STAGED INDEPENDENT LEARNING: TOWARDS DECENTRALIZED COOPERATIVE MULTI-AGENT REINFORCEMENT LEARNING

Hadi Nekoei *
Université de Montréal, Mila

Akilesh Badrinaaraayanan
Université de Montréal, Mila

Amit Sinha
McGill University, Mila

Mohammad Amini
McGill University, Mila

Janarthanan Rajendran
Université de Montréal, Mila

Aditya Mahajan
McGill University, Mila

Sarath Chandar
École Polytechnique de Montréal, Mila

ABSTRACT

We empirically show that classic ideas from two-time scale stochastic approximation (Borkar, 1997) can be combined with sequential iterative best response (SIBR) to solve complex cooperative multi-agent reinforcement learning (MARL) problems. We first start with giving a multi-agent estimation problem as a motivating example where SIBR converges while parallel iterative best response (PIBR) does not. Then we present a general implementation of staged multi-agent RL algorithms based on SIBR and multi-time scale stochastic approximation, and show that our new methods which we call Staged Independent Proximal Policy Optimization (SIPPO) and Staged Independent Q-learning (SIQL) outperform state-of-the-art independent learning on almost all the tasks in the epymarl (Papoudakis et al., 2020) benchmark. This can be seen as a first step towards more decentralized MARL methods based on SIBR and multi-time scale learning.

1 INTRODUCTION

Multi-agent reinforcement learning (MARL) has emerged as an important learning framework for many applications such as playing Poker (Brown & Sandholm, 2018) and StarCraft (Vinyals et al., 2019) to robotics (Kober et al., 2013) and autonomous driving (Shalev-Shwartz et al., 2016). This framework includes strategic interaction of multiple RL agents in a shared environment in order to compete for resources or cooperate to solve a common problem.

The pioneering work in MARL that used the model of Markov/stochastic games (Shapley, 1953) as a framework was Littman (1994). MARL in Markov games has been the subject of numerous works since then (Busoniu et al., 2008; Zhang et al., 2021). Markov games mostly are assumed to be perfectly observable, and these systems are very well studied. However, most of the real-world decision making problems are partially observable such as in robotics (Wang & de Silva, 2008) and economics (Fudenberg et al., 1998). Moreover, from a single agent's perspective, the environment is non-stationary due to the simultaneous learning of other agents. This makes it difficult to provide convergence guarantees for the learnt policies of all the agents.

The current MARL algorithms can generally be categorized into two types: Centralized and Decentralized. With the former type, it is assumed that there is a central controller overseeing the agents,

*Correspondence at nekoeihe@mila.quebec

who has access to their joint actions as well as local observations. With full awareness of the game setup, the central controller coordinates the agents to determine their optimal policies and computes an equilibrium. Centralized training reduces the issues of partial observability and non-stationarity of the environment. This centralized training (though possibly decentralized execution) known as Centralized Training Decentralized Execution (CTDE) has become a common practice in empirical MARL (Oliehoek et al., 2008; Sunehag et al., 2017; Lowe et al., 2017; Rashid et al., 2018; Hostallero et al., 2019; Mao et al., 2020). However, it must handle joint action spaces that grow exponentially with the number of agents. More importantly, this centralized controller does not exist in many real-world scenarios unless training is done offline in a laboratory. On the other hand, decentralized agents make updates independently based only on local observation and action histories and do not need to be coordinated by a central controller. Therefore, decentralized training still is critical for online learning while interacting with other agents.

Recently, centralized algorithms have experienced significant theoretical advances, including provable finite-sample guarantees (Zhang et al., 2019; Bai & Jin, 2020), but it is more difficult to find theoretical guarantees for decentralized multi-agent reinforcement learning. Very recently, a few papers have provided some theoretical guarantees using two-time scale optimization (Borkar, 1997) for decentralized value-based (Sayin et al., 2021) and policy gradient based (Daskalakis et al., 2020) MARL algorithms. However, these are focused only on two player zero-sum games. In this paper, we aim to investigate the possibility of extending the idea of two-time scale learning to decentralized cooperative settings.

In section 2.1, we introduce two decentralized learning schemes: Parallel iterative best response (PIBR), where all the agents update their local policies simultaneously and sequential iterative best response (SIBR), where agents update their policies in stages. This means that we find the best response for a single agent while keeping the policies of the other agent fixed, sequentially for all agents in each iteration. While PIBR is currently the most common scheme used due to its efficiency in terms of implementation, we give a practical example where PIBR does not converge to the optimal solution while SIBR does. Moreover, we formally prove that SIBR always converges to an agent-by-agent equilibrium. Intuitively, by keeping all but one of the agents fixed, there is no non-stationarity from the other agents at every update. Therefore, by going towards the SIBR, we expect to alleviate some of the challenges of non-stationarity.

Motivated by theoretical guarantees of SIBR and the convergence properties of two-time scale learning, we propose two practical decentralized MARL algorithms named Staged IPPO (SIPPO) and Staged IQL (SIQL) that approximate SIBR. Through rigorous experiments, we evaluate our hypothesis that agents learning sequentially with two-time scales improve cooperative MARL compared to agents learning independently in one time scale. We show that SIPPO and SIQL outperform IPPO and IQL respectively for almost all of the tasks in epymarl benchmark (Papoudakis et al., 2020). To the best of our knowledge, this is the first work successfully applying the idea of SIBR and two-time scale learning to decentralized cooperative MARL.

2 Staged Multi-agent learning

2.1 Sequential versus Parallel iterative best response

In what follows, $J(\theta_t)$ is the performance in the cooperative MARL problem with the policy parameters $\theta_t = \{\theta_t^1, \cdots, \theta_t^n\}$ representing the policy parameters of the n agents at iteration t. Also, θ_t^{-i} consists of the tuple of all agent parameters excluding θ_t^i. One of the simplest schemes for decentralized learning is what we call *parallel* iterative best response policy update:

Definition 2.1. Parallel iterative best response(PIBR) is a policy update scheme where each agent maintains a local policy $\pi_{\theta_t^i}$, where θ_t^i is the parameters of an agent i at iteration t. All agents compute their best response to the policies of other agents, and all agents update their local policy to the computed best response simultaneously and in parallel: $\theta_{t+1}^i = \arg\max_{\theta^i} J(\theta^i, \theta_t^{-i})$.

In practice, instead of computing the best response, agents can compute a noisy gradient and use gradient ascent to update their policy parameters. Independent IPPO (de Witt et al., 2020) is an example of such a PIBR policy update. In general, there are a few convergence guarantees for such

parallel update schemes. To circumvent this difficulty, we consider what we call *sequential* or *staged* iterative best response:

Definition 2.2. Sequential iterative best response(SIBR) is a policy update scheme where each agent maintains a local policy and the policy update works in stages; at a stage, one designated agent computes the best response to the current policy of the other agents and updates its local policy to the computed best response: $\theta^i_{t+1} = \arg\max_{\theta^i} J(\theta^i, \theta^{-i}_t)$; all other agents $j \neq i$ do not change their policy, i.e., $\theta^j_{t+1} = \theta^j_t$. This same update is then sequentially carried out for each agent, and this process is repeated in a cycle.

In practice, instead of computing the best response, agents can compute a noisy gradient and use gradient ascent to update their policy parameters. In SIBR, the overall system performance is guaranteed not to decrease at each stage. So, if the rewards are uniformly bounded, SIBR is guaranteed to converge.

Proposition 1. *SIBR converges to an agent-by-agent optimal solution when the rewards of the game are bounded.*

Proof. We consider a 2 player team game for simplicity. The same procedure generalizes to n players. Consider for all t, θ^1_t, θ^2_t to be the parameters of player 1 and player 2 at iteration t and $J(\theta^1_t, \theta^2_t)$ be the performance of the team. Without loss of generality, consider that player 1 first updates its policy, followed by player 2. Player 1 at iteration $t+1$ plays its best response to player 2's policy at time t, i.e., $\theta^1_{t+1} = \arg\max_{\theta^1} J(\theta^1, \theta^2_t)$ and similarly, $\theta^2_{t+1} = \arg\max_{\theta^2} J(\theta^1_{t+1}, \theta^2)$, so that

$$J(\theta^1_t, \theta^2_t) \leq J(\theta^1_{t+1}, \theta^2_t) \leq J(\theta^1_{t+1}, \theta^2_{t+1}) \leq \ldots < \infty,$$

where $J(\theta^1, \theta^2)$ is always finite for all (θ^1, θ^2) because the rewards are bounded. Since $J(\theta^1_t, \theta^2_t)$ is a non-decreasing sequence upper bounded by a finite value, it must converge to a limit, say $(\tilde{\theta}^1, \tilde{\theta}^2)$. Thus, at this limit, $\tilde{\theta}^1$ is the best response to $\tilde{\theta}^2$ and vice-versa. This establishes that $(\tilde{\theta}^1, \tilde{\theta}^2)$ is an agent-by-agent optimal solution. □

Now that we showed SIBR is guaranteed to converge, we present an example to illustrate that SIBR converges when PIBR does not.

Example 1. Consider a multi-agent estimation problem for minimizing team mean-squared error (Afshari & Mahajan, 2021), where there are three agents, indexed by $i \in \{1, 2, 3\}$, which observe the state of nature $x \sim \mathcal{N}(0, 1)$ with noise. In particular, the observation $y_i \in \mathbb{R}$ of agent i is $y_i = x + v_i$, where $v_i \sim \mathcal{N}(0, 2)$ and (x, v_1, v_2, v_3) are independent.

Agent i generates an estimate $\hat{z}_i = \mu_i(y_i) \in \mathbb{R}$ based on its local observations. The objective is to minimize the estimation error

$$\mathbb{E}\Big[\sum_{i=1}^{3}(x - \hat{z}_i)^2 + \sum_{i=1}^{3}\sum_{j\neq i}(x - \bar{z}_{ij})^2\Big]$$

where $\bar{z}_{ij} = (\hat{z}_i + \hat{z}_j)/2$.

As shown in Afshari & Mahajan (2021), the optimal estimation policy is linear, i.e., $\hat{z}_i = K_i y_i$, where the gains K_i are given by the solution of the following system of linear equations

$$\begin{bmatrix} \frac{3}{2} & \frac{5}{4} & \frac{5}{4} \\ \frac{5}{4} & \frac{3}{2} & \frac{5}{4} \\ \frac{5}{4} & \frac{5}{4} & \frac{3}{2} \end{bmatrix} \begin{bmatrix} K_1 \\ K_2 \\ K_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix}$$

which we write as $\Gamma K = \eta$ for short. The optimal gains are $K_i = \frac{1}{2}$ for all agents. For ease of notation, we will write $\Gamma = D + L + U$ where D is the diagonal entries, L is the lower triangular entries (excluding the diagonal) and U is the upper triangular entries (excluding the diagonal).

Note that the above equation is derived by writing the first order optimality conditions for the total expected cost and setting the derivative to zero. Iterative best response corresponds to solving the system $\Gamma K = \eta$ iteratively as $K^{(n+1)} = M^{-1}(NK^{(n)} + \eta)$. In parallel iterative best response, all

agents update their policy at the same time. So, for this example, parallel iterative best response is the same as the Jacobi method for solving a system of linear equations. Thus, $M = D$ and $N = -(L + U)$. Hence

$$A_{Jacobi} \coloneqq -D^{-1}(L+U) = \begin{bmatrix} 0 & -\frac{5}{6} & -\frac{5}{6} \\ -\frac{5}{6} & 0 & -\frac{5}{6} \\ -\frac{5}{6} & -\frac{5}{6} & 0 \end{bmatrix}.$$

Note that the eigenvalues of A_{Jacobi} are $\{-\frac{5}{3}, \frac{5}{6}, \frac{5}{6}\}$. Thus, the spectral radius of A_{Jacobi} is $\frac{5}{3} > 1$. Hence, the parallel best response iteration does not converge.

In sequential iterative best response, agents update their policies one by one. So, for this example, sequential iterative best response is the same as the Gauss Seidel method for solving a system of linear equations. Thus, $M = (D + L)$ and $N = -U$. Hence,

$$A_{GS} \coloneqq -(D+L)^{-1}U \begin{bmatrix} 0 & -\frac{5}{6} & -\frac{5}{6} \\ 0 & -\frac{25}{36} & -\frac{5}{36} \\ 0 & -\frac{25}{216} & -\frac{175}{216} \end{bmatrix}.$$

Note that the eigenvalues of A_{GS} are $\{0, \frac{1}{432}(325 \pm \sqrt{95}i)\}$. Thus, the spectral radius of A_{GS} is $5\sqrt{30}/36 < 1$. Hence, the sequential best response iteration converges.

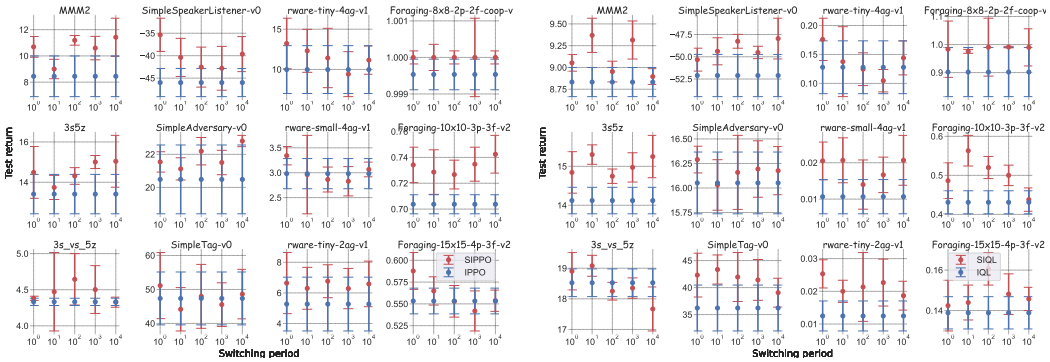
2.2 Staged-learning for deep multi-agent RL

Our main idea is to combine SIBR and multi-time scale learning to design deep decentralized MARL algorithms, which we refer to as staged independent learning. That is, rather than only one agent updating its policy at each stage, all agents update their policy parameters, but at different time scales. Let us assume there are n agents $\{\theta^i\}_{i=1}^n$ getting trained with H levels of learning rates $\{\lambda^h\}_{h=0}^{H-1}$. We can divide the agents to clusters of $\{\boldsymbol{c}^h\}_{h=0}^{H-1}$ where $\boldsymbol{c}^h$ are the agents trained with learning rate λ^h. Switching period (s) controls how frequently agents rotate among different clusters (time scales). For example, in the case of 3 agents with $H = 2$ and $s = 100$, the agents in the clusters $\boldsymbol{c}^0$ and $\boldsymbol{c}^1$ change as follows: $\boldsymbol{c}^0 = \{\theta^0\}$ and $\boldsymbol{c}^1 = \{\theta^1, \theta^2\}$ for the first 100 training steps (t), then $\boldsymbol{c}^0 = \{\theta^1\}$ and $\boldsymbol{c}^1 = \{\theta^0, \theta^2\}$ for $100 < t \leq 200$, and $\boldsymbol{c}^0 = \{\theta^2\}$ and $\boldsymbol{c}^1 = \{\theta^0, \theta^1\}$ for $200 < t \leq 300$, and this pattern repeats. All agents in $\boldsymbol{c}^0$ will be trained with λ^0, while all the agents in $\boldsymbol{c}^1$ will be trained with λ^1. Note that, for s equal to total number of training steps, this approach is reduced to having two different time scales but fixed throughout training.

In this paper, we propose Staged Independent Proximal Policy Optimization (SIPPO) based on the IPPO algorithm and Staged Independent Q-Learning (SIQL) based on the IQL (Tampuu et al., 2017) alogrithm. IPPO is a variant of the commonly used PPO algorithm (Schulman et al., 2017) for decentralized training in multi-agent systems in which each agent estimates its own local value function. de Witt et al. (2020) show that IPPO performs competitively on various StarCraft Multi-Agent Challenge (SMAC) tasks and also IPPO seems to have the best performance between all decentralized MARL algorithms in various tasks (Papoudakis et al., 2020). Note that staged learning can be applied to even centralized methods like multi-agent PPO (MAPPO) (Yu et al., 2021). In this work, we focus on evaluating SIPPO and SIQL and we leave evaluating staged learning with other algorithms and more time scales for future work.

3 Experiments

We evaluate our hypothesis that agents learning in stages in different time scales improve cooperative MARL compared to agents learning independently in one time scale through rigorous experiments. To this end, we evaluate the performance of Independent Proximal Policy Optimization (IPPO) and Independent Q-Learning (IQL) without parameter sharing on four different and complex MARL *epymarl* testbeds (Papoudakis et al., 2020): Multi-Agent Particle Environment (MPE) (Lowe et al., 2017), StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019), Level-Based Foraging (LBF) (Albrecht & Ramamoorthy, 2015), and Multi-Robot Warehouse (RWARE) (Christianos et al.,



(a) Staged IPPO (b) Staged IQL

Figure 1: Performance of IPPO, IQL, and their staged version vs switching period for each task. Error bars in each plot represent the variation over 5 seeds. SIPPO and SIQL performs better than IPPO and IQL respectively in most tasks.

2020). de Witt et al. (2020) show that IPPO performs competitively on various SMAC tasks. Hence, we build on IPPO for our staged learning implementation in all our experiments. (Papoudakis et al., 2020) show that IQL has performance comparable to IPPO, hence we also consider SIQL in our experiments. Some important experimental details are highlighted here: a) Switching period is after s critic training steps, b) We use Adam (Kingma & Ba, 2014) optimizer in all experiments (we only change the learning rate hyperparameter), c) We change the learning rates of both the actor and critic. For all the tasks, we consider five switching periods $(1, 10, 100, 1000, 10000)$. We sample a pair of (lr_0, lr_1) from $L \times L$ learning rates set. For each environment, we construct L by considering the learning rates around the best hyperparameter reported in Papoudakis et al. (2020). In the case of two agents, each agent learns with the respective learning rate while in the case of more than two agents, one agent learns with lr_0 and the rest with lr_1. Note that, to report SIPPO and SIQL performance, we exclude pairs where $lr_0 = lr_1$.

Multi-Agent Particle Environment (MPE): We included three tasks from the MPE environment: Speaker-Listener, Adversary, and Tag. These are two-dimensional navigation tasks that require coordination. The observations of the agent include high-level feature vectors like relative agent and landmark locations. For these experiments, we consider $L = \{1.25 \times 10^{-5}, 2.5 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}, 2 \times 10^{-4}, 4 \times 10^{-4}, 8 \times 10^{-4}\}$. Figure 1 shows the performance of both SIPPO (red) and IPPO (blue) across different switching periods we considered corresponding to the best learning rate. It is clear that in the case of Speaker-Listener and Tag, staged learning almost always performs better than IPPO while in the case of Adversary, benefits of staged learning are evident with more frequent switching.

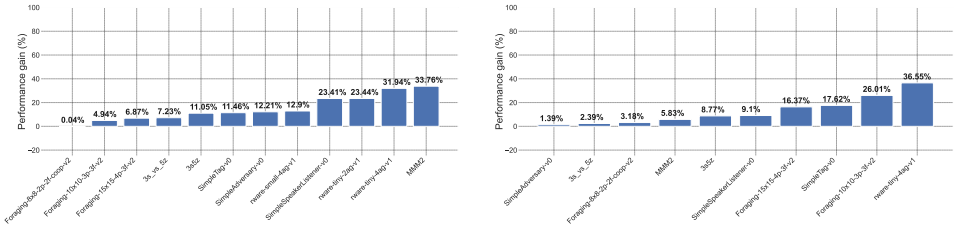
Level-Based Foraging (LBF): In LBF, agents should collect food items that are scattered randomly in a grid-world. Agents and items are assigned levels such that a group of agents can collect an item only if the sum of their levels is greater or equal to the level of the item. We include three tasks from LBF environment: 8v8-2p-2f-c, 10v10-3p-3f, and 15v15-4p-3f with varying world-size, number of agents and food items. The convention for environment name is {grid_size}×{grid_size}-{player count}p-{food locations}f. For these experiments, we consider $L = \{1.25 \times 10^{-5}, 2.5 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}, 2 \times 10^{-4}, 4 \times 10^{-4}, 8 \times 10^{-4}\}$. SIPPO performs better than IPPO in all these environments across different switching periods highlighting the relevance of staged learning with two-time scale even for tasks with more than 2 agents (Figure 1).

Multi-Robot Warehouse (RWARE): We included three tasks from RWARE environment: tiny-4p, tiny-2p, and small-4p. The convention for environment name is {grid-size}-{player count}p. For these experiments, we consider $L = \{6.25 \times 10^{-5}, 1.25 \times 10^{-4}, 2.5 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}, 2 \times 10^{-3}, 4 \times 10^{-3}\}$. As we can observe from Figure 1, SIPPO almost always performs better than IPPO.

StarCraft Multi-Agent Challenge (SMAC): We included three tasks from SMAC environment: MMM2 (a symmetric scenario where each team controls seven marines, two marauders, and one

medivac unit), 3s5z (a symmetric scenario where each team controls three stalkers and five zerglings for a total of eight agents), and 3s_vs_5z (team of three stalkers is controlled by agents to fight against a team of five game-controlled zerglings). For these experiments, we consider 5 learning rates $L = \{1.25 \times 10^{-4}, 2.5 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}, 2 \times 10^{-3}\}$ due to its higher computational requirement. In all three tasks, SIPPO performs consistently better than IPPO across different switching periods, although the variance seems high for certain s values (Figure 1).

Figure 2 shows the performance gain of SIPPO relative to IPPO as well as SIQL relative to IQL across the 12 tasks. SIPPO always either improves or performs as good as IPPO with highest gains in MMM2 and RWARE-tiny-4ag. Similar performance gains can be seen with SIQL as well across all tasks with maximum gains in RWARE-tiny-4ag and LBF-10v10-3p. Figure 3 shows the learning curves of both SIPPO and IPPO suggesting that staged learning can not only improve final performance but can also lead to faster convergence in most cases. For this figure, we pick SIPPO with best combination of (lr_0, lr_1) and s for each task. The learning curves for SIQL and SIPPO corresponding to other s values and heat maps for all the tasks are available in the Appendix.



(a) Staged IPPO

(b) Staged IQL

Figure 2: Performance gain of best performing SIPPO and SIQL relative to regular IPPO and IQL on all 12 tasks. (IQL and SIQL almost have zero return on small-4ag and tiny-2ag. That's why these two tasks are excluded from (b))

4 Related work

Seminal work of Borkar (1997) provided theoretical guarantees on the convergence of concurrent approximation processes with learning parameters that approach zero at different rates. Having two differently paced processes motivates the fast process to correspond with the best response to the slow process. Inspired by it, Borkar & Konda (1997) cast actor-critic algorithms as a two-time scale stochastic approximation and provide its convergence analysis.

While most of the theoretical work in the MARL space have been focusing on centralized training (Bai & Jin, 2020; Wei et al., 2017; Xie et al., 2020), there have been some recent works on decentralized training in zero-sum games providing convergence guarantees for the first time. Sayin et al. (2021) develop a decentralized two-time scale learning dynamics where each agent updates its local Q-function and value function estimates concurrently, the latter happening at a slower time scale without even requiring asymmetric update rules. Also, Daskalakis et al. (2020) show that in a zero-sum game, when two competitive Policy Gradient agents learn simultaneously and their learning rates follow a two-time scale rule, their policies converge to a min-max equilibrium. A classic work by Bowling & Veloso (2002) focus specifically on varying learning rate on a restricted class of iterated matrix games.

Mao et al. (2022) designed a *stage-based V-learning* agent which can learn coarse correlated equilibria (CCE) and correlated equilibria (CE) in general-sum Markov games. They also proposed policy gradient algorithms that learn Nash Equlibria (NE) in Markov potential games. Around the same time, Mao & Başar (2022) came up with an algorithm in which each agent runs optimistic V-learning (a variant of Q-learning) in an independent setting to efficiently explore the environment, through deploying a stabilized online mirror descent (OMD) subroutine for policy updates.

Centralized training approaches have become a common practice in empirical MARL (Oliehoek et al., 2008; Sunehag et al., 2017; Lowe et al., 2017; Rashid et al., 2018; Hostallero et al., 2019; Mao et al.,

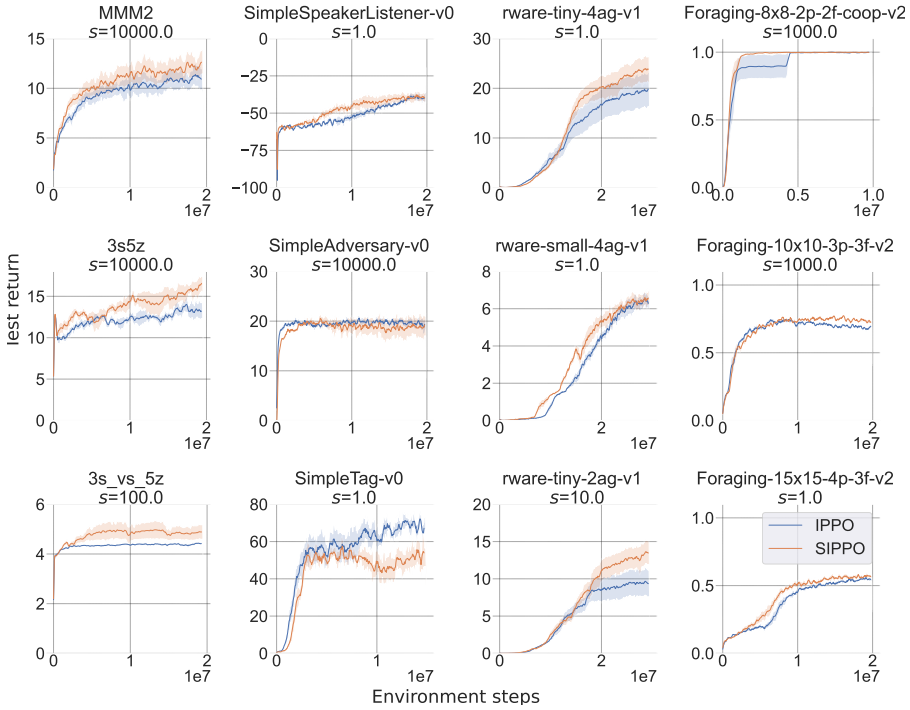


Figure 3: Learning curves for each task. SIPPO leads to faster convergence than IPPO in many tasks. Solid lines are mean test returns over 100 test episodes averaged over 5 independent seeds. Shaded regions indicates the standard-error. Smoothing with window size = 5 is used.

2020). Tan. (1993) proposed Independent Q-learning (IQL), an extension of Q-learning to multi-agent games. Then Tampuu et al. (2017) implemented IQL using deep neural networks as the function approximator in decentralized training of two agents. Foerster et al. (2017) proposed a method based on importance sampling to reduce the effect of non-stationarity in IQL and reports promising results on StarCraft unit micro-management (Samvelyan et al., 2019). There have been other works which are mostly based on optimistic heuristics for updating the learning rates in cooperative environments. Work by Matignon et al. (2007) proposes hysteretic Q-learning in which the Q-values are updated with a higher rate when getting a reward better than the expected state-action value. Omidshafiei et al. (2017) implemented Deep Hysteretic Q-learning. Moreover, Panait et al. (2006) and Palmer et al. (2017) store decaying temperate values for each state-action pair which controls the degree of leniency towards policy updates sampled from the buffer. Palmer et al. (2018) extends the leniency approach to scale to more complex domains by discarding episodes yielding cumulative rewards outside the range of expanding intervals.

5 Conclusions and Future Work

Our goal in this paper was to study the possibility of extending the SIBR algorithm to deep cooperative MARL with the help of two-time scale stochastic approximation ideas (Borkar, 1997) which we call staged learning. We started with a multi-agent estimation problem as a motivating example where SIBR converges when PIBR does not. Then we presented a general implementation of staged multi-agent RL algorithms based on multi-time scale stochastic approximation using different learning rates. Using this protocol, we then proposed Staged IPPO (SIPPO) and Staged IQL (SIQL) which are based on training IPPO and IQL agents with multi-time scales. Finally, we empirically show that with two-time scale learning, SIPPO and SIQL outperforms IPPO and IQL respectively for most of the epymarl benchmark tasks.

Even though in this paper, we focused on decentralized algorithms, Staged learning can be applied to even centralized methods like multi-agent PPO (MAPPO) (Yu et al., 2021). In this work, we only evaluated SIPPO and SIQL with two time-scales. Evaluation of staged learning with other algorithms and more time scales are left for future work. Moreover, in the current setup, the agents need to agree upon the learning rate schedule in advance (what learning rates to use and with what frequencies to switch). Although it is a reasonable assumption that agents can agree to follow some protocols in advance in many MARL scenarios, one potentially promising idea is to adaptively tune the learning rates and learn when to switch those learning rates.

REFERENCES

Mohammad Afshari and Aditya Mahajan. Multi-agent estimation and filtering for minimizing team mean-squared error. *IEEE Transactions on Signal Processing*, 69:5206–5221, 2021.

Stefano V Albrecht and Subramanian Ramamoorthy. A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems. *arXiv preprint arXiv:1506.01170*, 2015.

Yu Bai and Chi Jin. Provable self-play algorithms for competitive reinforcement learning. In *International conference on machine learning*, pp. 551–560. PMLR, 2020.

Vivek S Borkar. Stochastic approximation with two time scales. *Systems & Control Letters*, 29(5): 291–294, 1997.

Vivek S Borkar and Vijaymohan R Konda. The actor-critic algorithm as multi-time-scale stochastic approximation. *Sadhana*, 22(4):525–543, 1997.

Michael Bowling and Manuela Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

Noam Brown and Tuomas Sandholm. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

Filippos Christianos, Lukas Schäfer, and Stefano Albrecht. Shared experience actor-critic for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 10707–10717, 2020.

Constantinos Daskalakis, Dylan J Foster, and Noah Golowich. Independent policy gradient methods for competitive reinforcement learning. *Advances in neural information processing systems*, 33: 5527–5540, 2020.

Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip HS Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *International conference on machine learning*, pp. 1146–1155. PMLR, 2017.

Drew Fudenberg, Fudenberg Drew, David K Levine, and David K Levine. *The theory of learning in games*, volume 2. MIT press, 1998.

Wan Ju Kang David Earl Hostallero, Kyunghwan Son, Daewoo Kim, and Yung Yi Qtran. Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *Proceedings of the 31st International Conference on Machine Learning, Proceedings of Machine Learning Research. PMLR*, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pp. 157–163. Elsevier, 1994.

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.

Weichao Mao and Tamer Başar. Provably efficient reinforcement learning in decentralized general-sum markov games. *Dynamic Games and Applications*, pp. 1–22, 2022.

Weichao Mao, Kaiqing Zhang, Erik Miehling, and Tamer Başar. Information state embedding in partially observable cooperative multi-agent reinforcement learning. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 6124–6131. IEEE, 2020.

Weichao Mao, Lin F. Yang, Kaiqing Zhang, and Tamer Başar. On improving model-free algorithms for decentralized multi-agent reinforcement learning, 2022.

Laëtitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat. Hysteretic q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 64–69. IEEE, 2007.

Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P How, and John Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *International Conference on Machine Learning*, pp. 2681–2690. PMLR, 2017.

Gregory Palmer, Karl Tuyls, Daan Bloembergen, and Rahul Savani. Lenient multi-agent deep reinforcement learning. *arXiv preprint arXiv:1707.04402*, 2017.

Gregory Palmer, Rahul Savani, and Karl Tuyls. Negative update intervals in deep multi-agent reinforcement learning. *arXiv preprint arXiv:1809.05096*, 2018.

Liviu Panait, Keith Sullivan, and Sean Luke. Lenient learners in cooperative multiagent systems. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pp. 801–803, 2006.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. *arXiv preprint arXiv:2006.07869*, 2020.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 4295–4304. PMLR, 2018.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

Muhammed O. Sayin, Kaiqing Zhang, David S. Leslie, Tamer Basar, and Asuman Ozdaglar. Decentralized q-learning in zero-sum markov games, 2021.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

Lloyd S Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10): 1095–1100, 1953.

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.

Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4):e0172395, 2017.

Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Tenth International Conference on Machine Learning*, pp. 330–337, 1993.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Ying Wang and Clarence W de Silva. A machine-learning approach to multi-robot coordination. *Engineering Applications of Artificial Intelligence*, 21(3):470–484, 2008.

Chen-Yu Wei, Yi-Te Hong, and Chi-Jen Lu. Online reinforcement learning in stochastic games. *Advances in Neural Information Processing Systems*, 30, 2017.

Qiaomin Xie, Yudong Chen, Zhaoran Wang, and Zhuoran Yang. Learning zero-sum simultaneous-move markov games using function approximation and correlated equilibrium. In *Conference on learning theory*, pp. 3674–3682. PMLR, 2020.

Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021.

Kaiqing Zhang, Zhuoran Yang, and Tamer Basar. Policy optimization provably converges to nash equilibria in zero-sum linear quadratic games. *Advances in Neural Information Processing Systems*, 32, 2019.

Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, pp. 321–384, 2021.

A LEARNING CURVES

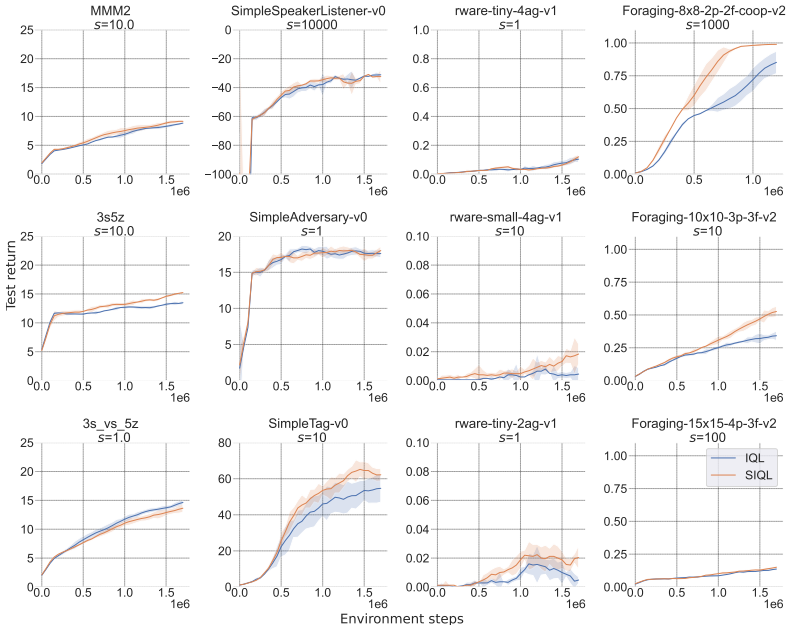
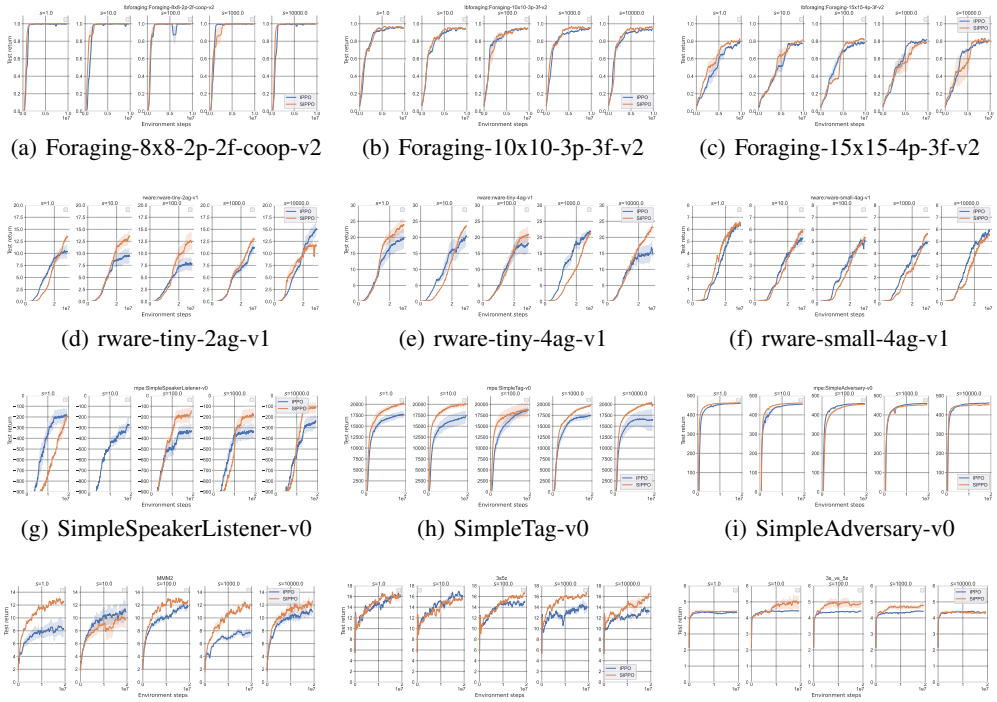


Figure 4: Learning curves for each task. SIQL leads to faster convergence than IQL in many tasks. Solid lines are mean test returns over 100 test episodes averaged over 5 independent seeds. Shaded regions indicates the standard-error. Smoothing with window size = 5 is used.



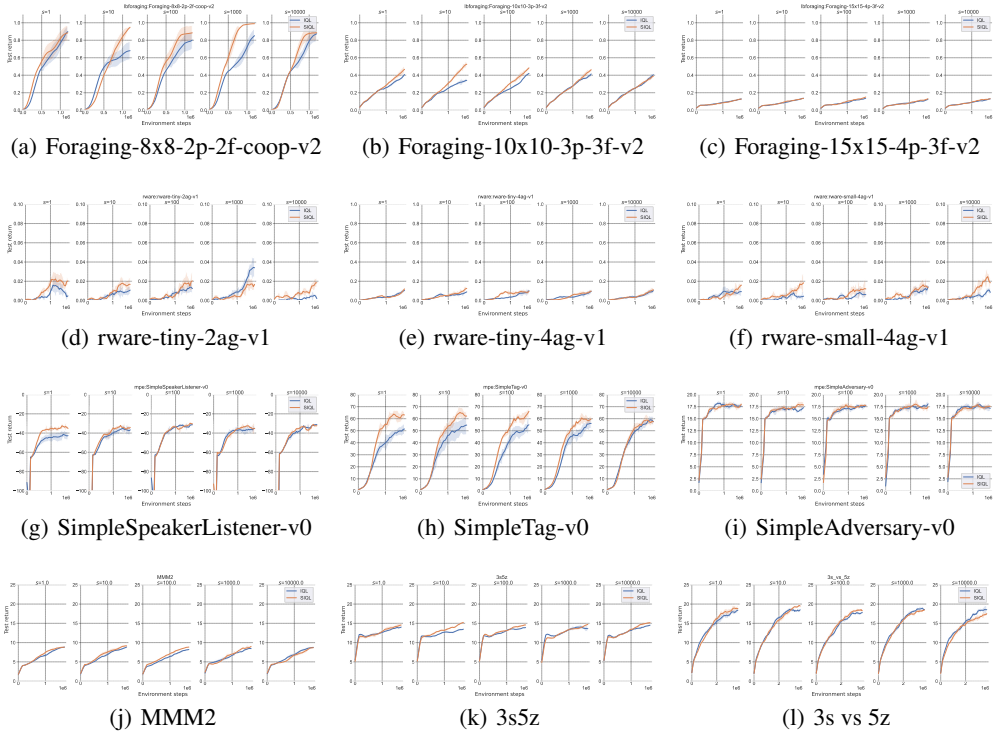
(a) Foraging-8x8-2p-2f-coop-v2 (b) Foraging-10x10-3p-3f-v2 (c) Foraging-15x15-4p-3f-v2

(d) rware-tiny-2ag-v1 (e) rware-tiny-4ag-v1 (f) rware-small-4ag-v1

(g) SimpleSpeakerListener-v0 (h) SimpleTag-v0 (i) SimpleAdversary-v0

(j) MMM2 (k) 3s5z (l) 3s vs 5z

Figure 5: Learning curves of SIPPO and IPPO for each task and different switching periods. Solid lines are mean test returns over 100 test episodes averaged over 5 independent seeds. Shadow region indicates the standard-error. Smoothing with window size = 5 is used.



(a) Foraging-8x8-2p-2f-coop-v2 (b) Foraging-10x10-3p-3f-v2 (c) Foraging-15x15-4p-3f-v2

(d) rware-tiny-2ag-v1 (e) rware-tiny-4ag-v1 (f) rware-small-4ag-v1

(g) SimpleSpeakerListener-v0 (h) SimpleTag-v0 (i) SimpleAdversary-v0

(j) MMM2 (k) 3s5z (l) 3s vs 5z

Figure 6: Learning curves of SIQL and IQL for each task and different switching periods. Solid lines are mean test returns over 100 test episodes averaged over 5 independent seeds. Shadow region indicates the standard-error. Smoothing with window size = 5 is used.

B HEATMAPS

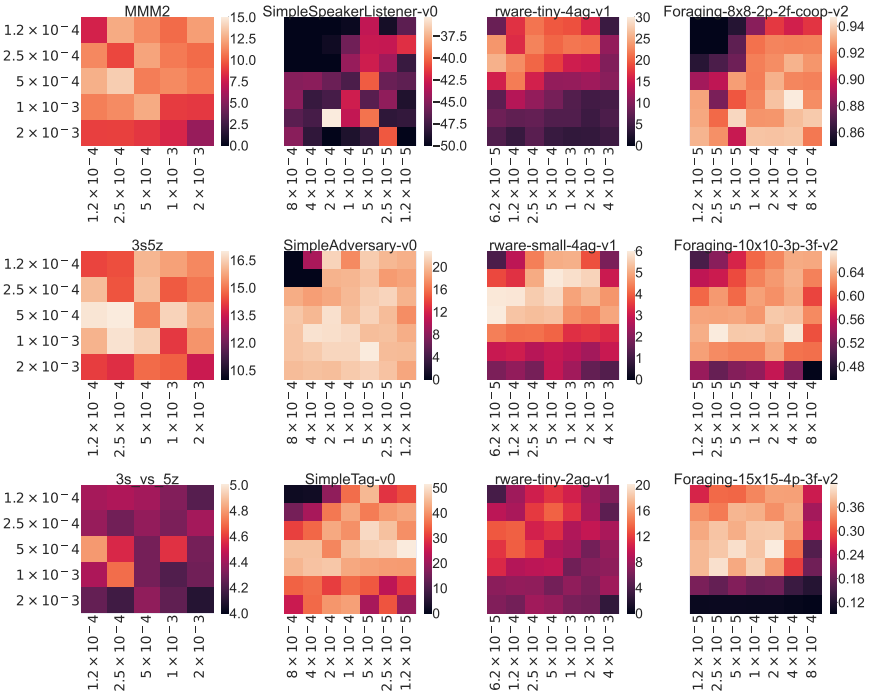


Figure 7: Final performance of IPPO with different learning rate combinations. These heatmaps are for the best switching period values. It's clear that in many tasks, non-diagonal values (SIPPO) have relatively better performance compared to diagonal values (IPPO).

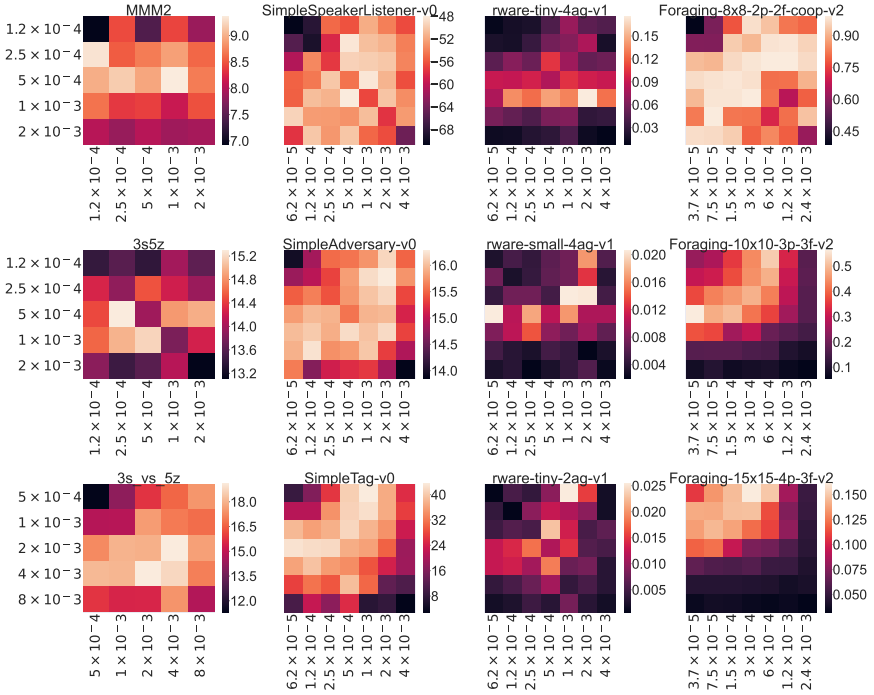


Figure 8: Final performance of IQL with different learning rate combinations. These heatmaps are for the best switching period values. It's clear that in many tasks, non-diagonal values (SIQL) have relatively better performance compared to diagonal values (IQL).

C HYPERPARAMETERS

Table 1: Hyperparameters for IPPO without parameter sharing.

	MPE	SMAC	LBF	RWARE
hidden dimension	128	64	128	128
learning rate	0.0001	0.0005	0.0001	0.0005
reward standardisation	True	True	False	False
network type	FC	FC	GRU	FC
entropy coefficient	0.01	0.001	0.001	0.001
target update	0.01 (soft)	0.01 (soft)	200 (hard)	0.01 (soft)
n-step	10	10	5	10

Table 2: Hyperparameters for IQL without parameter sharing.

	MPE	SMAC	LBF	RWARE
hidden dimension	128	64	64	64
learning rate	0.0005	0.0005	0.0003	0.0005
reward standardisation	True	True	True	True
network type	FC	GRU	GRU	FC
target update	0.01 (soft)	200 (hard)	200 (hard)	0.01 (soft)