Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

Open-Ended Coordination for Multi-Agent Systems Using Modular Open Policies

David Rother, Joni Pajarinen, Jan Peters, Thomas H Weisswange

2025

Preprint:

This is a post-peer-review, pre-copyedit version of an article published in Autonomous Agents and Multi-Agent Systems. The final authenticated version is available online at: https://doi.org/10.1007/s10458-025-09723-7

Auton Agent Multi-Agent Syst (2025) 39:40
https://doi.org/10.1007/s10458-025-09723-7

Open-ended coordination for multi-agent systems using modular open policies

David Rother[1,3] · Joni Pajarinen[2] · Jan Peters[1] · Thomas H. Weisswange[3]

Received: 19 February 2025 / Accepted: 17 September 2025
© The Author(s) 2025

Abstract
Significant multi-agent advances addressing the challenge of learning policies for acting in ad hoc teamwork have been made. In ad hoc teamwork, a team of agents must cooperate effectively without prior coordination or communication. Many existing approaches, however, struggle to perform well in open environments where the setting can change significantly during deployment. This paper presents a new reinforcement learning approach to tackle collaboration in open environments controlling one agent with a changing number of distinct other agents, each with an individual task. The approach uses policy blending based on an online goal inference module and a collection of learned policies modeling the individual interaction impact between the agent and populations of partners with different tasks. Blending is done using the estimated goals of others and a posterior-based action blending with entropy adjustment and regularization. Our approach addresses issues of existing policy blending mechanisms, such as handling conflicting modes in action distributions leading to oscillation and instability and adapting to uncertain states dynamically. In experiments in two collaborative open environments based on Overcooked and Level-based Foraging, our approach outperforms a baseline learner, trained with the joint reward of all agents, across changes to both agents and tasks. Ablation studies further highlight the importance of our posterior-based blending mechanism to achieve high rewards as well as the provided goal weighting. The proposed approach provides an important step towards the application of reinforcement learning to AI assistance beyond strictly closed worlds and towards more realistic scenarios.

Keywords Reinforcement learning · Multi-agent systems · Policy blending · Maximum entropy · Cooperative intelligence

1 Introduction

In real-world multi-agent environments, autonomous agents must interact with diverse partners, each potentially pursuing different tasks, all while adapting to new and evolving objectives over time. Consider a warehouse robot that must simultaneously coordinate with other

Extended author information available on the last page of the article

Published online: 10 October 2025

robots, human workers, and incoming delivery drones. Each of these entities has distinct goals, which may change as new tasks emerge. This complexity highlights the necessity of scalable and adaptable frameworks capable of handling variable team compositions, mixed motives, and continuously expanding task sets.

The challenge lies in managing these multi-agent, multi-task interactions effectively. While research in multi-agent reinforcement learning (MARL) has made progress in coordination strategies, such as centralized training with decentralized execution, policy distillation, and meta-learning techniques [1–3], these methods typically assume static task definitions or pre-defined agent interactions. In contrast, real-world environments demand that agents continually acquire new skills without sacrificing previously learned abilities. This requires balancing adaptation to new objectives with the retention of existing competencies, avoiding catastrophic forgetting, and coordinating with unfamiliar partners. Consequently, a framework is needed that not only meets traditional reinforcement learning objectives but also incorporates elements of lifelong learning, ensuring robust performance in open, dynamic multi-agent systems. While existing lifelong learning approaches focus on preserving knowledge through regularization or memory replay [4, 5], they do not address the need for zero-shot coordination in settings where agents must interact with unknown partners pursuing different objectives. Current work also does not adequately address the need for simultaneous coordination with multiple agents pursuing different objectives, and real-world challenges demand solutions that address both the dynamic nature of tasks and the inherent variability in agent interactions.

To tackle these challenges, we introduce Double Open Stochastic Bayesian Games (DOSBG), a novel Markov Decision Process (MDP) formulation describing a double open teamwork challenge, which extends the principles of open teamwork to multi-task environments as depicted in Fig. 1 with an open set of tasks. Environments may include both artificial agents and humans, along with varying tasks assigned to each of these agents.

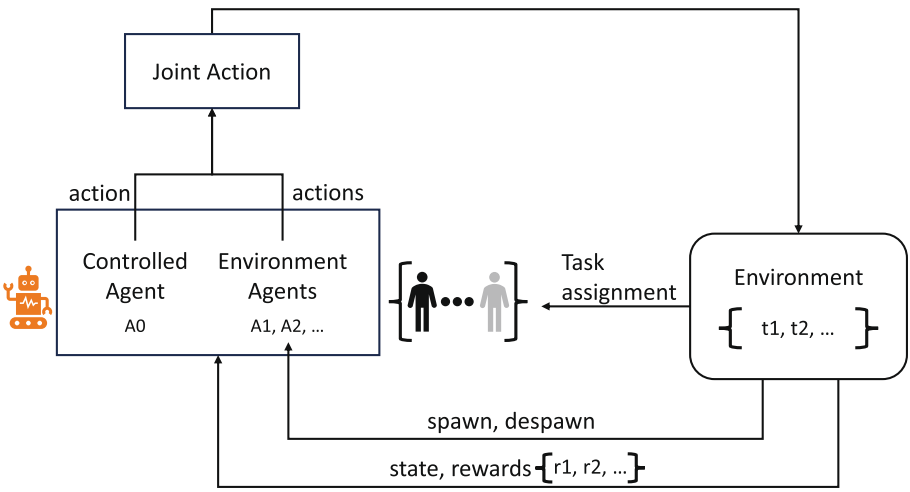


Fig. 1 The open multi-agent world loop. In an open loop, the other autonomous agents may spawn or despawn at any step, causing a variable team size. In a mixed-motive environment, all agents may have different reward functions expressed as different tasks in the world

In our work, the controlled agent does not pursue an independent task for itself but instead focuses on assisting other agents in achieving their goals. This cooperative role fundamentally shifts the focus to modeling and understanding other agents, as successful collaboration depends on accurately predicting their objectives, behaviors, and interactions. Consequently, effective agent modeling becomes a cornerstone of our approach, enabling the agent to provide meaningful support in a dynamic and open environment. This perspective amplifies the importance of handling open challenges, as the agent must dynamically adapt to new partners, changing goals, and evolving tasks while maintaining its ability to generalize across diverse scenarios.

We propose a new method that enables agents to learn and adapt to new interaction tasks incrementally in a continual learning setting, without retraining existing components or accessing past training data. Our approach involves learning a separate interaction policy for each potential task using Reinforcement Learning. For any scene involving multiple agents, we blend the outputs of the relevant interaction policies to form a combined probability distribution. Actions are then sampled from this blended distribution, allowing for seamless integration of new policies corresponding to new tasks or partners.

This modular design offers several advantages. It supports lifelong learning by allowing the agent to incorporate new interaction policies as they become relevant without affecting previously learned behaviors. It enables the agent to coordinate with unseen partners based on learned interaction patterns. In addition, the blending mechanism ensures that the agent's decision-making process is scalable, even as the number of interaction policies grows.

Our contributions are as follows:

1. **A Method for Open-Ended Open System Coordination**: We introduce a method that incrementally learns interaction policies for new tasks or agents, enabling lifelong adaptation without retraining existing policies.
2. **Blended Policy Mechanism**: We develop a blending method that combines the outputs of multiple interaction policies to guide action selection in multi-agent environments.
3. **Experimental Validation**: We demonstrate the effectiveness of our approach in complex coordination tasks.

The paper is organized as follows: Section 2 provides an overview of related work in open ad hoc teamwork, multi-task learning, modular reinforcement learning, as well as other adjacent literature that shares part of the introduced problem setting. Section 3 introduces the concept of Double Open Stochastic Bayesian Games and its inherent challenges. In Section 4, we describe the modular open policy architecture in detail. Section 5 presents the experiment environments and the training and test regimes. In Section 6, we present the results and show the effectiveness of the modular open policy approach in various environments, including multi-task, multi-agent scenarios, and open and closed-world scenarios. Finally, we conclude the paper in Section 7, providing a summary of our findings and discussing potential directions for future research.

2 Related work

Multiple existing approaches in different research directions in Reinforcement Learning (RL) tackle specific aspects of open-world challenges. We discuss Modular Reinforcement Learning (MRL), providing algorithmic techniques for scalability and lifelong learning. Multi-agent methods study how to coordinate multiple agents operating simultaneously. Ad-hoc teamwork research addresses flexible collaboration with changing sets of teammates. Interaction partner modeling allows for adaptation to the changing behavior of other agents. We build upon these ideas by considering temporal openness, in which agents can continue to act even after completing their tasks. This expands the requirements to handle shifting teammates and tasks and manage ongoing activity over time.

2.1 Modular reinforcement learning

Modular reinforcement learning has emerged as a pivotal approach to enhance the scalability and adaptability of RL agents in complex and dynamic environments. The fundamental idea is to decompose a monolithic RL problem into smaller, more manageable sub-problems, each addressed by a specialized module or subpolicy. Early work by Uchibe et al. [6] introduced behavior coordination for mobile robots using MRL, demonstrating that separate modules could learn distinct behaviors such as obstacle avoidance and goal achievement, which could then be coordinated to accomplish complex tasks. Samejima et al. [7] tackled the challenge of inter-module credit assignment in MRL, proposing methods to distribute the global reward signal appropriately among modules. Their work improved learning efficiency by ensuring that each module received feedback relevant to its specific function, a concept that aligns with our approach of training interaction policies with reward signals from other agents. But our approach, instead of learning all policies at the same time, learns them one by one in an open-ended fashion. In the realm of complex dynamic environments, Wang et al. [8] proposed a modular deep RL framework that enabled agents to navigate through intricate and changing scenarios. By decomposing the navigation task into modules handling different environmental aspects, their approach enhanced adaptability and generalization, which is essential in open multi-agent systems. Andreas et al. [9] introduced modular multitask RL with policy sketches, where policies are composed based on high-level task descriptions. This method allows for the reuse of policy components across tasks, significantly improving learning efficiency in multitask settings. Our work shares this philosophy by enabling the recombination of pretrained task and interaction policies to address new tasks without retraining. Simpkins and Isbell [10] discussed composable modular RL, emphasizing the importance of modularity in allowing agents to adapt to new tasks by reusing and recombining existing modules. Their findings support our multi-stage learning approach, where modular policies are trained separately and later combined to solve more complex multi-agent tasks. In multi-agent systems, Ming et al. [11] proposed cooperative modular RL for large discrete action space problems. Their method addresses scalability issues by decomposing the action space and enabling agents to learn cooperative behaviors through modular policies. This approach parallels our work in handling the scalability with the number of agents and tasks. Sarantopoulos et al. [12] applied MRL to robotic manipulation, specifically in total singulation tasks. Their approach demonstrated that modular policies could handle complex manipulation tasks requiring coordination of multiple behaviors,

highlighting the versatility of MRL in different domains. In the context of autonomous negotiating agents, Bakker et al. [13] introduced RLBOA, a modular RL framework that allows agents to learn negotiation strategies. Their work illustrates the applicability of MRL in strategic interaction scenarios, which is relevant to our focus on interaction policies in multi-agent settings. To learn a blending mechanism for newly learned and given policies [14] introduces an arbitrator that assigns a weight to each module dynamically in each state. The action space of the learned arbitrator then becomes the weight space of the learned modules, and the final aggregated action is taken with an argmax over the resulting Q-values. By choosing the magnitude of weights through an arbitrator,r they can handle different reward scales. Our approach relies on the entropy statistic to handle different modules and tailors the blending mechanism towards a goal without learning an additional component. This allows us to introduce new modules in a continuous manner instead of having to have a predetermined fixed number of modules.

Despite the advantages, MRL faces challenges such as the difficulty of integrating modules with different reward scales and keeping a stable policy. Bhat et al. [15] discussed these issues, including partial observability and coordination among modules. Our method addresses some of these challenges by employing a divergence minimization objective and a posterior-based blending mechanism that dynamically adjusts module contributions based on the current state and agent interactions, increasing stability.

2.2 Multi-agent reinforcement learning

Multi-agent reinforcement learning (MARL) was initially applied to settings where all agents were known in advance, often focusing on cooperative tasks with centralized training or control. This setup allowed researchers to address the credit assignment problem in a straightforward manner, but limited the scope of the scenarios considered. Over time, more flexible methods emerged, introducing decentralized control and learning so that each agent could adapt its own policy. Independent Q-Learning (IQL) [16] exemplifies this decentralized trend, although it suffered from the non-stationarity of the sampled data as agents' policies evolved in parallel.

Centralized solutions such as COMA [2] tackled the non-stationarity challenge by employing a centralized critic, while still executing decisions locally at each agent. Lowe et al. [1] extended these ideas to handle mixed cooperative and competitive objectives via multi-agent actor-critic frameworks. Value decomposition methods, including VDN [17] and QMIX [18], addressed scalability by factorizing a global value function into individual agent contributions. These methods work well when agent sets and tasks remain stable, typically in cooperative environments. Ming et al. [11] applied modular RL to large discrete action spaces, improving scalability.

Despite these advancements, most MARL techniques still assume a fixed or slowly varying number of agents and a single task or set of cooperative tasks. They do not inherently address frequent agent turnover or the continuous introduction of new tasks. By contrast, our approach treats interaction policies as modular components, allowing agents to be added or removed and tasks to change. This perspective expands the scope of MARL toward more open-ended domains, where agent populations and task requirements may shift continuously.

2.3 Interaction partner modeling

Modeling the behaviors, intentions, and strategies of other agents is a key challenge in multi-agent reinforcement learning. Modeling and learning effective interaction policies are essential in cooperative and competitive multi-agent settings. Albrecht and Ramamoorthy [19] studied game-theoretic models for learning to forage in multi-agent systems, providing insights into cooperative behaviors. Other earlier work in opponent or partner modeling examined how to infer hidden parameters of a counterpart's policy in competitive settings, often by maintaining explicit belief distributions over possible behaviors [20]. Subsequent research has extended such techniques to cooperative scenarios, highlighting the need for flexible methods that adapt to changes in a partner's strategies [1, 21]. Our approach to reward-correlated latent representations builds on these foundations by automatically extracting the features that are most relevant for effective coordination in multi-agent interactions. In doing so, we exploit the learned latent encoding to create modular components (i.e., the interaction policies) that can be recombined when the number of agents or tasks changes. This design principle aligns closely with the spirit of our modular open policy framework, in which each sub-policy is dedicated to a particular agent-task pair, yet can be blended into a unified decision mechanism at inference time. We therefore view our interaction modules as a form of partner modeling that is tightly integrated into the compositional structure of our multi-stage learning procedure.

2.4 Ad-hoc teamwork

Ad-hoc teamwork addresses the challenge of collaborating with previously unknown partners on a shared task [22]. In the basic formulation, there is a single cooperative task and a fixed number of potential teammates; each agent must adapt to whichever teammates are present in a given scenario. Early solutions focused on learning models of other agents' behaviors based on observations and introduced type-based reasoning [23, 24], allowing an ad-hoc agent to predict actions of specific partner types for more effective coordination.

Subsequent work investigated extensions where the size of the team might vary from one episode to another [25], while still assuming cooperation toward a single goal. Graph-based approaches and relational forward models [26] capture joint-action values pairwise for each agent-agent interaction, providing a more scalable way to handle different combinations of teammates. Further studies considered situations with multiple tasks, motivating architectures that can handle additional tasks sequentially rather than learning them all at once. Examples include the use of separately trained policies for each agent-task combination [27, 28].

Despite these advances, existing ad-hoc teamwork solutions typically focus on collaborative tasks and limit themselves to selecting the most appropriate policies for a single team objective. In contrast, the scenario we target incorporates potentially mixed motives among agents, an open-ended set of tasks, and ongoing interactions beyond the immediate completion of any one goal. This broader setting introduces challenges extending existing ad-hoc approaches, highlighting the need for algorithms capable of handling more complex temporal and task set openness forms.

3 Problem formulation

In this section, we introduce the Double Open Bayesian Stochastic Game (DOSBG), an extension of the Markov Decision Process (MDP) framework. The DOSBG formulation addresses the challenge of dynamic agent participation and open-task environments, where agents may join or leave episodes, pursue different goals, and adapt to a stream of new tasks over time in a continuous learning framework, where tasks serve as the primary mechanism to structure agent behavior in DOSBG. The double open environment framework is open with respect to the agent number and task presence, that is, the tasks of the agents currently in the environment. At any given time step, the number of agents may change without bound. The set of tasks that are pursued by agents and thus present at any given time can change.

3.1 Double open stochastic Bayesian game

The Double Open Bayesian Stochastic Game is defined by the 6-tuple $\langle N, \mathbb{S}, \mathrm{A}, P, R, \mathrm{T} \rangle$, where N is the set of agents, S is the set of states, A is the set of all possible joint-actions of all agents, R is the reward function, P is the transition function, and T is the set of tasks, enabling us to handle multi-task scenarios. Let $\mathcal{P}$ denote the power set. We define the joint agent-action space for a variable number of agents as $\mathbf{A_N} = \{a | a \in \mathcal{P}(N \times \mathbb{A}), \forall (i, a^i), (j, a^j) \in a : i = j \Rightarrow a^i = a^j\}$ with $a \in \mathbf{A_N}$ being joint agent-actions. Similarly, the joint agent-task space is defined as $\mathrm{T_N} = \{\tau | \tau \in \mathcal{P}(\mathrm{T} \times A), \forall (i, \tau^i), (j, \tau^j) \in \tau : i = j \Rightarrow \tau^i = \tau^j\}$ with $\tau \in \mathrm{T_N}$ being joint agent-tasks. We constrain each agent to be assigned to exactly one task. The policy function is described as π^i_τ to indicate that the *i*-th agent is working on task τ, allowing different agents to perform different tasks.

In our problem setting, we assume full observability of the state and rewards during training, as well as knowledge about the task of each agent. The joint agent-reward function is defined as $R : S \times \mathbf{A_N} \times \mathrm{T_N} \to \mathbb{R}^N$, with R_i being the reward function corresponding to agent *i*. The reward function in DOSBG directly depends on the tasks assigned to the agents. Tasks define the objectives that agents are expected to accomplish, and the reward function encodes the desirability of outcomes relative to those objectives. The transition function is formally defined as $P : S \times \mathbf{A_N} \to \Delta(S \times \mathrm{T_N})$. It gives the probabilities of the next state and joint agent-tasks, given the current state and joint agent-actions. $\Delta(X)$ denotes the set of all probability distributions over X. Note that technically, agents may change tasks dynamically, but we assume that they are fixed in this work.

At the beginning of a game, an initial state s_0, and an initial set of agents N_0, with associated tasks, τ^i_0 with $i \in N_0$, are sampled from an initial distribution $P_0 \in \Delta(S \times \mathrm{T_N})$. A task in the context of DOSBG represents a specific objective or goal that an agent or group of agents aims to achieve during an episode. A transition from state s_t is done as follows: From the set of existing Agents $N_t \subseteq N$ and tasks τ^i_t, $i \in N_t$ the joint-action a_t is constructed, by individually following the probability distributions from their respective policy π^i producing the agent actions $\mathrm{a}_t \sim \pi(s|\cdot)$. Given the state and joint-action, we can then sample from P the next set of Agents N_{t+1}, next tasks τ_{t+1}, and the next state s_{t+1}. Note that due to the task-assignment restriction, $\tau^i_{t+1} = \tau^i_t$ for all $i \in N_{t+1}$. While the task assignment to a single agent can not change, a new agent may enter at any time with a new task, allowing for present task changes as well as agent number changes.

3.2 Optimal policy estimation in double open stochastic Bayesian game

Let us assume that i denotes the agent we are interested in training, the optimal policy is defined below.

Definition 1 Let the joint actions and the joint policy of agents other than i at time t be denoted by a_t^{-i} and π_t^{-i}, respectively, in the following notation. We omit the other actions or policies at times for visual clarity, when their actions are not relevant, such as for the definition of optimality of the agent's own policy. Given a discount factor, $0 \leq \gamma \leq 1$, we define the action-value of policy π^i,$\hat{Q}_{\pi^i}(s, a^i)$, as

$$Q_{\pi^i}(s, a^i) \;=\; \mathbb{E}_{a_t^i \sim \pi^i, a_t^{-i} \sim \pi_t^{-i}, P}\left[\sum_{t=0}^{\infty} \gamma R_i(s_t, a_t, \tau_t) | s_0 = s, a_0^i = a^i, \tau_0 = \tau\right],$$

which denotes the expected discounted return after the learner executes a^i at s. A policy $\pi^{i,*}$ is optimal if

$$\forall\, \pi^i,\ \forall\, s \in \mathcal{S},\ \forall\, a^i \in \mathcal{A}^i : \quad Q_{\pi^{i,*}}(s, a^i) \;\geq\; Q_{\pi^i}(s, a^i),$$

implying that there is no other policy in any other state that can achieve a higher expected return. Given $Q_{\pi^{i,*}}(s, a)$, a DOSBG is solved by always choosing a^i with the highest action-value at s. Note that we can augment the reward with an entropy term $\mathcal{H}(\pi^i(\cdot|s))$, which was shown to produce better policies for blending [29], to obtain the maximum entropy objective of the problem formulation, which we will be using going forward.

4 Modular open policies

In this section, we first introduce a reformulation of the learning objective, making an additional assumption. We then show how we can leverage this reformulation using a learning curriculum on a stream of tasks and obtain an optimal policy at inference time for the multi-task multi-agent challenge afterward. We describe in detail the different modules and the respective architectures used. A general overview of the learning architecture is given in Fig. 2.

4.1 Divergence minimization objective

Our learning agent has access to the learning signal of each agent, as such we formulate our optimization problem as the sum of the expected rewards over all agents as

$$J(\pi^i) = \sum_{\substack{j=1 \\ j \neq i}}^{N} \mathbb{E}_{s' \sim P(s,a)}\left[R_j(s, \pi^i(s), \pi^{-i}(s), \tau^j) + \gamma V_{\pi^i}(s')\right]. \tag{1}$$

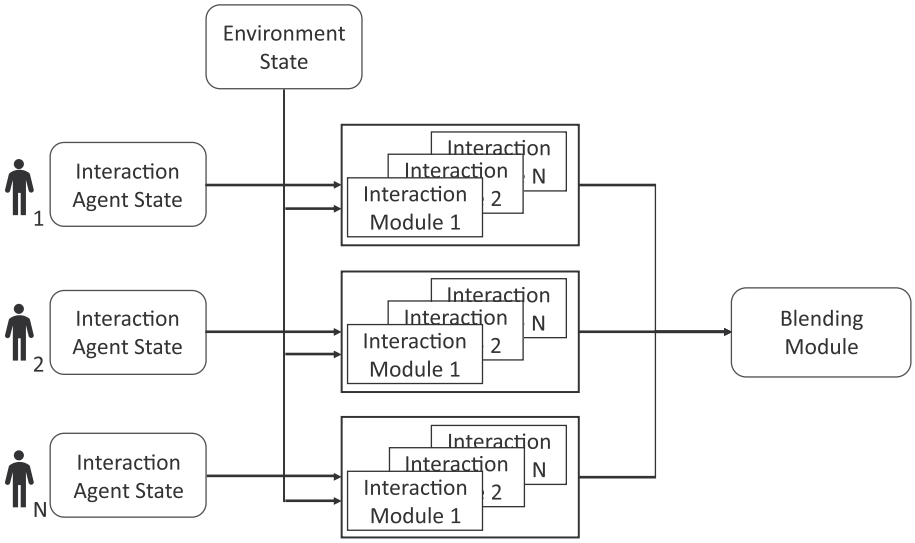


Fig. 2 Information flow of the modular open policy architecture. The state is preprocessed and separated into the information about the world state and separate information for each agent entity. For each agent entity, an interaction module computes the corresponding best response action distribution. The blending module then computes a final action distribution

For this work, the ego agent does not pursue its own task (i.e., $R_i(\cdot) = 0$); hence, its performance is evaluated solely based on the rewards of the other agents ($j \neq i$). Obtaining the policy π^i is a notoriously hard learning problem, exacerbated by the fact that the number of agents may change dynamically, and agents can have a variety of tasks. Additionally, we impose the constraint that a learning architecture should handle the addition of new tasks without retraining previously learned ones. To address these challenges, we define the policies π^i and π^c as probability distributions over actions

$$\pi^i(a^i \mid s, \tau^i) : \mathbb{S} \times \mathrm{T} \to \Delta(\mathbb{A}),$$

where $\pi^i(a^i \mid s, \tau^i)$ gives the probability of agent *i* selecting action a^i given the current state *s* and task τ^i. Similarly, the combined policy $\pi^c(a^i \mid s)$ outputs a distribution over actions without being tied to any specific task configuration, thereby allowing flexible integration and reassignment of policies.

This distinction allows us to decompose the optimization problem into task-specific components ($\pi^{i,j}$) and a combined policy (π^c). The intuition here is that $\pi^{i,j}$ represents an agent *i*'s hypothetical optimal policy for interacting with another agent *j*, given its task τ^j. By leveraging $\pi^{i,j}$ as a performance upper bound, we can design a tractable objective function to minimize the gap between π^c and the hypothetical optimum. The closest solution to this hypothetical optimum is the practical optimum. We rewrite our objective function using a hypothetical upper bound on the performance of our agent. The upper bound is given by

$$J^i_{\text{ub}}(s|\pi^{i,1},...,\pi^{i,N}) = \sum_{a^i \in \mathcal{A}^i} \sum_{\substack{j=1 \\ j \neq i}}^{N} \pi^{i,j}(a^i \mid s, \tau^j) \left[R_j(s, a^i, \pi^{-i}(s), \tau^j) + \gamma \, \mathbb{E}_{s' \sim P(s,a)} V_{\pi^i}(s') \right].$$

where $\pi^{i,j}$ is the optimal policy towards agent j by first optimizing for the Q-function

$$Q^{i,j}(s, a^i, \tau^j) \;=\; \mathbb{E}_{s' \sim P(s,a)} \left[R_j(s, a^i, \pi^{-i}(s), \tau^j) + \gamma V_{\pi^i}(s') \right],$$

which then gives us the policy as

$$\pi^{i,j}(a^i \mid s, \tau^j) \;=\; \frac{\exp\left(Q^{i,j}(s, a^i, \tau^j)\right)}{\sum_{a' \in \mathcal{A}^i} \exp\left(Q^{i,j}(s, a', \tau^j)\right)},$$

with $\pi^{i,j}$ optimizing the own reward if $i = j$. Given an upper bound over the expected reward we minimize the distance of the expected reward given some policy π^c only for the current time step at state s as

$$\begin{aligned}
&D(s|\pi^{i,1},...,\pi^{i,N},\pi^c) = J_{\text{ub}}(s|\pi^{i,1},...,\pi^{i,N}) - J(s|\pi^c) \\
&= \frac{1}{N-1} \sum_{a^i \in A^i} \sum_{\substack{j=1 \\ j \neq i}}^{N} \pi^{i,j}(a^i, s) Q_{\pi^{i,j}}(a^i, s, \tau^j) - \sum_{a^i \in A^i} \pi^c(a^i, s) Q_{\pi^{i,j}}(a^i, s, \tau^j) \\
&= \sum_{a^i \in A^i} \sum_{\substack{j=1 \\ j \neq i}}^{N} \pi^{i,j}(a^i, s) Q_{\pi^{i,j}}(a^i, s, \tau^j) - \pi^c(a^i, s) Q_{\pi^{i,j}}(a^i, s, \tau^j) \\
&= \sum_{a^i \in A^i} \sum_{\substack{j=1 \\ j \neq i}}^{N} (\pi^{i,j}(a^i, s) - \pi^c(a^i, s)) Q(a^i, s, \tau^j),
\end{aligned}$$

reducing this problem to minimizing the regret of using policy π^c over policy $\pi^{i,j}$. Solving for π^c can be seen as a weighted least squares regression problem, by assuming categorical probability distributions and by constructing normalized weights

$$C(a^i, s, \tau^j) = \frac{Q^{i,j}(s, a^i, \tau^j)}{\sum_{a' \in \mathcal{A}^i} Q^{i,j}(s, a', \tau^j)}.$$

The result of the optimal policy is

$$\pi^{c,*}(a^i \mid s) = \sum_{\substack{j=1 \\ j \neq i}}^{N} \pi^{i,j}(a^i \mid s, \tau^j) \, C(a^i, s, \tau^j).$$

It was previously shown that maximum entropy distributions are optimal towards their composability [30] and minimize the expected divergence between them [27]. Following the equivalence of soft-q learning and entropy regularized policy gradient methods in their learned objective [31], we argue that to get the optimal solution, both maximum entropy Q learning and policy gradient methods are viable training algorithms to obtain the policies $\pi^{i,j}$.

We want to emphasize that to be the optimal solution, the weighting of the policies through their respective assigned credit needs to be optimal. However, in practice we deal with sub-optimal estimates of the true value, especially when one policy explores an unknown state space for others, which cannot provide accurate estimates for these states and thus renders the weighted Q function inapplicable. Additionally, the estimates of the credit towards each task must be correctly scaled according to the user's belief system, posing a significant challenge in reward function design. To alleviate these problems, we propose additional mechanisms that include an adaptive credit weighting scheme and an uncertainty-aware correction term. The adaptive scheme dynamically adjusts the credit assigned to each policy based on confidence measures, while the correction term regularizes the weighted Q function toward more stable baseline estimates. Together, these enhancements enable our architecture to handle estimation errors better and adapt to the dynamic conditions of multi-agent interactions, ensuring that the final blended policy remains robust.

4.2 Multi stage learning

Algorithm 1 Two stage learning strategy for open-ended tasks in DOSBG.

Input: Task set T, agents N, reward function R
Output: Policies $\{\pi^{i,j}\}$ for each agent-task pair
for each task $\tau \in$ T **do**
  Phase 1: Train Task-Specific Policies
  for each agent $i \in N$ **do**
    Train task-specific policy $\pi^{i,\tau}$ using PPO
  end for
  Phase 2: Train Interaction Policies
  for each agent $i \in N$ **do**
    for each other agent $j \neq i$ in N **do**
      Augment observation o_i with state s_j of agent j
      Train interaction policy $\pi^{i,j}$ to maximize agent j's expected reward (2)
    end for
  end for
end for

Curriculum learning strategies have been employed to improve training efficiency by gradually increasing task difficulty [32]. For example, Yang et al. [33] improve training efficiency by incrementally increasing task difficulty, often through the use of modular designs that allow agents to build capabilities in a structured manner. Our approach is similar but not equivalent because we construct different policies in multiple stages. The different stages are depicted in Algorithm 1. First, we construct policies for potential tasks the learning agent has to solve in isolation. In the second phase, interaction policies and corresponding action likelihood estimators are learned by training solely on the received reward of one other agent, whom we conduct training with, effectively maximizing

$$J(\pi^{\star}_{i,j}) \;=\; \mathbb{E}_{\tau\sim\pi_{\theta_{i,j}},\mathcal{P}}\Big[\sum_{t=0}^{T-1}\gamma^t\, R_j\big(s_t, a^j_t\big) \;\Big|\; a^i_t \sim \pi_{\theta_{i,j}}\big(a^i_t \mid \tilde{o}^i_t\big)\Big]. \tag{2}$$

This procedure is done for every task that other agents can have, spawning a new policy for each task. This allows adding new tasks later on without having to retrain old policies addressing the open-ended task challenge imposed by the DOSBG. The training of the components and their architectures are described in the following subsections. We train interaction models with one agent at a time. Policies are trained using maximum entropy policy gradient with the common PPO implementation details [34, 35] as well as the Generalized Advantage Estimate (GAE). The agent receives a reward signal for training the reward signal of the other agent. For the interaction policies that are computing the interaction distribution for agent j we augment the observation o^i with the state s^j of that agent.

4.3 Goal estimation

We maintain, for each other agent i, a probability distribution over potential goals $G^i = (p^i_1, \ldots, p^i_g)$ where g is the size of the task set. Each task τ is associated with a pretrained action prediction module $f_g(s_t)$, which outputs a likelihood distribution over actions given the agent's state s_t. Since we do not alter existing modules when new goals arise, no retraining of other estimators is required when the task set changes. Instead, newly introduced goals simply extend the set of available modules.

Formally, let s_t be the relevant processed state for agent i at time t, and a_t the action observed from that agent. We store a prior over goals $G^i(t)$, initialized to a uniform distribution when no information is available or inherited from past updates otherwise. The action prediction module for task τ provides a likelihood for each task. We then perform the Bayesian update

$$\tilde{p}^i_g(t+1) \;=\; p^i_g(t)\, f_g(\mathrm{s}_t)$$

where $\tilde{p}^i_g(t+1)$ is the unnormalized posterior for goal g. We normalize the resulting vector

$$p^i_g(t+1) \;=\; \frac{\tilde{p}^i_g(t+1)}{\sum_{g'=1}^{G}\tilde{p}^i_{g'}(t+1)}. \tag{3}$$

If the total probability mass is zero or close to zero, a uniform distribution over all goals is restored to avoid numerical instabilities.

This procedure integrates seamlessly with the agent's interaction policies and final blending strategy. For any agent i, we update the goal probabilities p^i each time an action is observed. Subsequently, the interaction policy retrieves the current goal distribution to weight the corresponding action distributions. Because each goal's prediction module remains fixed once trained, these updates avoid retraining costs, even as new tasks or goals are introduced.

4.4 Posterior-based action blending with entropy adjustment and regularization

Algorithm 2 Deployment of the modular open–policy agent.

Input:
Pre-trained interaction policies $\{\pi^{\text{ego},\tau}\}_{\tau\in\mathrm{T}}$
Goal–likelihood modules $\{f_\tau\}_{\tau\in\mathrm{T}}$
Regularisation weight $\lambda \in [0, 1]$
discount factor for entropy decay β.
Begin:
Initialise an empty agent dictionary $\mathcal{A}$
for each timestep t **do**
  Observe global state s_t and the current set of other agents $Alive(t)$
  for each agent $j \in Alive(t)$ **do** ▷ **Goal inference & policy set–up**
    if $j \notin \mathcal{A}$ **then**
      $G_j \leftarrow \text{Uniform}(|\mathrm{T}|)$
      $\alpha_j \leftarrow \frac{1}{|Alive(t)|}$
    end if
    Extract agent-specific features $\tilde{s}_t^{\,j}$ and last observed action a_{t-1}^j
    for each goal $\tau \in \mathrm{T}$ **do**
      $p_{j,\tau} \leftarrow G_{j,\tau}\, f_\tau(s_t^j)$
    end for
    $\pi_t^j(a) \;=\; \sum_{\tau\in\mathrm{T}} G_{j,\tau}\, \pi^{\text{ego},\tau}(a \mid s_t)$
    $H_j \leftarrow -\sum_a \pi_t^j(a)\, \log\big(\pi_t^j(a)\big)$
  end for
  Posterior weight regularisation
    $\forall j:\; P_j \leftarrow (1-\lambda)\, P_j + \lambda\, \frac{1}{|Alive(t)|}$
  Blend and act
    $\pi_t^c(a) \;=\; \sum_{j\in Alive(t)} P_j\, \pi_t^j(a)\, e^{-H_j}$
  Sample ego action $a_t^{\text{ego}} \sim \pi_t^c(\,\cdot\,)$ and execute
  Store (G_j, P_j) back into $\mathcal{A}$ for every j
  Remove entries for agents that have despawned at t
end for

In this section, we formalise a sequential weighting mechanism that prevents multimodality or oscillation of the merged policy. When merging policies by adding them up, multiple modes may arise in the probability distribution. In sequential problems, this can occur when the optimal behavior involves reversing the previously taken action. For example, if an agent alternates between an action and its reverse, for example, moving forward then backward, the merged policy might exhibit two distinct peaks: one favoring the original action and one favoring its reversal. This bimodality (or multimodality) reflects conflicting local optima that arise from the inherent oscillation in the agent's strategy. To alleviate this problem, we propose a posterior-based method for blending action distributions in multi-agent systems, which dynamically adapts to the agents' behaviors and goals. The posterior probabilities for each agent represent their contribution to the likelihood of sampled actions. These probabilities are iteratively updated, incorporating factors such as the entropy of the agent's action distribution and regularization towards a uniform prior. This approach ensures robust adaptation while avoiding overfitting to transient observations.

At each step, the posterior probabilities are adjusted based on the probability of the sampled action and the entropy of the agent's blended action distribution. The entropy H_i of the blended distribution for agent i is

$$H_i(s) = -\sum_a \pi_i(s,a) \log_b \pi_i(s,a),$$

where $\pi_i(a)$ is the probability of action a for agent i, and b is the logarithmic base equal to the number of actions. We use as factor $\exp(-H_i)$, which is applied to reduce the contribution of agents with high-entropy (uncertain) distributions, while retaining higher weights for agents with peaked distributions. Additionally, a regularization term is incorporated to move the posterior probabilities towards a uniform prior between updates. This prevents the posterior from overly favoring specific agents based on transient observations. For each agent i, the posterior is regularized as

$$P_i \leftarrow (1-\lambda)P_i + \lambda \frac{1}{N},$$

where λ is the regularization weight, N is the total number of agents, and P_i is the posterior probability for agent i. This ensures the system retains adaptability while maintaining a balanced contribution from all agents. Putting everything together, the posterior update combines the entropy-adjusted action probability and regularization. After normalization, the posterior probabilities are used to compute a weighted combination of action distributions. For a system with N agents, the probability of sampling action a in state s is given by

$$P_{\text{blend}}(s,a) = \frac{1}{N} \sum_{i=1}^{N} P_i \cdot \pi_i(s,a) \cdot G_i \cdot \exp(-H_i),$$

where $P_i(a)$ is the probability of action a under agent i's distribution, and G_i is the goal alignment probability for agent i from . An action is sampled from the resulting blended distribution, and the posterior probabilities are iteratively refined based on the sampled action.

This approach incorporates both entropy-based adjustments and regularization to ensure stability and adaptability. By reducing the influence of uncertain agents and maintaining a balanced prior, the method provides a principled framework for multi-agent action blending that is robust to dynamic changes in the environment and agent behaviors.

5 Experiment setup

To investigate the effectiveness of controlling a single agent in a multi-agent environment with unknown objectives, we conducted a series of experiments. Our objective is to maximize the collective reward. We study whether a modular blending approach with learned policies can successfully be used to coordinate helpful interactions in an open environment. In detail, we aim to answer whether our approach can successfully learn to act in DOS-BGs and investigate the following sub-questions. (1) Does our approach work in traditional

closed-world environments? (2) How does performance change with an increasing number of agents? (3) How does performance change when new tasks become known? (4) How important is the proposed merging mechanism for the performance?

5.1 Environments

To evaluate our framework, we select environments based on two primary criteria. First, the environment must allow multiple independent tasks of sufficient complexity to be solved concurrently. This ensures that meaningful interactions among agents can be tested, enabling us to assess the generalizability of our approach. Second, the environment must support agent variability, including dynamic spawning and despawning of agents during an ongoing episode. This capability is essential for testing the scalability and adaptability of the proposed approach in open systems.

We use two distinct environments to test our framework and depict them in Fig. 3. The Cooking environment, inspired by the computer game Overcooked, provides a rich and dynamic setting where agents must coordinate to achieve shared objectives. Agents interact with both movable and static elements, such as ingredients and kitchen counters, while managing strict time constraints and penalties. To further align this environment with our

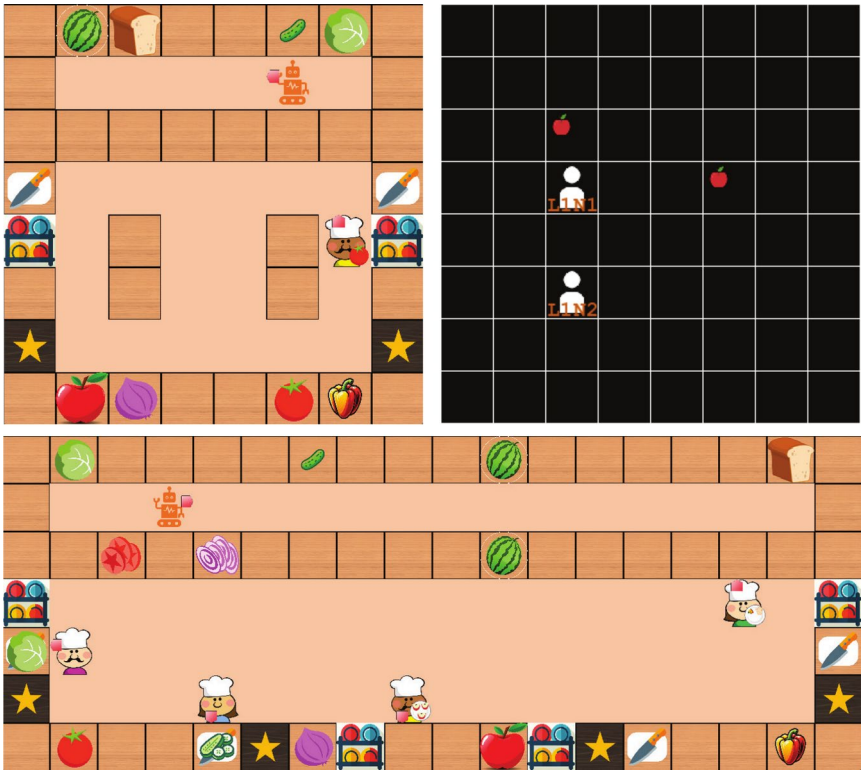


Fig. 3 On the top left is the tight kitchen environment layout, on the top right is the level-based foraging environment, and in the bottom is the wide kitchen layout. The wide kitchen layout has a long traversal time to help different agents, in contrast to a tight layout. The level-based foraging environment is a coordination-focused environment as agents need to gather fruit in the same time step

research needs, we extend it to support agent spawning and despawning during episodes. This modification allows us to evaluate the performance of our framework in scenarios with a fluctuating number of agents, simulating open multi-agent systems. Additionally, the environment introduces varying task complexities, such as different target recipes for each agent, making it a challenging and flexible testbed.

The Level-Based Foraging (LBF) environment is an established benchmark for collaborative multi-agent systems. It provides a controlled setting where agents with varying skill levels must cooperate to forage for resources while navigating shared spaces. The level-based mechanics of the environment, coupled with the dynamic interactions required between agents and objects, make it particularly suitable for testing the scalability and temporal consistency of modular policies. By comparing performance in this structured domain, we benchmark our approach against established baselines for cooperative multi-agent systems.

Both environments are implemented to support the stochastic game interface provided by the PettingZoo library [36], which facilitates the creation of multi-agent systems with dynamic agent creation and destruction. To ensure reproducibility and encourage standardization in the field, we publish these environments along with our codebase. This contribution enables the broader research community to validate and build upon our findings.

5.1.1 Cooking

We use and extend an existing version of the cookingzoo environment [27]. The environment provides rewards for the successful completion of recipes while incorporating a configurable time penalty for each step taken by the agent. A kitchen instance contains both movable components and static stations, such as counters or knives, and is designed to support dynamic task allocation and interaction among agents.

Each kitchen layout has a clear separation into distinct zones, influencing agent movement and collaboration dynamics. Agents can move simultaneously in all four cardinal directions or remain stationary. Each agent (except the ego agent) is assigned one of four possible recipes, ensuring no double assignments. Recipes consist of retrieving and processing (e.g., cutting) two distinct, non-overlapping objects and combining them on a plate for delivery. The second object can only be accessed by another agent on the other side of the separation, creating an asymmetric dependency that emphasizes collaborative behavior. After completing a task, agents are assigned to complete their recipe again, ensuring continuous interaction.

The maximum number of tasks and agents is constrained to four tasks for four agents and no task repetitions, plus the ego agent, resulting in a maximum of five agents per episode. For each new episode, recipes, ingredient locations, and agent starting positions are resampled, introducing variability in task allocation and spatial dynamics.

Episodes terminate either when all recipes have been successfully delivered or after 200 time steps. The state of the environment is represented using a tensor that encodes the positions and states of all objects and agents in a layered matrix. The ego agent is represented in a separate layer to facilitate specialized observation and decision-making, as depicted in Fig. 4.

In our experiments, we evaluate agent behavior using two distinct kitchen layouts, a tight layout and a wide layout. The tight layout features compact zones with ingredients

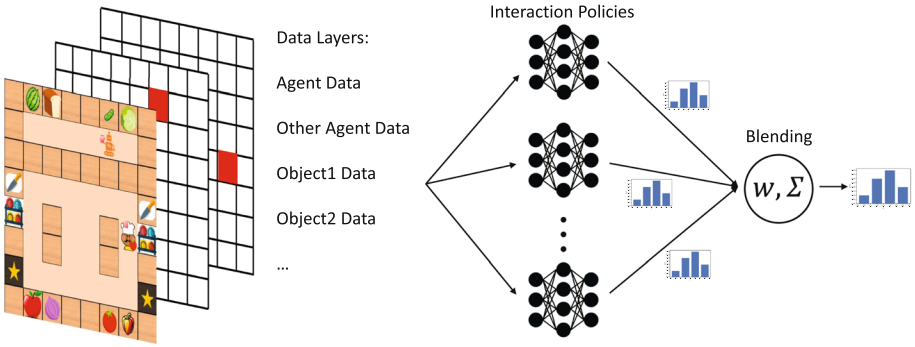


Fig. 4 The structure of the observation layout used in the experiments. Data is represented as a 3D tensor, and each entry of the tensor is a 2D matrix, with each matrix encoding the position and state of one specific agent or object in the world. The first data layer is the ego layer, encoding one's position. The second layer encodes the position of all other agents. The rest of the data layers encode every object individually. We use this scheme across both the kitchen and the extended LBF environment. The interaction policies then consume the tensor data, and their output is blended to achieve the final probability distribution

and tools placed in close proximity, minimizing the costs of agent interactions. This configuration enables efficient task-switching and frequent collaboration, as the overhead of assisting another agent is minimal. In contrast, the wide layout introduces greater spatial separation between zones, significantly increasing the cost of providing assistance. Agents in this layout face challenging trade-offs, as helping another agent often requires foregoing opportunities to complete local objectives, leading to increased interaction costs and more pronounced collaborative dynamics. These configurations allow us to examine agent performance with different levels of collaboration costs systematically.

5.1.2 LBF

We use and extend the Level-Based Foraging (LBF) environment [19], a cooperative multi-agent environment designed to study coordination challenges. In this environment, agents and objects (fruits) are placed on an 8×8 grid. Fruits and agents are assigned levels of 1, 2, or 3, and agents can move in the four cardinal directions, remain stationary, or attempt to collect adjacent objects. Successfully collecting an object requires the combined levels of the agents attempting the collection to meet or exceed the object's level. An episode ends either when all objects are collected or after 200 time steps.

In the original formulation, agents received rewards for successfully collecting any object, and only a single fruit type was present. We introduce several modifications to adapt the environment for our study. First, we add a second type of fruit, increasing the number of distinct objects and enabling multi-task scenarios. Second, we assign each agent a specific fruit type to collect, rewarding them only for gathering their assigned fruits and introducing penalties for collecting unassigned objects. These changes preserve the core mechanics of level-based coordination while allowing us to study task-specific behavior and multi-task scalability.

The modified environment supports a configurable number of agents and objects. In our experiments, we use up to five agents, with agent-to-fruit assignments randomized at the start of each episode. This setup generalizes to more than two fruit types, although our

evaluations are restricted to two for simplicity and focus. The inclusion of a step penalty further encourages agents to achieve faster task completion.

The observation space of the environment is represented as a tensor with four layers. Each layer encodes grid-based information about the agents, their levels, and the two fruit types, including their levels, as shown in Fig. 4. This representation extends the original environment by explicitly accommodating multi-task scenarios and supporting clear observation of assigned fruits.

In our experiments, we evaluate agent behavior using the modified environment to study the effects of task-specific assignments, multi-task scalability, and coordination efficiency under varying configurations. The modifications preserve the cooperative nature of the environment while introducing meaningful extensions for studying open-ended, multi-agent, multi-task dynamics.

5.2 Setup

We evaluate four strategies to analyze the performance of our modular policy framework. The Joint Learner serves as our baseline, employing Proximal Policy Optimization (PPO) [37] to train a single policy on the accumulated joint reward of all agents for all tasks. This approach is computationally straightforward but lacks flexibility and scalability in diverse or dynamic environments, making it a critical point of comparison for evaluating modular strategies. A simple modular baseline approach is average blending, which computes the mean of all individual policies to produce a unified decision. Inspired by the work of Haarnoja et al. [38], average blending is computationally simple, but it is fundamentally limited in its ability to account for task-specific goals or agent context. Our empirical results indicate that this approach is suboptimal for the tasks at hand and adds little benefit to the broader discussion of modular policies.

A comparison with a joint learner can highlight the benefits of our overall architecture compared to a general learning framework. To evaluate the benefits of the proposed merging scheme, we also compare results to an ablation that uses goal-weighted blending, which integrates task-specific goals into the weighting of individual policies using the same goal estimation, but leaving out the posterior term. Other arbitrator-based frameworks can not be adapted in a straightforward manner because their training paradigm presupposes access to the full distribution of agent counts and task combinations that later appear during testing, and for those to be static to produce relevant results. Our approach is designed for continual deployment in genuinely open environments where new tasks may emerge to train, and during deployment, agents may enter or leave at any time step. Instead, we report results for the standard joint-learner, which is the state of the art in closed-world learning. Goal-weighted blending and a sequential joint learner, which share the same information constraints as our algorithm, therefore provide a fairer assessment of relative performance and are well-performing algorithms in multi-agent settings [39]. For all approaches, we use a CNN layer as a neural network that takes the tensor representation of the environment and uses 16 kernels. After flattening the CNN output, we use two layers with sizes 2048 and 512 before passing them through the policy or value head, respectively. As a framework, we use PyTorch and TorchRL [40].

5.3 Training task, interaction, and joint policies

To ensure robustness and efficient learning, we pretrain task-specific policies in all environments. In the cooking environments we employ during task training, simple collaboration bots are employed that are programmed to always help the learning agent during its task, so the agent can solve it. In LBF, agents learn first to collect the fruits alone and then with a clone of themself to coordinate and pick up fruits together that need multiple agents. We handle different layouts in the kitchen environment separately and train different policies for each, as their input space is has different sizes. For each task, we pretrain five policies. The best-performing policy is selected for further experiments based on the highest average reward achieved across all agents. We show training graphs in Fig. 5 where we can see that the task as well as the interaction policies quickly pick up on the provided reward signal and learn to complete their respective tasks successfully.

Interaction policies are trained with one task policy agent at a time and receive the reward signal of the task agent. This allows us to conduct learning in an open-ended fashion, where additional policies can be incorporated and learned at any time.

Our joint agent PPO baseline is trained with the pre-trained agents using task policies. The joint agent learns to interact with exactly one agent at a time and for each task once.[1] We trained all policies five times, and in all our experiments, the same set of hyperparameters was employed for training the task, interaction, and joint policies as shown in Table 1. Training of each task, interaction, and goal inference module was done for 200000 steps. This results in a sample complexity of 2 times the number of tasks. We emphasize that the sample complexity of our method is linearly scaling with the sample complexity of training any policy in isolation. The joint reward PPO learner is training with the same continual task setting and has the same sample complexity.

6 Results and discussion

We begin by evaluating our framework in a more classical multi-agent reinforcement learning (MARL) setup, referred to here as closed-world, where a constant number of agents interact, and the episode terminates after tasks are completed, once there are only enough resources to complete each task once within a single episode. This setup provides a controlled environment for benchmarking and establishes a baseline comparison for the proposed modular policy framework. Following this, we examine its performance in open-world scenarios, which present additional challenges: (a) agents dynamically spawn and despawn during episodes, (b) the number of interaction partners is fixed and its effects are examined, and (c) the set of assigned tasks will be revealed to the learner progressively. Our measures are the average reward of all other agents as implied in the objective function of our objective in (1) and the fraction of help that agents receive, expressed as the percentage of their reward, ordered from lowest to highest. These experiments are designed to assess the framework's robustness to dynamic changes and its ability to maintain effective coordination and task allocation in environments with incomplete information and fluctuating

[1] We also evaluated a fully informed setup in which the joint agent was exposed to all agent–task combinations in training, but it did not add anything, as its performance did not differ from sequential joint PPO learning due to getting stuck in bad local optima and exhibiting the same issues.

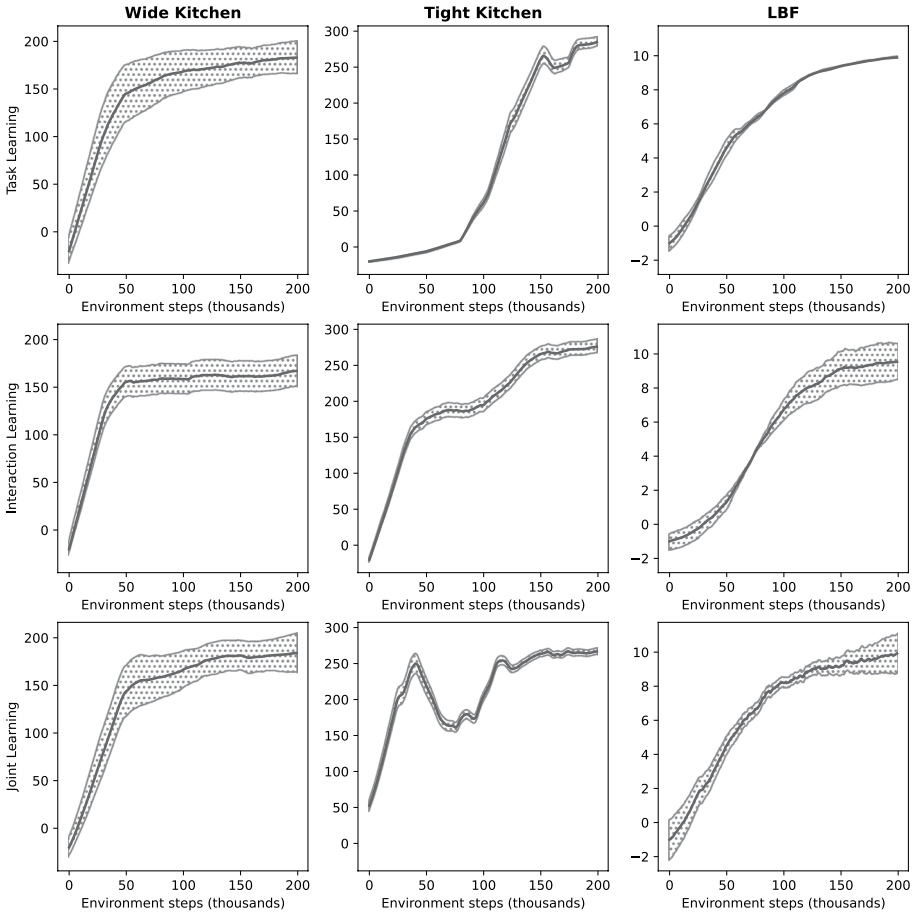


Fig. 5 Visualization of the training progress of the task policies and interaction policies across three environmental layouts. Each plot displays the mean performance curve as a function of training steps, with the standard deviation represented as a dotted area around the curve. The results are aggregated over all available tasks in each environment, each being trained 5 times. 200 training epochs were conducted in each run

agent populations. This section provides a structured analysis of these setups, connecting the experimental design to our research questions. We evaluate each scenario 100 times and report the average reward across all scenarios (Table 2).

6.1 Evaluation closed world

To assess our system under simpler conditions, we used a closed-world variant of the environment in which each agent completes exactly one assigned task. This shows baseline performance without the complexities introduced by dynamic arrivals or multiple task assignments. In our experiments, we tested 4 possible tasks in the cooking and 2 in the LBF environment. One agent per task was spawned at the beginning of the episode, performed its assigned task, and then the trial ended without introducing additional agents or tasks.

Table 1 Common hyperparameters for task, interaction, and joint policy training

Hyperparameter	Value
Number of Iterations (n_{iters})	200
Max Steps per Episode	200
Sub-batch Size	256
Frames per Batch	2,560
Total Frames	512,000
Number of Epochs per Batch	10
Learning Rate	1×10^{-4}
Max Gradient Norm	1.0
Clipping Epsilon	0.3
Discount Factor (γ)	0.99
GAE Lambda (λ)	0.90
Entropy Coefficient	0.01
Critic Coefficient	0.25
Critic Clipping Parameter	10.0
Optimizer	Adam (weight decay = 0.0001)

All training approaches were executed with these settings

Table 2 Comparison of Posterior Blending and Joint Learning in closed-world setups across 100 evaluation runs

Environment	Posterior Blending	Joint Learning
Wide Kitchen Layout	71.4 (8.1)	76.7 (1.9)
Tight Kitchen Layout	74.3 (5.4)	77.9 (1.5)
LBF	8.7 (1.9)	10.1 (0.6)

Reported are the average of the summed reward of other agents and its standard deviation

The joint learner serves as our baseline, achieving marginally higher cumulative rewards by optimizing over a fixed set of agents and tasks in a centralized manner. In contrast, Posterior Blending is designed to flexibly adapt to evolving conditions by dynamically integrating modular policies. Our experimental results show that Posterior Blending performs very similarly to the joint learner overall. The small performance gap observed is primarily due to a delay of a few steps when Posterior Blending slowly switches between task-specific policies. This delay occurs during the transition phases when a new one replaces a completed task. For the joint learner, the policy implicitly learns to handle these transitions.

Posterior Blending remains competitive in this closed-world scenario, demonstrating robust agent-task coordination. Though initially intended for dynamic settings, it handles static tasks without retraining.

6.2 Evaluation agent openness

We now consider an open environment in which agents are repeatedly assigned to their tasks and can leave and join the setting dynamically. Each time an agent completes its recipe, it may leave with probability equal to a despawn rate, and a new agent with its own task may arrive at a rate proportional to that despawn rate, where $\text{rate}_{\text{spawn}} = 0.05\,\text{rate}_{\text{despawn}}$. In contrast to the closed-world scenario, the episode does not end after a single successful task completion. Instead, tasks can be completed multiple times. For lower despawn rates, the environment remains stable because most agents stay throughout an episode. As the

rate increases toward 1, the episode becomes more variable. After nearly every completed recipe, agents will leave, and newly introduced agents appear at a matching pace, creating dynamic shifts in agent and task composition.

Figure 6 presents the average total cumulative reward for each kitchen layout across varying despawn rates. Additionally, we plot the percentage contributions of the tasks in order of most helped to least helped with different color tones. In the wide cooking layout, Posterior Blending outperforms the Joint Learner at higher despawn rates, where the presence of newly arriving agents creates frequent coordination challenges. Posterior Blending's policy weighting mechanism adapts effectively to these changes, whereas the Joint Learner, optimized under more stable conditions, exhibits a noticeable drop in performance. Goal-weighted blending improves as the despawn rate increases because the fact that fewer agents are present benefits this blending mechanism, as there is less confusion about whom to help. This effect shows that for stabilization with multiple agents, our proposed blending mechanism improves coordination ability as the policy is nudged toward consistency. This notion is supported by the measured performance when changing the number of agents, which will be discussed in the following section. The Posterior Blending does not have the same benefits from a lower average agent count, as the blending with posterior weighting mitigated the issues that caused the ablation to perform better when relieved. However, the joint learner baseline and the Goal-weighted blending approach remain below Posterior Blending.

Across both layouts, Posterior Blending also tends to balance rewards more evenly among the various agents in the system, suggesting that it shifts its assistance from one agent's task to the next as needed. In the wide layout, where helping another agent is costly, Posterior Blending scales back this behavior when the number of agents is stable but still switches effectively once individual tasks are complete. In the tight layout, where coordination costs are lower, Posterior Blending provides more frequent, varied support. These results underscore the value of Posterior Blending's dynamic weighting of specialized modules, allowing it to handle open-ended agent arrivals and departures while maintaining

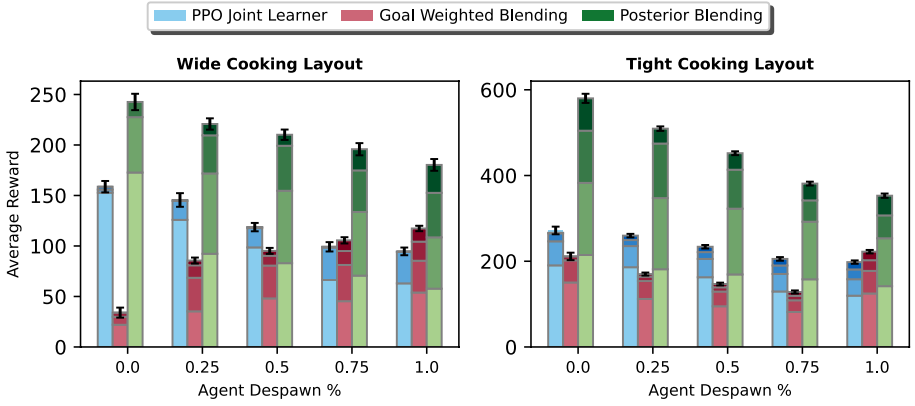


Fig. 6 Comparison of the average reward across different layouts and agent despawn percentages. The figure shows the performance of four different strategies, Joint Learner (blue), Goal Weighted Blending (red), and Posterior Blending (green), across three layouts. Error bars indicate the standard error of the reward. The different shades represent the percentage of the agent's reward

robust task completion. Our ablation of the Posterior Blending term highlights the necessity to stabilize the policy weighting process and shows that our newly introduced term is an important key mechanism to gain the desired flexibility in our approach.

6.3 Interaction partner count scaling

For this experiment, we test how the system performs in different setups with different static numbers of agents, which continuously perform their assigned tasks. All systems have learned all tasks that can be assigned to agents within the tested scenario, meaning the PPO agent has been trained to interact exclusively within the subset of tasks belonging to the agents present. The results are presented in Fig. 7, which shows the average reward for each strategy as a function of the number of agents in each environment. In the Tight Cooking Layout, performance improves with an increasing number of agents, with Posterior Blending consistently achieving the highest average reward across all agent counts. The Joint learner performs comparably for smaller numbers of agents but exhibits a slower rate of improvement as more agents are introduced. Goal Weighted Blending maintains moderate performance and is able to perform with a single agent, as the goal weighting stabilizes the blending process in that case. In the Wide Cooking Layout, similar trends are observed. The Posterior Blending continues to outperform the other strategies, achieving higher rewards as the number of agents increases. Joint Learning and Goal Weighted Blending perform moderately well, although their gains plateau with three or more agents. The Goal Weighted Blending strategy eventually lags behind in performance when more agents are involved. Unlike Posterior Blending, it lacks the posterior-based smoothing term that gradually locks in assistance toward one agent, even when there are multiple agents to assist. Since this stabilising signal is missing, the goal weights oscillate from step to step, so no single interaction policy can consistently determine the final action taken. The resulting multimodal mixed policy causes the acting agent to switch prematurely between helping different teammates, which breaks temporal consistency and reduces the accumulated reward.

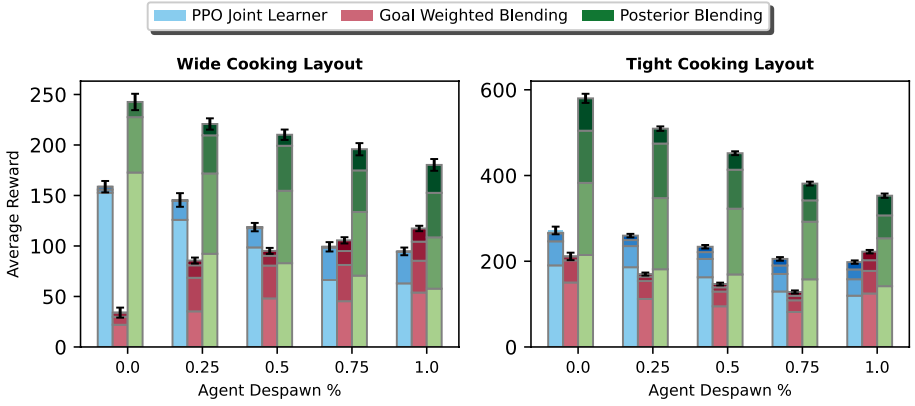


Fig. 7 Average Reward Across Layouts with Varying Agent Numbers. The figure shows the average reward for the Joint Learner (blue), Goal weighted blending (red), and Posterior Blending (green) strategies in three environments. The x-axis represents the number of interaction agents present, and the y-axis indicates the average reward. Error bars show the standard error

6.4 Evaluation task scaling

To assess how the performance changes as agents progressively learn new tasks, we measure the performance with four agents and four tasks present in the cooking environment layouts. Each agent is uniquely performing one of the possible tasks. However, the agent will initially only be trained with one random task from this set (and one partner). As the number of learned tasks increases, we expect overall rewards to rise because the agent can assist more tasks. In the tight kitchen layout, where providing help to others imposes only minor trade-offs, we would further anticipate stable or simultaneous support for multiple agents if assisting an additional task does not hinder ongoing efforts.

The baseline joint-agent is trained in the same manner as before, but only on those tasks present during evaluation. The blending baseline and our proposed blending solution each learn a new interaction policy for every additional task, with their total number of learned tasks always matching that of the joint-agent. We incrementally add tasks one by one and assess the respective performance.

The results, shown in Fig. 8, demonstrate that Posterior Blending achieves the highest average reward in all three environments as the number of learned tasks grows. In the results across both layouts, when only a single task is trained, all methods perform similarly and reach comparable reward levels, as the single-task case constrains each approach nearly identically. As additional tasks are introduced, however, the Posterior Blending approach experiences a substantial improvement in reward, especially on the third and fourth tasks, whereas Goal Weighted Blending exhibits notable drops in performance, especially in the wide cooking layout. This drop in performance shows that, especially for situations where switching interaction partners does incur more cost in the beginning (more time steps have to be spent moving towards another agent), necessitates our blending mechanism to stabilize policy selection. Although the Joint Learner retains a steady performance, it does not scale as effectively as Posterior Blending when the number of tasks increases.

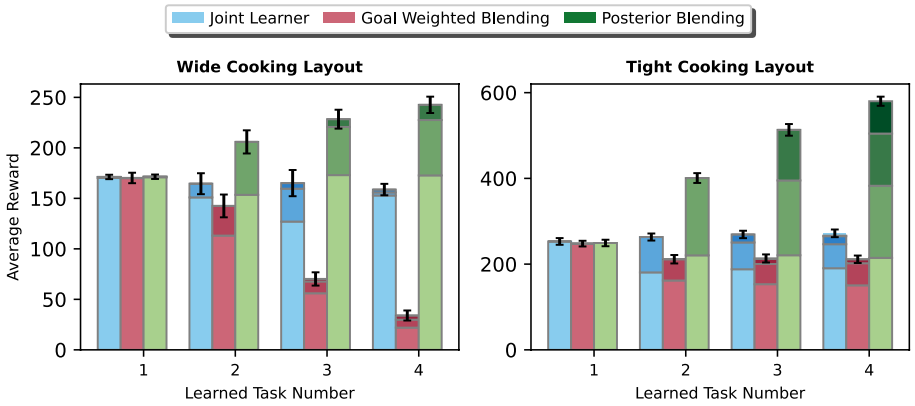


Fig. 8 Average Reward Across Layouts after learning different tasks. The figure shows the average reward for the Joint Learner (blue), Goal weighted blending (red), and Posterior Blending (green) strategies in two kitchen layouts. The x-axis represents the number of learned tasks, and the y-axis indicates the average reward. Error bars show the standard error. The different shades represent the percentage of the agent's reward

The distribution of agent rewards gets more even with learning additional tasks when using our Posterior Blending approach. Looking at which task the joint learner supports in detail shows that it is always the last one learned, a result in line with catastrophic forgetting that is encountered in multi-task or lifelong learning. In contrast, the learning order does not impact our modular blending approach by design. The goal weighted blending ablation, on the other hand, fails to incorporate the additional knowledge gained through other policies.

The results highlight that Posterior Blending consistently outperforms the other strategies across all environments and agent configurations. The average reward increases as more agents are added, with Posterior Blending achieving the most effective scaling. The Joint Agent performs well in environments with fewer agents but fails to maintain this advantage as the agent population grows. Goal weighted blending exhibits limited adaptability, particularly in the Tight Cooking Layout and LBF, where spatial constraints and task complexity appear to exacerbate their performance limitations.

Overall, these experiments demonstrate the scalability and robustness of Posterior Blending across varying numbers of agents and tasks. This blending strategy achieves higher rewards and adapts more effectively to increasing agent counts compared to other methods. The consistent trends across all three environments confirm the advantages of modular, task-aware policy strategies in dynamic multi-agent systems.

7 Conclusion and future work

Our work introduces and addresses a new problem formulation, the Double Open Bayesian Stochastic Game (DOSBG). DOSBG extends the traditional Markov Decision Process framework to capture the challenges of dynamic agent participation and open-task environments, where agents may join or leave episodes and pursue different objectives over time. By explicitly incorporating tasks as a structuring mechanism for agent behavior, DOSBG provides a flexible model for multi-agent coordination in evolving settings. The experimental findings show that modular open policies, when integrated with a flexible blending mechanism, are capable of consistently high performance in both closed-world and open-world multi-agent cooperation scenarios. The ability to reuse independently trained task- and interaction-specific policies appears to be an important factor when dealing with the demands of dynamic multi-agent systems, where agents frequently spawn and despawn. The results in the Cooking and LBF environments confirm that our approach supports a degree of adaptability that a centralized baseline struggles to match, especially when increasing the number of tasks or agents. A key observation is that blending methods differing only in how they aggregate sub-policies diverge considerably in performance. Posterior-based blending, which dynamically updates weights based on contextual factors such as entropy or task alignment, achieves high and stable rewards and maintains more equitable distributions among agents. We showed through the ablation with the Goal-weighted blending approach that the addition of the stabilization is necessary to maintain high performance. Although the joint learner baseline retains an advantage in strictly static closed-world domains, it cannot match the open-ended capacity of our proposed framework when agents or tasks are introduced incrementally. These findings validate our contributions: (1) a framework for open-ended, open-system coordination that enables lifelong adaptation without retraining, (2) a blended policy mechanism that robustly combines multiple interaction policies,

and (3) experimental evidence demonstrating superior performance in dynamic, multi-agent environments. The outcomes of the experiments underscore the importance of modular architectures combined with adaptive weighting for coping with evolving multi-agent system demands and show that this is a promising research direction to unify multiple separately trained models.

Author Contributions David Rother wrote the manuscript text and prepared the figures. All authors reviewed the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. Joni Pajarinen acknowledges funding by the Research Council of Finland (345521, 357301). This research is supported by the Honda Research Institute Europe GmbH.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Competing interests The authors declare no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Pieter Abbeel, O., & Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in Neural Information Processing Systems, 30*.
2. Foerster, J. N., Assael, Y. M., de Freitas, N., & Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. *Advances in Neural Information Processing Systems, 29*, 2137–2145.
3. Gupta, J. K., Egorov, M., & Kochenderfer, M. (2017). Cooperative multi-agent control using deep reinforcement learning. In G. Sukthankar & J. A. Rodriguez-Aguilar (Eds.), *Autonomous agents and multiagent systems* (pp. 66–83). Cham: Springer International Publishing.
4. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., & Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences, 114*(13), 3521–3526.
5. Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., & Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural Networks, 113*, 54–71.
6. Uchibe, E., Asada, M., & Hosoda, K. (1996). Behavior coordination for a mobile robot using modular reinforcement learning. In *Proceedings of IEEE/RSJ international conference on intelligent robots and systems*. IROS'96 (Vol. 3, pp. 1329–1336). IEEE.
7. Samejima, K., Doya, K., & Kawato, M. (2003). Inter-module credit assignment in modular reinforcement learning. *Neural Networks, 16*(7), 985–994.
8. Wang, Y., He, H., & Sun, C. (2018). Learning to navigate through complex dynamic environment with modular deep reinforcement learning. *IEEE Transactions on Games, 10*(4), 400–412.
9. Andreas, J., Klein, D., & Levine, S. (2017). Modular multitask reinforcement learning with policy sketches. In *International conference on machine learning* (pp. 166–175). PMLR.

10. Simpkins, C., & Isbell, C. (2019). Composable modular reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 33, pp. 4975–4982).
11. Ming, F., Gao, F., Liu, K., & Zhao, C. (2023). Cooperative modular reinforcement learning for large discrete action space problem. *Neural Networks, 161*, 281–296.
12. Sarantopoulos, I., Kiatos, M., Doulgeri, Z., & Malassiotis, S. (2021). Total singulation with modular reinforcement learning. *IEEE Robotics and Automation Letters, 6*(2), 4117–4124.
13. Bakker, J., Hammond, A., Bloembergen, D., & Baarslag, T. (2019). RLBOA: A modular reinforcement learning framework for autonomous negotiating agents. In *AAMAS* (pp. 260–268).
14. Gupta, V., Anand, D., Paruchuri, P., & Kumar, A. (2021). Action selection for composable modular deep reinforcement learning. In *Proceedings of the 20th international conference on autonomous agents and multiagent systems*. AAMAS '21 (pp. 565–573). Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.
15. Bhat, S., Isbell, C. L., & Mateas, M. (2006). On the difficulty of modular reinforcement learning for real-world partial programming. In *Proceedings of the national conference on artificial intelligence* (Vol. 21, p. 318). Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.
16. Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning* (pp. 330–337). Morgan Kaufmann Publishers Inc.
17. Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., & Graepel, T. (2018). Value-decomposition networks for cooperative multi-agent learning based on team reward. AAMAS '18 (pp. 2085–2087). Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.
18. Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., & Whiteson, S. (2018). Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International conference on machine learning* (pp. 4295–4304). PMLR.
19. Albrecht, S. V., & Ramamoorthy, S. (2013). A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems. In *Proceedings of the 2013 international conference on autonomous agents and multi-agent systems*. AAMAS '13 (pp. 1155–1156). Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.
20. He, H., Boyd-Graber, J., Kwok, K., & Daumé III, H. (2016). Opponent modeling in deep reinforcement learning. In *International conference on machine learning* (pp. 1804–1813). PMLR.
21. Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., & Whiteson, S. (2018). Counterfactual multi-agent policy gradients. *Proceedings of the AAAI Conference on Artificial Intelligence, 32*(1). https://doi.org/10.1609/aaai.v32i1.11794
22. Stone, P., Kaminka, G., Kraus, S., & Rosenschein, J. (2010). Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 24, pp. 1504–1509).
23. Albrecht, S. V., & Stone, P. (2017). Reasoning about hypothetical agent behaviours and their parameters. In *Proceedings of the 16th conference on autonomous agents and multiagent systems*. AAMAS '17 (pp. 547–555). Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.
24. Barrett, S., Rosenfeld, A., Kraus, S., & Stone, P. (2017). Making friends on the fly: Cooperating with new teammates. *Artificial Intelligence, 242*, 132–171.
25. Rahman, M. A., Hopner, N., Christianos, F., & Albrecht, S. V. (2021). Towards open ad hoc teamwork using graph-based policy learning. In *International conference on machine learning* (pp. 8776–8786). PMLR.
26. Tacchetti, A., Song, H. F., Mediano, P. A., Zambaldi, V., Rabinowitz, N. C., Graepel, T., et al. (2019). Relational forward models for multi-agent learning. In *International conference on learning representations*.
27. Rother, D., Weisswange, T. H., & Peters, J. (2023). Disentangling interaction using maximum entropy reinforcement learning in multi-agent systems. In *ECAI* (pp. 1994–2001).
28. Rother, D., Herbert, F., Kalter, F., Koert, D., Pajarinen, J., Peters, J., & Weisswange, T. H. (2025). Entropy based blending of policies for multi-agent coexistence. *Autonomous Agents and Multi-Agent Systems, 39*(1), 1–28.
29. Haarnoja, T., Pong, V., Zhou, A., Dalal, M., Abbeel, P., & Levine, S. (2018). Composable deep reinforcement learning for robotic manipulation. In *2018 IEEE international conference on robotics and automation (ICRA)* (pp. 6244–6251). IEEE.
30. Haarnoja, T., Tang, H., Abbeel, P., & Levine, S. (2017). Reinforcement learning with deep energy-based policies. In *International conference on machine learning* (pp. 1352–1361). PMLR.
31. Schulman, J., Chen, X., & Abbeel, P. (2017). Equivalence between policy gradients and soft q-learning. arXiv:1704.06440

32. Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning* (pp. 41–48).
33. Yang, J., Nakhaei, A., Isele, D., Fujimura, K., & Zha, H. (2020) Cm3: Cooperative multi-goal multi-stage multi-agent reinforcement learning. *International conference on learning representations*.
34. Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., et al. (2020). Implementation matters in deep policy gradients: A case study on PPO and TRPO. *International Conference on Learning Representations*.
35. Huang, S., Dossa, R. F. J., Raffin, A., Kanervisto, A., & Wang, W. (2022). The 37 implementation details of proximal policy optimization. *The ICLR Blog Track 2023*.
36. Terry, J., Black, B., Grammel, N., Jayakumar, M., Hari, A., Sullivan, R., Santos, L., Perez, R., Horsch, C., Dieffendahl, C., Williams, N. L., Lokesh, Y., & Ravi, P. (2021). Pettingzoo: Gym for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems, 34*, 15032–15043.
37. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv:1707.06347
38. Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning* (pp. 1861–1870). Pmlr.
39. Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., & Wu, Y. (2022). The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems, 35*, 24611–24624.
40. Bou, A., Bettini, M., Dittert, S., Kumar, V., Sodhani, S., Yang, X., De Fabritiis, G., & Moens, V. (2023). TorchRL: A data-driven decision-making library for PyTorch. arXiv:2306.00577

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

David Rother[1,3] · Joni Pajarinen[2] · Jan Peters[1] · Thomas H. Weisswange[3]

✉ David Rother
work.rother@gmail.com

Joni Pajarinen
joni.pajarinen@aalto.fi

Jan Peters
jan.peters@tu-darmstadt.de

Thomas H. Weisswange
thomas.weisswange@honda-ri.de

[1] Intelligent Autonomous Systems (IAS), TU Darmstadt, Darmstadt, Germany

[2] Aalto University, Espoo, Finland

[3] Honda Research Institute Europe GmbH, Offenbach, Germany