Autonomous Agents and Multi-Agent Systems (2025) 39:27
https://doi.org/10.1007/s10458-025-09707-7

Entropy based blending of policies for multi-agent coexistence

David Rother[1,3] · Franziska Herbert[1] · Fabian Kalter[1] · Dorothea Koert[1] · Joni Pajarinen[1,2] · Jan Peters[1] · Thomas H. Weisswange[4]

Accepted: 28 April 2025 / Published online: 16 May 2025

Abstract
Research on multi-agent interaction involving humans is still in its infancy. Most approaches have focused on environments with collaborative human behavior or a small, defined set of situations. When deploying robots in human-inhabited environments in the future, the diversity of interactions surpasses the capabilities of pre-trained collaboration models. "Coexistence" environments, characterized by agents with varying or partially aligned objectives, present a unique challenge for robotic collaboration. Traditional reinforcement learning methods fall short in these settings. These approaches lack the flexibility to adapt to changing agent counts or task requirements without undergoing retraining. Moreover, existing models do not adequately support scenarios where robots should exhibit helpful behavior toward others without compromising their primary goals. 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 using a goal inference model for task estimation. We create impact-aware agents and linearly scale the cost of training agents with the number of agents and available tasks. To this end, a weighting function blending action distributions for individual interactions with the original task action distribution is proposed. To support our claims we demonstrate that our framework scales in task and agent count across several environments and considers collaboration opportunities when present. The new learning paradigm opens the path to more complex multi-robot, multi-human interactions.

✉ David Rother
david.rother@tu-darmstadt.de

Dorothea Koert
dorothea.koert@tu-darmstadt.de

Joni Pajarinen
joni.pajarinen@aalto.fi

Jan Peters
jan.peters@tu-darmstadt.de

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

1 Intelligent Autonomos Systems (IAS), TU Darmstadt, Darmstadt, Germany

2 Aalto University, Espoo, Finland

3 Honda Research Institute EU, Offenbach, Germany

4 Honda Research Institute Europe GmbH, Offenbach, Germany

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

1 Introduction

Deployment of robots in environments shared with humans is a trending research topic [1, 2] as there is an increased push toward robots that work alongside humans. The applications studied in the scientific literature vary and include topics such as teaching [3, 4], assistance [5–7], entertainment [8], delivery [9], service [10] or elderly support [11]. The earliest example of robots deployed to a human environment only cared about not colliding with them [12]. Nowadays, modeling of an interaction aims for an explicit collaboration with a human or a group of humans to achieve a common goal [13, 14]. 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 [15]. The essence of these interactions lies not just in the robots' ability to perform tasks but also in their exposure to 'other' humans who are not the primary users but share the environment and may have independent objectives. How to (co-)operate when facing other people while performing a task is an important question that needs to be addressed in human-robot interaction.

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 a second household member starts cooking something for themself. The robot and the cook both require access to some shared resources, like the dish rack, but 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.

We term such situations "coexistence" environments, where there are at least two agents present, that do not share a goal and are not jointly controlled. Figure 1 depicts the relation to other types of environments used in the literature. In coexistence environments, agents have an impact on other agents' performances, most likely through sharing space or resources, while

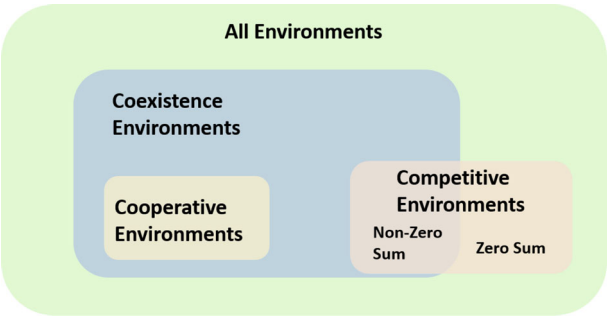

Fig. 1 Classification of multi-agent environment structures. Coexistence environments constrain the performance of multiple agents to be interdependent but not exclusive. Cooperative environments and some competitive environments are considered sub-sets

both agents can in principle reach their goals. If there exists an explicit interdependence to reach a certain goal, the joint task is classified as 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 are related to the Ad-Hoc Teamwork (AHT) field where agents have to collaborate with unknown teammates. Classical Ad-Hoc teamwork operates within the definition of cooperative environments as agents have an overarching joint goal. AHT agents always share some part of the reward function or goal, even in mixed-motive situations, where agents still share a collaborative goal [16, 17]. A coexisting agent in contrast does not have to share a goal or part of the reward function (but certainly can). This distinction places every AHT framework within the realm of coexistence, extending the application space of AHT. 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 single defined optimization target in the joint reward, with which possible algorithms are evaluated. On the other hand, in coexistence environments, defining an optimization target is not as straightforward as there are two distinct, possibly opposing evaluation criteria. The first is the reward of the focal agent, and the second is the joint reward of all other agents.

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 based on how our actions influence other agents while they perform their respective tasks. Task policies learn based on a reward function how to solve a given task and the second type of policies is called interaction policies, which learn in self-play with task policies how to assist them solving a task. Additionally, we derive an entropy-based mechanism for blending these policies. Finally, we employ a goal inference approach, to estimate the tasks of other agents during evaluation and include uncertainty.

Our method is based on the assumption that the environment is fully observable with categorical action spaces. We assume further that every task can be (non-optimally) solved by one agent alone. Each task that our agent wants to support needs to be solvable by the agent alone, and it has to be able to simulate the task for self-play.

In addition to showcasing the scalability of this approach for agents with diverse intentions, we analyze the advantages of employing this approach towards enhancing the task performance of the population, while simultaneously minimizing any negative impact on the focal agent's performance. Scalability in our framework is achieved by decoupling task-specific learning from interaction dynamics, allowing each component to be learned independently. Once the task policies and interaction policies are trained, they can be recombined dynamically for new agent-task scenarios without the need for retraining. The entropy-based blending mechanism enables the seamless integration of these policies, adjusting to changing agent populations and tasks. To support these claims we intend to answer the following research question and sub-questions:

(1) Does incorporating entropy-based blending in policies enhance scalability and benefit both population-level and individual focal agent behaviors?

(1.1) Does the combination of task and interaction policies enable an agent to solve the given task while helping others under task uncertainty?

(1.2) Do we see a scaling benefit of the framework over traditional joint learning approaches with an increasing number of agents?

(1.3) Does our system perform well when incorporating task uncertainty?

The rest of the paper is structured as follows: We start by giving an overview over the related work and show algorithms that achieve parts of our problem setting but fall short in other aspects. Afterward, we present our algorithm with a new goal-predictive component to deal with unknown goals of other agents extending previously published work [18]. Additionally, we extend our analysis to more complex scenarios with more agents, including a new environment in our evaluation, and add a comparison to another method in a baseline cooperative scenario. We conclude our paper by summarizing our findings and pointing out interesting future research directions.

2 Related work

Coexistence environments provide several challenges, such as modeling other agents' policies and intentions, evaluating the impact of one's 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 [19] framework. Decision frameworks using this model engage in explicit reasoning about humans [20, 21] and use the mental model of others to improve their 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 task.

Multi-agent reinforcement learning The issues usually faced with problems involving multiple agents are that learning a single network based on a joint reward signal faces a credit assignment problem and has to deal with the combinatorial space over the task-agent space, which leads to catastrophic forgetting and performance degradation. For this reason, multi-agent approaches commonly deal with cooperative environments, where a common goal is present such that individually optimal behavior is preferable in comparison to egoistic behavior to achieve the best possible reward [22] and the aforementioned problems do not occur. Such problems explore the difficulty of coordinating to achieve the best reward [23] but do not generalize to problems where the focal agent's best strategy does not align with the strategy that achieves, given successful coordination, the highest population reward. Integrating other agents from the population in the learning process of a policy is a central aspect of Multi-Agent Reinforcement Learning (MARL), where all agents are jointly trained together [22, 24]. One approach 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 [25–27]. The resulting policies are restricted to a specific set of tasks for the agents, which restricts the generalizability of the system to new tasks, which we do consider in this work. In [28], agents are trained in a centralized learning, decentralized execution regime and compute a credit score by taking other's perspectives but need to retrain the complete policy if their own goal or that of the other agent changes. Previous work has also considered Multi-Agent systems that have to solve multiple tasks [29]. 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 but is strictly limited to these areas, whereas we do not rely on shared goals between agents.

Ad-hoc Teamwork Another multi-agent problem setting is Ad-hoc teamwork, where, in contrast to traditional multi-agent reinforcement learning approaches, one agent is controlled instead of many during evaluation. Ad-hoc teamwork is a single-agent learning problem where the agent has to be capable of cooperating on the fly with other agents without prior coordination [17]. A popular approach is to compute Bayesian posteriors over predefined teammate types, which are then used in other models, such as reinforcement learners [30, 31]. Others use transfer learning intending to reuse knowledge across agents to enable faster adaptation to new agent types [32, 33]. Melo and Sardinha [34] learn a teammate's task from a set of predefined tasks 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 joint policies that do not scale beyond a few select tasks are learned.

Almost all approaches in the current literature are restricted to a joint goal formulation of all agents and do not consider open environments [35]. The framework of [36] models action impact on policies using Graph-Based Policies (GPL) to adapt to open environments. However, GPL only considers the impact of other agents' actions on their reward not how much impact one has on everyone else in the environment. Additionally, GPL is trained on one task distribution for the present agents and can not deal with a different task distribution during evaluation by design. Our framework addresses the problem of learning to coexist in environments with a changing number of agents that each perform their tasks, addressing openness in the task space as described in [35].

Policy Blending Maximum entropy methods have been shown to produce policies that are robust to minor changes in the environment or other agents' behaviors [37–40]. Combining energy-based policies as a product of experts has also been shown as an effective measure to solve problems that include multiple sub-tasks [41–43]. Other policy blending measures in the literature are used for shared control with an arbitration mechanism between policies [44–46]. They either use the human policy or a generated one, which is decided based on some computed factor or the presence of human inputs. On the other hand, we look at interaction with others instead of sharing control and blending policies by combining them instead of arbitrating which one to use.

Goal Inference To solve the problem of not knowing other agents' intentions, various methods and approaches exist to tackle goal inference. Some approaches try to reimplement human Theory of Mind (ToM) models from Cognitive Science on an agent [47–51] or make use of perspective-taking to infer a ToM [52–54]. In our work, we also take the perspective of the other agent to infer their goals. Other approaches are more computationally inspired and make use of Inverse Reinforcement Learning [55–57] or Bayesian Inference [58–60] to implement a ToM model. In recent years several approaches made use of deep neural networks to estimate the mental state of an agent [61–63] and we use a fully connected neural net as well in our work to estimate a distribution of goals because of the simplicity of the approach.

Prior work has not dealt with the complexities that come with coexistence and, as a result, only solves the problem in part. Our work creates a new framework to view multi-agent interaction in varied tasks as a decomposable problem and solves it through this inherent structure.

3 Entropy-based policy blending interaction

We propose to model a coexisting agent that navigates the world using separate task and impact policies. In addition to modular policies, we use a goal inference model and a recombination mechanism to infer the action distribution to sample from during evaluation (Fig. 2) to avoid both of these problems.

We start this section with the formal definition of the problem setting. We describe how to learn task policies and interaction policies. We follow with a section on how we construct our goal inference module. To close off the section, we explain how our policy recombination works and derive its formal properties.

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. We introduce an action that does not change the world state as a_0 and can be seen as the neutral action (i.e. standing still) in each environment. $r_t^i : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$ is the reward function that returns a scalar value to the i-th agent for a transition from $(s, a^i, \boldsymbol{a}^{-i})$ in a given timestep t, where $\boldsymbol{a}^{-i}$ is the action vector of all agents except agent i. 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 $\pi \sim \pi(\cdot|s) \quad | \quad s \in S$. An agent can only control its actions and receive the corresponding reward signal. We refer to policies learned to solve a given task as "task policies" and policies learned to improve the interaction with another agent with a given task as "interaction policies". We do not consider planning on the level of continuous trajectories but rather on a level of categorical actions. Based on this abstraction level, we also assume that our agent learns to solve the other agent's task conceptually. We

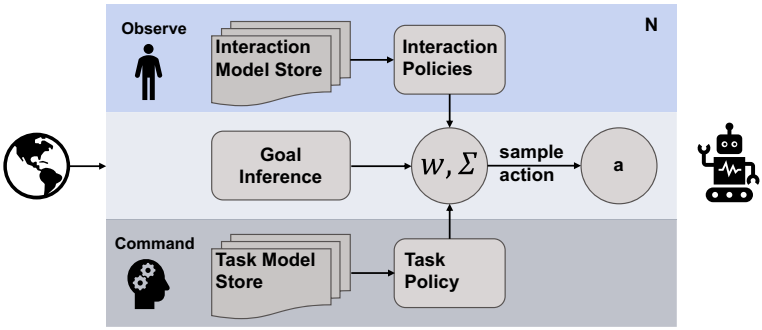

Fig. 2 Architecture of the entropy-based blending framework. It consists of task policies, that are trained to solve the task of the acting agent and the interaction policies model the influence of actions on other agents' possible goals. The goal inference model estimates the distribution over goals for the other agents, which is then used together with an entropy measure between policies to create a combined policy and act based on it

Algorithm 1 Action selection of the entropy-based blending framework.

Task policy: π_t
Interaction policies: $\pi_1, ..., \pi_n : n =$ Number other agents
Goal inference model: θ
for $tin1 : T$ **do**
  Observe state $s \in S$
  Compute action distributions $\pi_t(\cdot|s), \pi_i(\cdot|s) \quad | \quad i \in 1, ..., n$
  Compute goal distributions $g_i = \theta(s) \quad | \quad i \in 1, ..., n$
  Combine action distributions using weights based on (9), (8), (5)
  Sample action a from π_c using (7)
  return a
end for

motivate these assumptions by reasoning that a robot cannot do exactly the same as a human physically due to the limited capabilities of the robot. Instead, our method focuses on being able to reproduce the same goal on an abstract, predefined level, where the robot comes up using simulation with a theoretical plan of how the human would solve a problem. Multiple individually learned task policies will be kept in a task model store, to be activated based on the current task and likewise for interaction policies. Once an agent is deployed, their task and all relevant interaction policies are recombined into a single policy to solve the compound system (Fig. 2) using an entropy-based blending mechanism.

3.1 Learning task policies

We derive our theoretical work based on the maximum entropy reinforcement learning (MaxEnt RL) framework [38, 64]. Maximum entropy policies have many desirable properties, such as good exploration and possibly finding multiple solutions to a problem, something that is desirable in coexistence environments because it allows agents to be flexible in their decision-making. We learn our task and impact policies using PPO and use the connection of policy gradient learning to Maximum Entropy RL. Schulman et al. [65] proves the equivalence of entropy regularized policy gradient methods and Soft Q learning. Due to the usage of PPO with an entropy regularization term, we can view it as a Boltzmann distribution, despite using additional implementation details to stabilize the training process, such as gradient clipping. We show the derivations regarding maximum entropy RL and then show how they can be practically applied to PPO policies when discussing the combination of the policies. 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. In PPO the entropy is added as an additional loss term, which is maximized similarly. The optimal policy function for MaxEnt RL with finite horizon T is

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

where temperature parameter α 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 π. The optimal policy π can be estimated by a Q-function as an energy-based model (EBM) [38]

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

with the Q-function taking on the role of negative energy. The resulting Boltzmann policy is shown to be similar to policies produced by policy gradient algorithms such as PPO. The Q-values are the basis for drawing samples from a Boltzmann distribution, which offers the advantage of being able to derive the distribution we are sampling from analytically. The adoption of this method constrains our work to discrete action settings, such as robots using an action library. The method is applicable to any environment, where policies that output categorical distributions for action sampling can be trained. Following [18, 38] one can define the optimal soft Q as

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

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

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 as

$$I^i_j(s_t, a^i_t) = \mathbb{E}_{s_{t+1:T}\sim\rho_{\pi_j}, a^j_l\sim\pi_j(s_l)}\left[\sum_{l=t}^{T}\gamma^l\, r^j_l(s_l, a^i_l, a^j_l) \;\middle|\; a^i_{t+1:T} = a^i_0\right] - \mathbb{E}_{s_{t+1:T}\sim\rho_{\pi_j}, a^j_l\sim\pi_j(s_l)}\left[\sum_{l=t}^{T}\gamma^l\, r^j_l(s_l, a^i_0, a^j_l)\right].$$

where we assume a finite horizon 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, where the actions a^j and a^i are made simultaneously. We measure only the impact this one intervention action has by setting all hypothetical subsequent actions not to change the world state. This formulation quantifies the marginal contribution of agent i's action to the expected future reward of agent j, capturing how much agent j's expected performance changes due to agent i's decision. As such we rewrite the impact in terms of the value function of agent j

$$I^i_j(s_t, a^i_t) = r^j(s_t, a^i_t, a^j_t \sim \pi_j(s_t)) + V^j(s_{t+1}) - V^j(s_t),$$

where $P(s_{t+1}|s_t, \boldsymbol{a}_t)$. Computing the impact requires estimating the value function of the given task of agent j. We assume the agent knows how to solve the other agent's task as the agent has learned all tasks that other agents might be doing and that we want to be able to assist later. During self-play later this assumption allows us to initialize the other agent with previously learned task policies. We take the other agent's perspective and use our internal Q-model to estimate the values with the internal Q model corresponding to the task that agent j is currently pursuing. Using this impact measure, the agent learns an impact-aware policy in the maximum entropy fashion, where the reward is now given by I.

We formulate our impact training objective as

$$J_i(\pi) = \sum_{j=1, j\neq i}^{N}\sum_{t=0}^{T-1}\mathbb{E}_{(s_t, \mathbf{a_t})\sim\rho}[I(s_t, a^i_t) + \alpha\mathcal{H}(\pi_i(\cdot|s_t)]. \tag{3}$$

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

$$Q_i(s,a) \leftarrow I(s,a^i) + \gamma \mathbb{E}_{s' \sim p(s'|s,\mathbf{a})} \left[V_i(s') \right].$$

The agent uses self-play to learn the impact models, eliminating the need to train with others with policies unknown to the agent.

3.3 Learning a goal inference model

In order to compute the impact that one agent has on the other agent, we need to know which goal the other agent is currently pursuing and learn a Goal Inference (GI) model. Since agents in a coexistence environment can interact on the fly without prior communication, the goal of the other agent might not be known a priori and need to be estimated. We propose a dedicated model to infer the goal for each other agent in the current environment, which is learned from past episodes. We focus inference on the goal g_j pursued by a given agent j. The model predicts a probability distribution $p(g)$ over all n_g possible discrete goals $\mathcal{G}$ given a sequence of the past states of the environment $s_{t,t-1,t-2}$. We record episodes with agents solving their tasks in an environment and train on a fully connected neural net on the goal-labeled data with the cross-entropy loss

$$L(\theta_{\mathrm{GI}}) = -\sum_{g \in \mathcal{G}} y_g \log p(g|s_{t,t-1,t-2}; \theta_{\mathrm{GI}}).$$

We train to minimize the average loss

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(\theta_{\mathrm{GI}})_i. \tag{4}$$

The resulting distribution

$$p(g|s_{t,t-1,t-2}; \theta_{\mathrm{GI}}) = \mathrm{Softmax}(\theta_{\mathrm{GI}}(s_{t,t-1,t-2}))$$

can be used as a weight factor for each present agent for every task as a weight vector

$$w_g = p(g|s_{t,t-1,t-2}; \theta_{\mathrm{GI}}). \tag{5}$$

During deployment, the model selects the task policy corresponding to its goal, and for each agent in the scene, the impact-aware interaction policies match the respective agents' goal distribution as computed by the goal-predictive model.

3.4 Recombination of policies

Our framework's final and integral building block is the combination of the resulting distributions from interaction and task policies. We propose a blending mechanism that bounds the experienced regret of blending. In Fig. 3 we depict the weighting for a simple case with only one interaction policy. Combining policies is also known as policy blending and prior work has shown that compositionality arises naturally when using maximum entropy reinforcement learning [40] and that this approach is also known as a product of experts [41, 42]. Policy blending enables a clear separation between the policies directed toward other agents and those 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. In the following equations,

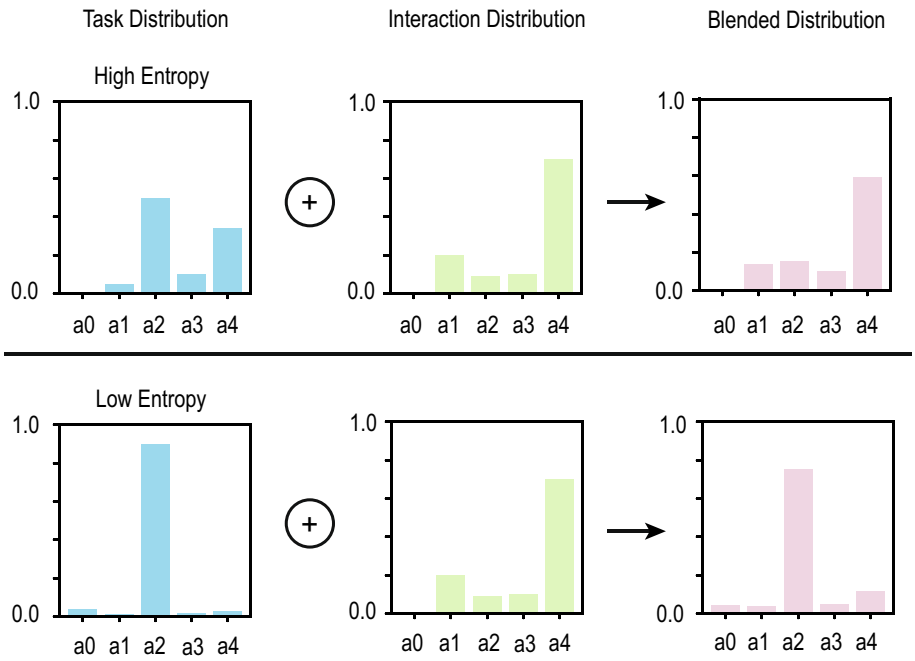

Fig. 3 Visualization of the weight blending. Example of two different distributions produced by task policies with the same distribution by an interaction policy. The combined policies blend according to the entropy of the task distribution

we show the composition based on Q-values but note that this can be done in an analog manner for categorical distributions to sample actions from a policy gradient method such as PPO by establishing that the sum of Q-values at any inference step for every network is normalized. We 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 $Q_{g,i}$

$$Q_C = w_t Q_t + (1 - w_t) \sum_{i \in I} \sum_{g \in \mathcal{G}} \frac{w_{i,g}}{\sum_{j \in I} \sum_{k \in \mathcal{G}} w_{j,k}} Q_{g,i}. \tag{6}$$

The weighting of q-values aims to reduce the negative impact of incorporating interaction-aware elements into the task policy. When using policy gradients that directly compute a distribution over actions, we need to adapt the combination of policies. By taking (2) we derive that

$$\begin{aligned} \exp(Q_c) &= \exp\left(\sum_n^N w_n Q_n\right) = \prod_n^N \exp(w_n Q_n) \\ &\propto \prod_n^N \pi_n^{w_n} = \exp\left(\sum_n^N w_n \log(\pi_n)\right). \end{aligned} \tag{7}$$

Thus, we conclude that we can directly sum the weighted policies in log space and that they are not bound to value-based methods through the previously mentioned equivalence in resulting action distributions. The objective of our weighting scheme is to automatically find a good trade-off between the focal agent's reward and the reward of the other agents. 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 using the entropy's complement $H \in [0, 1]$ to weigh the task distribution π_t and the compound interaction distribution $\pi_c \propto Q_c$. First, we introduce a lower bound of the policy blending process using an arbitrary weight $w \in [0, 1]$, extending the proof of [40].

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 A. □

Corollary 1 *Following Lemma 1 we can deduce that the regret of using policy π_2 instead of policy π_1 is proportional to the Rényi divergence $K(\pi_1, \pi_2) \propto \mathcal{D}_w(\pi_1 || \pi_2)$.*

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

$$\mathbb{E}_{\pi_c \sim Dir(1,1,\ldots,1)} [JSD(\pi_t || \pi_c)] \propto -\mathcal{H}(\pi_t).s$$

This establishes that we have a regret term that is bound by the Rényi divergence of the two policies. In the following section, we use the fact that the JSD is the Rényi divergence with order 2, and since the Renyi divergence is strictly increasing with its order [66], the JSD is an upper bound to the previously established bound of the regret. We continue to 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

$$K(\pi_t, \pi_c) \leq \mathbb{E}_{\pi_c \sim \mathrm{Dir}(1,1,\ldots,1)} [\mathrm{JSD}(\pi_t || \pi_c)] \propto -\mathcal{H}(\pi_t),$$

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 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. The entropy of a policy encodes the uncertainty of which action to take and lets us 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 inversely proportional to the entropy. Therefore, we propose to compute the weight w_t of the task policy depending on the entropy

$$w_t = 1 - \mathcal{H}_{|A|}(\pi_t), \tag{8}$$

where $|A|$ is the number of actions in the action space. 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 π_i as the JSD to the task distribution times the respective goal's assumed probability as computed by the goal predictive model

$$w_{i,g} = \sum_{g \in \mathcal{G}} w_g \text{JSD}(\pi_t || \pi_{i,g}). \tag{9}$$

These weights are then used in (7). 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 and obtain categorical distributions. The weighting of distributions is done using the entropy and JSD between the task action distribution and the impact action distribution. Finally, the distributions are combined as a weighted sum, and an action from the resulting categorical distribution is sampled.

3.5 Training procedure

We propose to use PPO as a training algorithm but emphasize that any algorithm, that produces a maximum entropy policy and can learn the tasks at hand, is compatible with the framework. 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. The agent optimizes its positive impact according to the impact reward formulation in (3). The interaction policies are updated using the same methodology as is standard for PPO. For optimization we use the ADAM optimizer [67] with weight decay. 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 the focal agents' performance and the team's performance.

We use two fully connected layers with 64 units for each policy in the particle environment and an additional CNN layer with 3x3 kernels and 16 channels in the Level-Based Foraging and cooking-zoo environment. In each environment, we use a policy head with the number of actions as output. We train on 6000 episodes in the cooking environment, 1000 in the Level-Based Foraging environment, and 200 in the particle environment for each task.

4 Experimental evaluation

We test our framework in coexistence and cooperative environments, namely a navigation multi-particle environment [68], a cooking environment[1], and a gathering environment [69]. We use CookingZoo and show the scalability of our method to complex environments and tasks, even when the intentions of others are unknown. We analyze the model's performance compared to the ablation of our blending methods and the baseline.

CookingZoo Previously established cooking environments were limited to a single joint goal of cooperating agents, modeling only strictly cooperative tasks, whereas our environment allows each agent to have separate recipes. Our environment supports the latest versions of the gymnasium[2] and pettingzoo[3] libraries, enabling easy usage of common RL frameworks.

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

[2] https://gymnasium.farama.org/

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

We support different reward schemes with rewards for sub-goals or 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 dish racks). Agents can move in all four directions or stand still and all agents move simultaneously. Moving against a counter or tool picks up, puts down, or uses an object or tool. An episode is concluded once all agents have delivered their recipe or after a defined amount of time steps have elapsed. Positive rewards are given either only for completing recipes or for each correct step within a recipe (such as cutting the correct ingredient). In this work, we used the latter scheme. In addition, each time step/action comes with a small negative reward. Finally, we penalize actions that revoke recipe progress (i.e., placing a wrong ingredient on a plate along with the correct ingredients). The observation space is represented by a tensor that consists of stacked layers containing information about each object type, where the layers have the dimensions of the grid.

Figure 4 depicts the cooking setup used in our experiments and two possible recipes. The environment is divided into two separate spaces to focus on interactions through active actions (placing an ingredient) and avoid other influences such as collision. One space is for the focal agent that is separately trained according to the corresponding training regime, and the other is on the right for the other agents that are task-specific trained PPO agents. A specification of the hyperparameters and network used during training can be found in Table 1. We sample during training of the task policies from all valid starting positions, placing any ingredient anywhere reachable for that agent if it is needed for its recipe. The

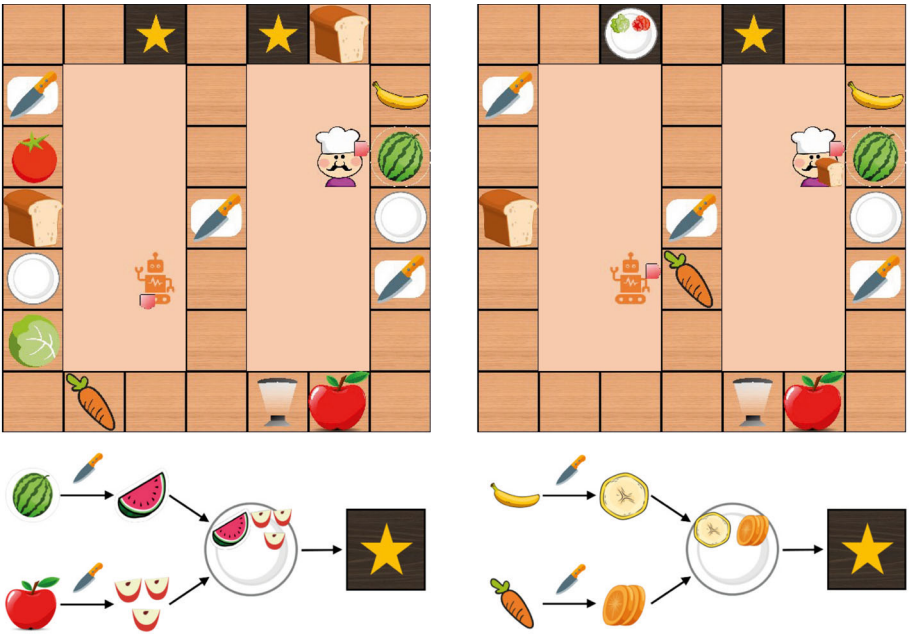

Fig. 4 The cooking environment used in the experiments. CookingZoo is a sparse reward environment with distinct goals in the form of recipes. Recipes require a combination of ingredients, which might have to be processed using tools. Two examples are shown in the lower part of the figure. The placement of tools, ingredients, and layout are freely configurable to create diverse scenarios. Two example states are shown with separate workspaces for two agents. While the human (right) picks up bread, the robot (left) moves the carrot to a space accessible by the human

Table 1 Customized hyperparameters for proximal policy optimization

Hyperparameter	Description	Standard Value
γ	Discount factor	0.99
λ	GAE (Generalized Advantage Estimation) parameter	0.95
ϵ	Clipping parameter	0.2
K	Number of epochs per PPO update	10
T	Number of steps per environment rollout	2048
N	Number of actors (parallel environments)	1
α	Learning rate	3×10^{-4}
c_1	Value function loss coefficient	0.25
Entropy start	Initial entropy coefficient	0.1
Entropy end	Final entropy coefficient	0.001
Annealing steps	Entropy coefficient annealing steps	4000
Batch size	Size of batch sampled from buffer	100
Buffer size	Size of experience buffer	4000
Gradient clipping	Maximum gradient norm	0.3
Weight decay	Use of L2 regularization	0.0001
c_2	Entropy loss coefficient	0.01

design of the counter and tools remains fixed. For training interaction policies and evaluation, we move one ingredient the agent on the right side needs to the left side, making it dependent. On the left side, there are three possible recipes to be completed, while on the right side, there are four recipes to be completed, since there are more agents on that side during our experiments. The structure of the recipes is depicted in Fig. 4. An agent has to prepare two different ingredients using the cupboard, place them in any order on a plate, which he has to get from the dish rack, and then put those on a star-marked square. We train one policy for each of the 7 recipes available in our framework's environment. Subsequently, we train a goal inference (GI) model to recognize the recipe other agents are cooking. At last, we train the interaction policies for the recipes. The training plots of the policies Fig. 5. To evaluate our system we look at scenarios with two, three, and four agents present. The position of the tools is fixed, while the ingredients are randomly sampled, with the only restriction during evaluation being that one ingredient per agent on the right side is only accessible to the left

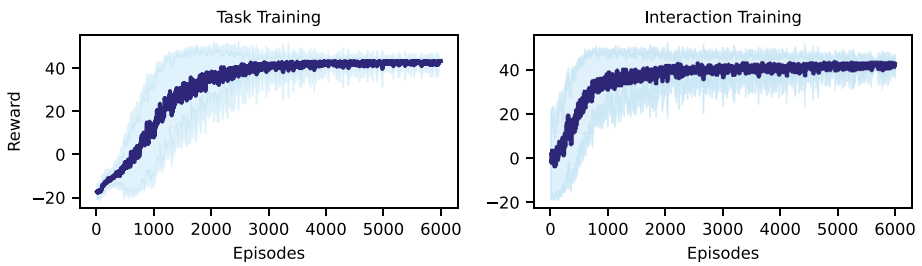

Fig. 5 Average reward of the task learner on the left and the interaction learner on the right during training plotted over the number of training episodes in the cooking environment for all recipes using 10 random seeds. The shaded area shows the standard deviation

agent and that all required ingredients of the left agent are reachable. This allows us to test only relevant scenarios, where an interaction of the focal agent is useful and necessary, and no symmetric reliance on others is ensured.

Particle Navigation We evaluate a target area reaching task on the particle environment as seen in Fig. 6, as navigating in a space with multiple other agents present is one of the most prominent tasks anyone has to solve when roaming in the real world. We explore different combinations of target areas for our agents to reach. Compared to the original, the twist in our environment version is that agents are additionally rewarded based on the others' position, such that, within an agent's target area, there is an optimal area for the other. Starting positions are randomly sampled within the entire area. The reward function is the distance to its goal area summed with the others' distance to the second area times -1 to penalize being further away. The observation space is a tensor consisting of four layers, where each layer is a grid over the world space and contains information about the speed or position of an agent at that position.

We train the task and interaction policies on 35 goal areas and their corresponding interaction areas in separate training processes. We rasterize the environment to have a sufficient number of tasks available in a single environment to be able to evaluate how a system scales with an increase in the number of available tasks. As in the cooking environment, we trained a goal inference model that predicts a distribution over the expected goal area. We conduct experiments on 5 scenarios with four other agents present and compare the results of our framework with and without entropy weighting with a joint learning system, which is learned on the final tasks with four other present agents.

Level-Based Foraging In the LBF environment [69], agents and apples are placed on an 8x8 grid as shown on the right in Fig. 6. This environment is a cooperative environment as all agents have the same goal. This environment allows us to compare the performance in a previously established domain. Apples have levels 1, 2, or 3. The main objective for the agents is to gather all the objects on the grid. Agents can move in any of the primary four directions, remain stationary, or collect objects that are next to them. To successfully collect an object, the combined levels of the agents trying to collect it must be at least equal to

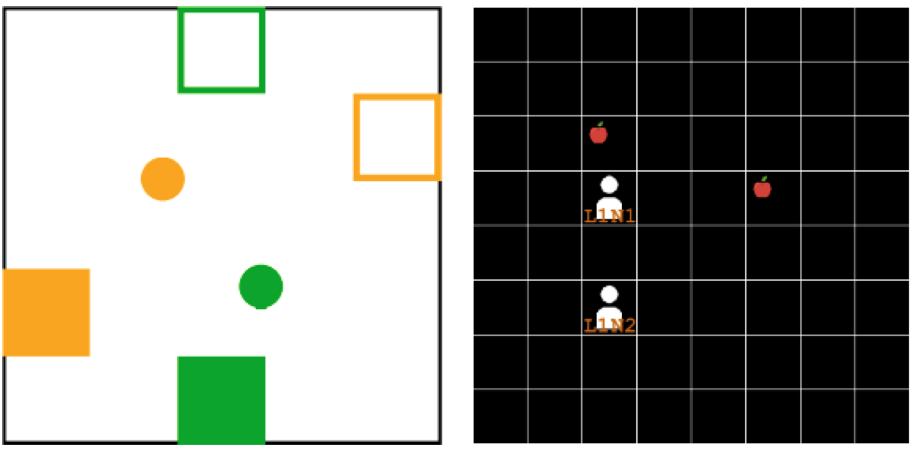

Fig. 6 The particle (left) and level-based foraging (right) environments

the object's level. When an agent gathers an object, it earns a reward corresponding to the object's level. An episode concludes once all objects are collected, or after the maximum number of time steps (400) are exceeded. We add a step penalty to the original environment to encourage faster completion of the task. The observation space is a tensor consisting of four layers, where each layer is a grid over the world with either the level information of an agent or the level information of a fruit type.

We train a baseline agent to solve the task with pre-trained agents on the joint reward, as well as an agent using our proposed system, who gets a separate reward for collecting apples himself for the task policy and the interaction policy only gets rewards for apples the other agent collects. We also compare our method in the cooperative to the state-of-the-art baseline GPL [36].

4.1 Evaluation of coexistence environments

To answer our research questions we first show the performance in coexistence environments. We compare our proposed entropy-based blending method to a baseline joint learning model and the maximum performance of a single-agent learner. An episode for an agent counts as completed once its recipe has been finished or the target location has been reached. The joint reward learner explicitly optimizes for the combined reward of the team and shows potential gaps in either one's own completion rate or those of others.

The experiments were conducted using ten different random seeds, with each seed sampling 100 environment setups for three different task combinations each. The primary metric for evaluation was the completion rate.

We show in Fig. 7 that the entropy-based blending method for both individual agents and the team reward outperformed the joint learning method in the cooking environment. Notably, the joint learning method demonstrated an inability to learn effective coexistence strategies, performing much worse than the entropy-based blending method.

In the particle environment, similar trends can be observed. The entropy-based method surpassed the joint learning model, suggesting its effectiveness scales with an increased number of agents. While performing better than in the cooking environment, the joint learning method did not match the entropy-based blending method's performance. The entropy-based method and the joint reward learner were able to perform their own tasks at a high rate. In the particle environment, both approaches perform nearly the same as in a dense reward

Fig. 7 Average completion rate over 3000 testing episodes on 10 training seeds of the entropy-based blending approach and the joint rewardlearner in the cooking and particle environment with a single other agent present. The red line shows the average completion rate of an agent acting only based on its task policy

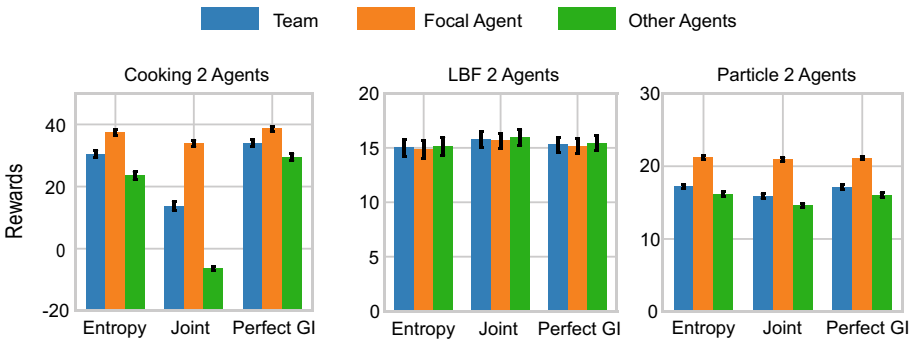

Fig. 8 Average reward over 3000 testing episodes on 10 training seeds of the entropy-weighted blending, the joint learning, and the perfect goal inference with entropy blending with task uncertainty

environment. In the cooking environment, the entropy-based method outperforms the joint reward learner, which we found to be significant when using a two-sample t-test ($p = 0.0$) for the team performance results. The performance of both approaches is close to an ego agent, indicated by the red line in both plots, showing that we do not have to sacrifice to achieve our goals in order to coexist. The results are within expectation, except for the joint reward learner not being able to learn a meaningful joint policy that helps the other agent. The joint reward agent does not discover the optimal joint solution but only finds suboptimal local solutions, which benefit either himself or others, highlighting the difficulty of the exploration problem with off-the-shelf RL methods. This highlights the complexity of the credit assignment problem [70–73], when other actors cause a reward signal and when the reward of a good interaction is delayed, as in the sparse reward cooking environment.

4.2 Evaluation with task uncertainty

In the next evaluation, we tested whether our framework can effectively handle the uncertainty of which tasks others are doing. We compared our model's performance with and without perfect goal prediction and the joint learning strategy, which has access to the other agent's tasks. The comparisons were made in the cooking and particle navigation environment, as shown in Fig. 8. The accuracy of our goal inference in the cooking environment is shown in Fig. 9. At the start of the episode, when little information is available about the other agent,

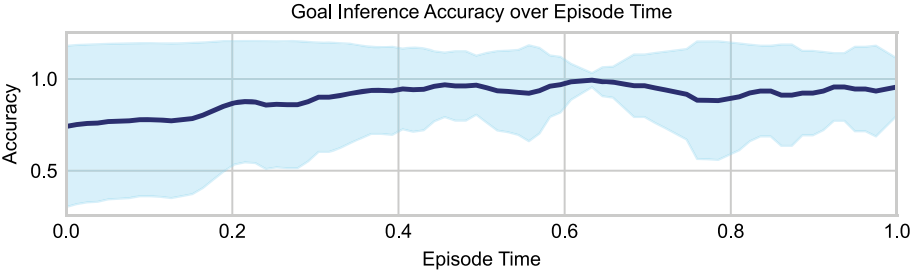

Fig. 9 Goal inference accuracy over episode time in the cooking environment evaluated on 3000 testing episodes on 10 training seeds using one agent to interact with. The episode times are normalized between 0 and 1 to account for episodes of different lengths. The shaded area is the standard deviation from the mean

accuracy is lowest and then increases over time until about 60% of the episode time. This shows that meaningful interaction is only possible after some time, and opportunities to have a positive impact, especially at the start of an episode, might be missed. The decrease in the later half of the episode in accuracy can be attributed to the other or focal agent not being able to finish the episode and something unexpected happening, hindering task inference. We use the same setup for the first results but use reward as a metric to highlight differences in total performance.

In the cooking scenario, our framework with the entropy-based combination performed very well, even when compared to the version with perfect goal prediction, with only slight differences in the achieved reward. As observed before, the joint learning model did not perform as well, failing to learn effective ways for the agents to interact and collaborate while completing tasks. In the particle environment, the differences between all the models were minor, with the joint learning and perfect prediction module showing only a small 0.08% improvement over our framework. This suggests that having a perfect prediction or complete information in these environments does not significantly impact the overall performance. The only noticeable difference was found in the cooking environment, where a t-test showed that the difference between the perfect goal prediction and our entropy-based model was significant($p = 0.02$).

Overall, the results confirm that our framework is capable of handling task uncertainty effectively. The improvements that a perfect goal inference yields in performance are minor, underlining the stability of our approach. We also see that a joint reward learner, even with complete information, does not always lead to better outcomes, especially in complex tasks, where good interactions earn delayed rewards.

4.3 Scaling to multiple agents

This section examines the scalability of the entropy-based blending approach when the number of agents increases. We extended our evaluation to include setups with four agents in the cooking environment and five in both the particle navigation and LBF environments. In the LBF environment, we additionally compared against a state-of-the-art method for cooperative environments graph policy learning (GPL) [36]. Results are presented in Fig. 10.

In the coexistence tasks represented by the cooking and particle environments, our method demonstrated an ability to outperform the joint learning approach, reinforcing the scalabil-

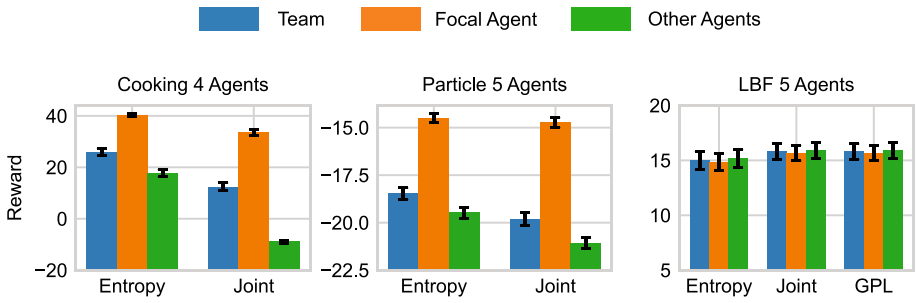

Fig. 10 Average reward over 3000 testing episodes on 10 training seeds of the entropy blending, the joint learning, and the perfect goal inference approach reached during evaluation in the cooking, particle, and level-based foraging environment with 4, 5 and 5 agents respectively

ity of our entropy-based blending method when handling multiple agents. The observed effectiveness is attributed to our system's dynamic adaptation to the increased interaction complexity that more agents introduce.

Contrastingly, in the cooperative LBF environment, our approach showed a lower performance compared to GPL and the joint reward learner, which we found to be significant ($p = 0.0$). The decrement in efficiency can be attributed to the higher entropy levels within the agents' policy distributions in our framework, as seen in Fig. 3. The increased entropy caused slower execution times during the task execution phase, subsequently affecting the overall reward.

The inherent increase in entropy is a trade-off for the enhanced exploration and robustness against uncertainty in agent behaviors. While this trait is advantageous in scenarios requiring diverse solution strategies, it becomes a limitation in environments where the speed of execution is directly correlated to the reward, such as the LBF setup.

The joint learning approach fared well in the LBF environment, where the agent's individual success naturally aligns with the team's performance. In such contexts, the direct coupling of individual rewards to collective success proves beneficial, as evidenced by the superior results of the joint reward learner when paired with GPL.

The results confirm that our entropy-based blending framework can effectively scale to a larger number of agents in multi-agent systems. Even though there is a performance trade-off in cooperative environments where execution speed is crucial, our system maintains competence in solving the given tasks. These findings suggest that our framework, with its ability to balance exploration and exploitation, holds promise for diverse multi-agent applications, especially in complex coexistence scenarios. Future work may explore optimizing the entropy levels to enhance execution speed without compromising the quality of exploration.

4.4 Effects of entropy-based weighting

We establish whether an entropy-based weighting approach for combining interaction and task policies in multi-agent environments has empirical benefits over an equal weighting scheme. Again, we tested the entropy-based blending across the three environments: cooking, particle navigation, and LBF. The findings, illustrated in Fig. 11, show the results of our entropy-based method in comparison to the equal-weighting baseline.

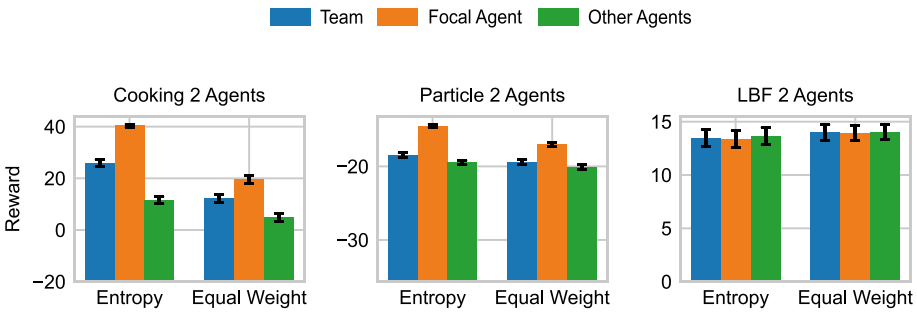

Fig. 11 Average reward over 3000 testing episodes on 10 training seeds of the entropy-weighted and equal-weight blending in the cooking, particle, and level-based foraging (LBF) environments

The cooking and particle navigation environments show a performance difference between the entropy-based and equal weighting approaches. Empirically, the entropy-based method accomplishes its task objectives with a higher reward and demonstrates, in addition, more efficient interaction with other agents. The faster execution times allow the agent additional time to assist others, further compounding the advantage of the entropy-based blending. Conversely, the equal weighting method sometimes encounters situations where interaction and task policies command the agent to go in different directions, leading to indecision or suboptimal actions.

In contrast, within the purely cooperative LBF environment, our analysis indicates no statistically significant performance difference between the entropy-based and equal weighting schemes. This outcome aligns with our expectations, as the LBF's cooperative nature renders the policy alignment less impactful on the overall performance. A possible reason is that the challenge within the LBF environment lies in the collective discovery of an effective strategy rather than the reward structure. The reward structure causes the optimal strategy for a single agent to be closely related to the optimal strategy of an agent. This is in strong contrast with our coexistence environments, where the discovery of strongly coordinated behavior is not necessary but the reward structure inherent to the problem makes joint learning hard.

5 Conclusion & future work

This paper introduced a formulation and methodology to systematically learn a framework of task and interaction policies separately from each other and to combine them under task uncertainty using an entropy-based blending. We showed that we can learn these policies and a goal inference model for task uncertainty. The interaction policies worked with more than one other agent and increased the group's overall performance in multiple environments across various tasks. Our entropy-based policy blending shows that our system keeps the regret drastically lower than a simple policy averaging approach. Despite never seeing the test distribution of tasks with specific other agents, our framework achieved good results in all tested scenarios even with multiple interaction partners. We demonstrated the extrapolation ability by recombining policies to new task combinations, allowing our system to scale much more favorably than a joint training approach while keeping and even exceeding performance. Experiments demonstrated an ability to scale to a complex, sparse environment with multiple agents. Our goal inference model performed reasonably and supported finding good interaction policies with increased accuracy over time. Our comparison to established methods using joint learning and GPL in a cooperative environment highlights that we can learn cooperative behavior and that the difficulties one encounters in coexistence challenges lie in the reward structure of the presented tasks. Our work is limited to discrete action spaces because the computation of the entropy factor requires a bounded range. Additionally, our method is constrained to environments where tasks can be done independently or when coordination with others does not exceed one additional player, since we trained with only one partner in our experiments. This limitation could be worked on in future work by incorporating policies that know how to interact with a group in environments where coordinated actions of groups larger than two agents are crucial. Another possible extension of this work could extend the approach to account for different types of co-existing agents, particularly human players, to establish robustness.

Appendix A: Proof of Lemma 1

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 We will prove Lemma 1 by induction on the number of Bellman updates and by using properties of soft Q-functions. The idea is to show that the soft Q-function for the combined reward r_C can be bounded by the weighted sum of the individual Q-functions Q_1^* and Q_2^*, with an error term that depends on the divergence between the policies π_1^* and π_2^*.

Step 1: Base Case $k = 0$

For the base case (no Bellman updates), we start with $Q_C^{(0)}(s,a) = Q_\sum(s,a) = wQ_1^*(s,a) + (1-w)Q_2^*(s,a)$. This holds trivially by definition of $Q_\sum$.

Thus, at $k = 0$,

$$Q_\sum(s,a) = Q_C^{(0)}(s,a).$$

Step 2: Inductive Hypothesis

Assume that after k Bellman updates, the following inequality holds:

$$Q_\sum(s,a) \geq Q_C^{(k)}(s,a) \geq Q_\sum(s,a) - C^{(k)}(s,a),$$

where $C^{(k)}$ is the cumulative regret up to step k.

Step 3: Bellman Update and Inductive Step

We apply the soft Bellman update at step $k+1$. The soft Q-function for the combined reward r_C after one more Bellman update is:

$$Q_C^{(k+1)}(s,a) = r_C(s,a) + \gamma \mathbb{E}_{s' \sim p(s'|s,a)} \left[\log \sum_{a'} \exp(Q_C^{(k)}(s',a')) \right].$$

For the weighted sum of soft Q-functions $Q_\sum$, after a Bellman update, we have:

$$Q_\sum^{(k+1)}(s,a) = r_\sum(s,a) + \gamma \mathbb{E}_{s' \sim p(s'|s,a)} \left[\log \sum_{a'} \exp(Q_\sum^{(k)}(s',a')) \right],$$

where $r_\sum = wr_1 + (1-w)r_2$. Since the soft Bellman update is a contraction, we know that the updated soft Q-function $Q_C^{(k+1)}(s,a)$ is bounded by:

$$Q_\sum^{(k+1)}(s,a) \geq Q_C^{(k+1)}(s,a).$$

Thus, we have the upper bound:

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

Step 4: Lower Bound with Divergence Term

To derive the lower bound, we look at the gap between the weighted sum $Q_{\sum}^{(k+1)}$ and the optimal Q-function $Q_C^{(k+1)}$. This gap is due to the divergence between the individual policies π_1^* and π_2^*. The regret from using $Q_{\sum}$ instead of the optimal Q_C^* can be bounded by the expected divergence between the two policies π_1^* and π_2^*. Specifically, the gap is proportional to the Rényi divergence $\mathbb{D}_w(\pi_1^*||\pi_2^*)$, which measures the difference between the two policies. Thus, the lower bound can be written as:

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

where $C^*(s,a)$ is the fixed point of:

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

Since we can switch the order of the probability distributions π_1^* and π_2^* as part of the calculations, and the weight ω controls their contribution to the combined policy, we always choose the order where ω is the lower value. Specifically, we calculate the Rényi divergence $\mathbb{D}_\omega(\pi_1^*||\pi_2^*)$ using the smaller weight, which ensures that the contribution of the divergence remains bounded.

Given that ω is restricted to the range [0, 1] and the maximum value for the smaller ω occurs at 0.5, we can further guarantee that the Rényi divergence, which is a non-decreasing function of ω, will bound the regret between the policies within the interval [0, 0.5]. This is because, as ω approaches 0.5, the divergence reaches its highest value, and beyond this point, the contribution of one policy over the other diminishes.

Thus, by always selecting the smaller ω, we limit the regret to this bounded range, ensuring that the blending of the two policies remains stable. This property of the Rényi divergence provides a safeguard against excessive regret as the policies are combined, keeping the performance loss within manageable bounds.

Appendix B: Proof of Lemma 2

The expected Jensen-Shannon distance to a fixed policy π_t for a policy π_c drawn by a Dirichlet process DP with a non-informative prior is proportional to the negative entropy of the policy π_t. π_t is the task policy and π_c is the compound policy.

$$\mathbb{E}_{\pi_c \sim \text{Dir}(1,1,\ldots,1)} [\text{JSD}(\pi_t||\pi_c)] \propto -\mathcal{H}(\pi_t).$$

Following we prove lemma 2. We consider the Δ^{n-1} simplex for categorical distributions, which represents all possible probability distributions over n outcomes. We aim to show that the expected Euclidean distance from any point on the simplex to all other points is minimized at the centroid of the simplex. For simplicity, we choose the Euclidian distance and note that this proof directly applies to all geometric distance measures (such as the JSD).

The probability simplex Δ^{n-1} is defined as

$$\Delta^{n-1} = \left\{ (p_1, \ldots, p_n) \in \mathbb{R}^n : \sum_{i=1}^{n} p_i = 1,\ p_i \geq 0 \, \forall i \right\}.$$

The Euclidean distance between two points x and y on the simplex is given by

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}.$$

The centroid c of the simplex is the point where each coordinate is equal, representing the uniform distribution

$$c = \left(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}\right).$$

The expected squared distance from a point p to a random point X uniformly distributed over Δ^{n-1} is

$$E[d^2(p, X)] = \sum_{i=1}^{n} E[(p_i - X_i)^2] = \sum_{i=1}^{n}(p_i^2 - 2p_i E[X_i] + E[X_i^2]),$$

where $E[X_i] = \frac{1}{n}$ given the uniform distribution of X over Δ^{n-1}. By substituting $p = c$ into the expected squared distance equation, we find

$$E[d^2(c, X)] = \sum_{i=1}^{n}\left(\frac{1}{n^2} - \frac{2}{n^2} + E[X_i^2]\right),$$

which simplifies to a minimal value, showing that the centroid minimizes the expected squared distance to all points on the simplex.

Now that we have established that the distribution that minimizes the expected JSD on a probability simplex given a Dirichlet distribution with all $\alpha_i = 1$, we only need to show that the JSD of a distribution a and the uniform distribution is proportional to the entropy of a.

Proof Assume a and b are two categorical distributions over the same finite set, S, with c as the uniform distribution over S representing the maximum entropy distribution and $M = \frac{1}{2}(p + q)$. We prove that if $H(a) > H(b)$, then $\text{JSD}(a||c) < \text{JSD}(b||c)$. The mixed distribution M when comparing to the uniform distribution c is

$$M = \frac{1}{2}(p + c).$$

Given $c_i = \frac{1}{n}$, we have

$$M_i = \frac{1}{2}\left(p_i + \frac{1}{n}\right).$$

The KL divergence $D_{KL}(p||M)$ simplifies to

$$D_{KL}(p||M) = \sum_{i \in S} p_i \log \frac{p_i}{M_i} = \sum_{i \in S} p_i \log \frac{p_i}{\frac{1}{2}(p_i + \frac{1}{n})}.$$

And similarly for c to M

$$D_{KL}(c||M) = \sum_{i \in S} \frac{1}{n} \log \frac{\frac{1}{n}}{M_i}.$$

Thus, JSD for p and c becomes

$$\text{JSD}(p||c) = \frac{1}{2}\left(\sum_{i \in S} p_i \log \frac{p_i}{\frac{1}{2}(p_i + \frac{1}{n})} + \sum_{i \in S} \frac{1}{n} \log \frac{\frac{1}{n}}{\frac{1}{2}(p_i + \frac{1}{n})}\right).$$

Both components involve expressions dependent on the values of p_i relative to $\frac{1}{n}$. As p becomes more uniform (i.e., p_i approaches $\frac{1}{n}$ and hence its entropy $H(p)$ increases), both terms in the JSD expression decrease because the partial second derivates of p_i increases. This shows that given the JSD function

$$\mathrm{JSD}(p||c) = \frac{1}{2}\left(\sum_{i\in S} p_i \log \frac{p_i}{\frac{1}{2}(p_i + \frac{1}{n})} + \sum_{i\in S} \frac{1}{n} \log \frac{\frac{1}{n}}{\frac{1}{2}(p_i + \frac{1}{n})}\right),$$

the overall second derivative is

$$\mathrm{JSD}''(p_i) = \frac{1}{2}(f''(p_i) + g''(p_i)),$$

where

$$f''(p_i) = \frac{1}{n}\frac{1}{p_i(p_i + \frac{1}{n})} + \frac{1}{(p_i + \frac{1}{n})^2},$$

$$g''(p_i) = \frac{1}{2}\frac{1}{(p_i + \frac{1}{n})^2}.$$

The second derivative of JSD increases as p_i diverges from $\frac{1}{n}$, indicating increased sensitivity and variability in the divergence as probabilities move away from uniformity. Thus we conclude that if $H(a) > H(b)$, then a is closer to uniform than b, meaning the terms involving a in the JSD expression are smaller than those involving b as demonstrated above. Therefore, $\mathrm{JSD}(a||c) < \mathrm{JSD}(b||c)$. This result confirms the principle that higher entropy in a distribution leads to a smaller JSD to the uniform distribution, indicating closer similarity.

Author Contributions David Rother and Thomas Weisswange prepared the draft of the manuscript; All authors contributed to the conception of this work. David Rother, Thomas Weisswange, Fabian Kalter, and Franziska Herbert contributed to the architectural design. David Rother built and test the framework; David Rother, Thomas Weisswange, Dorothea Koert, Joni Pajarinen, and Jan Peters reviewed the Manuscript. The work was supervised by Thomas Weisswange and Jan Peters.

Funding Open Access funding enabled and organized by Projekt DEAL. Joni Pajarinen acknowledges funding by the Research Council of Finland (345521, 353198). Dorothea Koert was funded by the German Federal Ministry of Education and Research (project IKIDA 01IS20045).This research is supported by the Honda Research Institute Europe.

Data Availability Reinforcement Learning Environments, which support the findings of this paper are made available under https://github.com/DavidRother/cooking_zoo ; https://github.com/DavidRother/lb-foraging.

Declarations

Competing interests This work was supported by the Honda Research Institute Europe, Germany Dorothea Koert was funded by German Federal Ministry of Education and Research (project IKIDA 01IS20045) Joni Pajarinen was supported by Research Council of Finland (formerly Academy of Finland) (decision 345521) Thomas H. Weisswange is an employee of the Honda Research Institute Europe GmbH.

References

1. Goodrich, M. A., & Schultz, A. C. (2008). Human-robot interaction: A survey. *Foundations and Trends in Human-Computer Interaction, 1*(3), 203–275.
2. Sheridan, T. B. (2016). Human-robot interaction: Status and challenges. *Human Factors, 58*(4), 525–532.
3. Akgun, B., Cakmak, M., Yoo, J. W., & Thomaz, A. L. (2012). 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).
4. Du, G., Chen, M., Liu, C., Zhang, B., & Zhang, P. (2018). Online robot teaching with natural human-robot interaction. *IEEE Transactions on Industrial Electronics, 65*(12), 9571–9581.
5. Carros, F., Meurer, J., Löffler, D., Unbehaun, D., Matthies, S., Koch, I., Wieching, R., Randall, D., Hassenzahl, M., & Wulf, V. (2020). 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).
6. Mast, M., Burmester, M., Graf, B., Weisshardt, F., Arbeiter, G., Španěl, M., Materna, Z., Smrž, P., & Kronreif, G. (2015). Design of the human-robot interaction for a semi-autonomous service robot to assist elderly people. In *Ambient assisted living* (pp. 15–29). Springer.
7. Zhu, J., Gienger, M., & Kober, J. (2022). Learning task-parameterized skills from few demonstrations. *IEEE Robotics and Automation Letters, 7*(2), 4063–4070.
8. Aaltonen, I., Arvola, A., Heikkilä, P., & Lammi, H. (2017). 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).
9. Boysen, N., Fedtke, S., & Schwerdfeger, S. (2021). Last-mile delivery concepts: A survey from an operational research perspective. *OR Spectrum, 43*(1), 1–58.
10. Gonzalez-Aguirre, J. A., Osorio-Oliveros, R., Rodríguez-Hernández, K. L., Lizárraga-Iturralde, J., Morales Menendez, R., Ramírez-Mendoza, R. A., Ramírez-Moreno, M. A., & de Jesús Lozoya-Santos, J. (2021). Service robots: Trends and technology. *Applied Sciences, 11*(22), 10702.
11. Bedaf, S., Gelderblom, G. J., & De Witte, L. (2015). Overview and categorization of robots supporting independent living of elderly people: What activities do they support and how far have they developed. *Assistive Technology, 27*(2), 88–100.
12. Asama, H., Ozaki, K., Itakura, H., Matsumoto, A., Ishida, Y., & Endo, I. (1991). Collision avoidance among multiple mobile robots based on rules and communication. In *IROS* (Vol. 91, pp. 1215–1220).
13. Buehler, M. C., & Weisswange, T. H. (2020). Theory of mind based communication for human agent cooperation. In *2020 IEEE International Conference on Human-Machine Systems (ICHMS)* (pp. 1–6). IEEE.
14. Sendhoff, B., & Wersing, H. (2020). Cooperative intelligence-a humane perspective. In *2020 IEEE International Conference on Human-Machine Systems (ICHMS)* (pp. 1–6). IEEE.
15. Street, C., Lacerda, B., Staniaszek, M., Mühlig, M., & Hawes, N. (2022). Context-aware modelling for multi-robot systems under uncertainty. In *20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2022)* (pp. 1228–1236). International Foundation for Autonomous Agents and Multiagent Systems.
16. Grosz, B. J., & Kraus, S. (1999). The evolution of SharedPlans. In *Foundations of rational agency* (pp. 227–262). Springer.
17. Mirsky, R., Carlucho, I., Rahman, A., Fosong, E., Macke, W., Sridharan, M., Stone, P., & Albrecht, S. V. (2022). A survey of Ad Hoc teamwork research. In *European Conference on Multi-Agent Systems (EUMAS)* (pp. 275–293). Springer International Publishing.
18. Rother, D., Weisswange, T., & Peters, J. (2023). Disentangling interaction using maximum entropy reinforcement learning in multi-agent systems. *26th European Conference on Artificial Intelligence (ECAI 2023)*.
19. Gmytrasiewicz, P. J., & Doshi, P. (2005). A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research, 24*, 49–79.
20. Doshi, P., Qu, X., Goodie, A., & Young, D. (2010). 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).
21. Hoang, T. N., & Low, K. H. (2013). Interactive POMDP Lite: Towards practical planning to predict and exploit intentions for interacting with self-interested agents. In *Proceedings of the 23rd international joint conference on Artificial Intelligence (IJCAI 2013* (pp. 2298–2305).
22. Oroojlooy, A., & Hajinezhad, D. (2023). A review of cooperative multi-agent deep reinforcement learning. *Applied Intelligence, 53*(11), 13677–13722.

23. Christianos, F., Papoudakis, G., & Albrecht, S. V. (2023). Pareto actor-critic for equilibrium selection in multi-agent reinforcement learning. *Transactions on Machine Learning Research*.
24. Yang, Y., & Wang, J. (2020). An overview of multi-agent reinforcement learning from game theoretical perspective. arXiv:2011.00583
25. 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.
26. 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. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '18)* (pp. 2085–2087).
27. Zhang, T., Li, Y., Wang, C., Xie, G., & Lu, Z. (2021). Fop: Factorizing optimal joint policy of maximum-entropy multi-agent reinforcement learning. In *International conference on machine learning* (pp. 12491–12500). PMLR.
28. Yang, J., Nakhaei, A., Isele, D., Fujimura, K., & Zha, H. (2019). CM3: Cooperative multi-goal multi-stage multi-agent reinforcement learning. In *International conference on learning representations*.
29. Omidshafiei, S., Pazis, J., Amato, C., How, J. P., & Vian, J. (2017). Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *International conference on machine learning* (pp. 2681–2690). PMLR.
30. 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).
31. Barrett, S., Rosenfeld, A., Kraus, S., & Stone, P. (2017). Making friends on the fly: Cooperating with new teammates. *Artificial Intelligence, 242*, 132–171.
32. Barrett, S., Stone, P., Kraus, S., & Rosenfeld, A. (2012). Learning teammate models for ad hoc teamwork. In *AAMAS Adaptive Learning Agents (ALA) Workshop* (pp. 57–63). Citeseer.
33. Barrett, S., Stone, P., Kraus, S., & Rosenfeld, A. (2013). Teamwork with limited knowledge of teammates. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 27, pp. 102–108).
34. Melo, F. S., & Sardinha, A. (2016). Ad hoc teamwork by learning teammates' task. *Autonomous Agents and Multi-Agent Systems, 30*(2), 175–219.
35. Eck, A., Soh, L. K., & Doshi, P. (2023). Decision making in open agent systems. *AI Magazine, 44*(4), 508–523.
36. 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.
37. Wilmers, G. (2015). A foundational approach to generalising the maximum entropy inference process to the multi-agent context. *Entropy, 17*, 594–645. https://doi.org/10.3390/e17020594
38. 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.
39. Wang, Z., Zhang, Y., Yin, C., & Huang, Z. (2021). Multi-agent deep reinforcement learning based on maximum entropy. In *2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)* (Vol. 4, pp. 1402–1406). IEEE.
40. 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.
41. Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation, 14*(8), 1771–1800.
42. Le, A. T., Hansel, K., Peters, J., & Chalvatzaki, G. (2022). Hierarchical policy blending as optimal transport. In *5th annual learning for dynamics and control conference* (pp. 211:797–211:812). PMLR.
43. Hansel, K., Urain, J., Peters, J., & Chalvatzaki, G. (2023). Hierarchical policy blending as inference for reactive robot control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 10181–10188). IEEE.
44. Tange, Y., Kiryu, S., & Matsui, T. (2020). Mild action blending policy on deep reinforcement learning with discretized actions for process control. In *2020 59th annual conference of the Society of Instrument and Control Engineers of Japan (SICE)* (pp. 587–592). IEEE.
45. Singh, S., & Heard, J. (2023). Probabilistic policy blending for shared autonomy using deep reinforcement learning. In *2023 32nd IEEE International conference on robot and human interactive communication (RO-MAN)* (pp. 1537–1544). IEEE.
46. Dragan, A. D., & Srinivasa, S. S. (2013). A policy-blending formalism for shared control. *The International Journal of Robotics Research, 32*(7), 790–805.

47. Hiatt, L. M., Harrison, A. M., & Trafton, J. G. (2011). Accommodating human variability in human-robot teams through theory of mind. In *Twenty-second international joint conference on artificial intelligence* (pp. 2066–2071).
48. Gallese, V., & Goldman, A. (1998). Mirror neurons and the simulation theory of mind-reading. *Trends in Cognitive Sciences, 2*(12), 493–501.
49. Nguyen, T., & Gonzalez, C. (2021). Theory of mind from observation in cognitive models and humans. *Topics in Cognitive Science*.
50. Gray, J., Breazeal, C., Berlin, M., Brooks, A., & Lieberman, J. (2005). Action parsing and goal inference using self as simulator. In *ROMAN 2005. IEEE international workshop on robot and human interactive communication, 2005* (pp. 202–209). IEEE.
51. Breazeal, C., Gray, J., & Berlin, M. (2009). An embodied cognition approach to mindreading skills for socially intelligent robots. *The International Journal of Robotics Research, 28*(5), 656–680.
52. Trafton, J. G., Cassimatis, N. L., Bugajska, M. D., Brock, D. P., Mintz, F. E., & Schultz, A. C. (2005). Enabling effective human-robot interaction using perspective-taking in robots. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 35*(4), 460–470.
53. Berlin, M., Gray, J., Thomaz, A. L., Breazeal, C. (2006). Perspective taking: An organizing principle for learning in human-robot interaction. In *Association for the advancement of artificial intelligence* (Vol. 2, pp. 1444–1450).
54. Talamadupula, K., Briggs, G., Chakraborti, T., Scheutz, M., & Kambhampati, S. (2014). Coordination in human-robot teams using mental modeling and plan recognition. In *2014 IEEE/RSJ international conference on intelligent robots and systems* (pp. 2957–2962). IEEE.
55. Jara-Ettinger, J. (2019). Theory of mind as inverse reinforcement learning. *Current Opinion in Behavioral Sciences, 29*, 105–110.
56. Choudhury, R., Swamy, G., Hadfield-Menell, D., & Dragan, A. D. (2019). On the utility of model learning in hri. In *2019 14th ACM/IEEE international conference on Human-Robot Interaction (HRI)* (pp. 317–325). IEEE.
57. Javdani, S., Srinivasa, S. S., & Bagnell, J. A. (2015). Shared autonomy via hindsight optimization. *Robotics science and systems: online proceedings*.
58. Baker, C., Saxe, R., & Tenenbaum, J. (2011). Bayesian theory of mind: Modeling joint belief-desire attribution. In *Proceedings of the annual meeting of the cognitive science society* (Vol. 33, pp. 2069–2074).
59. Wu, S. A., Wang, R. E., Evans, J. A., Tenenbaum, J. B., Parkes, D. C., & Kleiman-Weiner, M. (2021). Too many cooks: Bayesian inference for coordinating multi-agent collaboration. *Topics in Cognitive Science, 13*(2), 414–432.
60. Yuan, L., Fu, Z., Zhou, L., Yang, K., & Zhu, S. C. (2019). Emergence of theory of mind collaboration in multiagent systems. *Emergent Communication Workshop, 33rd Conference on Neural Information Processing Systems (NeurIPS)*.
61. Rabinowitz, N., Perbet, F., Song, F., Zhang, C., Eslami, S. A., & Botvinick, M. (2018). Machine theory of mind. In *International conference on machine learning* (pp. 4218–4227). PMLR.
62. Oguntola, I., Hughes, D., & Sycara, K. (2021). Deep interpretable models of theory of mind. In *2021 30th IEEE international conference on robot & human interactive communication (RO-MAN)* (pp. 657–664). IEEE.
63. Zhu, H., Neubig, G., & Bisk, Y. (2021). Few-shot language coordination by modeling theory of mind. In *International conference on machine learning* (pp. 12901–12911). PMLR.
64. Ziebart, B. D. (2010). *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University.
65. Schulman, J., Chen, X., & Abbeel, P. (2017). Equivalence between policy gradients and soft q-learning. arXiv:1704.06440
66. Nielsen, F. (2020). On a generalization of the Jensen-Shannon divergence and the Jensen-Shannon centroid. *Entropy, 22*(2), 221.
67. Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd international conference for learning representations, San Diego*.
68. 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*.
69. 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).

70. Zhou, M., Liu, Z., Sui, P., Li, Y., & Chung, Y. Y. (2020). Learning implicit credit assignment for cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems, 33*, 11853–11864.
71. Feng, L., Xie, Y., Liu, B., & Wang, S. (2022). Multi-level credit assignment for cooperative multi-agent reinforcement learning. *Applied Sciences, 12*(14), 6938.
72. Zhou, T., Zhang, F., Shao, K., Li, K., Huang, W., Luo, J., Wang, W., Yang, Y., Mao, H., Wang, B., Li, D., Liu, W., & Hao, J. (2021). Cooperative multi-agent transfer learning with level-adaptive credit assignment. arXiv:2106.00517
73. Nguyen, D. T., Kumar, A., & Lau, H. C. (2018). Credit assignment for collective multiagent RL with global rewards. *Advances in Neural Information Processing Systems, 31*.