value-based methods. In the 6h_vs_8z and 3s5z_vs_3s6z tasks, not all the baselines show the winning rate and our method can achieve acceptable results. It is worth noting that we adjust the parameter size of

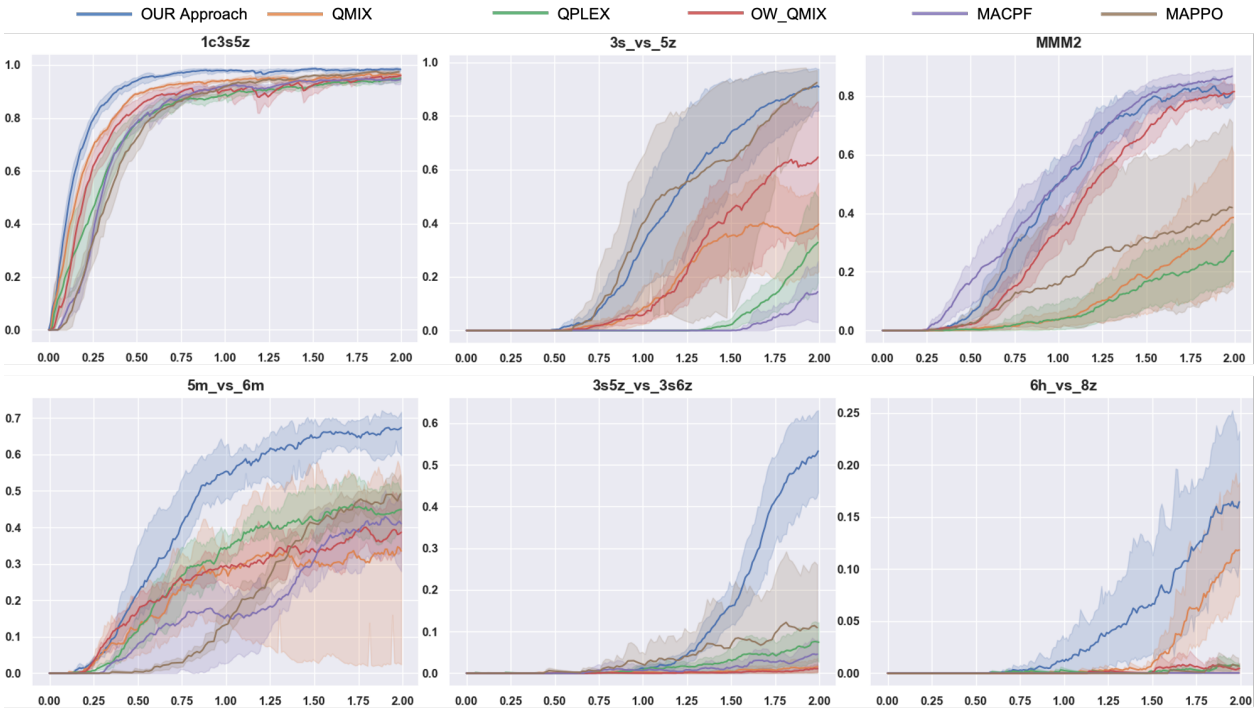


Figure 5: The winning rate curves evaluated on the 6 SMAC tasks with 2 difficulties. The x-axis represents the time steps (1e6) being evaluated and the y-axis is the mean of the winning rate.

the mixing network of QMIX and also apply both the original setting and the adjustment setting to other baselines. The better results of the two settings are shown in the graph. Other hyper-parameters are in Appendix C.

5.3 Ablation Studies

Compatibility: We implement a trajectory tree to provide current existing optimal trajectories for training and rollout and our stable prefix policy module is entirely based on mixing networks, our method can be regarded as a plugin that can be adapted to other value-based MARL methods with minor changes. To test the compatibility of our work, we apply our method on QPLEX, and OW_QMIX algorithms in 1c3s5z, 3s_vs_5z, and MMM2 scenarios correspondingly.

According to Figure 6, in the 1c3s5z scenario and MMM2 task, both the QMIX with stable prefix and QPLEX with stable prefix outperform their original algorithms and OW_QMIX with stable prefix can compete with its origin. In the 3s_vs_5z scenario, all of the algorithms with our proposed stable prefix policy outperform the algorithms without prefix policy. By adding a large replay buffer and assembling the target critic value, we also apply the idea of our approach to the MAPPO algorithm to restrict early exploration. Figure 7 shows that our method can also improve the policy-based MARL algorithms with critic networks.

Effectiveness: During the rollout process, our proposed MCT^2 provides a potential optimal trajectory for agents to follow. Agents select actions according to their utility network and might encounter disagreements with the template in some time steps. Therefore, we record the portion of time steps that agents drop out from the template with the average length of an episode and analyze the influence of the dropout time step on the performance in three scenarios.

According to Figure 8 and the task specifications, 1c3s5z is an easy task for agents to focus fire on correct enemies, so agents have more probability to agree with the stable prefix trajectories. In the 3s_vs_5z task, agents should walk and attack, which is difficult for stable prefix policy to predict when to walk and attack. The important way to win MMM2 task is the control of Medivac and the ally to sacrifice, so the ratio of dropout length is high. According to the trend from the graph, as the policy network converges and the value of each node in our MCT^2 becomes accurate, the dropout ratio becomes higher in later training time steps.

6 Discussion

Performance Enhancement: According to the main experiment result in Figure 5, our method can compete with or outperform other baseline algorithms in most tasks. Our method can also outperform other value-based algorithms in environments where policy-based algorithms are dominant. The existing optimal template trajectories provide agents currently the best route with the highest return. The Q value assembling mechanism within a batch of trajectories reduces the error between the assembled Q_{tot} and its true value.

Adaptability: We adapt our method on value-based MARL algorithms with mixing networks and assemble the Q_{tot} for training. The essence of our work is providing a potential trajectory to agents and assembling a more accurate Q value.

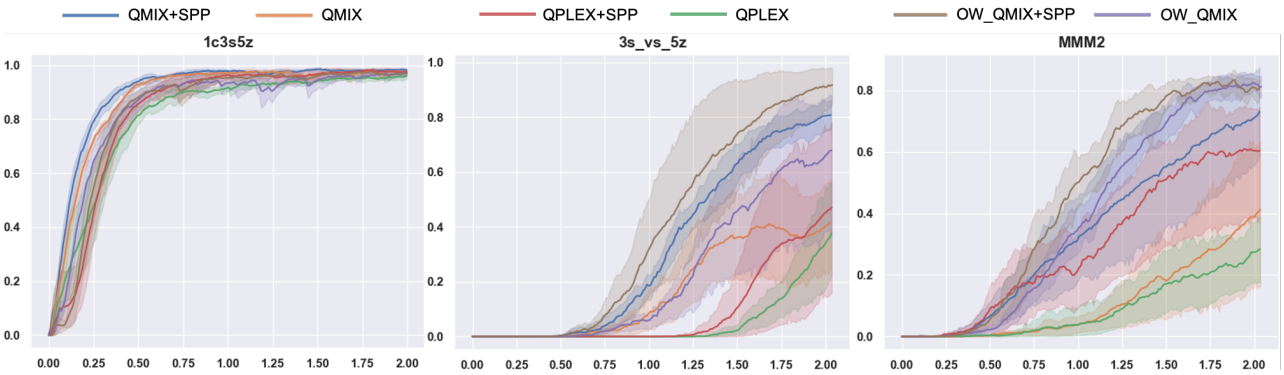


Figure 6: The winning rate curves evaluated on 1c3s5z (Easy), 3s_vs_5z (Hard), and MMM2 (Super-hard) scenarios. The x-axis is the time steps (1e6) that algorithms are evaluated at and the y-axis is the average value of the winning rate among 5 different seeds.

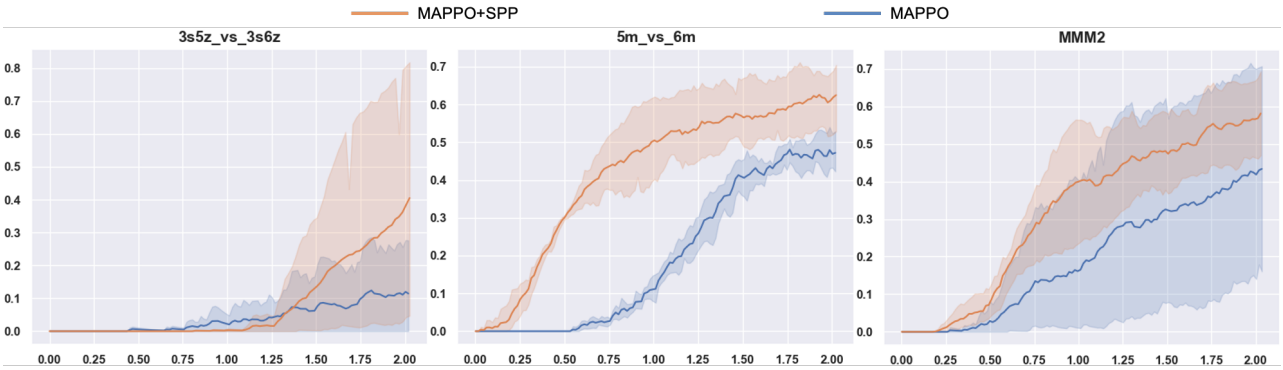


Figure 7: The winning rate curves evaluated on the three SMAC tasks of MAPPO and our proposed MAPPO+SPP methods. The x-axis represents the time steps (1e6) being evaluated and the y-axis is the mean of the winning rate.

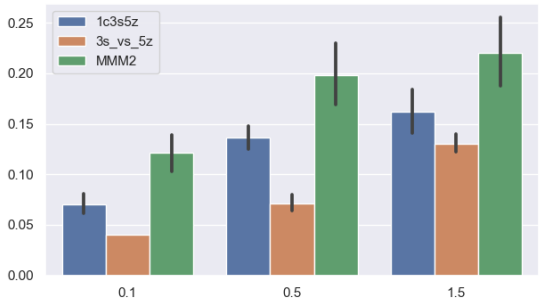


Figure 8: The drop out time step ratio of our QMIX+SPP algorithm on the 1c3s5z, 3s_vs_5z, and MMM2 scenarios in 100k, 500k, and 1500k time steps.

Therefore, value-based MARL algorithms without centralized training, such as IQL, should also be suitable for our method. As for Actor-Critic MARL algorithms, the training of the critic modules is a value-based process, so our proposed method should be suitable for the critic training.

Effectiveness: We aim to find the optimal value without trembling hands when a sub-optimal policy can be obtained from historical interactions. Therefore, the stability of the prefix policy influences the dropout time step, the time step agents encounter disagreements with the provided template. According to Figure 8, the dropout time step is lower in the task where agents need to explore more during the early time steps. When the task is easy enough or the policy is near convergence, the dropout time step will rise during the rollout process. In summary, the dropout time step is empirically positively correlated to training time steps and negatively correlated to the task difficulty.

7 Conclusion and Future Work

In this work, we consider the dilemma between the need for exploration and sub-optimal decision-making exploitation. To solve the problem, we propose a plugin that consists of a stable prefix trajectory provider, the Monte-Carlo Trajectory Tree, and a selective assemble function. We show that the usage of our stable prefix policy can improve MARL algorithms' performance when their utility network is close to optimal and offer exploration budget to later time steps by restricting early exploration according to the templates. SMAC experimental results indicate that our method can be adapted to value-based MARL methods in terms of implementation and offers significant improvements to value-based MARL methods. In the future, we might focus on the prescription that early exploration is vital and update a solution without early dropout in this paper.

Acknowledgements

This work was supported by the Zhejiang Provincial Natural Science Foundation of China under Grant No. LZ23F020009, the NSFC project (No. 62072399), MoE Engineering Research Center of Digital Library, China Research Centre on Data and Knowledge for Engineering Sciences and Technology, the Fundamental Research Funds for the Central Universities, and the Ministry of Education Industry-Academia Cooperation and Collaborative Education for Tencent Project Sponsorship-Tencent AI Arena. We also express our sincere gratitude to anonymous reviewers for their invaluable feedback and constructive comments.

References

[Agarwal *et al.*, 2021] Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *The Journal of Machine Learning Research*, 22(1):4431–4506, 2021.

[Bu *et al.*, 2020] Yuxuan Bu, Xuechen Chen, T. D. Kulkarni, Amir Saeedi, and Joshua B. Tenenbaum. Individual-global-maximum: A new framework for multi-agent reinforcement learning. *arXiv preprint arXiv:2006.07689*, 2020.

[Cao *et al.*, 2012] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics*, 9(1):427–438, 2012.

[Foerster *et al.*, 2017] Jakob Foerster, Guillaume Assael, Michael White, Timothy P. Lillicrap, and John D. Schulman. Learning to communicate with deep multi-agent reinforcement learning. *arXiv preprint arXiv:1706.03762*, 2017.

[Foerster *et al.*, 2018] Jakob Foerster, Guillaume Assael, Michael White, Timothy P. Lillicrap, and John D. Schulman. Counterfactual multi-agent policy gradients. *arXiv preprint arXiv:1802.01561*, 2018.

[Hu *et al.*, 2021] Jian Hu, Siyang Jiang, Seth Austin Harding, Haibin Wu, and Shih wei Liao. Rethinking the implementation tricks and monotonicity constraint in cooperative multi-agent reinforcement learning. 2021.

[Jin *et al.*, 2018] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is q-learning provably efficient? *Advances in neural information processing systems*, 31, 2018.

[Koenig and Simmons, 1993] Sven Koenig and Reid G Simmons. Complexity analysis of real-time reinforcement learning. In *AAAI*, volume 93, pages 99–105, 1993.

[Lowe *et al.*, 2017] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[Mahajan *et al.*, 2019] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. Maven: Multi-agent variational exploration. *Advances in Neural Information Processing Systems*, 32, 2019.

[Oliehoek *et al.*, 2016] Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

[Ouyang *et al.*, 2017] Yi Ouyang, Mukul Gagrani, Ashutosh Nayyar, and Rahul Jain. Learning unknown markov decision processes: A thompson sampling approach. *Advances in neural information processing systems*, 30, 2017.

[Rashid *et al.*, 2018] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4292–4301, 2018.

[Rashid *et al.*, 2020] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 33:10199–10210, 2020.

[Samvelyan *et al.*, 2019] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043, 2019.

[Son *et al.*, 2019] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 5887–5896. PMLR, 2019.

[Sunehag *et al.*, 2017] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[Sutton and Barto, 1998] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction. *MIT press*, 1998.

[Uchendu *et al.*, 2023] Ikechukwu Uchendu, Ted Xiao, Yao Lu, Banghua Zhu, Mengyuan Yan, Joséphine Simon, Matthew Bennice, Chuyuan Fu, Cong Ma, Jiantao Jiao, et al. Jump-start reinforcement learning. In *International Conference on Machine Learning*, pages 34556–34583. PMLR, 2023.

[Wang *et al.*, 2019a] Tonghan Wang, Jianhao Wang, Yi Wu, and Chongjie Zhang. Influence-based multi-agent exploration. *arXiv preprint arXiv:1910.05512*, 2019.

[Wang *et al.*, 2019b] Tonghan Wang, Jianhao Wang, Chongyi Zheng, and Chongjie Zhang. Learning nearly decomposable value functions via communication minimization. *arXiv preprint arXiv:1910.05366*, 2019.

[Wang *et al.*, 2020a] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling

multi-agent q-learning. *arXiv preprint arXiv:2008.01062*, 2020.

[Wang *et al.*, 2020b] Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. Roma: Multi-agent reinforcement learning with emergent roles. *arXiv preprint arXiv:2003.08039*, 2020.

[Wang *et al.*, 2023] Jiangxing Wang, Deheng Ye, and Zongqing Lu. More centralized training, still decentralized execution: Multi-agent conditional policy factorization. In *International Conference on Learning Representations (ICLR)*, 2023.

[Xie *et al.*, 2022] Tengyang Xie, Dylan J Foster, Yu Bai, Nan Jiang, and Sham M Kakade. The role of coverage in online reinforcement learning. *arXiv preprint arXiv:2210.04157*, 2022.

[Yang *et al.*, 2020] Yaodong Yang, Ying Wen, Jun Wang, Liheng Chen, Kun Shao, David Mguni, and Weinan Zhang. Multi-agent determinantal q-learning. In *International Conference on Machine Learning*, pages 10757–10766. PMLR, 2020.

[Ye *et al.*, 2021] Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. *Advances in Neural Information Processing Systems*, 34:25476–25488, 2021.

[Yu *et al.*, 2022] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

[Zhang and Lesser, 2011] Chongjie Zhang and Victor Lesser. Coordinated multi-agent reinforcement learning in networked distributed pomdps. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[Zhang *et al.*, 2021] Tianhao Zhang, Yueheng Li, Chen Wang, Guangming Xie, and Zongqing Lu. Fop: Factorizing optimal joint policy of maximum-entropy multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 12491–12500. PMLR, 2021.