DEALING WITH NON-STATIONARITY IN DECENTRALIZED COOPERATIVE MULTI-AGENT DEEP REINFORCEMENT LEARNING VIA MULTI-TIMESCALE LEARNING

Hadi Nekoei[*]
Mila, Université de Montréal

Akilesh Badrinaaraayanan
Mila, Université de Montréal

Amit Sinha
Mila, University of McGill

Mohammad Amini
Mila, University of McGill

Janarthanan Rajendran
Mila, Université de Montréal

Aditya Mahajan
Mila, University of McGill

Sarath Chandar
Mila, Polytechnique Montréal

ABSTRACT

Decentralized cooperative multi-agent deep reinforcement learning (MARL) can be a versatile learning framework, particularly in scenarios where centralized training is either not possible or not practical. One of the critical challenges in decentralized deep MARL is the non-stationarity of the learning environment when multiple agents are learning concurrently. A commonly used and efficient scheme for decentralized MARL is independent learning in which agents concurrently update their policies independently of each other. We first show that independent learning does not always converge, while sequential learning where agents update their policies one after another in a sequence is guaranteed to converge to an agent-by-agent optimal solution. In sequential learning, when one agent updates its policy, all other agent's policies are kept fixed, alleviating the challenge of non-stationarity due to simultaneous updates in other agents' policies. However, it can be slow because only one agent is learning at any time. Therefore it might also not always be practical. In this work, we propose a decentralized cooperative MARL algorithm based on multi-timescale learning. In multi-timescale learning, all agents learn simultaneously, but at different learning rates. In our proposed method, when one agent updates its policy, other agents are allowed to update their policies as well, but at a slower rate. This speeds up sequential learning, while also minimizing non-stationarity caused by other agents updating concurrently. Multi-timescale learning outperforms state-of-the-art decentralized learning methods on a set of challenging multi-agent cooperative tasks in the *epymarl* (Papoudakis et al., 2020) benchmark. This can be seen as a first step towards more general decentralized cooperative deep MARL methods based on multi-timescale learning.

1 INTRODUCTION

In many emerging reinforcement learning (RL) applications, multiple agents interact in a shared environment. There are three types of such multi-agent environments: (i) Environments where agents are competitive and have an objective of maximizing their individual rewards: examples include games such as Poker (Brown & Sandholm, 2018), online auctions, and firms interacting in a market; (ii) Environments where agents are cooperative and have an objective of maximizing a common reward: examples include games such as Hanabi (Bard et al., 2020), multi-agent robotics (Kober et al., 2013), networked control systems (Yüksel & Basar, 2013), power-grid systems (Mai et al., 2023), and self-driving cars; (iii) Environments where agents have mixed strategies and can be both cooperative and competitive: examples include games such as Football (Kurach et al., 2019) and Starcraft (Vinyals et al., 2019). In this paper, we focus on cooperative multi-agent environments.

When the system dynamics and reward function are known, cooperative multi-agent environments are studied using team theory (Marshack & Radner, 1972), cooperative game theory (Shapley, 2016), or decentralized stochastic control (Nayyar et al., 2013). When the system dynamics and reward function are unknown, they are investigated using multi-agent reinforcement learning (MARL). One of the main challenges in cooperative MARL is the *non-stationarity*

[*] Correspondence to: nekoeihe@mila.quebec.

of the learning environment. When multiple agents are learning and updating their policies concurrently, the transition dynamics and rewards are not stationary from a single agent's point of view since the next state of the environment is a function of the joint action of all agents and not only that agent's own action. The problem of non-stationarity becomes even more severe when the environment is partially observable which results in incomplete and asymmetric information across agents.

One setting which is commonly used in the literature to circumvent the challenge of non-stationarity is to assume that agents are trained in an environment where a centralized critic can access the observations and actions of all agents (and potentially some or all components of the state). This centralized critic computes a centralized action-value function, which is then used by all agents to determine policies that can be executed in a decentralized manner by agents using just the local information available to them. This learning paradigm is called *centralized training and decentralized execution* (CTDE). Although CTDE is able to circumvent the conceptual challenges of non-stationarity of the environment and partial observability, it is not an ideal solution in all scenarios. CTDE is only applicable when there is a centralized critic which has access to the observations and actions of all the agents. It is not always possible to construct such a centralized critic, especially in online real-world settings. For example, self-driving cars cannot share their policies and observations with other cars on the road in real time.

An alternative that does not suffer from the limitations of CTDE is decentralized training, which is the focus of this work. In decentralized training, each agent has access to only its local observations and actions. Here, the challenge of non-stationarity becomes more pronounced. A commonly used scheme for decentralized cooperative deep MARL is to approximate what we call *independent iterative best response* (IIBR) (a form of independent learning) where agents independently and concurrently try to find the best response strategy with respect to other agents' policies. Examples of applying independent learning to MARL include independent PPO (IPPO) (de Witt et al., 2020) and independent Q-learning (IQL) (Tampuu et al., 2017).

An alternate scheme one could use for decentralized cooperative deep MARL is to approximate what we call *sequential iterative best response* (SIBR) (a form of sequential learning), where instead of all agents learning simultaneously, they learn sequentially one after another. The idea of SIBR goes back to fictitious play (Brown, 1951) and has been used by the game theory community widely, but mostly overlooked by the deep MARL community. In SIBR, an agent updates its policy until convergence to the Best Response (BR) strategy for other agents' fixed policies. The updated agent's policy is then fixed and the next agent updates its policy to the BR strategy of the other agents' fixed policies and the cycle continues. This way, only one agent learns at any time, while all other agent's policies remain fixed, circumventing the non-stationarity caused by other agents' policies changing while learning. We showcase an example in section 2 where IIBR does not converge while SIBR converges and prove that this is a general property of SIBR that is guaranteed to converge to an agent-by-agent optimal solution (also called team-Nash equilibrium).

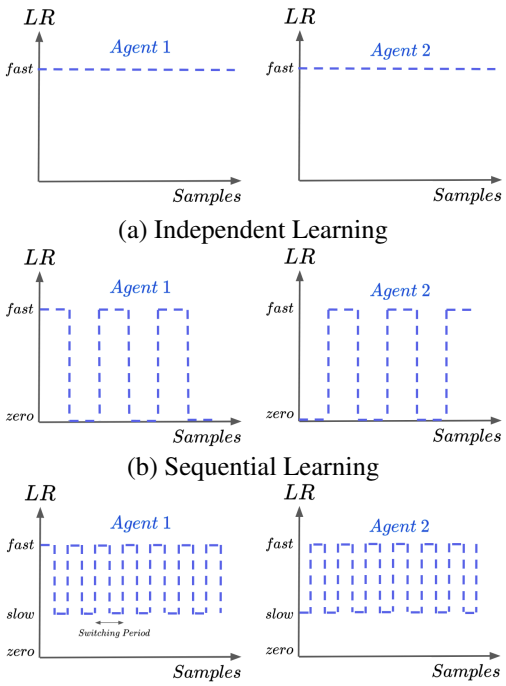


Figure 1: *Difference in learning rate schedules between (a) independent learning, (b) sequential learning, and (c) an instance of multi-timescale learning.*

Although sequential learning can completely side-step the challenge of non-stationarity introduced by other agents learning concurrently, it slows down the learning process because only one agent is learning at any time. To address this issue, we introduce *multi-timescale learning* framework where all agents learn simultaneously, but at different learning rates. In our proposed method, instead of keeping the "non-learning" agents stationary, we allow them to learn using a slower learning rate. The difference between independent learning, sequential learning, and multi-timescale learning is illustrated in Figure 1.

Our hypothesis is that using multi-timescale learning in the way described above can help the currently used independent learning algorithms to deal with non-stationarity better and thereby improve the performance on cooperative decentralized deep MARL tasks. In this work we propose Multi-timescale PPO (MTPPO) and Multi-timescale Q-learning (MTQL) as multi-timescale versions of the two commonly used decentralized deep MARL algorithms: Independent PPO (IPPO) (de Witt et al., 2020) and Independent Q-learning (IQL) (Tampuu et al., 2017). We evaluate MTPPO and MTQL on 12 complex cooperative MARL environments from the *epymarl* (Papoudakis et al., 2020) testbed. Our results

show that multi-timescale algorithms outperform both independent and sequential deep MARL algorithms in most of the tasks. We perform a detailed analysis to understand the performance of multi-timescale learning. Multi-timescale learning is a simple idea that has been usually overlooked by deep MARL practitioners, but it can have a significant improvement in performance and therefore should be one of the tools used in decentralized cooperative deep MARL.

2 IIBR vs SIBR

In this section, we provide an example where IIBR does not converge while SIBR converges and prove that this is a general property of SIBR that is guaranteed to converge to an agent-by-agent optimal solution.

Consider an n-agent MARL problem. Let θ^i denote the policy parameters of agent i and let $\theta = (\theta^1, \cdots, \theta^n)$ denote the policy parameters of the n-agents team. Given policy parameters θ, we use $J(\theta)$ to denote the performance of the team. Furthermore, the notation θ^{-i} refers to the policy parameters of all agents other than that of agent i's. We consider iterative methods of the form: $\theta_{t+1} = F_t(\theta_t)$ to update policy parameters, where θ_t is the policy parameters at iteration t and F_t is some generic update function. Best Response (BR) dynamics is a popular class of update scheme and the most common form of iterative BR dynamics are IIBR and SIBR.

IIBR is an iterative policy update scheme where at iteration t, for all $i \in \{1, \cdots, n\}$, agent i chooses its policy parameters to be the BR to θ_t^{-i}.

$$\theta^i_{t+1} = \arg\max_{\theta^i} J(\theta^i, \theta_t^{-i}) \ .$$

SIBR is an iterative policy update scheme where at iteration t, only agent $j = (t \bmod n) + 1$ updates its policy parameters to be the BR to θ_t^{-j}, and all other agents remain frozen.

$$\theta^i_{t+1} = \begin{cases} \arg\max_{\theta^i} J(\theta^i, \theta_t^{-i}), & \text{if } i = (t \bmod n) + 1, \\ \theta^i_t & \text{otherwise.} \end{cases}$$

Note that IIBR suffers from non-stationarity of the environment because all agents are updating in parallel but SIBR does not suffer from that non-stationarity. We now present an example which shows that the non-stationarity can lead to non-convergence of IIBR while SIBR converges.

Example 1. Consider a multi-agent estimation problem for minimizing team mean-squared error introduced by Afshari & Mahajan (2021). 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 $v_i \sim \mathcal{N}(0, 0.5)$ 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 in the multi-agent estimation problem is to minimize the team mean-squared estimation error $\mathbb{E}\big[\big(x - \frac{1}{3}\sum_{i=1}^3 \hat{z}_i\big)^2\big]$. Minimizing team mean-squared estimation error requires the agents to cooperate to minimize the distance between the average of their estimations and the true state of nature.

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 derived by writing the first-order optimality conditions for the total expected error and setting the derivative to zero (for more details refer to appendix A).

Iterative best response corresponds to solving the system $\Gamma K = \eta$ iteratively as $K^{(t+1)} = M^{-1}(NK^{(t)} + \eta)$ for appropriate choice of M and N. This may be viewed as a linear system $K^{(t+1)} = AK^{(t)} + B\eta$, which is stable when the eigenvalues of A lie within the unit circle.

We now compute the A-matrix for IIBR and SIBR. 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). In IIBR, all agents update their policy at the same time. So, for this example, IIBR is same as the Jacobi method for solving a system of linear equations for which $M = D$ and $N = -(L + U)$. Hence $A_{IIBR} = -D^{-1}(L + U)$. Note that the eigenvalues of A_{IIBR} are $\{-\frac{4}{3}, \frac{2}{3}, \frac{2}{3}\}$. Thus, the spectral radius of A_{IIBR} is $\frac{4}{3} > 1$ which is outside of the unit circle. Hence, IIBR does not converge.

In SIBR, agents update their policies one by one. So, for this example, the sequential iterative best response is the same as the Gauss Seidel method for solving a system of linear equations for which $M = (D + L)$ and $N = -U$. Hence, $A_{SIBR} = -(D + L)^{-1}U$. Note that the eigenvalues of A_{SIBR} are $\{0, \frac{1}{27}(14 \pm \sqrt{20}i)\}$. Thus, the spectral radius of A_{SIBR} is $6\sqrt{6}/27 < 1$. Hence, SIBR converges.

This example illustrates that SIBR which circumvents the problem of non-stationarity converges, while IIBR does not. We now show that this is a general property of SIBR, i.e., SIBR is guaranteed to converge.

Proposition 1. *When the per-step rewards are bounded, then the performance of SIBR converges. Moreover, the policy parameters of the agents converge to an agent-by-agent policy. In particular, let θ^i_t denote the policy parameters of agent i at time t. Let $\theta^{i\star}$ be any limit point of $\{\theta^i_t\}$ along the time-steps when agent i updates its policy parameters. Then $(\theta^{1\star}, \theta^{2\star})$ is agent-by-agent optimal.*

Proof. For the simplicity of exposition, we consider a 2 player team game. The same argument applies to a general n player team game. Let (θ^1_t, θ^2_t) denote the parameters of player 1 and player 2 at iteration t and let $J(\theta^1_t, \theta^2_t)$ denote the performance of the team. We assume that players update their policies following SIBR in the order $1 \to 2 \to 1 \to 2 \to \cdots$. At odd iterations $\theta^1_{(2t+1)} = \arg\max_{\theta^1} J(\theta^1, \theta^2_{(2t)})$ and $\theta^2_{(2t+1)} = \theta^2_{(2t)}$. Similarly, at even iterations $\theta^1_{(2t)} = \theta^1_{(2t-1)}$ and $\theta^2_{(2t)} = \arg\max_{\theta^2} J(\theta^1_{(2t-1)}, \theta^2)$.

Therefore, we have

$$J\left(\theta^1_0, \theta^2_0\right) \leq J\left(\theta^1_1, \theta^2_1 = \theta^2_0\right) \leq J\left(\theta^1_2 = \theta^1_1, \theta^2_2\right) \leq \cdots.$$

For any iteration, we have

$$J\left(\theta^1_{(2t)} = \theta^1_{(2t-1)}, \theta^2_{(2t)}\right) \quad \leq \quad J\left(\theta^1_{(2t+1)}, \theta^2_{(2t+1)} = \theta^2_{(2t)}\right) \quad \leq \quad J\left(\theta^1_{(2t+2)} = \theta^1_{(2t+1)}, \theta^2_{(2t+2)}\right) \quad \leq \quad \cdots.$$

$J(\theta^1_t, \theta^2_t)$ is a non-decreasing sequence and is bounded from above (because the rewards are bounded). Hence, it must converge to a limit. Let $J^\star$ denote this limit. Moreover, let $\theta^{1\star}$ be any limit point of $\{\theta^1_{2t+1}\}_{t\geq 1}$ and $\theta^{2\star}$ be any limit point of $\{\theta^2_{2t}\}_{t\geq 1}$. Then, it must be the case that

$$\max_{\theta^1} J(\theta^1, \theta^{2\star}) = J^\star \quad \text{and} \quad \max_{\theta^2} J(\theta^{1\star}, \theta^2) = J^\star.$$

Thus, $(\theta^{1\star}, \theta^{2\star})$ is an agent-by-agent optimal solution. □

Note that SIBR is guaranteed to converge only to an agent-by-agent optimal solution, where unilateral deviations by an agent do not improve performance. This is a weaker notion of a solution than global optimality. However, under certain assumptions (such as when J is concave in (θ^1, θ^2)), agent-by-agent optimality implies global optimality. See Marshack & Radner (1972); Yüksel & Basar (2013) for a discussion.

3 Multi-Timescale Learning for Decentralized Cooperative Deep MARL

In settings of consideration in this paper, where the true model is unknown, and the environment is large and complex that the policy is parameterized using deep neural network function approximator, it is not possible to calculate the exact BR to other agents' policies. Agents approximate BR by computing noisy gradients and using gradient ascent iteratively to update their policy parameters. In a gradient-based implementation of SIBR, there is only one active agent that performs gradient ascent at any time. All agents other than the active agent keep their policy parameters frozen which slows down the overall learning process.

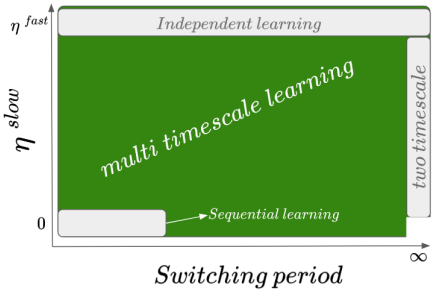


Figure 2: *Multi-timescale learning as a unified framework for independent learning, sequential learning, and two-timescale stochastic approximation. The x-axis indicates how fast agents switch between timescales and the y-axis indicates the learning rate of the slower agent.*

Our key insight is that we can speed up overall learning while still minimizing the perceived non-stationarity by allowing the non-active agents to update their policy parameters as well but at a slower timescale. This is an instance of what we call *multi-timescale learning*, where all agents update their policies concurrently, but at different timescales (learning rates). Inspiration for using multi-timescale learning comes from two-timescale stochastic approximation methods (Borkar, 1997), which are recursive algorithms in which some of the parameters are updated using smaller step-sizes compared to the remaining parameters (Konda & Tsitsiklis, 2004). This principle has been also widely used to train actor-critic algorithms (Konda & Tsitsiklis, 1999). However, we use the idea of multi-timescale algorithms in a different manner, as we explain below.

To illustrate multi-timescale learning, let us consider n agents $\{\theta^i\}_{i=1}^n$ getting trained with H levels of learning rates $\{\eta^h\}_{h=0}^{H-1}$. 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 η^h. The

switching period (s) controls how frequently agents rotate among different clusters (timescales). 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 η^0, while all the agents in $\boldsymbol{c}^1$ will be trained with η^1.

Even though multi-timescale learning can be implemented with more than two timescales, in this work, we focused on having only two timescales. In any period, one agent $\boldsymbol{c}^0 = \{\theta^i\}$ learns at rate η^{fast} and all other agents $\boldsymbol{c}^1 = \{\theta^{-i}\}$ learn at rate η^{slow}. If we set $\eta^{\text{slow}} = \eta^{\text{fast}}$, then multi-timescale learning reduces to independent learning. Similarly, if we set $s = \infty$, multi-timescale learning reduces to a standard two-timescale stochastic approximation, where agents are learning at different learning rates, but the learning rates do not switch over time. Furthermore, if we set $\eta^{\text{slow}} = 0$ multi-timescale learning reduces to sequential learning, where only one agent at a time updates its gradient. Thus, independent learning, sequential learning, and two-timescale stochastic approximation can all be considered special cases of multi-timescale learning. This is illustrated in Figure 2.

In this paper, we propose multi-timescale versions of two commonly used decentralized cooperative deep MARL algorithms: IPPO and IQL. They are the methods with the best performance among all decentralized MARL algorithms on the various tasks in the *epymarl* benchmark (Papoudakis et al., 2020). We propose Multi-timescale Proximal Policy Optimization (MTPPO) which is based on the IPPO algorithm and Multi-timescale Q-Learning (MTQL) which is based on the IQL algorithm. The pseudo-code for MTPPO is shown in Algorithm 1; the pseudo-code for MTQL can be expressed similarly.

Algorithm 1 Multi-timescale PPO

Input learning rate schedules including $\{\eta^{\text{fast}}, \eta^{\text{slow}}\}$ and switching period s
Initialize actors $\pi^i(\theta)$ and critics $Q^i(\phi)$, $i = \{1, \ldots, n\}$
Initialize faster agent $i^* = 1$
for *t=1 to max-train-steps* **do**
  if $t \pmod s == 0$ **then**
    set faster agent $i^* = (i^* + 1 \mod n) + 1$
  end
  $\pi^{i^*}(\theta), Q^{i^*}(\phi) \leftarrow$ PPO update step with η^{fast}
  $\pi^{i}(\theta), Q^{i}(\phi) \leftarrow$ PPO update step with $\eta^{\text{slow}}, \forall i \neq i^*$
end

4 Experiments

We evaluate our hypothesis that agents learning at different timescales improve decentralized cooperative deep MARL compared to agents learning independently at one timescale through rigorous experiments. This section is organized as follows: in section 4.1, we explain the experimental setup used, in section 4.2, we compare the performance of using multi-timescale learning with independent learning, and in section 4.3, we provide a detailed analysis of the experimental results using multi-timescale learning.

4.1 Experimental Setup

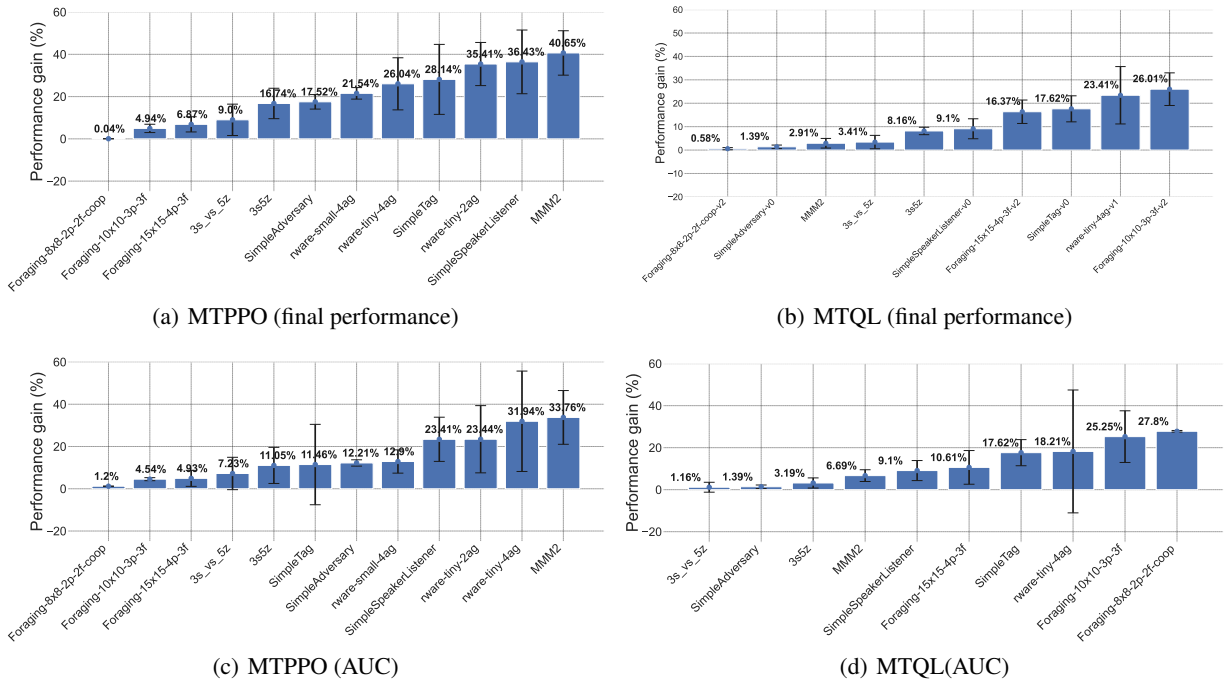
We consider 12 tasks from four different and complex cooperative MARL environments from *epymarl* benchmark (Papoudakis et al., 2020): Multi-Agent Particle Environment (MPE) (Lowe et al., 2017), StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019), Level-Based Foraging (LBF) (Albrecht & Ramamoorthy, 2015), and Multi-Robot Warehouse (RWARE) (Christianos et al., 2020). A brief description of the environments and tasks is provided below.

Multi-Agent Particle Environment (MPE) (Lowe et al., 2017): MPE environment comprises two-dimensional navigation tasks that require coordination to be solved. We include three tasks from the MPE environment: Speaker-Listener, Adversary, and Tag.

Level-Based Foraging (LBF) (Albrecht & Ramamoorthy, 2015): In the LBF environment, agents should cooperate to collect food items that are scattered randomly in a grid-world. We include three tasks from LBF environment: 8×8-2p-2f-c, 10×10-3p-3f and 15×15-4p-3f with varying world-size, number of agents and food items.

Multi-Robot Warehouse (RWARE) (Christianos et al., 2020): RWARE simulates a grid-world warehouse in which agents (robots) must locate and deliver requested shelves to workstations and return them after delivery. We include three partially observable tasks from RWARE environment: tiny-4ag, tiny-2ag and small-4ag. The convention for environment name is {grid-size}-{player count}ag.

StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019): SMAC simulates battle scenarios in which a team of controlled agents must destroy an enemy team. We include three tasks from SMAC environment: MMM2 (10 agents), 3s5z (8 agents) and 3s_vs_5z (3 agents) with a different number of agents and levels of difficulty. For all these environments, refer to Appendix B for detailed descriptions of the tasks and hyperparameters.



(a) MTPPO (final performance)

(b) MTQL (final performance)

(c) MTPPO (AUC)

(d) MTQL(AUC)

Figure 3: *Performance gain of MTPPO and MTQL relative to IPPO and IQL on all 12 tasks. IQL and MTQL almost have zero return on small-4ag and tiny-2ag. That's why these two tasks are excluded from (b). MTPPO always either improves or performs as well as IPPO with the highest gains in MMM2, RWARE-tiny-{4ag, 2ag}, and MPE-Speaker-Listener. Similar performance gains can be seen with MTQL as well across all tasks with maximum gains in RWARE-tiny-4ag and Foraging-10×10-3p-3f. Error bars represent the standard error over 5 seeds.*

4.2 Multi-Timescale vs Independent Learning

In all our multi-timescale experiments, we have two timescales with learning rates lr_0 and lr_1. 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. We perform hyperparameter search by varying the learning rates (lr_0, lr_1) over $L \times L$, where L is chosen by considering the learning rates around the best hyperparameters reported in Papoudakis et al. (2020). Refer to Appendix B for details. We vary the switching period hyperparameter $s \in \{1, 10, 10^2, 10^3, 10^4\}$. In IPPO, critic and actor updates might be done with different frequencies. In such a case switching is done based on critic training steps.

We compare the performance of MTPPO and MTQL with those of IPPO and IQL. Note that, by definition, multi-timescale learning includes $lr_0 = lr_1$ as well. Therefore, to ensure that any improvement in the performance reported is the result of having *different* learning rates, we do not report the results of $lr_0 = lr_1$ for MTPPO and MTQL.

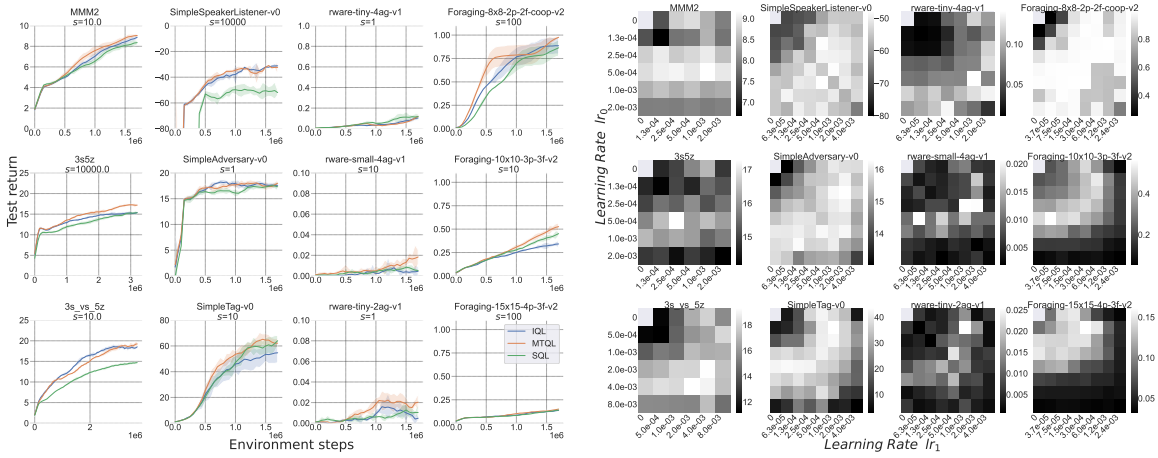
Following Papoudakis et al. (2020), we normalize the returns of all algorithms in each task in the [0, 1] range using the following formula: $(G_t^a - min(G_t))/(max(G_t) - min(G_t))$ where G_t^a is the return of algorithm a in task t, and G_t is the returns of all algorithms in task t. As shown in Table 1, we report the aggregate performance of the algorithms across all the 12 environments.

Table 1: Aggregate performance across all 12 tasks.

	Mean	Median
IQL	0.363	0.464
MTQL	0.404	0.507
IPPO	0.534	0.578
MTPPO	0.599	0.714

Figure 3 shows the performance gain of MTPPO relative to IPPO as well as MTQL relative to IQL across the 12 tasks. MTPPO always either improves or performs as good as IPPO with highest gains in MMM2, RWARE-tiny-{4ag, 2ag}, and MPE-Speaker-Listener. Similar performance gains can be seen with MTQL as well across all tasks with maximum gains in RWARE-tiny-4ag and Foraging-10×10-3p-3f. This also shows that multi-timescale learning can be effective for tasks with different varying numbers of agents (from 2 - 10 in our tasks). Detailed observations on each environment are as follows:

MPE: In the case of Speaker-Listener and Adversary, MTPPO almost always performs better than IPPO while in the case of Tag, MTPPO and IPPO have comparable performance. MTQL clearly performs better than IQL in all tasks.



(a) (b)

Figure 4: *(a) Learning curves for IQL, MTQL, and SQL. MTQL improves the performance in almost all the tasks compared with Sequential Q-learning (SQL) (where one of the learning rates is zero). SQL sometimes outperforms IQL, for example, in SimpleTag and Foraging* 10×10 *while still worse than MTQL. (b) Final performance of IQL (diagonal), SQL (top row and leftmost column), and MTQL (the entire grid) with different learning rate combinations. These heatmaps are for the best switching period values. In many tasks, the best performance results from an off-diagonal learning rate combination, which is possible only through multi-timescale learning.*

LBF: MTPPO performs better than IPPO in all these environments. A similar trend is seen with MTQL and IQL.

RWARE: As we can observe from Figure 3, MTPPO almost always performs better than IPPO in the case of tiny-4ag and tiny-2ag, while the performance is comparable in small-4ag. IQL and MTQL almost have zero return on small-4ag and tiny-2ag. To avoid reporting misleading performance gains, these two tasks are excluded. However, MTQL clearly performs better than IQL on tiny-4ag.

SMAC: In all three tasks, MTPPO performs consistently better than IPPO. MTQL outperforms IQL in MMM2 and 3s5z, while we see comparable performance in 3s_vs_5z.

4.3 Analysis

We perform a detailed analysis of our experiments to study the following questions:

Does multi-timescale learning accelerate sequential learning? Figure 4(a) shows the learning curves for the best version of IQL, MTQL, and Sequential Q-learning (SQL). We implement sequential learning by choosing a zero learning rate for the slower agent. The switching period is still optimized for sequential learning. MTQL improves both the performance and sample complexity in almost all the tasks compared with SQL, and also IQL. Interestingly, sequential learning sometimes outperforms independent learning, for example, in SimpleTag and Foraging 10×10-3p-3f while still worse than multi-timescale learning. For more learning curves, see Appendix C.

How does multi-timescale learning's performance vary with different timescales (learning rates)? We report the performance results of MTQL, SQL, and IQL for each combination of learning rates across all switching periods in Figure 4. Diagonal values with the same learning rates denote IQL. The top row and the leftmost column of the heatmaps show the performance of SQL, and the remaining off-diagonal values represent multi-timescale learning excluding independent learning and sequential learning. In several tasks, there seems to be a pattern with higher performance resulting from these off-diagonal learning rates corresponding to multi-timescale learning rates which are non-zero and different from each other. For the heatmaps of MTPPO, refer to Appendix C.

How does multi-timescale learning's performance vary with different switching periods? Figure 5 shows MTQL's performances across different switching periods (Refer to Figure 14 for MTPPO's results). Overall multi-timescale learning seems to be robust with respect to the choice of the switching period. There are two cases where finding a suitable switching period seems necessary for good performance. Firstly, in tasks where the difference between the best combination of learning rates is high, such as Foraging-10×10-3p-3f, and Foraging-15×15-4p-3f (Figure 4), we observe in Figure 5 that there is a sweet spot for switching.

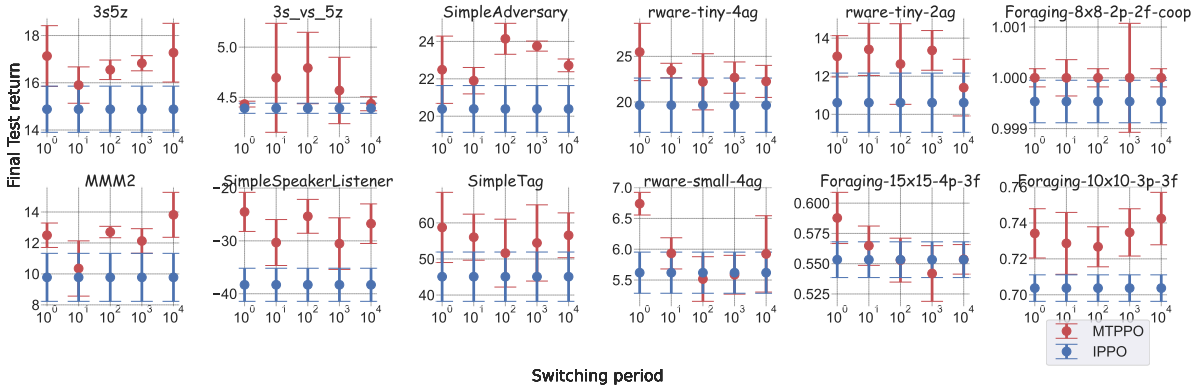


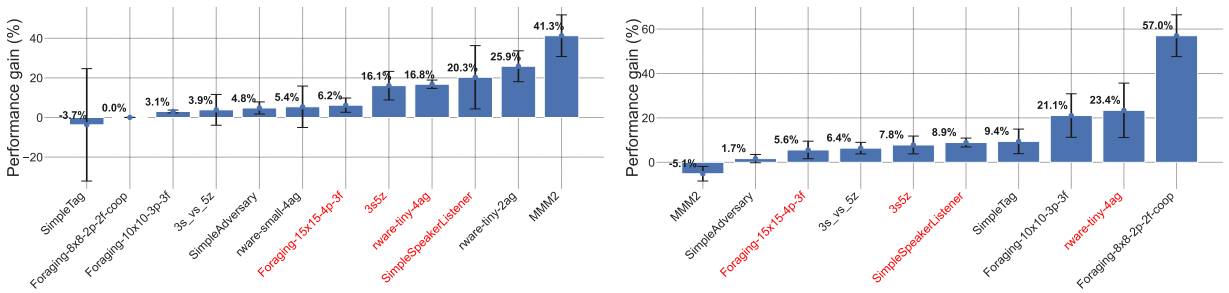
Figure 5: *Performance of IPPO, IQL, and their Multi-timescale version vs switching period for each task. At each switching period, the best-performing multi-timescale learning with different learning rates is compared with the best-performing independent learning with the same learning rates. Error bars represent the standard error over 5 seeds.*

Table 2: Gap between CTDE and DT recovered by Multi-timescale learning. For MTPPO, the considered gap is between IPPO and max(MAPPO, MAAC2) and for MTQL the gap is between IQL and max(VDN, QMIX).

	MPE	SMAC	LBF	RWARE
MTPPO	15.63%	57.65%	29.82%	19.09%
MTQL	56.75%	22.60%	18.26%	–

On which tasks does multi-timescale learning help the most? We observe that multi-timescale learning helps more in environments where there is a gap between CTDE and independent learning. For example, environments such as MPE Speaker-Listener, hard SMAC, and RWARE, where CTDE outperform independent learning (as shown in Papoudakis et al. (2020)), are also environments where multi-timescale learning helps a lot compared to their independent learning counterparts. We hypothesize that while CTDE helps reduce non-stationarity by sharing information about other agents observations and actions, thereby improving coordination and reducing variance (Lowe et al., 2017; Yang et al., 2018; Das et al., 2019; Li et al., 2019), multi-timescale learning improves performance by also reducing non-stationarity, but by controlling non-stationarity that arise from other agents learning concurrently. We would like to emphasize that we do not expect our method to recover all the gap between Decentralized Training (DT) and CTDE since CTDE has access to more information during training than our method that uses DT. However, to provide a picture of where the performance of different methods across different training schemes stand, we computed the percentage of the performance gap between DT and the best CTDE method among Multi-agent PPO (MAPPO)(Yu et al., 2021), Multi-agent Actor-Critic (MAAC2)(Papoudakis et al., 2020), Value Decomposition Networks (VDN) (Sunehag et al., 2017), and QMIX (Rashid et al., 2018) for each environment, that our proposed Multi-timescale DT bridges as $\frac{(MDT-DT)*100}{CTDE-DT}$. As shown in Table 2, MTPPO and MTQL managed to recover some part of the gap between CTDE and DT.

How robust multi-timescale learning is with respect to the hyper-parameteres? To assess the robustness of multi-timescale learning in relation to hyperparameters (HPs), we employed the HP tuning approach described by Papoudakis et al. (2020). Rather than tuning the HPs individually for each task, we conducted HP tuning for a selected subset of tasks. Specifically, we determined the optimal learning rates for both multi-timescale learning and independent learning

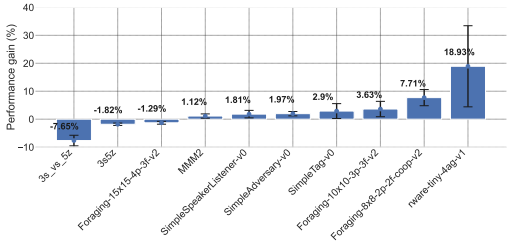


(a) MTPPO (b) MTQL

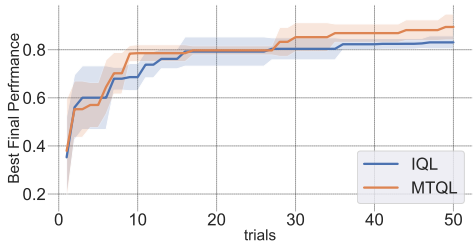
Figure 6: *Performance gain of MTPPO and MTQL relative to IPPO and IQL on all 12 tasks.* ***HPs are tunned only for the tasks shown in red.*** *These results show that the improvements of MTPPO and MTQL over the baselines are relatively robust to HP tunning. Error bars represent the standard error over 5 seeds.*

on Speaker-Listener, Foraging-15×15-4p-3f, RWARE-tiny-4ag, and 3s5z. Subsequently, we evaluated the performance of these methods on other tasks within the same environment, utilizing the identified hyperparameters. Figure 6 presents the corresponding results. The outcomes are varied: while some tasks showed a decline in relative performance gain, others exhibited a higher performance gain for multi-timescale learning. Overall, our approach consistently outperforms the baseline in the majority of tasks, demonstrating the robustness of multi-timescale learning.

How does multi-timescale learning perform compared to independent learning under low computation budget? Here, we test how multi-timescale learning performs relative to independent learning when having access to a fixed low computation budget. To do so, we use hyperparameter tuning tools and show that multi-timescale learning has competitive performance even under a low computation budget. To this end, we perform hyperparameter tuning for both the baselines (only one hyperparameter for IQL and IPPO: $lr0$) and their multi-timescale versions (with three hyperparameters: $lr0$, $lr1$, s) with Orion (Bouthillier et al., 2022). In particular, we used the Tree-structured Parzen Estimator (TPE) algorithm which is one of the Sequential Model-Based Global Optimization (SMBO) algorithms. In these experiments, we start with 20 random initial samples of hyperparameters. For each trial, learning rates are sampled from a *loguniform* distribution and the switching period is chosen from a categorical distribution. The final evaluation return is averaged over 3 independent runs and returned to the TPE algorithm to propose the next set of hyperparameters. TPE proposes 30 extra hyperparameters sequentially so the total number of trials is 50 per method. Then we let both the methods run this optimization process 5 independent times and take the average over these runs to get the final curves like in Fig][(b). We report the results of MTQL on three different environments and 10 tasks in Fig 7(a). Since the IQL agent almost achieves zero performance on two out of three RWARE tasks, we did not include the results since any amount of improvement had a high variance. See Appendix C.2 for other optimization curves and more results. With the limited budget of 50 trials, MTQL performs better than IQL in 7 out of 10 tasks. Note that multi-timescale learning scales better with more compute. As more compute budget is provided, the results will get closer to the result reported in Figure 3 (b), with significantly more performance improvement on all 10 tasks, and also independent learning is a special case of multi-timescale learning.



(a) MTQL performance gain



(b) Optimization curve

Figure 7: *(a) Performance gain of MTQL relative to IQL for a budget of 50 trials. In each run, both methods start with the same initial of 20 points, and the TPE method is used for optimization. The error bars represent the standard error across 5 optimization processes. (b) optimization curve for Foraging-8x8-2p-2f task*

5 Related Work

Handling non-stationarity in cooperative MARL: A lot of work in the cooperative MARL literature has focused on *centralized training and decentralized execution* (CTDE) (Sunehag et al., 2017; Lowe et al., 2017; Rashid et al., 2018; Hostallero et al., 2019; Mao et al., 2020). Although CTDE is able to circumvent the conceptual challenges of non-stationarity of the environment, it may not always be possible to perform centralized training in the first place. For example, in online real-world settings like self-driving cars. Moreover, centralized critic suffers from the curse of agents (Mao et al., 2022; Wang et al., 2020) — the size of joint action space increases exponentially with the number of agents.

However, there have been relatively few efforts in handling non-stationarity in decentralized cooperative MARL settings. Foerster et al. (2017) propose a method to reduce the effect of non-stationarity in IQL by conditioning each agent's value function on a time-dependent fingerprint and report promising results on StarCraft unit micro-management (Samvelyan et al., 2019). A theoretical justification for the decentralized IPPO is provided in Sun et al. (2022), which guarantees monotonic improvements by forming a trust region over joint policies and provides some insight on how to form the trust region for each agent individually. However, these guarantees do not extend to IQL because it is not based on trust regions. Self-play for zero-sum games deals with the non-stationarity problem by playing against several past versions of itself. However, in the case of cooperative environments, self-play does not exploit the fact that the agents can cooperate to optimize the common objective. Self-play also has been shown to learn arbitrary policies that do not generalize when cooperating with novel partners (Bard et al., 2020; Nekoei et al., 2021).

With respect to agents learning at different rates, there have been some works that are 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 learning rate when getting a reward better than the expected state-action value and Omidshafiei et al. (2017) implemented Deep Hysteretic Q-learning. Note that this approach is complementary to multi-timescale learning and will be an interesting future work to evaluate multi-timescale hysteretic Q-learning. A classic work by Bowling & Veloso (2002) focuses specifically on varying the learning rate on a restricted class of iterated matrix games.

Sequential learning and two-timescale learning: The idea of sequential learning goes back to fictitious play (Brown, 1951). However, the combination of Fictitious Self-Play (FSP) with deep RL was proposed recently by Heinrich et al. (2015). Lanctot et al. (2017) proposed Policy-Space Response Oracles (PSRO) to generalize the IIBR, SIBR, and fictitious play methods. PSRO is focused on tackling overfitting in MARL while our approach aims to tackle non-stationarity in the Decentralized Training setting. Moreover, PSRO does not explore this idea of soft learning where we allow a fast learner and a slow learner simultaneously instead of IIBR (both fast) and SIBR (one fast and one not learning). Finally, in contrast to PSRO, our method is conceptually much simpler since it does not require the creation of a population of BR agents and the computation of meta-strategies.

Recently, sequential learning has been studied by a series of works (Bertsekas, 2020; 2021), laying down the theoretical foundations for agent-by-agent policy iteration, value iteration methods, and their optimality guarantees. These works show the promise of SIBR but they are still limited to fully-observable settings. In our work, we are proposing a new setting with switching learning rates and we believe more theoretical work on switching Ordinary Differential Equations (ODEs) is needed, which are beyond the scope of this work, but certainly should be done in future works to understand the method that shows very promising empirical results.

There has been some recent work on competitive decentralized training using two-timescale optimization providing convergence guarantees. A two-timescale decentralized algorithm was developed for zero-sum games by Sayin et al. (2021), where each agent updates its local Q-function and state-value function estimates concurrently, the latter happening at a slower timescale without requiring asymmetric update rules. Also, Daskalakis et al. (2020) show that in a zero-sum game, when two competitive policy gradient-based agents learn simultaneously and their learning rates follow a two-timescale rule, their policies converge to a min-max equilibrium. However, these results are all still limited to zero-sum games.

6 Conclusion and Future Work

The commonly used training scheme for decentralized cooperative deep MARL has been independent learning based on IIBR, which suffers from the non-stationarity of other simultaneous learning agents. Sequential learning on the other hand can circumvent this issue, but it is slow since only one agent learns at any time. In this work, we proposed using the framework of multi-timescale learning, where different agents are learning concurrently at different learning rates for decentralized cooperative deep MARL. In our proposed instantiation of multi-timescale learning, agents learn one after another like in sequential learning, but while one agent learns, all other agents also concurrently update their policies but at a slower learning rate, minimizing the issue of non-stationarity, while not making the overall learning very slow. Our evaluation of Multi-timescale PPO (MTPPO) and Multi-timescale QL (MTQL) on 12 complex cooperative MARL tasks from the *epymarl* benchmark showed that multi-timescale versions outperform both their independent and sequential counterparts in most of the tasks.

This work empirically presented multi-timescale learning as a promising framework for decentralized cooperative deep MARL. Conducting more theoretical work to understand the learning dynamics in multi-timescale learning can be an exciting future work. Even though in this paper, we focused on two standard decentralized algorithms, multi-timescale learning can be applied to other decentralized and even centralized methods like multi-agent PPO (MAPPO) (Yu et al., 2021). In this work, we evaluated MTPPO and MTQL with only two timescales. We assumed that in the case of more than two agents, only one agent is learning at a different timescale while other agents are learning at the same timescale. Clearly, there are more ways to cluster the agents, which might be useful especially if the environment and task are such that there are dependencies and independencies between a certain subset of agents. Evaluation of multi-timescale learning with other algorithms, more timescales, different clustering protocols, and in non-cooperative settings are very interesting 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 protocol in advance in many MARL scenarios, one potentially promising idea is to adaptively tune the learning rates. Overall, we hope this work is a first step towards more decentralized cooperative deep MARL methods based on multi-timescale learning.

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.

Nolan Bard, Jakob N Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, et al. The hanabi challenge: A new frontier for ai research. *Artificial Intelligence*, 280:103216, 2020.

Dimitri Bertsekas. Multiagent value iteration algorithms in dynamic programming and reinforcement learning. *Results in Control and Optimization*, 1:100003, 2020.

Dimitri Bertsekas. Multiagent reinforcement learning: Rollout and policy iteration. *IEEE/CAA Journal of Automatica Sinica*, 8(2):249–272, 2021.

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

Xavier Bouthillier, Christos Tsirigotis, François Corneau-Tremblay, Thomas Schweizer, Lin Dong, Pierre Delaunay, Fabrice Normandin, Mirko Bronzi, Dendi Suhubdy, Reyhane Askari, Michael Noukhovitch, Chao Xue, Satya Ortiz-Gagné, Olivier Breuleux, Arnaud Bergeron, Olexa Bilaniuk, Steven Bocco, Hadrien Bertrand, Guillaume Alain, Dmitriy Serdyuk, Peter Henderson, Pascal Lamblin, and Christopher Beckham. Epistimio/orion: Asynchronous Distributed Hyperparameter Optimization, March 2022. URL https://doi.org/10.5281/zenodo.3478592.

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

George W Brown. Iterative solution of games by fictitious play. *Act. Anal. Prod Allocation*, 13(1):374, 1951.

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

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.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In *International Conference on Machine Learning*, pp. 1538–1546. PMLR, 2019.

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.

Johannes Heinrich, Marc Lanctot, and David Silver. Fictitious self-play in extensive-form games. In *International conference on machine learning*, pp. 805–813. PMLR, 2015.

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.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

Vijay R Konda and John N Tsitsiklis. Convergence rate of linear two-time-scale stochastic approximation. *The Annals of Applied Probability*, 14(2):796–819, 2004.

Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zając, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. Google research football: A novel reinforcement learning environment. *arXiv preprint arXiv:1907.11180*, 2019.

Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Shihui Li, Yi Wu, Xinyue Cui, Honghua Dong, Fei Fang, and Stuart Russell. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4213–4220, 2019.

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

Vincent Mai, Philippe Maisonneuve, Tianyu Zhang, Hadi Nekoei, Liam Paull, and Antoine Lesage-Landry. Multi-agent reinforcement learning for fast-timescale demand response of residential loads. *arXiv preprint arXiv:2301.02593*, 2023.

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.

J Marshack and R Radner. Economic theory of teams. *Cowles Foundation for Research in Economics, 22*, 1972.

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.

Ashutosh Nayyar, Aditya Mahajan, and Demosthenis Teneketzis. Decentralized stochastic control with partial history sharing: A common information approach. *IEEE Transactions on Automatic Control*, 58(7):1644–1658, July 2013.

Hadi Nekoei, Akilesh Badrinaaraayanan, Aaron Courville, and Sarath Chandar. Continuous coordination as a realistic scenario for lifelong learning. In *International Conference on Machine Learning*, pp. 8016–8024. PMLR, 2021.

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.

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.

L. S. Shapley. *A Value for n-Person Games*, pp. 307–318. Princeton University Press, 2016. doi: doi:10.1515/9781400881970-018. URL https://doi.org/10.1515/9781400881970-018.

Mingfei Sun, Sam Devlin, Katja Hofmann, and Shimon Whiteson. Monotonic improvement guarantees under non-stationarity for decentralized PPO. *arXiv preprint arXiv:2202.00082*, 2022.

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.

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.

Lingxiao Wang, Zhuoran Yang, and Zhaoran Wang. Breaking the curse of many agents: Provable mean embedding q-iteration for mean-field reinforcement learning. In *International Conference on Machine Learning*, pp. 10092–10103. PMLR, 2020.

Jiachen Yang, Alireza Nakhaei, David Isele, Kikuo Fujimura, and Hongyuan Zha. Cm3: Cooperative multi-goal multi-stage multi-agent reinforcement learning. *arXiv preprint arXiv:1809.05188*, 2018.

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.

Serdar Yüksel and Tamer Basar. Stochastic networked control systems. *AMC*, 10:12, 2013.

APPENDICES

A MULTI-AGENT ESTIMATION PROBLEM

Consider a general three player minimum team mean squared optimization problem with $x \sim \mathcal{N}(0,1)$, $y_i = x + w_i$ where $w_i \sim \mathcal{N}(0, \sigma^2)$ where the objective is to choose $\hat{z}_i = \mu_i(y_i)$ to minimize an estimation cost of the form

$$\mathbf{E}[(x\mathbf{1} - \hat{z})^T S(x\mathbf{1} - \hat{z})]$$

where $\hat{z} = \text{vec}(\hat{z}_1, \hat{z}_2, \hat{z}_3)$ and

$$S = \begin{bmatrix} p & q & q \\ q & p & q \\ q & q & p \end{bmatrix}$$

According to (Afshari & Mahajan, 2021, Theorem 1), the team optimal estimation strategies are linear and of the form $\hat{z}_i = K_i y_i$, where $K = \text{vec}(K_1, K_2, K_3)$ is given by the solution of the linear system of equations

$$\Gamma K = \eta$$

where

$$\Gamma = \begin{bmatrix} p(1+\sigma^2) & q & q \\ q & p(1+\sigma^2) & q \\ q & q & p(1+\sigma^2) \end{bmatrix} \quad \text{and} \quad \eta = \begin{bmatrix} p+2q \\ p+2q \\ p+2q \end{bmatrix} \tag{1}$$

Now, the cost function in the example described in Sec. 2, the estimation cost may be written as

$$\frac{1}{9}\mathbf{E}\left[\begin{bmatrix} x-\hat{z}_1 \\ x-\hat{z}_2 \\ x-\hat{z}_3 \end{bmatrix}^T \begin{bmatrix} p & q & q \\ q & p & q \\ q & q & p \end{bmatrix} \begin{bmatrix} x-\hat{z}_1 \\ x-\hat{z}_2 \\ x-\hat{z}_3 \end{bmatrix} \right]$$

where $p = 1$ and $q = 1$. In the model, it is also assumed that $\sigma^2 = 0.5$. Thus, equation 1 simplifies to:

$$\Gamma = \begin{bmatrix} \frac{3}{2} & 1 & 1 \\ 1 & \frac{3}{2} & 1 \\ 1 & 1 & \frac{3}{2} \end{bmatrix} \quad \text{and} \quad \eta = \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix}.$$

Iterative best response corresponds to solving the system $\Gamma K = \eta$ iteratively as $K^{(t+1)} = M^{-1}(NK^{(t)} + \eta)$ for appropriate choice of M and N. This may be viewed as a linear system $K^{(t+1)} = AK^{(t)} + B\eta$, which is stable when the eigenvalues of A lie within the unit circle.

We now compute the A-matrix for IIBR and SIBR. 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). In IIBR, all agents update their policy at the same time. So, for this example, IIBR is same as the Jacobi method for solving a system of linear equations for which $M = D$ and $N = -(L+U)$. Hence $A_{IIBR} = -D^{-1}(L+U)$.

$$A_{IIBR} \coloneqq -D^{-1}(L+U) = \begin{bmatrix} 0 & -\frac{2}{3} & -\frac{2}{3} \\ -\frac{2}{3} & 0 & -\frac{2}{3} \\ -\frac{2}{3} & -\frac{2}{3} & 0 \end{bmatrix}.$$

Note that the eigenvalues of A_{IIBR} are $\{-\frac{4}{3}, \frac{2}{3}, \frac{2}{3}\}$. Thus, the spectral radius of A_{IIBR} is $\frac{4}{3} > 1$ which is outside of the unit circle. Hence, IIBR does not converge.

In SIBR, agents update their policies one by one. So, for this example, the sequential iterative best response is the same as the Gauss Seidel method for solving a system of linear equations for which $M = (D+L)$ and $N = -U$. Hence, $A_{SIBR} = -(D+L)^{-1}U$.

$$A_{SIBR} \coloneqq -(D+L)^{-1}U = \begin{bmatrix} 0 & -\frac{2}{3} & -\frac{2}{3} \\ 0 & \frac{4}{9} & -\frac{2}{9} \\ 0 & \frac{4}{27} & \frac{16}{27} \end{bmatrix}.$$

Note that the eigenvalues of A_{SIBR} are $\{0, \frac{1}{27}(14 \pm \sqrt{20}i)\}$. Thus, the spectral radius of A_{SIBR} is $6\sqrt{6}/27 < 1$. Hence, SIBR converges.

B ENVIRONMENT DETAILS AND HYPERPARAMETERS

In this section, we give an overview of the tasks and environments we used for our experiments. Then, we list all the important hyper-parameters.

MPE: 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.

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. The convention for environment name is {grid_size}×{grid_size}-{player count}p-{food locations}f.

RWARE: The convention for environment name is {grid-size}-{player count}ag.

SMAC: 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 SMAC experiments, we only consider 5 learning rates due to its higher computational requirement.

Some important experimental details are listed below:

- For an IPPO agent, we change the learning rate of both the actor and the critic.
- In IPPO, critic and actor updates might be done with different frequencies. In such a case switching is done based on critic training steps.
- We use the Adam (Kingma & Ba, 2014) optimizer in all experiments (we only change the learning rate hyperparameter). Even though Adam adaptively changes the gradient signal, we can still control its scale with changing the learning rate coefficient directly.
- In the case of IPPO, we change the learning rates of both the actor and critic.
- d) 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.

Table 3: Hyperparameters for IPPO without parameter sharing.

	MPE	SMAC	LBF	RWARE
Hidden dimension	128	64	128	128
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 4: Learning rates for MTPPO without parameter sharing.

Environment	Learning rates
MPE	$\{1.25 \times 10^{-5}, 2.5 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}, 2 \times 10^{-4}, 4 \times 10^{-4}, 8 \times 10^{-4}\}$
LBF	$\{1.25 \times 10^{-5}, 2.5 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}, 2 \times 10^{-4}, 4 \times 10^{-4}, 8 \times 10^{-4}\}$
RWARE	$\{6.25 \times 10^{-5}, 1.25 \times 10^{-4}, 2.5 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}, 2 \times 10^{-3}, 4 \times 10^{-3}\}$
SMAC	$\{1.25 \times 10^{-4}, 2.5 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}, 2 \times 10^{-3}\}$

Table 5: Hyperparameters for IQL without parameter sharing.

	MPE	SMAC	LBF	RWARE
Hidden dimension	128	64	64	64
Reward standardisation	True	True	True	True
Network type	FC	GRU	GRU	FC
Target update	0.01 (soft)	200 (hard)	200 (hard)	0.01 (soft)

C More results

Regarding the experimental setup, 5 seeds may not be enough due to high variance in the performance of the baseline algorithms in certain tasks. We did the following to make sure that improvements due to using multi-timscale learning is indeed significant: For environments with high variance in the performance, we ran the experiments for 25 seeds and the results are reported in 7. Compared with the original results with 5 seeds, we can see that the improvements due to multi-timescale learning are statistically significant indeed.

C.1 Learning curves and Heatmaps

In this section, we provided the best learning curves for MTPPO and MTQL in Figures 9 and 10. We also included the learning curves across switching periods in Figures 11 and 12.

We also provide heatmaps of the final performance for MTPPO for all combination of learning rates in Figure 8.

Table 6: Learning rate for MTQL without parameter sharing.

Environment	Learning rates
MPE	$\{6.25 \times 10^{-5}, 1.25 \times 10^{-4}, 2.5 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}, 2 \times 10^{-3}, 4 \times 10^{-3}\}$
LBF	$\{3.7 \times 10^{-5}, 7.5 \times 10^{-5}, 1.5 \times 10^{-4}, 3 \times 10^{-4}, 6 \times 10^{-4}, 1.2 \times 10^{-3}, 2.4 \times 10^{-3}\}$
RWARE	$\{6.25 \times 10^{-5}, 1.25 \times 10^{-4}, 2.5 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}, 2 \times 10^{-3}, 4 \times 10^{-3}\}$
SMAC	$\{\{1.25 \times 10^{-4}, 2.5 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}, 2 \times 10^{-3}\}$

Table 7: Rerunning experiments with 25 seeds for the tasks with high variance. The improvements due to multi-timescale learning are still statistically significant.

	Foraging-10*10-3p-3f	SimpleTag	RWARE tiny4ag
IQL	$0.42(\pm 0.01)$	$57.36(\pm 1.57)$	$0.1243(\pm 0.0158)$
MTQL	$\mathbf{0.48}(\pm 0.02)$	$\mathbf{64.03}(\pm 2.00)$	$\mathbf{0.1405}(\pm 0.0172)$

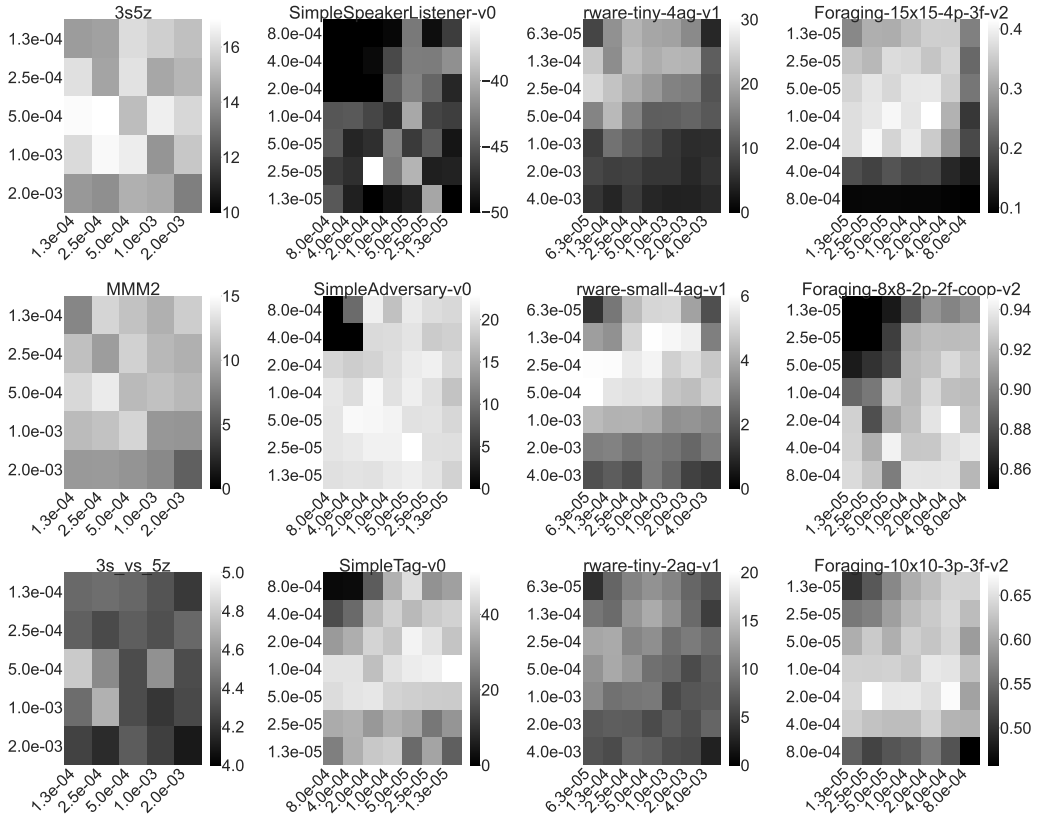


Figure 8: *Final performance of IPPO with different learning rate combinations. These heatmaps are for the best switching period values. It's clear that in many tasks, non-diagonal values (MTPPO) have relatively better performance compared to diagonal values (IPPO).*

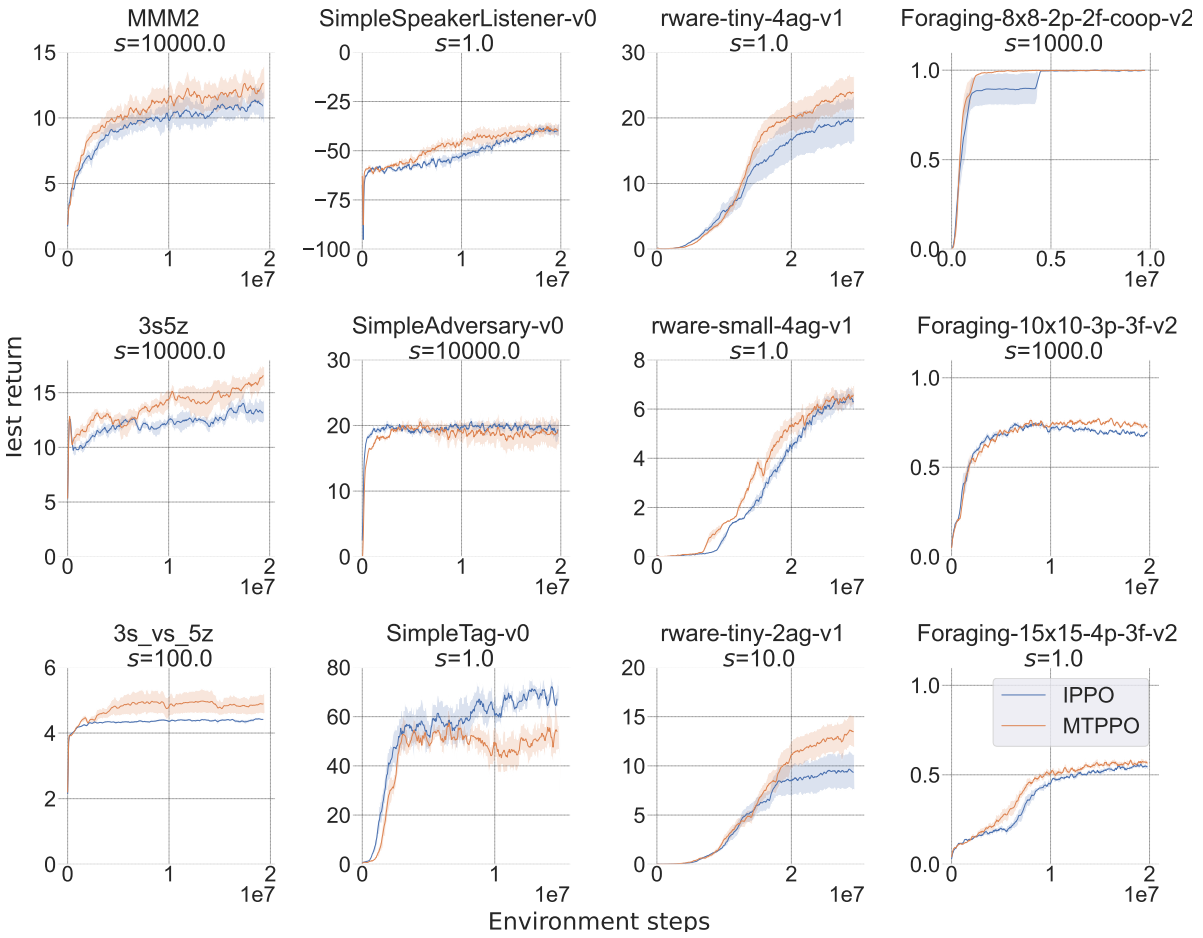


Figure 9: *Learning curves for each task. MTPPO leads to faster convergence than IPPO in many tasks. Solid lines are mean test returns over 100 test episodes averaged over 5 independent seeds. Shaded regions indicates the standard-error. Smoothing with window size = 5 is used.*

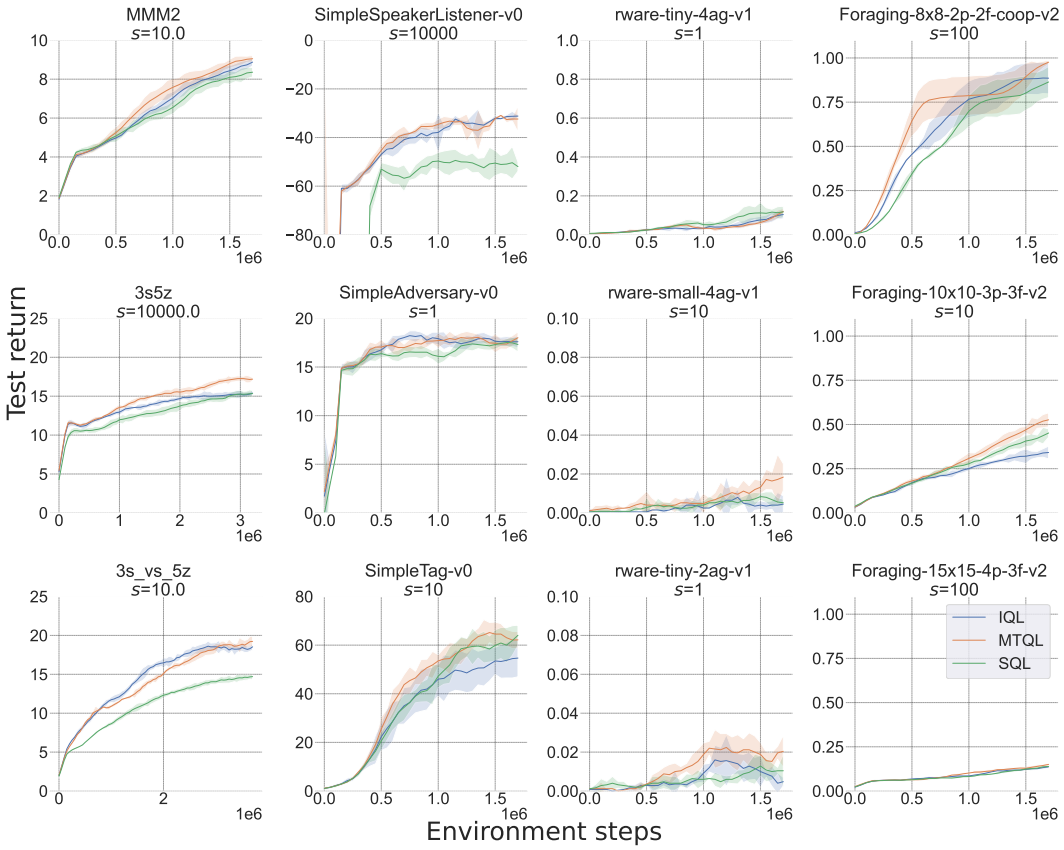
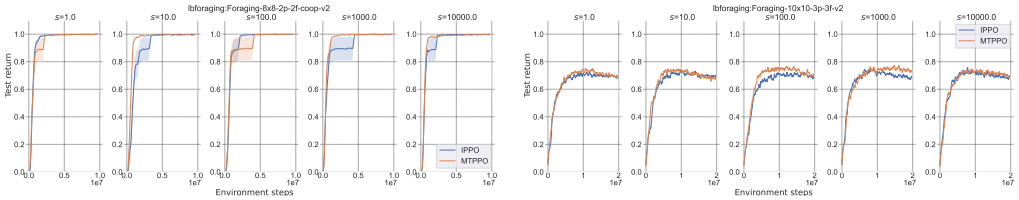
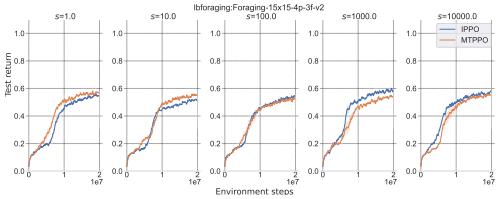


Figure 10: *Learning curves for each task. MTQL leads to faster convergence than IQL in many tasks. Solid lines are mean test returns over 100 test episodes averaged over 5 independent seeds. Shaded regions indicates the standard-error. Smoothing with window size = 5 is used.*

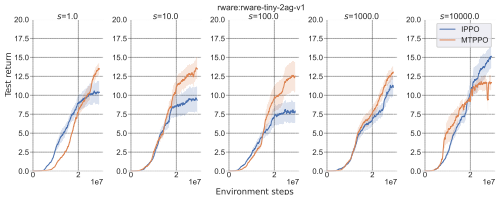


(a) Foraging-8x8-2p-2f-coop-v2

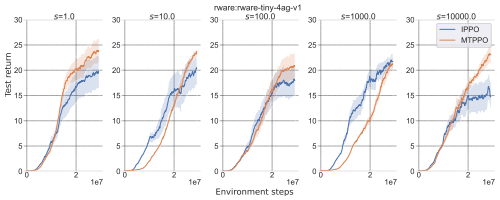
(b) Foraging-10x10-3p-3f-v2



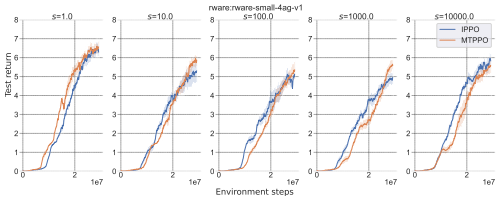
(c) Foraging-15x15-4p-3f-v2



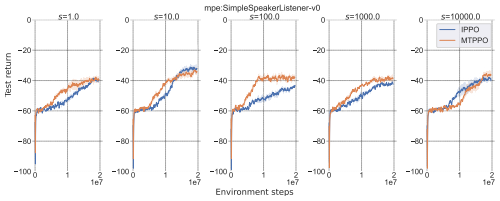
(d) rware-tiny-2ag-v1



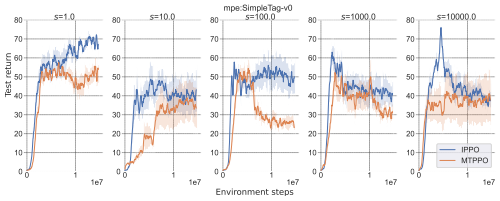
(e) rware-tiny-4ag-v1



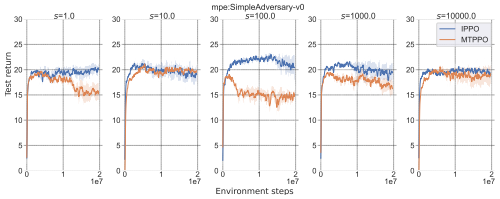
(f) rware-small-4ag-v1



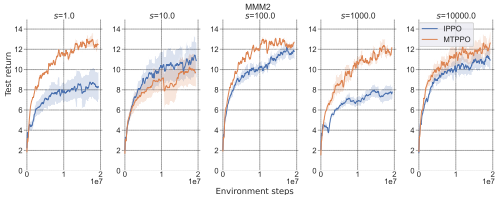
(g) SimpleSpeakerListener-v0



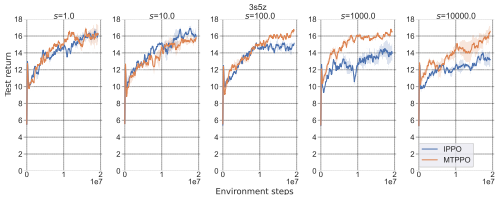
(h) SimpleTag-v0



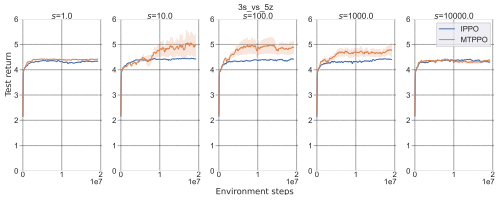
(i) SimpleAdversary-v0



(j) MMM2

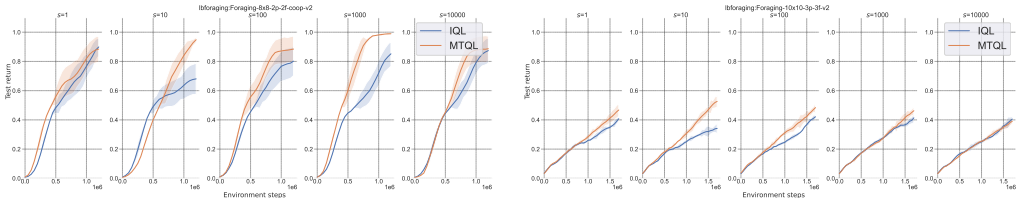


(k) 3s5z



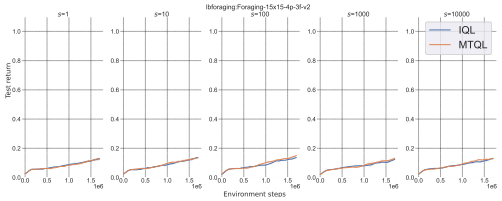
(l) 3s vs 5z

Figure 11: *Learning curves of MTPPO 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. Differnece in IPPO's performance across different switching periods is due to the variance in the results.*

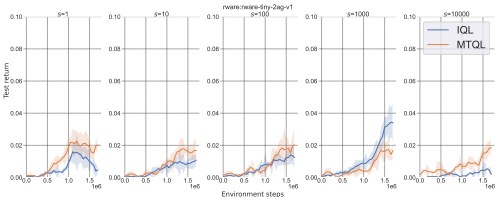


(a) Foraging-8x8-2p-2f-coop-v2

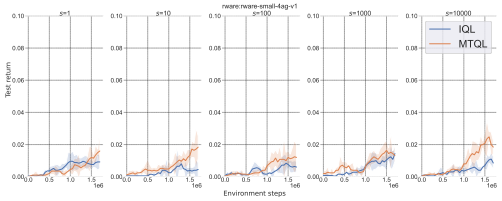
(b) Foraging-10x10-3p-3f-v2



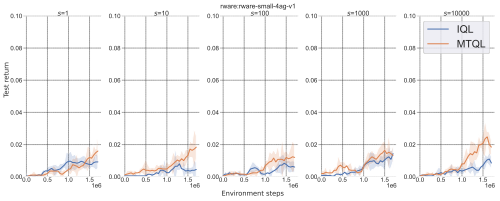
(c) Foraging-15x15-4p-3f-v2



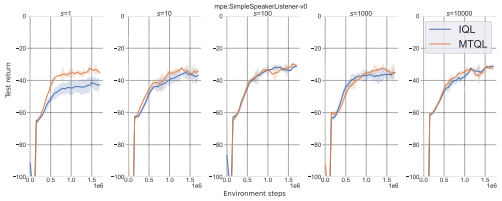
(d) rware-tiny-2ag-v1



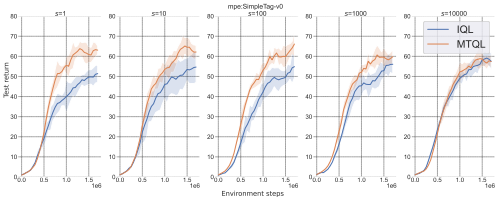
(e) rware-tiny-4ag-v1



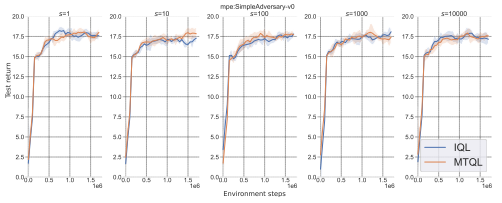
(f) rware-small-4ag-v1



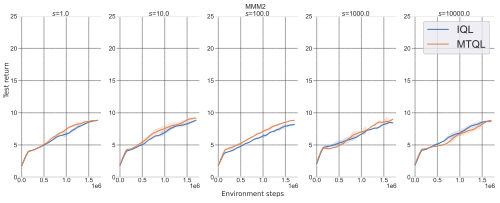
(g) SimpleSpeakerListener-v0



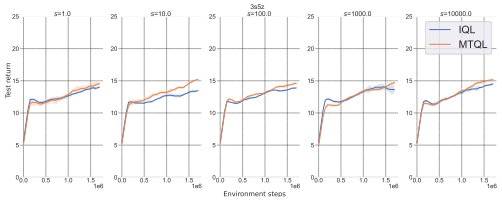
(h) SimpleTag-v0



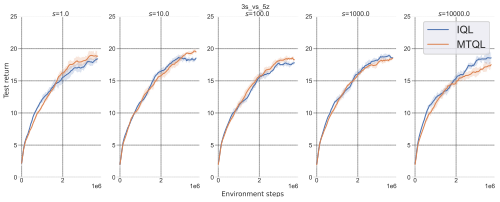
(i) SimpleAdversary-v0



(j) MMM2



(k) 3s5z

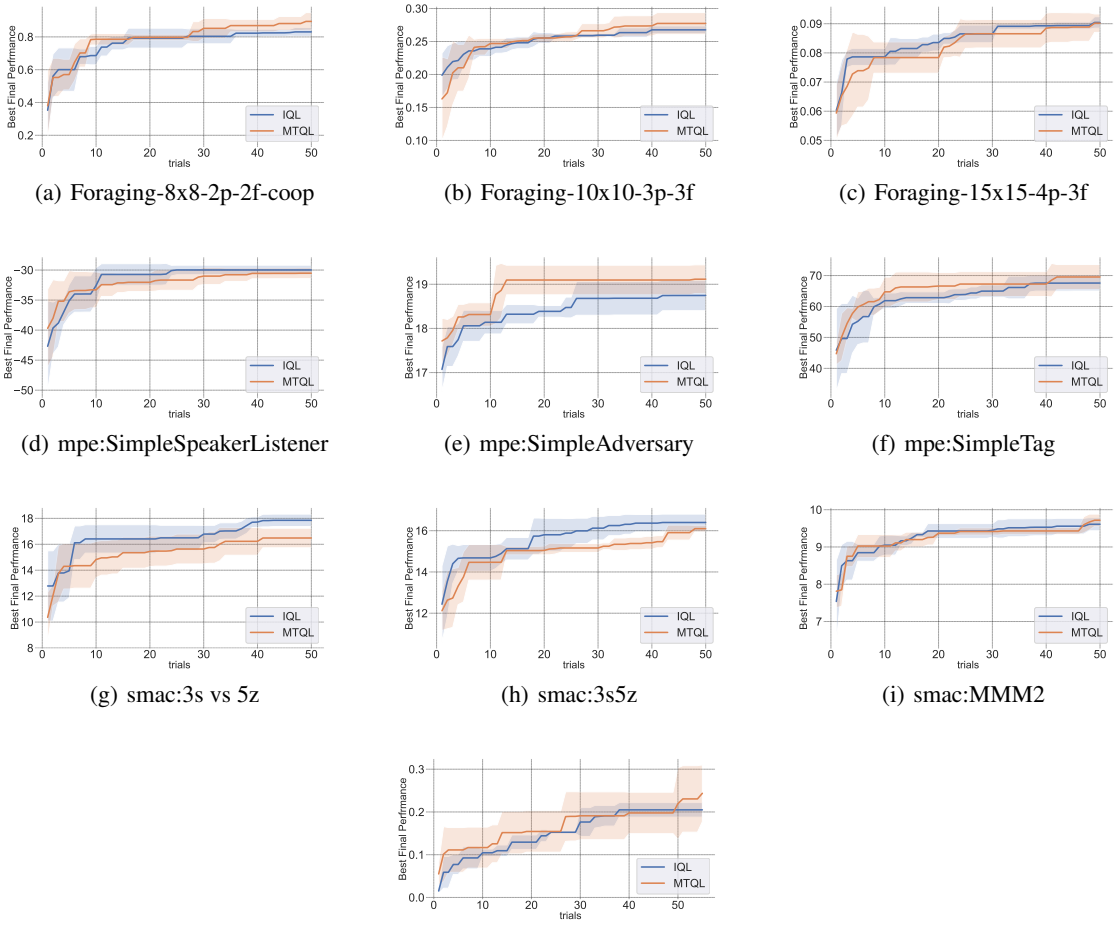


(l) 3s vs 5z

Figure 12: *Learning curves of MTQL 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. Differnece in IQL's performance across different switching periods is due to the variance in the results.*

C.2 ORION EXPERIMENTS

For orion experiments, all the hyperparameters are similar to table 5. However, learning rates and switching periods are sampled from distributions mentioned in table 8.



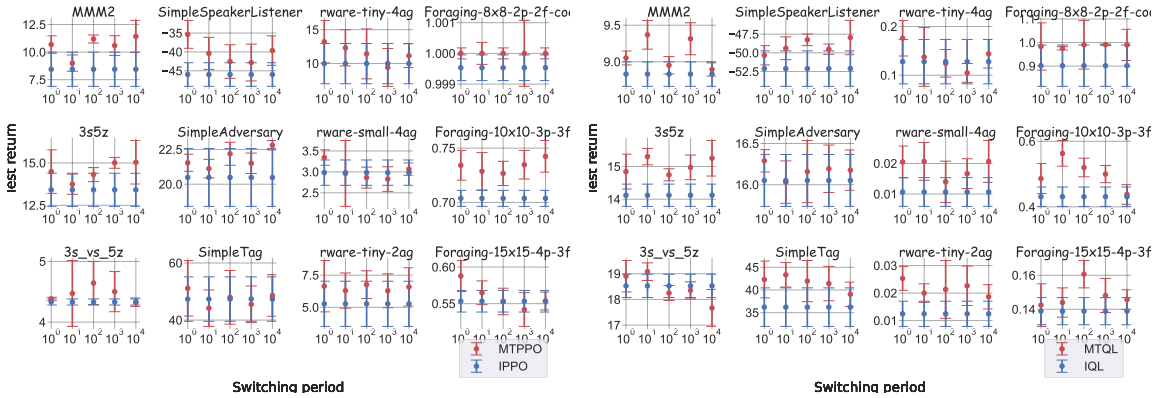
(a) Foraging-8x8-2p-2f-coop (b) Foraging-10x10-3p-3f (c) Foraging-15x15-4p-3f

(d) mpe:SimpleSpeakerListener (e) mpe:SimpleAdversary (f) mpe:SimpleTag

(g) smac:3s vs 5z (h) smac:3s5z (i) smac:MMM2

(j) rware-tiny-4ag

Figure 13: *Orion optimization curves of MTQL and IQL for each task. Solid lines are the best final test return found by TPE algorithm averaged over 5 independent seeds. Shadow region indicates the standard-error.*



(a) MTPPO (b) MTQL

Figure 14: *Performance of IPPO, IQL, and their Multi-timescale version vs switching period for each task. At each switching period, the best-performing multi-timescale learning with different learning rates is compared with the best-performing independent learning with the same learning rates. Error bars represent the standard error over 5 seeds.*

Table 8: Orion hyper parameters.

Environment	Learning rates	Switching period
MPE	$loguniform(1e-05, 0.005)$	$[10, 1000, 100000]$
LBF	$loguniform(5e-06, 0.005)$	$[10, 1000, 100000]$
RWARE	$loguniform(1e-05, 0.005)$	$[10, 1000, 100000]$