*ECAI 2023*
*K. Gal et al. (Eds.)*

*doi:10.3233/FAIA230491*

# Disentangling Interaction Using Maximum Entropy Reinforcement Learning in Multi-Agent Systems

**David Rother** [a;*], **Thomas H. Weisswange**[b] **and Jan Peters**[a]

[a]TU Darmstadt
[b]Honda Research Institute Europe GmbH

**Abstract.** Research on multi-agent interaction involving both multiple artificial agents and humans is still in its infancy. Most recent approaches have focused on environments with collaboration-focused human behavior, or providing only a small, defined set of situations. When deploying robots in human-inhabited environments in the future, it will be unlikely that all interactions fit a predefined model of collaboration, where collaborative behavior is still expected from the robot. Existing approaches are unlikely to effectively create such behaviors in such "coexistence" environments. To tackle this issue, we introduce a novel framework that decomposes interaction and task-solving into separate learning problems and blends the resulting policies at inference time. Policies are learned with maximum entropy reinforcement learning, allowing us to create interaction-impact-aware agents and scale the cost of training agents linearly with the number of agents and available tasks. We propose a weighting function covering the alignment of interaction distributions with the original task. We demonstrate that our framework addresses the scaling problem while solving a given task and considering collaboration opportunities in a co-existence particle environment and a new cooking environment. Our work introduces a new learning paradigm that opens the path to more complex multi-robot, multi-human interactions.

## 1 Introduction

Human-Robot Interaction (HRI) is a trending research topic [12, 31] as there is an increased push toward robots that work alongside humans. Applications of HRI work in the scientific literature vary and include topics such as teaching [2, 10], assistance [8, 19, 40] or entertainment [1]. Often modelling of the interaction aims for an explicit collaboration with a human or a group of humans to achieve a common goal [7, 30]. However, when sharing an environment with multiple humans, a robot may not interact with only a single, dedicated user. Other humans or robots may be bystanders that coexist in the same environment without being directly involved in the current task of the robot [32]. How to (co-)operate when facing other people while performing a task is an important question that needs to be addressed in HRI.

Consider the situation of a robot for assisted living within a group of older adults. The robot should be able to perform supportive tasks for a given person, while other humans with independent intentions are sharing the same space. For example, the robot could be asked by one person to prepare a sandwich while at the same time another person starts cooking something for herself. The robot and the cook both require access to some shared resources, like the fridge, while they do not share a common goal. However, a robot that does not acknowledge the right of a bystander to approach its own goals will likely not be accepted by society but neither will one that ignores its given task while other humans are close-by.

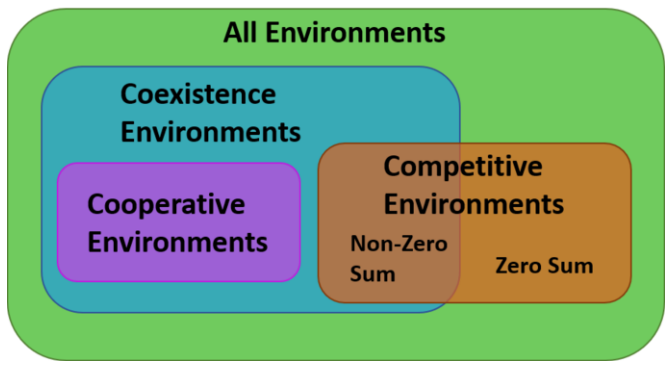


**Figure 1**: Classification of multi-agent environment structures. Coexistence environments constrain performance of multiple agents to be interdependent but not exclusive. Cooperative environments and some competitive environments are considered sub-sets.

We term such situations "coexistence" environments. Figure 1 depicts the relation to other types of environments used in the literature. In coexistence environments agents have an impact on other agent's performance, most likely through sharing space or resources, while both agents are able to theoretically reach their goals. If impact involves an explicit interdependence to reach a certain goal, this can be called a cooperative scenario. Competitive environments feature an explicit resource conflict. If this conflict results in sub-optimal solutions for at least one agent, it can still be considered a coexistence environment. However, zero-sum games, which result in only one agent being able to reach its goal at all will not allow the coexistence of the agents. Coexistence environments can also be related to the Ad-Hoc Teamwork (AHT) field where agents have to collaborate with unknown teammates [21]. Both AHT and coexistence environments share common assumptions, including no prior coordination between agents, no control over teammates, and no zero-sum competition. However, an important difference between the two is that AHT environments have a defined "optimal" solution, which is the joint reward. On the other hand, in coexistence environments, defining an optimal solution is not as straightforward. From an outside perspec-

* Corresponding Author. Email: david.rother@tu-darmstadt.de

*Please check ArXiv or contact the authors for any appendices or supplementary material mentioned in the paper.*

tive, fairness may be considered the optimal solution, whereas, from an egoistic perspective, social investment may be the way to go. Despite these differences, it's worth noting that AHT can still be considered a subset of cooperative environments, while coexistence defines a more general setting.

In this work, we propose a novel learning framework to solve coexistence environments. Our approach entails obtaining two distinct sets of policies: one set for completing tasks and another set for influencing other agents while they perform their respective tasks. Additionally, we derive an entropy based mechanism for blending these policies. In addition to showcasing the scalability of this approach for numerous agents with diverse intentions, we analyze the advantages of employing this approach towards enhancing task performance, while simultaneously minimizing any negative impact on the agent's own performance.

## 2 Related Work

Coexistence environments provide a number of challenges, such as modeling other agent's policies and intentions, evaluating the impact of own actions, integrating others in the learning of a policy or interacting with unknown agents, working with many different tasks, and scaling with the combination of single agent tasks. Many of the individual aspects have been addressed by prior work.

An example of modeling other agents in the environment explicitly is the I-POMDP [11] framework. Decision frameworks using this model engage in explicit reasoning about humans [9, 16] and use the mental model of others to improve their own outcomes. Our approach does not require a model of the environment during execution and uses the model of others to find the best solution for the population while solving the agent's own task.

Integrating others in the learning of a policy is a main part of Multi-Agent Reinforcement Learning (MARL) where all agents are explicitly trained together [25, 38]. One possibility in MARL is to train a set of agents by extracting joint action values as a complex non-linear combination of single agent values to act on decentralized local observations [27, 33, 39]. The resulting policy will however be restricted to a specific set of tasks for the agents. In [37], agents are trained in a centralized learning, decentralized execution regime and compute a credit score by taking other's perspective. This work deals with cooperative scenarios, where the function estimation process needs to train on each goal combination explicitly.

Maximum entropy methods have been shown to produce policies that are robust to minor changes in the environment or other agents' behaviors [35, 14, 34, 13]. Combining energy based policies as a product of experts has also be shown as an effective measure to solve problems that include multiple sub-tasks [15, 18]. Previous work has also considered Multi-Agent systems that have to solve multiple tasks [24]. They introduce a monolithic learning regime to learn over multiple tasks using a single policy without providing task identities. This approach scales well for cooperative tasks only.

Ad-Hoc teamwork is a single agent learning problem where the agent has to be capable to cooperate on the fly with other agents without prior coordination. [21] A popular approach is to compute Bayesian posteriors over predefined teammate types, which then are subsequently used in other models, such as reinforcement learners [3, 4]. Another approach uses transfer learning with the goal to reuse knowledge across agents to enable faster adaptation to new agent types [5, 28, 6]. [20] learn a teammate's task from a set of predefined tasks in addition and learn to cooperate in any of them. However, this does not factor in the possibility of changing the task of the learning agent and only learns joint policies that do not scale beyond a few select tasks.

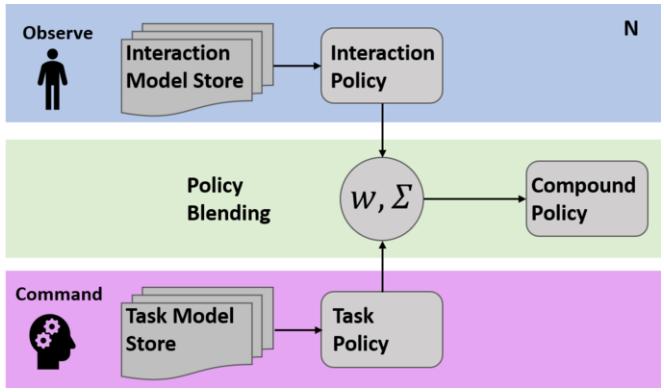


**Figure 2**: Architecture of our Action Aligned Interaction Learner.

The framework of [26] models action impact on policies using Graph-Based Policies (GPL) to adapt to open environments. However, GPL only considers the impact of other agents' action on the own reward but not how much impact one has on everyone else in the environment.

Almost all approaches in the current literature are restricted to a joint goal formulation of all agents. Our framework addresses the problem of learning to coexist in environments with a variable amount of agents that each perform their own tasks.

## 3 Action Aligned Interaction Learning

We propose to model a coexisting agent that navigates the world using separate task and impact policies (Figure 2). After deriving how to learn those policies we propose a method to unify them while weighting the relative importance of own and other's goals.

We define the problem where each policy attempts to solve a stochastic game given as the tuple $\langle N, \mathbb{S}, \{\mathbb{A}^i\}_{i\in\{1,\dots,N\}}, P, \{r^i\}_{i\in\{1,\dots,N\}}, \gamma\rangle$. $N$ refers to the number of agents where $N = 1$ is the standard single-agent MDP. $\mathbb{S}$ is the set of states of the world. $\mathbb{A}^i$ is the set of actions available to agent $i$ with $\mathbb{A} := \mathbb{A}^1 \times \cdots \times \mathbb{A}^N$. $P : \mathbb{S} \times \mathbb{A} \to \Delta(\mathbb{S})$ defines for a time step $t \in \mathbb{N}$ the transition probability to go from state $s \in \mathbb{S}$ to state $s' \in \mathbb{S}$ in the next time step. $r_t^i : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \to \mathbb{R}$ is the reward function that returns a scalar value to the $i$-th agent for a transition from $(s, a^i, \boldsymbol{a}^{-i})$ to $s'$ in a given timestep $t$. For formulations where only the action of the ego agent is relevant, we abbreviate our notation to a classic single-agent MDP formulation and omit to mention other agents. We define the policy function that outputs an action given a world state as $\pi : \mathbb{S} \to \mathbb{A}$. We also use $\rho_\pi(s_t)$ and $\rho_\pi(s_t, a_t)$ to denote the state and state-action marginals of the trajectory induced by a policy $\pi(a_t|s_t)$. Individual success is defined as $R^i = \sum_t r_t^i$, while coexistence success can be measured through the sum of rewards $\sum_i^N R^i$ of all agents in a given episode. For abbreviation and clarity purposes, when talking about distributions in the following section, it is implied that the distribution of a policy is given by $\pi \sim \pi(\cdot|s)|s \in S$.

An agent can only control its own actions as well as receive its own individual reward. We refer to policies learned to solve a given task as "task policies" and policies that are learned to improve the interaction with another agent with a given task as "impact policies". Multiple individually learned task policies will be kept in a task model store, to be activated based on the current task and likewise for impact poli-

cies. Once an agent is deployed, their task and all relevant impact policies are recombined into a single policy to solve the compound system (Figure 2) using an entropy-based blending mechanism.

### 3.1 Learning Task Policies

We learn our task and impact policies based on the maximum entropy reinforcement learning (MaxEnt RL) framework [14, 41]. MaxEnt RL adds an entropy term to the reward and maximizes the entropy over the policy distribution over each time step in addition to the traditional reward. The optimal policy function for MaxEnt RL with finite horizon $T$ is

$$\pi^*_{\text{MaxEnt}} = \underset{\pi}{\operatorname{argmax}} \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))], \quad (1)$$

where temperature parameter $\alpha$ controls the importance of the entropy in relation to the reward. By using this adapted objective, the policy automatically explores more of the state-space. The expected future return includes the future entropy for taking an action $a$ in state $s$ and following policy $pi$. The optimal policy $\pi$ can be estimated by a Q-function as an energy-based model (EBM) [14]

$$\pi^*(a|s) \propto \exp\left(\frac{1}{\alpha}Q^*(a,s)\right), \quad (2)$$

with the Q-function taking on the role of negative energy. In this work, we employ the Q-values as a basis for drawing samples from a Boltzmann distribution, which offers the advantage of being able to derive the distribution we are sampling from analytically. We are aware that the adoption of this method constrains our work to discrete action settings. The method is able to deal with large state spaces like any other Q-learning-based method relying on deep neural networks, where the network represents the energy function through the Q-values. Following [14] we define our optimal soft Q and soft V functions as

$$Q^*_{\text{soft}}(s_t,a_t) = r_t + \mathbb{E}_{(s_{t+1},\ldots)\tilde{\rho}_\pi}\left[\sum_{t=1}^{T}\gamma^t(r_{t+1} + \alpha\mathcal{H}(\pi^*(\cdot|s_{t+1})))\right]$$

with $\gamma \in [0,1)$ as the temporal discount factor, and

$$V^*_{\text{soft}}(s_t) = \alpha \log \sum_{a\in A} \exp\left(\frac{1}{\alpha}Q^*_{\text{soft}}(s_t,a)\right). \quad (3)$$

Our optimal policy is then given by

$$\pi^*_{\text{MaxEnt}}(a_t|s_t) = \exp\left(\frac{1}{\alpha}Q^*_{\text{soft}}(s_t,a_t) - V^*_{\text{soft}}(s_t)\right).$$

Using the notion of target and online network [23], parameterized respectively by $\bar{\theta}$ and $\theta$, the objective function that our Q-function neural network optimizes becomes

$$J_Q(\theta) = \mathbb{E}\left[\frac{1}{2}\left(Q^{\bar{\theta}}_{\text{soft}}(s_t,a_t) - Q^{\theta}_{\text{soft}}(s_t,a_t)\right)^2\right]. \quad (4)$$

We compute the target value as

$$Q^{\bar{\theta}}_{\text{soft}}(s_t,a_t) = r_t + \gamma\mathbb{E}_{s_{t+1}\tilde{p}_s}[V^{\bar{\theta}}_{\text{soft}}(s_t)], \quad (5)$$

where $V^{\bar{\theta}}_{\text{soft}}(s_t)$ is evaluated with the target parameters $\bar{\theta}$ according to Equation (3). Using real samples from the policy in past rollouts, one can apply gradient descent to optimize the objective in (4).

### 3.2 Learning Impact-Aware Interaction Policies

Formally, we define the impact $I$ to be the value/reward an action of one agent $i$ provides for another agent $j$.

$$I^i_j(s_t,a^i_t) = \mathbb{E}_\pi[R^j_{t+1}] - \mathbb{E}_\pi[R^j_t],$$

For an action $a$ taken by agent $i$ in state $s$, the impact is the change in expected return for agent $j$ between the original state and the subsequent state. As such we can rewrite the impact in terms of the value function

$$I^i_j(s_t,a^i_t) = V^j(s_{t+1}) - V^j(s_t). \quad (6)$$

Computing the impact requires estimating the value function of the given task of agent $j$. We assume that agents know how to solve approximately the task of the other agent themselves, that is, they have learned a task policy for it. We then take the perspective of the other agent and make use of our internal Q-model to estimate the values using (3) with the internal Q model corresponding to the task that agent $j$ is currently pursuing. If the state representation is specific to the agent's point of view, we transform the state. Using this impact measure, the agent learns an impact-aware policy in the maximum entropy fashion, where the reward is now given by $I$ to guarantee compositionality with the base task policy. We formulate our impact training objective as

$$J_c(\pi) = \sum_{t=0}^{T-1} \mathbb{E}_{(s_t,\mathbf{a_t})\sim\rho}[I(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t)].$$

From that formulation, it is straightforward to derive a soft Bellman operator as with the task policies

$$Q_i(s,a) \leftarrow I(s,a) + \gamma\mathbb{E}_{s'\sim p(s'|s,\mathbf{a})}[V(s')]. \quad (7)$$

To eliminate the need to train with others, the agent uses self-play to learn the impact models. During deployment, the model selects the task-corresponding policy and, for each agent in the scene, the impact-aware policy matches the respective agents' goal. For now, we assume to have full knowledge of these goals to isolate the effects of the separation of task and interaction objectives. Note, that the framework also allows to probabilistically combine policies according to the estimated likelihood of different possible goals.

### 3.3 Recombination of Policies

The final building block of our model is the combination of the resulting Q-values from impact and task policies. Combining policies is also known as policy blending and prior work has shown that compositionality arises naturally when using maximum entropy reinforcement learning [13] and that this approach is also known as a product of experts [15, 18]. Policy blending enables a clear separation between the policies that are directed toward other agents and those that are directed toward one's own goal. This is a crucial factor in enabling scalability to a greater number of agents and a wider range of task combinations. We propose to compose an agent's final policy with Q-function $Q_C$ based on the (weighted) sum of a given task policy $Q_T$ and the impact policies $Q_I$:

$$Q_C = w_t Q_t + (1-w_t)\sum_{i\in I}\frac{w_i}{\sum_{j\in I} w_j}Q_i. \quad (8)$$

The objective of the weighting of q-values is to reduce the negative impact of incorporating interaction-aware elements into the task policy. We motivate our entropy-based weighting approach and present

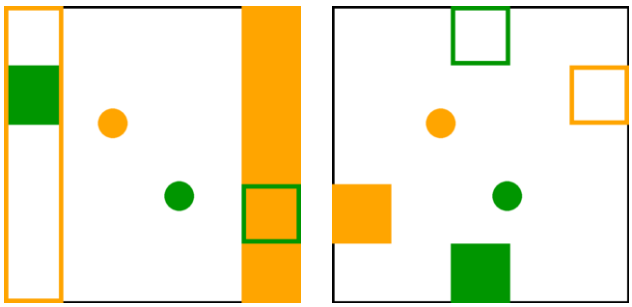

**Figure 3**: Two example scenarios in the Particle Environment with two agents. Filled areas are the primary target area of the agent of the respective color. Hollow areas define an influence area, where the agent of the respective color wants to locate other agents. Left: Primary areas of agents align with the influence area of the other agent. Right: Target and influence areas do not overlap.

the resulting bounds on the regrets we obtain. The weighting scheme prioritizes the task distribution when following the policy is critical and assigns bigger weights to the interaction policies, if we can blend them in without incurring significant costs. We propose to use the complement of the entropy $H \in [0, 1]$ to weigh the task distribution $\pi_t$ and the compound interaction distribution $\pi_c$. First, we introduce a lower bound of the policy blending process using an arbitrary weight $w \in [0, 1]$, extending the proof of [13].

**Lemma 1.** *Let $Q_1^*$ and $Q_2^*$ be the soft Q-functions corresponding to the optimal policies for reward functions $r_1$ and $r_2$, respectively. Define $Q_\sum \triangleq wQ_1^* + (1-w)Q_2^*$, where $w$ is a weight parameter. Then, the optimal soft Q-function $Q_C^*$ for the combined reward function $r_C \triangleq wr_1 + (1-w)r_2$ satisfies the following inequalities for all $s \in \mathbb{S}$ and $a \in \mathbb{A}$:*

$$Q_\sum(s,a) \geq Q_C^*(s,a) \geq Q_\sum(s,a) - C^*(s,a),$$

*where $C^*$ is the fixed point of*

$$C(s,a) \leftarrow \gamma \mathbb{E}_{s' \sim p(s'|s,a)} \left[ \mathbb{D}_w(\pi_1(\cdot|s') || \pi_2(\cdot|s')) + \max_{a' \in \mathbb{A}} C(s',a') \right]$$

*and $\mathcal{D}_w$ is the Rényi divergence of order $w \in [0, 1]$.*

*Proof.* See Appendix. □

**Corollary 1.** *Following Lemma 1 we can deduce that the regret of using policy $\pi_2$ instead of policy $\pi_1$ is proportional to the Rényi divergence $K(\pi_1, \pi_2) \propto \mathcal{D}_w(\pi_1 || \pi_2)$.*

We propose to compute the weight $w_t$ of the task policy depending on the entropy

$$w_t = 1 - \frac{|A|^{\mathcal{H}_{|A|}(\pi_t)}}{|A|}. \tag{9}$$

The entropy of a policy encodes the uncertainty of which action to take. We assign a small weight to distributions with high uncertainty and a large weight to those with low uncertainty. Entropy is a sensible choice for blending two policies as the regret is proportional to the entropy.

**Lemma 2.** *The expected Jensen-Shannon distance to a fixed policy $\pi_t$ for a policy $\pi_c$ drawn by a Dirichlet process DP with a non-informative prior is proportional to the negative entropy of the policy $\pi_t$.*

$$\mathbb{E}_{\pi_c \sim X, X \sim DP(G,\alpha)} \left[ JSD(\pi_t || \pi_c) \right] \propto -\mathcal{H}(\pi_t),$$

*where the base distribution $G$ is chosen to be uniform and $\alpha \in \mathcal{R}^+$.*

**Algorithm 1** Training Procedure for Task or Impact Policies

```
Initialize random parameters θ
Assign target parameters θ̄ ← θ
D ← Empty replay buffer
Assign goal g
for each epoch do
    for each t do
        Sample actions for each agent a_{t,n} ~ π_n(s_t)
        Sample the next state s_{t+1} ~ p_s(s_{t+1}|s_t, a_t)
        Save transition to replay memory
        D ← D ∪ (s_t, a_t, r_t, s_{t+1})
        Update network parameters
        if t mod update_interval = 0 then
            Update target parameters θ̄ ← θ
        end if
        Sample minibatch m := s_t, a_t, r_t, s_{t+1}
        if g is ego task then
            Compute Q_soft and V_soft according to 5 and 3
        else
            Compute Q_soft and V_soft according to 7 and 3
        end if
        Compute ∇J(θ) according to 4
        Update parameters θ using ADAM
    end for
end for
```

*Proof.* See Appendix. □

This establishes that we have a regret term that is bound by the Rényi divergence of the two policies and we can show that in the expectation the JSD is proportional to the entropy of the task policy and is an upper bound of the regret $K$, establishing

$$K(\pi_t, \pi_c) \leq \mathbb{E}_{\pi_c \sim X, X \sim \mathrm{DP}(G,\alpha)} \left[ \mathrm{JSD}(\pi_t || \pi_c) \right] \propto -\mathcal{H}(\pi_t),$$

By utilizing the relationship between the entropy of the task policy and the amount of blend-in of the compound interaction policy, we can ensure that incorporating it does not increase the regret. The entropy of the distribution $\mathcal{H}_{|A|}(\pi)$ is computed based on the number of actions and the max entropy is normalized to one.

Since the regret of using the compound policy is influenced by the distance of the task distribution to the interaction policies, we use the pairwise Jensen-Shannon Distance (JSD) as the weights of the interaction policies in the blending process. We define the weight $w_i$ for policy $\pi_i$ as the JSD to the task distribution

$$w_i = \mathrm{JSD}(\pi_t || \pi_i). \tag{10}$$

These weights are then used in (8).

Finally, to ensure equal contribution every distribution of Q-values has to be normalized to guarantee weight-based contribution to the compound policy. To summarize, we compute the Q-values of the given task for each action, then compute the Q-values for each action with respect to the impact on each other agent. We obtain the weighting of Q-values using the entropy and JSD between the task action distribution and the impact action distribution. Finally, combine the Q-values as a weighted sum and sample an action from the resulting action distribution.

### 3.4 Training Procedure

We propose the soft Q-Impact learning algorithm for learning in discrete action settings with multiple agents working on different in-

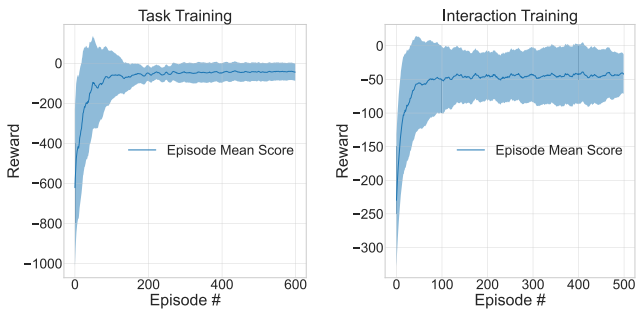


**Figure 4**: Reward of an agent during task learning (left) and impact learning (right) on 35 different tasks. Plotted is the mean with standard deviation.

dependent goals. The algorithm trains a set of ego task policies and uses them to simulate a second agent in self-play to learn a separate impact policy for each task it has previously learned. During impact learning the agent performs actions but uses the reward of the second agent for training. Through the impact reward formulation in Equation (7) the agent optimizes its positive impact. The Q-networks are updated using sampled mini-batches of experience from a replay buffer $\mathcal{D}$ [22] as is standard in deep Q-learning. Additionally, our update rule uses a target network that is periodically updated [23]. For optimization of the Q-networks, the ADAM optimizer [17] is used. After training has finished, we test the framework by simulating our agent in coexistence with agents trained on their given task with an independent method and measure its individual performance as well as the team's performance. We summarize our training procedure in Algorithm 1.

## 4 Experiments

The experiments are designed to answer the following research questions: (1) Can the agent learn impact-aware policies? (2) Can the agent combine task and interaction policies to solve the given task and help others? (3) Is the framework robust to setups with opposing goals? (4) How does the framework scale with the number of tasks and agents? (5) Does the algorithm scale to complex sparse tasks?

We test our framework in coexistence environments and start by introducing the navigation multi-particle environment and the different scenarios we trained and tested on, as well as explaining the baselines we have used. In a new variant of cooking-gym [36] named cooking-zoo, we show the scalability of our method to complex environments and tasks. We analyze the model's performance in comparison to the ablations of our methods and these baselines in both environments. We publish the new cooking environment to encourage future research in coexistence environments and to provide a complex simulation with social aspects that can model general-sum games[1].

### 4.1 Multi-Particle Navigation Environment

We created a multi-particle environment with continuous observation space and discrete action space in a fully observable world as depicted in Figure 3. In contrast to existing environments, the reward structure of our specific multi-particle setup has the properties of a coexistence environment. It supports independent goals that can be reached without specific actions of others and provides explicit influence on the rewards of others, modeling a general-sum game.

[1] https://github.com/DavidRother/cooking_zoo

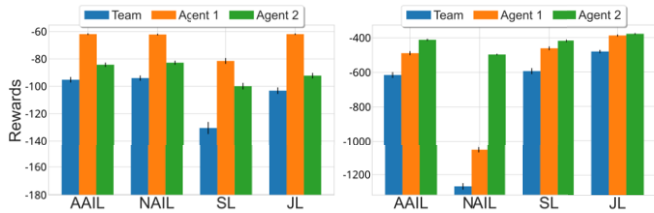


**Figure 5**: Single agent and team performance as accumulated reward of the Interaction learner with (AAIL) and without (NAIL) action alignment in comparison to a single-agent reward learner (SL) and a joint reward learner (JL). Left: goal-aligned scenario. Right: non-goal-aligned scenario. Errorbars show the standard error.

We define the world as a normalized square with agent positions in $[0, 1]$. The state of an agent consists of its x and y coordinates as well as its velocity. The state observation of a given agent is the concatenation of all agent states ($|\mathcal{O}| = 4 \times N$ with $N$ being the number of agents). The action space consists of five acceleration actions (rest, up, down, left, right) of which one can be chosen at any time step. The goal is to reach the space within a specific cell, row, or column in a regular 5x5 grid covering the environment. In addition, there is a second area influencing the reward of the corresponding agent depending on the distance of other agents to it. The reward function is the negative absolute distance of the agent toward its target area plus the negative distance of all other agents toward its influence area

$$r_i(\mathbf{x}, \mathbf{y}, T_i^x, T_i^y, I_i^y, I_i^y) = -\sqrt{(T_i^x - x_i)^2 + (T_i^y - y_i)^2} - \sum_{j \in N, j \neq i} \sqrt{(I_i^x - x_j)^2 + (I_i^y - y_j)^2},$$

where $T$ describes the closest point of the target area to the agent and $I$ is the closest point to the respective other agent of the influence area. An episode is concluded once all agents have reached their target area or after 1000 time steps have elapsed.

The environment can be categorized as having mixed objectives because the reward has an interactive as well as a single-agent component. We call environment instantiations where the best position of the agent in each state maximizes both team and single-agent reward as "goal-aligned scenarios" (Fig. 3 left) and otherwise "non-goal-aligned scenarios" (Fig. 3 right). In goal-aligned scenarios, the target area and the influence area of all other agents have to overlap. The interaction mechanism allows the incorporation of many different desirable aspects for testing an ad hoc framework, namely the independence in the reachability of a goal and the direct influence on the total reward between agents. Also, since we can specify different goals, being either a cell, row, or column, we get 35 different navigation tasks that model various degrees of disagreement in the optimal position concerning each agent. Combined with an interaction task we get a collaborative task space of $35 \times 35 = 1225$ combinations to test with only two agents, scalable to 5 agents with $52521875$ combinations.

### 4.2 Results

As baselines we use a Single-agent Learner (SL), that is the task policy alone, effectively neglecting effects on the other agent, and a Joint Learner (JL), trained explicitly on the sum of rewards of both agents. JL needs to be trained anew with an individual Q-network for each task combination of own and other tasks. We deploy our algorithm in two modes: The first mode uses simple averaging to combine

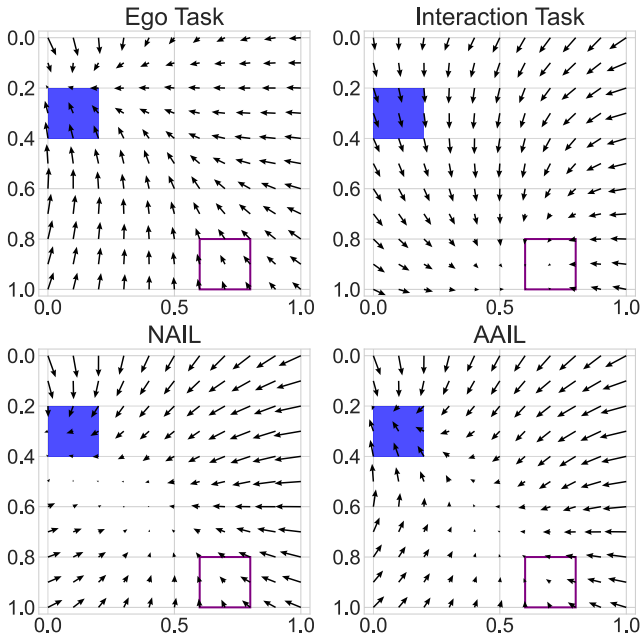


**Figure 6**: Average direction of acceleration of learned policies for a given scenario. Policies based on learning the ego task ($Q_C$), the impact on the other agent's task ($Q_I$) and the combination of both using The Non-Aligned Impact Learner (NAIL) and our proposed weighting (AAIL).

the task and impact Q-functions and is termed Non-Aligned Interaction Learner (NAIL). This highlights the effects of the architecture versus the weighting. The second is termed Action Aligned Interaction Learner (AAIL), which uses weights based on equations (9) and (10). We train the impact-aware agents on all 35 tasks and correspondingly 35 impact tasks, resulting in 70 different Q-networks, covering all task combinations the environment has to offer. The JL baseline would need 52521875 individual networks to cover the same variety.[2] The other agents in the environment were trained on the individual task only and come from different populations such that no agent had the possibility for joint training with other agents. We confirm a statistically significant difference in each scenario with 200 test runs using the Wilcoxon Signed-Rank test. The training performance over all 35 individual tasks can be seen in Figure 4, showing convergence for both task and impact policies.

The first test we conduct is performed in a goal-aligned scenario with two agents, with the twist that the target goal area of the controlled agent, in this and all following experiments named agent 1, is larger than the influence area of the other agent. We expect to see that all agents can solve the task, however, team performance should be lowest for the SL baseline. Figure 5 left depicts results for one sample task combination.

It shows the respective accumulated rewards of the individual agents as well as the team reward. The Action Aligned Impact Learner (AAIL) and the Non-Aligned Impact Learner (NAIL) show similar performance r($p \sim 0.16$). The Joint Reward Learner (JL) is close in performance but statistically different with $p < 6.3e-12$ and the single-task learner (SL) is by far the worst ($p < 2.0e-19$). The weighting of Q-values does not make a significant difference

[2] It might also be possible to learn a single network with the task encoded as input but to avoid interference between tasks we do implement single networks for each goal.

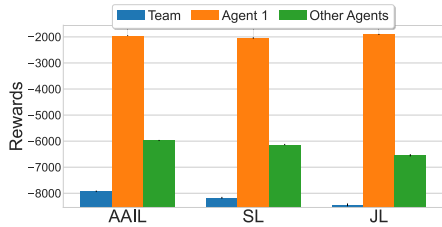


**Figure 7**: Single agent and team performance as an accumulated reward for one scenario with 5 agents (averaged over 200 runs).

between AAIL and NAIL as the target and influence areas overlap. AAIL and NAIL are able to find the best behavior in this case while already showing benefits over SL-trained agents demonstrating that blending the different policies worked. The second experiment we conduct is the non-goal-aligned case. The target and influence areas do not overlap in these scenarios. The agent has to navigate toward their own goal but may choose to do so in the best possible way for the other agent. We expect SL to perform similarly to the first set of scenarios, JL should perform best since it is directly trained on the team reward of both agents and AAIL should have a slight performance loss due to the contradictory nature of actions to navigate either the target or influence area but still manage to complete the task. NAIL should in contrast face massive problems in successfully navigating to either the target or influence area since the policies might favor acceleration in opposing directions in the worst case. Results are shown in Figure 5 right. The joint learner performs best, followed by the single-task learner and AAIL. As expected only NAIL does not manage to solve the task. It is noteworthy that for NAIL the performance of the second agent does not plummet as much, despite NAIL never reaching its target area to end the episode early, indicating that NAIL overvalues the input of the impact distribution. AAIL still finishes the task but lacks in performance behind SL and JL and is statistically different with $p << 1e-10$ from both. This is due to the previously mentioned discrepancy between the impact-aware and ego task policies. We visualize the average direction an agent wants to accelerate in Figure 6. It can be seen that the naive combination of task and impact-aware policies leads to an overweighting of the latter. This happens as the absolute Q-values of policies do not necessarily scale the same, resulting in an inherently unbalanced policy combination. For AAIL it is visible that the task policy dominates but the contributions of the impact-aware policy are not completely negated. We can conclude that policy blending is integral to the success of combined policies when dealing with opposing goals.

In our third experiment, we test the scaling capabilities of our algorithm by placing five agents into the environment, one of which is controlled by the evaluated algorithms in their respective test runs. The goals in this scenario are not aligned. We expect the joint learner to perform the best followed by AAIL, and SL being the worst. Results are presented in Figure 7. Surprisingly JL is the worst-performing algorithm with $p < 1.4e-11$, however only when looking at the scores of the other agents in the environment. This is due to the complexity of the reward function with four other agents present, not only preventing JL from finding the best possible solution but being even worse than SL towards other agents on average. The confidence that AAIL is statistically different from SL is given with $p = 0.048$. In general, this demonstrates the capabilities of the overall framework to scale beyond two agents with many different tasks.

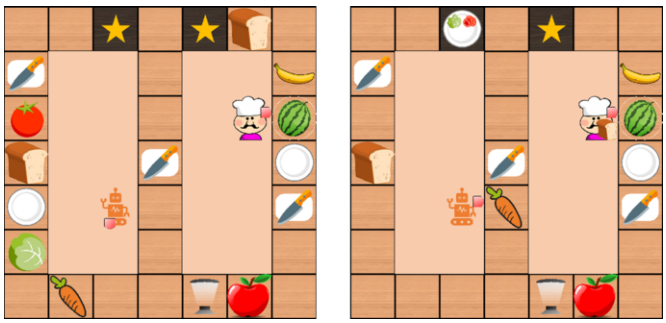

**Figure 8**: The cooking Environment.

### 4.3 Cooking Environment

The second environment we used is a new cooking environment, inspired by the game Overcooked as well as previous work using another cooking environment [36]. The previous work was limited to a single joint goal, modeling only strictly cooperative tasks, whereas our environment allows each agent to have separate recipes. We offer a fixed vector length input representation of the state space between different layouts of the environment. Our environment supports the latest versions of the gymnasium [3] and pettingzoo [4] libraries, enabling easy usage of common RL frameworks. We support a reward scheme with rewards for sub-goals and only for complete dishes as well as a configurable time penalty.

Kitchens in our environment consist of movable components and static stations (such as counters, knife stations, and blenders). Agents can move in all four directions or stay still and they move simultaneously. An episode is concluded once all agents have delivered their recipe or after 400 time steps have elapsed. The state space consists of a vector describing the state of all objects and their relative position to the agent. We chose to give rewards only for complete dishes (20) and subtract a time penalty for each step (0.01). To cook a dish an agent has to prepare the correct ingredients with the correct tool. Then the ingredients have to be placed on the same plate and the plate has to be placed on a square marked with a star. In our experiment we assign two spatially separated agents to two different recipes, each requiring two ingredients as depicted in Figure 8. The right agent needs the carrot from the left agent, whereas the left agent has its ingredients available without needing help. We test if our agent manages to deliver the carrot to the other agent and cook his own dish in comparison to the task-policy agent.

### 4.4 Results

We use the AAIL agent and an SL agent for our evaluation in the cooking environment alongside trained Proximal Policy Optimization (PPO) agents [29] for the other cook and report in the shown setup over 100 runs in Figure 9. When the agents act independently in the interaction-free scenario we expect the SL agent to be as good as the AAIL agent, while the performance of the second agent should be unaffected. The results in Figure 9 show the dish completion ratio of the AAIL algorithm and the SL agent in the same scenarios. The AAIL agent completes the dish slightly less often (4%) than the SL counterpart. But the PPO agent profits massively from the impact-aware AAIL agent. We can see that the agent does deliver the carrot only after having cooked its own recipe, which results in a slightly lower reward for the own agent on average ($\sim -0.78 \pm 1.32$). The incorporation of the interaction distribution is considered after the dish has been done, as the task distribution does not assign high probability mass to single actions, resulting in the interaction distribution taking over. However, because of the noise in this unseen state space of the task policy, it is evident that the success ratio of the right agent, even if greatly improved, is still far from optimal.

[3] https://gymnasium.farama.org/
[4] https://pettingzoo.farama.org/

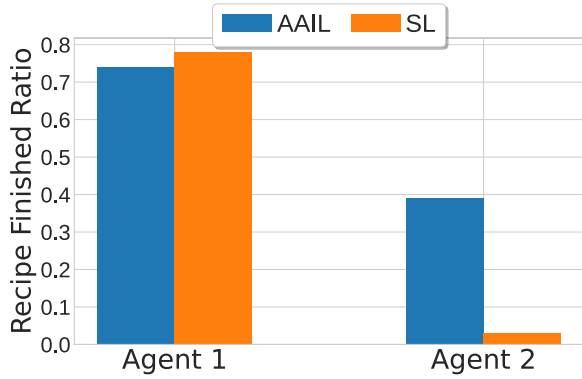


**Figure 9**: Ratio of finished recipes over 100 runs for both agents in the environment using AAIL in contrast to SL for the left agent, which has to interact. Agent 1 is controlled by AAIL/SL and Agent 2 is controlled by PPO.

## 5 Conclusion & Future Work

In summary, we were able to learn separate tasks and impact policies (Research Question 1). The impact policies worked with more than one other agent and increased the overall performance of the group (2). Furthermore, the regret of using a compound policy was low in scenarios, in which helping and fulfilling the own task was not compatible (3). Through our action alignment, we show that our system manages to keep the regret drastically lower than with a simple policy averaging. Despite never seeing the test distribution of tasks with specific other agents, our framework managed to achieve good results in all tested scenarios, including the very challenging cooking environment. We demonstrated the extrapolation ability through the recombination of policies to new task combinations, allowing our system to scale much more favorably than a joint training approach (4). Experiments in a new cooking environment demonstrated an ability to scale to a complex sparse environment. (5).

This paper introduced a formulation and methodology to systematically learn a disentanglement of tasks and interactions with other agents separately from each other. Our approach combines maximum entropy reinforcement learning with a Multi-Agent system model and a new method to combine policies. It was demonstrated to solve a navigation environment and a cooking environment in simulation without ever seeing the test scenarios during training. The flexible approach allows to scale in the number of tasks in the environment and the number of agents and lays the foundation for architectures suited for more complex domains. An extension of this work could test the approach with different types of co-existing agents, and in particular with human players, and consider epistemic uncertainty.

## Acknowledgements

David Rother gratefully acknowledges the financial support from Honda Research Institute Europe.

## References


[1] Iina Aaltonen, Anne Arvola, Päivi Heikkilä, and Hanna Lammi, 'Hello pepper, may i tickle you? children's and adults' responses to an entertainment robot at a shopping mall', in *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pp. 53–54, (2017).

[2] Baris Akgun, Maya Cakmak, Jae Wook Yoo, and Andrea Lockerd Thomaz, 'Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective', in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pp. 391–398, (2012).

[3] Stefano V Albrecht and Peter Stone, 'Reasoning about hypothetical agent behaviours and their parameters', *arXiv preprint arXiv:1906.11064*, (2019).

[4] Samuel Barrett, Avi Rosenfeld, Sarit Kraus, and Peter Stone, 'Making friends on the fly: Cooperating with new teammates', *Artificial Intelligence*, **242**, 132–171, (2017).

[5] Samuel Barrett, Peter Stone, Sarit Kraus, and Avi Rosenfeld, 'Learning teammate models for ad hoc teamwork', in *AAMAS Adaptive Learning Agents (ALA) Workshop*, pp. 57–63. Citeseer, (2012).

[6] Samuel Barrett, Peter Stone, Sarit Kraus, and Avi Rosenfeld, 'Teamwork with limited knowledge of teammates', in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 27, pp. 102–108, (2013).

[7] Moritz C Buehler and Thomas H Weisswange, 'Theory of mind based communication for human agent cooperation', in *2020 IEEE International Conference on Human-Machine Systems (ICHMS)*, pp. 1–6. IEEE, (2020).

[8] Felix Carros, Johanna Meurer, Diana Löffler, David Unbehaun, Sarah Matthies, Inga Koch, Rainer Wieching, Dave Randall, Marc Hassenzahl, and Volker Wulf, 'Exploring human-robot interaction with the elderly: results from a ten-week case study in a care home', in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–12, (2020).

[9] Prashant Doshi, Xia Qu, Adam Goodie, and Diana Young, 'Modeling recursive reasoning by humans using empirically informed interactive pomdps', in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pp. 1223–1230, (2010).

[10] Guanglong Du, Mingxuan Chen, Caibing Liu, Bo Zhang, and Ping Zhang, 'Online robot teaching with natural human–robot interaction', *IEEE Transactions on Industrial Electronics*, **65**(12), 9571–9581, (2018).

[11] Piotr J Gmytrasiewicz and Prashant Doshi, 'A framework for sequential planning in multi-agent settings', *Journal of Artificial Intelligence Research*, **24**, 49–79, (2005).

[12] Michael A Goodrich, Alan C Schultz, et al., 'Human–robot interaction: a survey', *Foundations and Trends® in Human–Computer Interaction*, **1**(3), 203–275, (2008).

[13] Tuomas Haarnoja, Vitchyr Pong, Aurick Zhou, Murtaza Dalal, Pieter Abbeel, and Sergey Levine, 'Composable deep reinforcement learning for robotic manipulation', in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 6244–6251. IEEE, (2018).

[14] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine, 'Reinforcement learning with deep energy-based policies', in *International conference on machine learning*, pp. 1352–1361. PMLR, (2017).

[15] Geoffrey E Hinton, 'Training products of experts by minimizing contrastive divergence', *Neural computation*, **14**(8), 1771–1800, (2002).

[16] Trong Nghia Hoang and Kian Hsiang Low, 'Interactive pomdp lite: Towards practical planning to predict and exploit intentions for interacting with self-interested agents', *arXiv preprint arXiv:1304.5159*, (2013).

[17] Diederik P Kingma and Jimmy Ba, 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980*, (2014).

[18] An T Le, Kay Hansel, Jan Peters, and Georgia Chalvatzaki, 'Hierarchical policy blending as optimal transport', *arXiv preprint arXiv:2212.01938*, (2022).

[19] Marcus Mast, Michael Burmester, Birgit Graf, Florian Weisshardt, Georg Arbeiter, Michal Španěl, Zdeněk Materna, Pavel Smrž, and Gernot Kronreif, 'Design of the human-robot interaction for a semi-autonomous service robot to assist elderly people', in *Ambient assisted living*, 15–29, Springer, (2015).

[20] Francisco S Melo and Alberto Sardinha, 'Ad hoc teamwork by learning teammates' task', *Autonomous Agents and Multi-Agent Systems*, **30**(2), 175–219, (2016).

[21] 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 *European Conference on Multi-Agent Systems (EUMAS)*, (2022).

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller, 'Playing atari with deep reinforcement learning', *arXiv preprint arXiv:1312.5602*, (2013).

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al., 'Human-level control through deep reinforcement learning', *nature*, **518**(7540), 529–533, (2015).

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

[25] Afshin OroojlooyJadid and Davood Hajinezhad, 'A review of cooperative multi-agent deep reinforcement learning', (8 2019).

[26] Muhammad A Rahman, Niklas Hopner, Filippos Christianos, and Stefano V Albrecht, 'Towards open ad hoc teamwork using graph-based policy learning', in *International Conference on Machine Learning*, pp. 8776–8786. PMLR, (2021).

[27] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson, 'Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning', in *International conference on machine learning*, pp. 4295–4304. PMLR, (2018).

[28] Sannidhya Sandheer, Tanya Sangtiani, Aditya Singh, Niharika Varshney, and Piyush Soni, 'Using transfer learning in an ad hoc team', (2019).

[29] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov, 'Proximal policy optimization algorithms', *arXiv preprint arXiv:1707.06347*, (2017).

[30] Bernhard Sendhoff and Heiko Wersing, 'Cooperative intelligence-a humane perspective', in *2020 IEEE International Conference on Human-Machine Systems (ICHMS)*, pp. 1–6. IEEE, (2020).

[31] Thomas B Sheridan, 'Human–robot interaction: status and challenges', *Human factors*, **58**(4), 525–532, (2016).

[32] Charlie Street, Bruno Lacerda, Michal Staniaszek, Manuel Mühlig, and Nick Hawes, 'Context-aware modelling for multi-robot systems under uncertainty', (2022).

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

[34] Zihao Wang, Yanxin Zhang, Chenkun Yin, and Zhiqing Huang, 'Multi-agent deep reinforcement learning based on maximum entropy', pp. 1402–1406. Institute of Electrical and Electronics Engineers Inc., (6 2021).

[35] George Wilmers, 'A foundational approach to generalising the maximum entropy inference process to the multi-agent context', *Entropy*, **17**, 594–645, (2015).

[36] Sarah A. Wu, Rose E. Wang, James A. Evans, Joshua B. Tenenbaum, David C. Parkes, and Max Kleiman-Weiner, 'Too many cooks: Coordinating multi-agent collaboration through inverse planning', *Topics in Cognitive Science*, **n/a**(n/a), (2021).

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

[38] Yaodong Yang and Jun Wang, 'An overview of multi-agent reinforcement learning from game theoretical perspective', (11 2020).

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

[40] Jihong Zhu, Michael Gienger, and Jens Kober, 'Learning task-parameterized skills from few demonstrations', *IEEE Robotics and Automation Letters*, **7**(2), 4063–4070, (2022).

[41] Brian D Ziebart, *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*, Carnegie Mellon University, 2010.