Learning Complex Teamwork Tasks using a Sub-task Curriculum

Citation for published version:
Fosong, E, Rahman, A, Carlucho, I & Albrecht, SV 2023 'Learning Complex Teamwork Tasks using a Sub-task Curriculum'. https://doi.org/10.48550/arXiv.2302.04944

Digital Object Identifier (DOI):
10.48550/arXiv.2302.04944

Link:
Link to publication record in Heriot-Watt Research Portal

Document Version:
Early version, also known as pre-print

Learning Complex Teamwork Tasks using a Sub-task Curriculum

Elliot Fosong[1] , **Arrasy Rahman**[2] , **Ignacio Carlucho**[1] , **Stefano V. Albrecht**[1]
[1]University of Edinburgh
[2]University of Texas at Austin
{e.fosong,ignacio.carlucho,s.albrecht}@ed.ac.uk, arrasy@utexas.edu

Abstract

Training a team to complete a complex task via multi-agent reinforcement learning can be difficult due to challenges such as policy search in a large policy space, and non-stationarity caused by mutually adapting agents. To facilitate efficient learning of complex multi-agent tasks, we propose an approach which uses an expert-provided curriculum of simpler multi-agent sub-tasks. In each sub-task of the curriculum, a subset of the entire team is trained to acquire sub-task-specific policies. The sub-teams are then merged and transferred to the target task, where their policies are collectively fined tuned to solve the more complex target task. We present MEDoE, a flexible method which identifies situations in the target task where each agent can use its sub-task-specific skills, and uses this information to modulate hyperparameters for learning and exploration during the fine-tuning process. We compare MEDoE to multi-agent reinforcement learning baselines that train from scratch in the full task, and with naïve applications of standard multi-agent reinforcement learning techniques for fine-tuning. We show that MEDoE outperforms baselines which train from scratch or use naïve fine-tuning approaches, requiring significantly fewer total training timesteps to solve a range of complex teamwork tasks.

1 Introduction

In cooperative multi-agent reinforcement learning (MARL) [Papoudakis *et al.*, 2021], the goal is to have a team of autonomous agents learn to complete a task, by having the team gather and learn from experiences in that task. Although MARL techniques have been used successfully to solve a range of team-based tasks, there are still challenges in complex tasks. These challenges include multi-agent credit assignment, non-stationarity due to simultaneously adapting agents, difficulty searching over a large joint action space, and equilibrium selection problems [Papoudakis *et al.*, 2019]. These problems typically worsen when the number of agents increases, or when complex coordinated behaviours are required.

We propose addressing these problems and solving complex multi-agent tasks by using a curriculum of sub-tasks. In this proposed approach, the team is divided up into sub-teams, where each sub-team is trained on a sub-task. Each sub-task is simpler, typically with a smaller number of agents, but it allows the agents to acquire skills relevant to the target task. We then recombine the sub-teams to form a full team, and fine-tune their existing policies on the complex target task in order to learn any skills not obtainable from the sub-tasks alone. Intuitively, we expect that the existing policies of the agents can bootstrap policy search in the complex target task, as the initial stages of random search are bypassed.

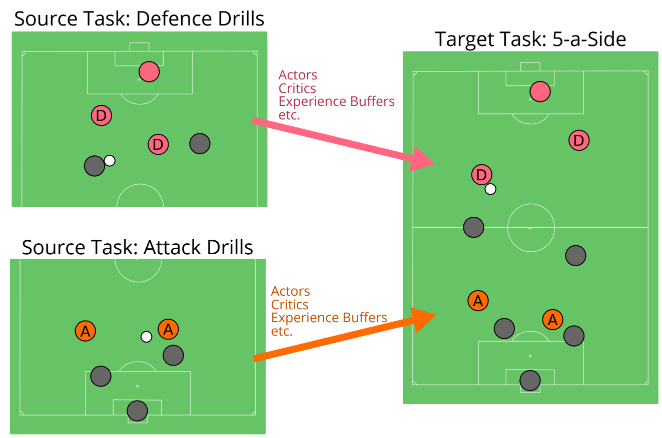


Figure 1: Sub-task Curriculum Diagram for 5-a-side Football

For example, consider training five agents to play 5-a-side football by breaking the problem up into attack drills with two attackers, and defence drills with two defenders and one goalkeeper (Figure 1). The attackers learn skills including "shooting on target" and "avoiding being tackled" which are useful in the full 5-a-side football game. Likewise, the defenders learn defensive skills useful in 5-a-side football. When the attackers and defenders are recombined, extra fine-tuning is required in the full 5-a-side football game, for example to teach defenders that they ought to pass to their attacker teammates.

However, applying standard MARL algorithms to fine-tune sub-task-skilled agents can actually slow training. One problem is overconfidence in out-of-distribution scenarios: a seemingly "skilled" agent might confidently take inappropriate actions, when it should instead be exploring. However, using a high exploration rate in MARL can itself exacerbate equilibrium selection problems and reduce stability of the training process. Therefore, exploration effort should be concentrated in situations which need it most, i.e., when agents' existing policies are inadequate. Another problem that arises particularly in sparse-reward settings is forgetting: while the agents learn to coordinate in the complex target task, they may forget the useful skills they obtained during the sub-task training.

In this paper we present an approach that, given a curriculum of sub-tasks, is able to train a team to complete a complex multi-agent task. Our method, Modulating Exploration and Training via Domain of Expertise (MEDoE), uses a domain of expertise (DoE) classifier to determine when each agent's existing policy is likely to be adequate to solve the complex task. Returning to the football example, a DoE classifier might classify a defensive scenario in 5-a-side football (such as that shown on the right of Figure 1) as belonging to each defender's DoE, as the policies the defenders learned during defence drills are also good in the 5-a-side game in this situation. When an agent should exploit its existing policy, MEDoE reduces non-stationarity by reducing the policy learning rate. On the other hand, when an agent needs to learn new skills, MEDoE increases the exploration temperature to encourage the agent to explore new behaviours. MEDoE also controls the rate at which each agent forgets ineffective behaviours by modulating the entropy regularisation coefficient. Finally, MEDoE uses behaviour priors [Tirumala *et al.*, 2020], controlling the rate at which each agent retains useful skills by modulating the behaviour prior regularisation coefficient. To our knowledge, this paper is the first to devise a method for accelerating MARL for complex teamwork tasks using a sub-task curriculum.

We evaluate MEDoE in two different multi-agent environments, *Chainball* and *Overcooked*. Our experiments show that MEDoE can enable the use of a sub-task curriculum to significantly accelerate MARL, even where naïve approaches which use standard MARL techniques for fine-tuning fail. MEDoE is able to solve the Chainball task (Section 4.1) where training from scratch converges to a suboptimal policy. Additonally, MEDoE learns to solve the complex Overcooked task (Section 4.1), which is not solved by other baselines even when using 5 times as many training timesteps as MEDoE. MEDoE can extend any actor-critic method, and can be used with any number of agents. MEDoE can also be used in situations where the size of the team in the complex target task may not be known during training in the simple sub-tasks, allowing for flexibility with respect to the target team composition, and re-use of sub-tasks in curricula for different target tasks.

2 Problem Formulation

In this section, we present the problem formulation providing the framework for our approach. We first discuss our sub-task curriculum formulation, in Sections 2.1 and 2.2. Then in Section 2.3, we formalise the domain of expertise (DoE), which encodes the relationship between the simple source tasks in the curriculum to the complex target task. Information about the DoE is then used to inform our method, MEDoE, as discussed in Section 3.

2.1 Sub-task Curriculum

We define a Few-Shot Teamwork (FST) problem, which consists of two stages: a *source stage*, where sub-teams of agents are trained with respect to simple source tasks, and an *adjustment stage*, where teams from the source stage are combined and fine-tuned to complete the complex target task. This provides the framework for our sub-task curriculum approach to accelerating MARL.

We model a set of M source tasks, $\mathcal{T}^1, \ldots, \mathcal{T}^M$, and a single target task $\mathcal{T}^T$. Each task $\mathcal{T}^m$ is a common-reward task, and can be modelled by a Dec-POMDP:

$$\mathcal{T}^m = \langle P^m, \mathcal{S}^m, \{\mathcal{A}_i^m\}_{i\in P^m}, T^m, \{\Omega_i^m\}_{i\in P^m}, O^m, R^m, \gamma\rangle, \quad (1)$$

where P^m is the set of agents (team); $\mathcal{S}^m$ is the state space; $\mathcal{A}_i^m$ is the action space for agent i; T^m is the state transition probability density function; Ω^m is the observation space for agent i; O^m is the observation probability density function; R^m is the reward function; and γ is the discount factor.

In the source stage, M disjoint sub-teams are created $P^1, \ldots, P^M$: one for each source task. Each team P^m trains for N^m training steps in its source task $\mathcal{T}^m$ using learning algorithm $\mathbb{L}^S$. This training generates a joint task policy π^m, and optionally other data or functions derived from the training process, such as a buffer of source stage experiences, which may be used during the adjustment stage. In this paper we assume the sub-task curriculum (the source tasks and the sub-team partitioning) is provided by an expert.

In the adjustment stage, sub-teams are combined to form the target task's team, $P^T \subseteq \bigcup_m^M P^m$, and then learn to coordinate by practising in the target task for a limited number of training steps, N^A, after which the team is evaluated. During this stage, agents use learning algorithm, $\mathbb{L}^A$, designed to promote coordination and exploration of the new task using the skilled source stage policies π^m and other data or functions derived during the source stage. At the end of the N^A training steps, the performance of the new team is evaluated on task $\mathcal{T}^T$, forming the optimisation objective of the FST problem.

2.2 Objective

The overall objective of the FST problem is to maximise the mean returns of the final team on the target task $\mathcal{T}^T$ in a limited number of training steps. The main research problem is to maximise this objective by designing the source and adjustment stage learning algorithms, $\mathbb{L}^S$ and $\mathbb{L}^A$ respectively.

In this work, we focus on finding an adjustment stage learning algorithm, $\mathbb{L}^A$ which requires less training experience to achieve desired performance on the target task than MARL baselines which train from scratch. If the best baseline approach requires $N^\odot$ training steps to reach performance $G^\odot$ from scratch on task $\mathcal{T}^T$, then a successful approach to FST should reach performance $G^\odot$ with $N^A + \sum_m^M N^m \ll N^\odot$.

2.3 Domain of Expertise

It is impossible to find an algorithm which improves generalisation performance on average across an unconstrained set of source and target tasks [Wolpert and Macready, 1997]. Therefore, we must introduce a relationship between the source and target tasks. We introduce the notion of a domain of expertise (DoE), which intuitively defines the set of observations in a target task for which a given agent is skilled. We impose the restriction that in the target task, there should typically be at least one agent who is already skilled at some aspects of the target task — i.e., that the DoEs of the new team in the target task should occupy a large fraction of the visitation space of the optimal policy on the target task.

Let $\pi_i^{*,m}$ be agent i's policy in the optimal decentralised policy of source task $\mathcal{T}^m$. Let $\Pi^{*,T}$ be the set of optimal decentralised policies of the target task. Assume a mapping $\phi_i^m : \Omega_i^T \mapsto \Omega_i^m$ between agent i's observation in the target task, and in the source task. Then, we consider an observation $o_i \in \Omega_i^T$ to be in the *domain of expertise*, $\mathcal{E}_i^{m,T}$, of agent i of task $\mathcal{T}^m$ iff

$$\exists \pi^{*,T} \in \Pi^{*,T},\ D_{KL}(\pi_i^{*,m}(\cdot|\phi_i^m(o_i)) \parallel \pi_i^{*,T}(\cdot|o_i)) < \tau, \tag{2}$$

where τ is a chosen similarity threshold, and D_{KL} is the KL divergence. We present this formalisation of the DoE to provide context for later discussion. However, the formalisation is not used directly in the derivation of our method.

3 Modulating Exploration and Training via Domain of Expertise (MEDoE)

In this section, we introduce our novel approach, *Modulating Exploration and Training via Domain of Expertise (MEDoE)*, designed to facilitate efficient learning in the adjustment stage of a sub-task curriculum. MEDoE identifies situations in the target which do not require each agent to update its policy, and uses this information to modulate each agent's exploration and learning procedure. MEDoE achieves this with two key components: a method for classifying when a given target task observation is in an agent's domain of expertise (DoE); and a method for adapting and guiding exploration based on the output of that DoE classifier. We discuss these components in turn in the following subsections.

3.1 Domain of Expertise Classification

MEDoE relies on a domain of expertise classifier for each agent i, $D_i : \Omega_i^T \mapsto [0, 1]$, which classifies whether a given target task observation lies within agent i's DoE.

$$D_i(o_i) = \begin{cases} 1 & \text{if } o_i \in \mathcal{E}_i^{m,T}, \\ 0 & \text{if } o_i \notin \mathcal{E}_i^{m,T}. \end{cases} \tag{3}$$

In most cases, the ground truth DoE classifier D_i will be unknown, as in practice knowing the ground truth DoE classifier requires knowing the set of ϵ-optimal policies, Π^ϵ.

Therefore, we instead rely on an approximate DoE classifier, $\hat{D}_i$, which outputs probabilistic classifications. In this work, we propose two types of approximate DoE classifiers: i) a expert-provided classifier, and ii) a learned classifier. The first classifier is provided by an expert, as in our experiments in Section 4. We use these expert-provided DoE classifiers to act as an approximation to a ground-truth DoE classifier, allowing us to study the performance of MEDoE in isolation from the problem of learning a DoE classifier. However, in some applications, expert knowledge might be unavailable or might be difficult to obtain. In those cases, it may be possible to learn adequate DoE classifiers from training experience. We developed a simple binary learned DoE classifier as a proof-of-concept, and show the feasibility of this approach in Section 4.3.

The learned DoE classifier trains a multi-layer perceptron (MLP), $\hat{D}_i$, for each agent i in the target task team P^T. The classifier training is formulated as a binary classification problem where positive examples are taken from agent i's source task experience buffer, and negative examples are taken from the experience buffers of all agents trained in a different source task to i. In this way, the MLP learns to identify features of the observation which differ between different source tasks, which are likely to correspond to the conditions relevant to domain of expertise. For example, a classifier trained to distinguish between sample observations from attack drills and defence drills in football might identify the position of the ball on the pitch as a feature of interest. Such a classifier is likely to be adequate for DoE classification, as when the ball is near the opponents' goal, the attackers' policies learned during attack drills are near optimal, so only the defenders need to learn to, e.g., position correctly in the case of a counter-attack.

3.2 Exploration Modulation

The key aspect of MEDoE is the modulation of the exploration and training process in the adjustment stage, informed by the DoE classifier. In this paper, we focus on our variant of MEDoE based on proximal policy optimisation (PPO) [Schulman *et al.*, 2017], which modulates four quantities:

1. the policy entropy regularisation coefficient, α;
2. the policy KL regularisation coefficient, κ;
3. the PPO clipping coefficient, δ; and
4. the action selection temperature, T.

In this section, we discuss the intuition behind the modulation of these quantities, and provide a description of MEDoE. Throughout the section, for convenience we call agents *experts* when the current observation is in their DoE; and as *non-experts* otherwise. Each agent is an expert for some observations, and a non-expert for others. See Algorithm 1 (Appendix A) for our PPO variant of MEDoE.

Non-experts should forget irrelevant skills, and experts should be slow to forget useful skills. During the source task, agents learn skills which are relevant to the completion of the target task, but also skills which might be irrelevant. Such irrelevant skills can arise from differences in skill requirements between source and target tasks, or from extrapolation. Ideally, agents should quickly forget irrelevant behaviours. However, at the same time they must retain useful skills, which may be difficult in settings which require complex coordination or with sparse rewards, as forgetting can occur during extended low-reward periods.

To control the rate of forgetting skills, MEDoE modulates two parameters. Firstly, we use entropy-regularised policies, and encourage non-experts to forget irrelevant skills by increasing non-experts' entropy regularisation coefficient, setting $\alpha_i = \alpha_{\text{base}} \times \beta_\alpha^{(1-\hat{D}_i(o_i))}$, where α_{base} is the base entropy coefficient, and $\beta_\alpha > 1$ is the entropy boost coefficient.

Secondly, we use fixed behaviour priors [Tirumala *et al.*, 2020] to encourage experts to retain useful skills. This entails using KL-regularised policies (see Equation (5)), where we aim to minimise the KL divergence between the agent's current policy $\pi_i(a_i|o_i;\theta_i)$, and its frozen source stage policy $\pi_i(a_i|o_i;\overline{\theta}_i^{\text{BP}})$, thereby encouraging the agent to stay close to its source stage behaviour. We boost the KL regularisation coefficient for experts, setting $\kappa_i = \kappa_{\text{base}} \times \beta_\kappa^{\hat{D}_i(o_i)}$.

Non-experts should quickly adapt their behaviour, and experts should be slow to update their behaviour. In multi-agent systems, when multiple agents are learning and changing their behaviours over time, this presents a learning agent with a moving target in terms of optimal behaviour. This non-stationarity makes learning stable team policies more difficult than if only one agent were adapting at a time. MEDoE addresses this problem by using the fact that experts do not need to update their policies. In our PPO-based version of MEDoE, we modulate the PPO clipping coefficient, which controls by how much the policy may be updated in a given step. We limit the policy update size using a low baseline clipping coefficient, δ_{base} and set $\delta_i = \delta_{\text{base}} \times \beta_\delta^{(1-\hat{D}_i(o_i))}$.

Non-experts should explore to learn new skills, and experts should be predictable to other agents by exploiting existing skills. By definition, non-expert agents need to learn new behaviours. To do so they must explore. Exploration in multi-agent systems can have negative effects on learning, such as reducing training stability, and increasing the difficulty of selecting equilibrium which require stable coordination. We therefore aim to restrict exploration to situations where it is necessary, i.e., when agents are non-experts. MEDoE takes a simple approach: modulate an agent's exploration parameter using that agent's DoE classifier. For the PPO-based MEDoE, the relevant exploration parameter is the stochastic action selection temperature:

$$a_i \sim \pi(\cdot|o_i; T_i = T_{\text{base}} \times \beta_T^{(1-\hat{D}_i(o_i))}),\ \forall i \in P. \tag{4}$$

During evaluation, we fix the action selection temperature to T_{base}. We therefore apply an importance sampling reweighting w_i (Equation (7)) during training.

Ultimately, we minimise the following policy and value losses for each agent i in the target team:

$$\begin{aligned} \mathcal{L}(\theta_i) = & w_i \text{PPOClip}(A_i, \pi_i(a_i|o_i;\theta_i), \delta_i) \\ & - \alpha_i H(\pi_i(\cdot|o_i;\theta_i)) \\ & + \kappa_i D_{KL}(\pi_i(\cdot|o_i;\theta_i) \parallel \pi_i(\cdot|o_i;\overline{\theta}_i^{\text{BP}})), \end{aligned} \tag{5}$$

where PPOClip(A, π, δ) is the PPO policy ratio clipping function described by Schulman *et al.* [2017] with clipping coefficient δ, and

$$\mathcal{L}(\psi_i) = w_i \|G_{t:t+n} - V_i(o_i;\psi_i)\|_2^2. \tag{6}$$

In Equations (5) and (6), we compute the importance weight for agent i,

$$w_i = \frac{\pi_i(a_i|o_i;\theta_i, T = T_{\text{base}})}{\pi_i(a_i|o_i;\theta_i, T = T_i)}, \tag{7}$$

the advantage function for agent i,

$$A_i = (G_{t:t+n} - V_i(o_{i,t};\psi_i)), \tag{8}$$

and the n-step return for agent i,

$$G_{t:t+n} = \gamma^n V_i(o_{i,t+n};\psi_i) + \sum_{i=0}^{n-1} \gamma^i r_{t+i}. \tag{9}$$

4 Experiments

Our experiments aim to answer the following questions:

1. Can MEDoE aid in solving complex multi-agent tasks with fewer total training timesteps than training from scratch? Does MEDoE provide benefit over fine-tuning using standard MARL algorithms?
2. How sensitive is MEDoE to hyperparameter selection?
3. Is it feasible to obtain an approximate DoE classifier from source task experience which enables MEDoE to outperform baseline approaches?
4. To what extent are behaviour priors responsible for the performance of MEDoE?

4.1 Environments

To test our approach, we consider two environments with clear task decompositions: *Chainball*, a simple environment we introduce to provide insight into our method; and *Overcooked* [Wang *et al.*, 2020a], a complex environment common in MARL research. We define the observation space in each environment such that the observation translation mapping ϕ between source and target tasks can be the identity function. In each environment, we normalise the rewards such that the maximum *expected* episodic return lies in $[0, 1]$. Further details of each environment and the expert-provided DoE classifiers for each environment can be found in Appendix C.

Chainball

We introduce the *Chainball* environment as a simple example to test MEDoE. We design Chainball to mimic the compositional properties of our football motivating example, while allowing for simple evaluation, and use of tabular methods. "Chainball-N" (Figure 2) consists of N states, $s \in \{1, \ldots, N\}$. At timestep t, each of four agents chooses an action $a_{t,i} \in \{1, 2, 3, 4\}$. We define the *forward probability* of taking joint action $\mathbf{a}_t$ in state s as $f_s(\mathbf{a}_t) = T(S_{t+1} = s+1 | S_t = s, \mathbf{a}_t)$. For $s = N$, rather than transitioning to non-existent state $N+1$, the agents score and get a reward of +1, and the state transitions to a restart (kick-off) state (in Figure 2, the M state). If the state does not transition forward to state $s+1$, it transitions backwards to state $r < s$ with probability proportional to 1.5^{r-s}. This intuitively corresponds to an "opposing team" getting possession of the ball. For $s = 1$, if the state transition backwards, the

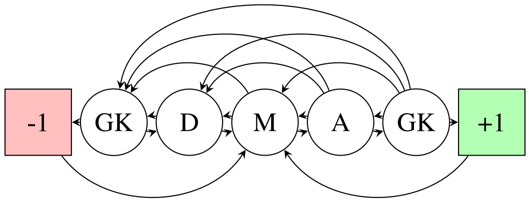


Figure 2: Chainball-5 Environment. Here states are re-labelled with GK (goalkeeper), D (defence), M (midfield), and A (attack).

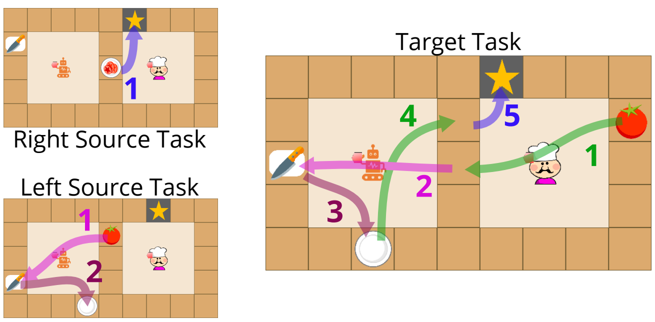


Figure 3: Overcooked Sub-task Curriculum. In the target task, agents must coordinate to pass and chop the tomato at the chopping board (1,2), put the chopped tomato on a plate (3), and pass the plate with chopped tomato back to serve at the starred counter (4,5). The skills to complete steps 2 and 3 can be learned in the "Right" source task; and skills to complete step 5 can be learned in the "Left" source task. Steps 1 and 4 require learning new behaviour in the target task.

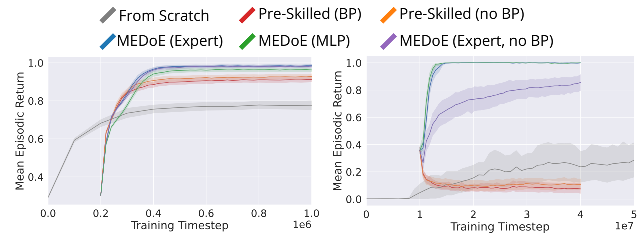
team concedes a goal, receiving a reward of -1, and the state transitions to the restart state. Chainball is an episodic task, which terminates after 90 timesteps. Chainball has two source tasks, Chainball-N-Att and Chainball-N-Def, to emulate attack and defence drills respectively. These source tasks have two agents each. Each source task consists of N states, but we make states $s < s_{\text{Att}}$ terminal states in Chainball-N-Att, and states $s > s_{\text{Def}}$ terminal states in Chainball-N-Def. Finally, both source tasks terminate if a terminal state is reached, or if a goal is scored or conceded, or after 90 timesteps. Our experiments use $N = 11$.

Overcooked

Overcooked [Wang *et al.*, 2020a] is a complex environment which requires multi-step coordination by agents. The goal of Overcooked is to complete a recipe by moving and processing foods in a grid world. Figure 3 shows the configuration of our Overcooked target task and sub-task curriculum. The team is rewarded for completing each step in the recipe, except steps 1 and 4 in the target task.

4.2 Protocol & Baselines

For each source task, we generate four seeds of skilled sub-teams by training agents using the standard independent PPO (IPPO) algorithm until convergence. As we consider fully-observable settings, IPPO is equivalent to the independent multi-agent PPO (MAPPO) algorithm [Papoudakis *et al.*, 2021]. In Chainball, we use tabular actors and critics; whereas



(a) Chainball-11 (b) Overcooked

Figure 4: Adjustment stage training returns for each environment. Mean episodic returns (100 episodes), averaged over 80 runs (5 seeds and 16 different teams per seed). The **from-scratch** baseline is averaged over 16 runs (16 seeds). Shaded area shows the 95% confidence interval of the mean over the runs. The **pre-skilled** and **MEDoE** baselines are shifted on the training step axis to account for the total training timesteps required to obtain the source task agents.

in Overcooked we use deep learning. The number of training timesteps in each source task is task-dependent. We report the number of timesteps used for each source task in Appendix B. We save the final actors, critics, and experience buffer from each source task seed. The experience buffer stores a number of source task observations for each agent: 40,000 observations per agent in Chainball and 320,000 in Overcooked.

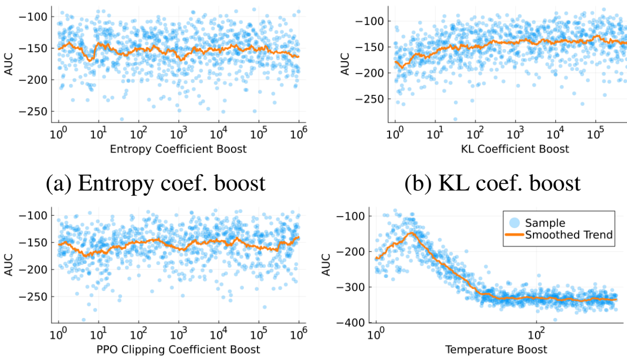
In our experiments, our **MEDoE** baselines employ the IPPO-based MEDoE approach described in Section 3 and Algorithm 1. We initialise each agent's actor and critic using an actor and critic from the source task, and use that same actor as the agent's behaviour prior. We consider two variants of our MEDoE algorithm: **MEDoE (Expert)**, which uses a expert-provided DoE classifier, as described in Section 4.1; and **MEDoE (MLP)**, which learns a classifier from source task experience according to the protocol described in Section 3.1.

As we are not aware of any existing methods that use a sub-task curriculum for accelerating MARL, we compare MEDoE to ablations and standard MARL baselines. The motivation for MEDoE is to accelerate learning a complex teamwork task relative to standard MARL approaches, which train from scratch on the complex task. We therefore consider the **from-scratch** baseline, which applies IPPO directly to the target task, starting from randomly initialised actor and critic networks. We aim to show that MEDoE is responsible for improving performance where naïve MARL fine-tuning approaches fail. Therefore, we create a **pre-skilled (BP)** baseline, which is initialised in the same way as **MEDoE**, but does not modulate training parameters, nor uses a DoE classifier. Finally, in Section 4.3 we investigate the role of behaviour priors in the performance of MEDoE. We therefore consider ablations of **MEDoE (Expert)** and **pre-skilled (BP)**. In these ablations, **MEDoE (Expert, no-BP)** and **pre-skilled (no-BP)** respectively, we do not use behaviour priors, setting the KL coefficient to zero.

4.3 Results

MEDoE Performance

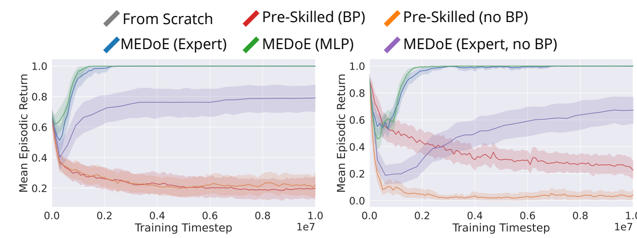
In Figure 4, we report the adjustment stage training episodic return curves for each of the baselines on the target task. For



(a) Entropy coef. boost (b) KL coef. boost

(c) PPO clip coef. boost (d) Temperature boost

Figure 5: Sensitivity to hyperparameters in Chainball. We fix each of the four MEDoE boost parameters, then choose one to randomly sample. We compute 1024 samples for each parameter, and report the AUC (the area under the mean episodic return training curve during the adjustment stage) for each parameter setting.



(a) Left Half (b) Right Half

Figure 6: Forgetting curve in Overcooked

baselines where we use source stage training data, we shift the training curves to account for the *total* source stage training timesteps of the target task team, in order to make a fair comparison to the **from-scratch** baseline. In each environment, we consider 16 pairing of source task agents and 5 seeds, for a total of 80 training runs per baseline. These results show that both MEDoE baselines, **MEDoE (Expert)** and **MEDoE (MLP)**, significantly outperforms the **from-scratch** baseline in both environments, supporting our claim that MEDoE can greatly reduce the amount of training timesteps required to solve complex teamwork tasks. The results also show that the naïve fine-tuning approaches (**pre-skilled** baselines) can fail to outperform the **from-scratch** baseline. In Chainball (Figure 4a), we see that naïve MARL fine-tuning performs comparably to the **from-scratch** baseline, whereas in Overcooked (Figure 4b), we find that the **pre-skilled** baselines actually perform significantly *worse* than the **from-scratch** baseline. This may in part explain why expert-based sub-task curriculum methods are not yet a popular approach in MARL. However, MEDoE opens new avenues of research in this direction.

Hyperparameter Sensitivty

Next, we examine the sensitivity of MEDoE to hyperparameters in the Chainball environment. Figure 5 shows the area under curve (AUC) for randomly sampled hyperparameter settings. For each sample, we fix each of the four MEDoE boost parameters, and then choose one to randomly sample. The AUC provides a measure that captures both the level of returns of that training run, in addition to how quickly that level was reached. In other words, quickly reaching high level of return leads to high AUC. MEDoE does not appear to be particularly sensitive to the entropy coefficient boost or to the PPO clipping coefficient boost, but larger values of the KL coefficient boost improve the performance of MEDoE. Lastly, MEDoE is sensitive to the temperature boost coefficient, with the highest performance at $\beta_T = 3$, but any setting between 1 and 5 outperforms the baseline performance of no temperature boost. These results suggest that MEDoE is not extremely sensitive to hyperparameters selection. As a consequence, we do not expect MEDoE to significantly increase the difficulty of hyperparameter search. However, when tuning, careful attention should be given to the temperature boost coefficient.

Learned Classifiers

Figure 4 shows that in both Overcooked and Chainball, MEDoE can perform just as well with a DoE classifier learned from source task data as when using our expert-based DoE classifiers. This performance is achieved even when using a simple MLP-based DoE classifier, as described in Section 3.1. Though MEDoE performance is sufficient to demonstrate the feasibility of obtaining classifiers from source task data, we further test the quality of the obtained binary MLP classifiers by examining two classifier loss metrics in Overcooked. Firstly, we consider the binary cross-entropy loss of the classifier when distinguishing between examples from different source tasks. Using a held-out evaluation set, we find that compared to a loss of 0.6931 $(\log 2)$ for a uniform random classifier, for the learned classifier the binary cross-entropy loss is 0.0045 ± 0.0001, which corresponds to a highly accurate classifier. Secondly, we compare the classifications of the learned classifier to that of the expert-provided classifier by computing the binary cross-entropy loss across a collection of 160,000 observations collected in the target task by a converged policy from our experiments in Section 4.3. We find that the learned classifier has a loss of 0.38 ± 0.01 for the "Left" task agent's DoE, and 3.44 ± 0.02 for the "Right" task agent. The classifier performs poorly for the "Right" task agent, suggesting further work to obtain higher quality DoE classifiers might be useful. However, by examining the learned classifier manually, we find that the "Right" task DoE classifier is confidently incorrect for only a small portion of the trajectory, which may explain why the relatively large cross-entropy loss does not correspond to reduced MEDoE performance.

Role of Behaviour Priors

We also examine the advantage of using behaviour priors as a part of MEDoE. Firstly, in Figure 4b we show the performance of the **MEDoE (Expert, no-BP)** ablation in Overcooked. While this ablation still significantly outperforms **pre-skilled** and **from-scratch** baselines, it performs poorly compared to **MEDoE (MLP)** and **MEDoE (Expert)** which use behaviour priors. This suggests MEDoE's benefits do not derive solely from the use of behaviour priors, but that without behaviour priors, MEDoE is more unreliable and slower to converge to optimal policies. Secondly, in Figure 6, we show the performance of each Overcooked agent in the *source task* in which that agent was trained, as both agents are fine-tuned during the adjustment stage. While it is possible that an agent could be making rapid progress on the target task while its source task evaluated episodic returns decreases, these returns provide intuitive insight into the role of behaviour priors in MEDoE. High source task return suggests the agent has not forgotten relevant sub-task skills. When behaviour priors are not used, agents are prone to forgetting sub-task skills.

5 Related Work

To address challenges in multi-agent reinforcement learning, prior methods also investigated modulating training parameters. In WoLF-PHC [Bowling and Veloso, 2001] and extensions [Bowling, 2004], each agent's policy learning rate is modulated according to the intuition that an agent's policy learning rate should be high when it is underperforming relative to its expectations, and low otherwise. More recently, MA2QL [Su *et al.*, 2022] focuses on a team learning setting. Instead of letting agents learn simultaneously, MA2QL tackles non-stationarity by allowing only one agent to learn at a time.

Work by Vrancx *et al.* [2011] considers transfer from simple tasks to complex target tasks in multi-agent systems. They train a classifier to distinguish between cases in which agents can learn individually, and cases in which they must learn to coordinate as part of a team. Though similar to our DoE classifier, one key difference is that our DoE classifier attempts to classify states in which further learning is not required, rather than states in which agents continue to learn without paying attention to other agents.

Wang *et al.* [2020c] also accelerate learning of complex multi-agent tasks using a curriculum. However, it focuses on cases in which the number of agents is progressively increased throughout the curriculum. By contrast, in our work we consider cases in which we decompose the task based not upon the number of agents, but the different skills required by agents in the target task. Similarly, recent work by Tang *et al.* [2022] considers scenarios in which agents join an unfamiliar team, and have to rapidly learn to adapt to coordinate with the new team to complete a known task. In contrast with our work, Tang *et al.* [2022] vary only the number of agents between source and target tasks, while the underlying dynamics remain the same. Taylor *et al.* [2019] consider *parallel transfer learning*, which transfers experiences collected in parallel by separate agents into a target agent, similarly to federated reinforcement learning [Qi *et al.*, 2021]. In contrast, our work does not directly transfer skills into individual agents, but instead attempts to accelerate the progress of the *team's* performance.

Several single-agent RL methods use a multi-agent approach to curriculum learning [Narvekar *et al.*, 2020; Dennis *et al.*, 2020; Parker-Holder *et al.*, 2022]. However, these methods focus on the curriculum design problem (i.e., generating the series of tasks that form the curriculum) by treating it as a two-player game. We instead focus on the problem of accelerating learning at each stage in the curriculum, when the target task and decomposed sub-tasks are given as input by an expert.

The problem of effective fine-tuning on new tasks often appears in the continual learning literature, typically in single-agent settings. Nekoei *et al.* [2021] propose using the card game Hanabi as a test-bed for continual learning in multi-agent settings. Liu *et al.* [2022] engineer an approach to train humanoid agents to play 2-vs-2 football, which, like MEDoE, uses behaviour priors [Tirumala *et al.*, 2020]. However, their solution is complex and specific to humanoid football, whereas MEDoE is a simple approach, applicable in a range of settings.

Finally, some works consider the assignment of agents into different roles, where agents assigned to the same role employ similar policies [Wang *et al.*, 2020b]. In our case, we assume roles do not have to be discovered, and are instead provided implicitly via the given sub-task decomposition.

6 Conclusion and Future Work

In this work, we presented Modulating Exploration and Training via Domain of Expertise (MEDoE): a method for solving complex MARL problems. MEDoE uses a curriculum of sub-tasks, given by an expert, and modulates training and exploration of each agent on the target task. Each agent's modulation is controlled by a domain of expertise classifier that provides information about whether the agent's policy is likely to be useful in the target task, given the current observation. This information is then used by MEDoE to adapt the rate at which each learns or forgets, and how much the agent should explore. Our experiments extend the IPPO algorithm, though any actor-critic method can be modified to produce a MEDoE version, allowing designers to choose the most appropriate learning algorithms for their task. As MEDoE acts on individual agents, the total computational cost of using MEDoE scales linearly with the number of agents. Furthermore, MEDoE does not require a fixed choice of number of agents in the source and target tasks, meaning it can be used in situations where teams are formed on an ad hoc basis [Mirsky *et al.*, 2022] without prior knowledge of the number of agents.

We evaluated MEDoE in two multi-agent environments – *Chainball* and *Overcooked*, and compared it to a number of baselines. Our results show that MEDoE is able to significantly speed up learning in complex multi-agent tasks. We also evaluated two types of domain of expertise classifiers: an expert based one and a learned classifier. Results show that a learned classifier with simple neural networks architectures is a feasible approach that can be readily integrated into MEDoE without significantly impacting the overall performance.

While MEDoE has many advantages, the work could be extended in multiple directions. Firstly, we assumed the target task and the sub-task curriculum was provided. In practice it might not be feasible to provide such a curriculum. For example, even human experts might not know how to decompose a task, or find it too time-consuming particularly when attempting to derive a multi-step curriculum. Automated sub-task decomposition based on experience in the complex task is an open challenge [Jeon *et al.*, 2022]. Secondly, while our experiments provide a proof-of-concept that DoE classifiers can be learned from source task data, the classifiers used are simple and might not be applicable to more realistic tasks. Alternative approaches to obtaining DoE classifiers could be investigated and tested. Future work could also study methods for updating DoE classifiers throughout the adjustment stage, in order to reflect new skills learned during the adjustment stage, or to correct for inaccuracies in the learned DoE classifier.

References

Michael Bowling and Manuela Veloso. Rational and convergent learning in stochastic games. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, volume 2 of *IJCAI'01*, pages 1021–1026, Seattle, WA, USA, August 2001. Morgan Kaufmann Publishers Inc.

Michael Bowling. Convergence and no-regret in multiagent learning. Technical report, University of Alberta Libraries, 2004.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, volume 34 of *NIPS'21*, pages 13049–13061, Online Conference, December 2020. Curran Associates Inc.

Jeewon Jeon, Woojun Kim, Whiyoung Jung, and Youngchul Sung. MASER: Multi-agent reinforcement learning with subgoals generated from experience replay buffer. *arXiv:2206.10607 [cs]*, June 2022.

Siqi Liu, Guy Lever, Zhe Wang, Josh Merel, S. M. Ali Eslami, Daniel Hennes, Wojciech M. Czarnecki, Yuval Tassa, Shayegan Omidshafiei, Abbas Abdolmaleki, Noah Y. Siegel, Leonard Hasenclever, Luke Marris, Saran Tunyasuvunakool, H. Francis Song, Markus Wulfmeier, Paul Muller, Tuomas Haarnoja, Brendan Tracey, Karl Tuyls, Thore Graepel, and Nicolas Heess. From motor control to team play in simulated humanoid football. *Science Robotics*, 7(69):eabo0235, August 2022.

Reuth Mirsky, Ignacio Carlucho, Arrasy Rahman, Elliot Fosong, William Macke, Mohan Sridharan, Peter Stone, and Stefano V. Albrecht. A survey of ad hoc teamwork research. In *Multi-Agent Systems: 19th European Conference*, pages 275–293, Düsseldorf, Germany, September 2022. Springer-Verlag.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

Hadi Nekoei, Akilesh Badrinaaraayanan, Aaron Courville, and Sarath Chandar. Continuous coordination as a realistic scenario for lifelong learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, Online Conference, June 2021. PMLR.

Georgios Papoudakis, Filippos Christianos, Arrasy Rahman, and Stefano V. Albrecht. Dealing with non-stationarity in multi-agent deep reinforcement learning. *arXiv:1906.04737 [cs, stat]*, June 2019.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, Online Conference, November 2021.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 17473–17498, Baltimore, MD, USA, June 2022. PMLR.

Jiaju Qi, Qihao Zhou, Lei Lei, and Kan Zheng. Federated reinforcement learning: Techniques, applications, and open challenges. *Intelligence & Robotics*, 1(1):18–57, October 2021.

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

Kefan Su, Siyuan Zhou, Chuang Gan, Xiangjun Wang, and Zongqing Lu. MA2QL: A minimalist approach to fully decentralized multi-agent reinforcement learning. *arXiv:2209.08244 [cs]*, September 2022.

Xuting Tang, Jia Xu, and Shusen Wang. Transferable multi-agent reinforcement learning with dynamic participating agents. *arXiv:2208.02424 [cs]*, August 2022.

Adam Taylor, Ivana Dusparic, Maxime Gueriau, and Siobhan Clarke. Parallel transfer learning in multi-agent systems: What, when and how to transfer? In *2019 International Joint Conference on Neural Networks*, pages 1–8, Budapest, Hungary, July 2019. IEEE.

Dhruva Tirumala, Alexandre Galashov, Hyeonwoo Noh, Leonard Hasenclever, Razvan Pascanu, Jonathan Schwarz, Guillaume Desjardins, Wojciech Marian Czarnecki, Arun Ahuja, Yee Whye Teh, and Nicolas Heess. Behavior priors for efficient reinforcement learning. *arXiv:2010.14274 [cs]*, October 2020.

Peter Vrancx, Yann-Michaël De Hauwere, and Ann Nowé. Transfer learning for multi-agent coordination. In *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence*, volume 1, pages 263–272, Rome, Italy, 2011. SciTePress - Science and and Technology Publications.

Rose E. Wang, Sarah A. Wu, James A. Evans, Joshua B. Tenenbaum, David C. Parkes, and Max Kleiman-Weiner. Too many cooks: Bayesian inference for coordinating multi-agent collaboration. *arXiv:2003.11778 [cs]*, July 2020.

Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. ROMA: Multi-agent reinforcement learning with emergent roles. In *Proceedings of the 37th International*

Conference on Machine Learning, volume 119, pages 9876–9886, Online Conference, November 2020. PMLR.

Weixun Wang, Tianpei Yang, Yong Liu, Jianye Hao, Xiaotian Hao, Yujing Hu, Yingfeng Chen, Changjie Fan, and Yang Gao. From few to more: Large-scale dynamic multiagent curriculum learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7293–7300, Online Conference, April 2020.

D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

Acknowledgments

Thanks to Filippos Christianos, Mahdi Kazemi Moghaddam, and Georgios Papoudakis for providing valuable feedback on drafts of this paper.

A PPO-MEDoE Algorithm

In this section we present our version of MEDoE based on the proximal policy optimisation (PPO) algorithm. We present a 1-step return version for clarity, but the extension to n-step is straightforward. MEDoE could also be used to extend other actor-critic algorithms in a similar manner.

Algorithm 1 PPO-MEDoE (1-step)

Require: Team of agents P with policies $\{\pi_i(a|o;\theta_i) : i \in P\}$ and critics $\{V_i(o;\psi_i) : i \in P\}$
Require: Domain of Expertise classifiers, $\left\{\hat{D}_i : i \in P\right\}$
Require: Hyperparameters T_{base}, δ_{base}, α_{base}, β_T, β_δ, β_α.
Set behaviour priors $\overline{\theta}_i^{\text{BP}}$ given initial policy parameters θ_i.
Observe initial state s_0
for $t = 0$ **to** $T_{\text{max}} - 1$ **do**
Compute boosted exploration coefficient, $\forall i \in P$

$$T_i = T_{\text{base}} \times \beta_T^{(1-\hat{D}_i(o_{i,t}))}$$

Sample action $a_i \sim \pi_i(\cdot|o_{i,t}; T = T_i)$, $\forall i \in P$
Take joint action $\mathbf{a}$, observe next state $o_{jt+1} \sim T(\cdot|s_t, \mathbf{a})$ and receive reward $r_t = R(s_t, \mathbf{a}_t)$
for $i \in P$ **do**
Compute importance weight (with stop grad)

$$w_i = \frac{\pi_i(a_i|o_i;\theta_i, T = T_{\text{base}})}{\pi_i(a_i|o_i;\theta_i, T = T_i)}$$

Compute 1-step return $G_{t:t+1} = r_t + \gamma V_i(o_{i,t+1};\psi_i)$
Compute critic loss

$$\mathcal{L}(\psi_i) = w_i \|G_{t:t+1} - V_i(o_{i,t};\psi_i)\|_2^2$$

Compute advantage $A_i = G_{t:t+1} - V_i(o_{i,t};\psi_i)$
Compute boosted learning coefficients

$$\alpha_i = \alpha_{\text{base}} \times \beta_\alpha^{(1-\hat{D}_i(o_{i,t}))}$$
$$\kappa_i = \kappa_{\text{base}} \times \beta_\kappa^{\hat{D}_i(o_{i,t})}$$
$$\delta_i = \delta_{\text{base}} \times \beta_\delta^{(1-\hat{D}_i(o_{i,t}))}$$

Compute actor loss

$$\begin{aligned}\mathcal{L}(\theta_i) = & w_i \text{PPOClip}(A_i, \pi_i(a_i|o_i;\theta_i), \delta_i) \\ & - \alpha_i H(\pi_i(\cdot|o_i;\theta_i)) \\ & + \kappa_i D_{KL}(\pi_i(\cdot|o_i;\theta_i) \parallel \pi_i(\cdot|o_i;\overline{\theta}_i^{\text{BP}}))\end{aligned}$$

Update θ_i and ψ_i using gradient descent
end for
end for
return Optimised target task policies $\{\pi_i(a|o;\theta_i) : i \in P\}$ and critics $\{V_i(o;\psi_i) : i \in P\}$

B Hyperparameter Settings for Experimental Results

Table 1 reports the hyperparameters used in our experiments. We choose the hyperparameters by using common hyperparameters for IPPO. We additionally perform an optimisation sweep to select the entropy coefficient and actor learning rates, and set the critic learning rate to be 2 times greater than the actor learning rate. We choose MEDoE hyperparameters in both environments on the basis of a hyperparameter sweep in the chainball environment.

C Additional Environment Details

C.1 Chainball

Environment Description

We introduce the *Chainball* environment as a simple example to test MEDoE. We designed Chainball to mimic the compositional properties of our football motivating example, while allowing for simple evaluation, and use of tabular methods. "Chainball-N" (Figure 7) consists of N states, $s \in \{1, \ldots, N\}$. At timestep t, each of four agents chooses an action $a_{t,i} \in \{1, 2, 3, 4\}$. We define the *forward probability* of taking joint action $\mathbf{a}_t$ in state s as $f_s(\mathbf{a}_t) = T(S_{t+1} = s + 1|S_t = s, \mathbf{a}_t)$. For $s = N$, rather than transitioning to non-existent state $N + 1$, the agents score and get a reward of +1, and the state transitions to a restart (kick-off) state (in Figure 7, the M state). If the state does not transition forward to state $s + 1$, it transitions backwards to state $r < s$ with probability proportional to 1.5^{r-s}. This intuitively corresponds to an "opposing team" getting possession of the ball. For $s = 1$, if the state transition backwards, the team concedes a goal, receiving a reward of -1, and the state transitions to the restart state. Chainball is an episodic task, which terminates after 90 timesteps. Chainball has two source tasks, Chainball-N-Att and Chainball-N-Def, to emulate attack and defence drills respectively. These source tasks have two agents each. Each source task consists of N states, but we make states $s < s_{\text{Att}}$ terminal states in Chainball-N-Att, and states $s > s_{\text{Def}}$ terminal states in Chainball-N-Def. Finally, both source tasks terminate if a terminal state is reached, or if a goal is scored or conceded, or after 90 timesteps. Our experiments use $N = 11$, $s_{\text{def}} = 6$ and $s_{\text{def}} = 6$.

Hyperparameter	Chainball	Overcooked
General hyperparameters		
Discount rate (γ)	0.99	0.99
GAE λ	0.95	0.95
n-steps	4	16
Optimiser	Adam	Adam
Adam ϵ	1×10^{-5}	1×10^{-5}
Gradient Clipping	False	False
Actor learning rate	1×10^{-2}	2×10^{-4}
Critic learning rate	2×10^{-2}	4×10^{-4}
Entropy coefficient (α)	1×10^{-5}	8×10^{-3}
Actor architecture	Tabular	FC, ReLU, hidden: [256, 128]
Critic architecture	Tabular	FC, ReLU, hidden: [256, 128]
PPO clip coef. (δ)	0.1	0.1
PPO epochs	2	2
PPO num. minibatches	1	1
PPO value clipping	False	False
Parallel environments	8	32
KL coefficient (κ)	8×10^{-3}	8×10^{-3}
MEDoE-specific hyperparameters		
Base temp. (T_{base})	1	1
Base KL coef. (κ_{base})	1.3×10^{-4}	3.2×10^{-3}
Base ent. coef. (α_{base})	1.6×10^{-6}	1.3×10^{-3}
Base clip coef. (δ_{base})	2.5×10^{-4}	2×10^{-4}
Temp. boost (β_T)	3	3
KL coef. boost (β_κ)	40	40
Ent. coef. boost (β_α)	40	40
Clip coef. boost(β_δ)	400	400

Table 1: Hyperparameter settings for baselines used in experiments

For each state, we store a forward probability table F, defined such that

$$F_s(a_1, a_2, a_3, a_4) = T(S_{t+1} = s + 1 | S_t = s, \mathbf{a}_t = (a_1, a_2, a_3, a_4)). \quad (10)$$

For each run seed, we generate each element of the table uniformly randomly in the interval $[0, 0.5]$, and then, for each state we set one of the $4^4 = 256$ elements to 0.8 to represent a known optimal joint action. To reduce the difficulty for the full task with 4 players, we make the optimal action in state 5 depend only on the joint action of (a_1, a_3), and the optimal action in state 7 depend only on the joint action of (a_2, a_4):

$$\forall a_1, a_3, a_1', a_3', F_5(a_1, a_2, a_3, a_4) = F_5(a_1', a_2, a_3', a_4),$$
$$\forall a_2, a_4, a_2', a_4', F_7(a_1, a_2, a_3, a_4) = F_7(a_1, a_2', a_3, a_4').$$

We apply a similar procedure to populate the forward probability tables for the attack and defence source tasks. However, for these source tasks, we do not reduce the difficultly of any states, as each state only has $4^2 = 16$ joint actions.

We design the optimal action in each source task to overlap with the target task. For example, in Chainball-11-Def, we set the optimal action in states 1,2,3, and 4 for agents 1 and 2 to be equal to the optimal action for agents 1 and 2 in states 1,2,3, and 4 of the target Chainball-11 task. Similarly, we set the optimal action for agents 1 and 2 in Chainball-11-Att in states 8,9,10,11 to be equal to the optimal action for agents 3 and 4 of the Chainball-11 task. Outside of these specified states, we require each agent to learn to take actions different to those which were optimal in its source task.

Despite the simplicity of the chainball task, it is difficult to solve due to the sparsity of reward and fact that in most states only one out of 256 joint actions is optimal.

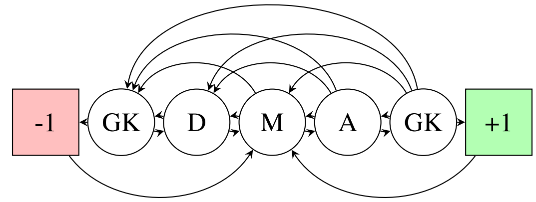


Figure 7: Chainball-5 Environment. Here states are re-labelled with GK (goalkeeper), D (defence), M (midfield), and A (attack).

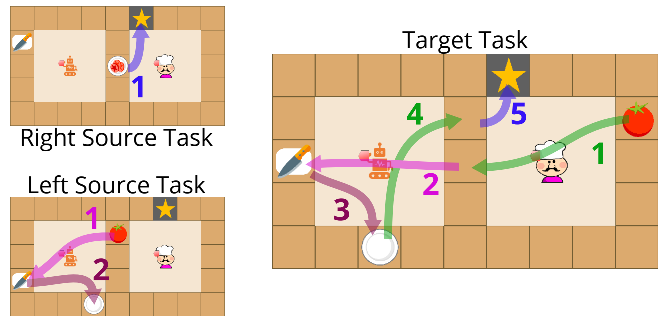


Figure 8: Overcooked Sub-task Curriculum. In the target task, agents must coordinate to pass and chop the tomato at the chopping board (1,2), put the chopped tomato on a plate (3), and pass the plate with chopped tomato back to serve at the starred counter (4,5). The skills to complete steps 2 and 3 can be learned in the "Right" source task; and skills to complete step 5 can be learned in the "Left" source task. Steps 1 and 4 require learning new behaviour in the target task.

Expert Domain of Expertise Classifier

As discussed in the previous section, we design the source task forward probability tables such that we know the defenders (agents 1 and 2 in the full task) are experts in states 1,2,3, and 4; and know that the attackers (agents 3 and 4 in the full task) are experts in states 8,9,10, and 11. This means we can specify ground-truth DoE classifiers:

$$\hat{D}_1(s) = \hat{D}_2(s) = \mathbb{1}[s \leq 4] \qquad \text{(defender DoE)}$$
$$\hat{D}_3(s) = \hat{D}_4(s) = \mathbb{1}[s \geq 8] \qquad \text{(attacker DoE)}$$

C.2 Overcooked

Environment Description

Overcooked is a complex environment which requires multi-step coordination by agents. The goal of Overcooked is to complete a recipe by moving and processing foods in a grid world. Figure 8 shows the configuration of our Overcooked target task and sub-task curriculum.

The action space in overcooked has 7 actions: 4 cardinal movement actions, a no-op action, and 2 interaction actions: one of which can be used to pick up objects, or put them down on counters; and the other which uses the chopping board when the tomato is placed on it. The object interacted with depends on the orientation of each agent (up/down/left/right).

We use an egocentric observation for each agent which has information about:

- the ego agent's current absolute location and absolute orientation,
- the other agent's current relative location and absolute orientation,
- the relative location and chopped state of the tomato,
- the relative location of the plate,
- the relative location of the chopping board,
- the relative location of the starred delivery tile.

We define rewards such that the maximum attainable return in each task is 1. In the full task, the team is rewarded with:

- +0.267 reward for chopping the tomato on the chopping board,
- +0.267 reward for putting chopped tomato on the plate,
- +0.476 reward for delivering the plate to the starred location.

In the "Right" source task, the team is rewarded with +1.0 reward for delivering the plate to the starred location. In the "Left" source task, the team is rewarded with +0.5 chopping the tomato, and +0.5 reward for placing the chopped tomato on the plate.

Overcooked terminates once all recipe steps have been completed, or after 100 timesteps.

Figure 9 shows how we initialise the position of objects in Overcooked. In each scenario, we uniformly randomly choose a side of the room to spawn the first agent in, placing that agent at the centre of the chosen side; then we spawn the second agent in the centre of the other side. The other objects are spawned uniformly randomly in:

- the plate in one of the 3 counter positions on the bottom of the left-hand half of the room,
- the starred service tile in one of the 3 counter positions on the top of the right-hand half of the room,
- the chopping board in one of the 3 counter positions on the left of the left-hand half of the room,
- (Right source task only): the tomato-on-plate on one of the 3 central counter positions,
- (Left source task only): the tomato on one of the 3 central counter positions,

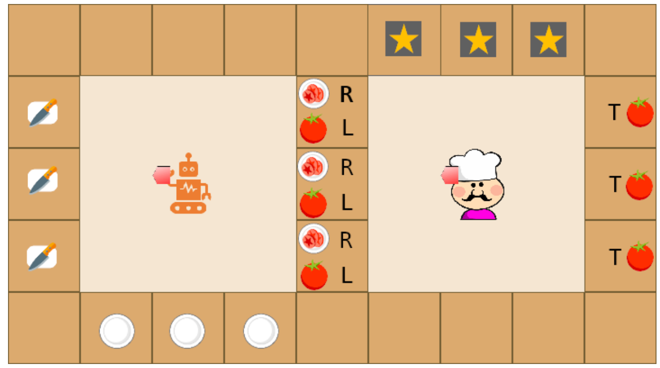


Figure 9: Spawn locations for objects in overcooked. Objects are labelled with "L" for "Left source task spawn position only", "R" for "Right source task spawn position only", and "T" for "Target task spawn position only".

- (Target source task only): the tomato on one of the 3 counter positions on the right of the right-hand half of the room.

Expert Domain of Expertise Classifier

The DoE classifier for Overcooked is based on the state and position of the tomato in the task. For the agent in the left-hand half of the room, $\hat{D}_{\text{left}}(s) = 1$ iff:

- the tomato is in the left half (including centre counters), *and*
- the plate is in the left half (including centre counters), *and*
- the tomato is *not* on the plate.

For the agent in the left-hand half of the room, $\hat{D}_{\text{left}}(s) = 1$ if:

- the tomato is chopped, *and*
- the tomato is on the plate, *and*
- the tomato is in the right half (including centre counters).

D Learned DoE Classifier

In this section, we provide further details about the learned DoE classifier used in our experiments.

Each agent's DoE classifier is represented by a In both Chainball and Overcooked, we use a fully-connected neural network with a single hidden layer of 128 units with ReLU activation functions. We use a learning rate of 1×10^{-2} with a batch size of 512, and train for 1 epoch. We minimise a binary cross-entropy loss function.

In Chainball, each agent stores 40,000 source task observations in its experience buffer. We label each observation with its source task, then concatenate and shuffle the experience buffer with an experience buffer from the other source task. For example, if agent i is trained in the defence source task, and agent j is trained in the attack source task, then we form a dataset of labelled examples which looks like:

$$\{\langle o_{i,734}, \texttt{def}\rangle, \langle o_{i,233}, \texttt{def}\rangle, \langle o_{j,1729}, \texttt{att}\rangle, \langle o_{i,333}, \texttt{def}\rangle, \dots\}.$$

This dataset has 80,000 labelled examples. We reserve 10% of this dataset as a test set, and train the network on the remaining 90%.

We follow a similar procedure for the Overcooked task, though in this case each experience buffer stores 320,000 source task observations, for a total dataset size of 640,000.

E Source Code

We will make our source code available in the camera-ready version of this paper.