frontiers | Frontiers in Robotics and AI

TYPE Original Research
PUBLISHED 16 April 2024
DOI 10.3389/frobt.2024.1229026

OPEN ACCESS

EDITED BY
Giovanni Iacca,
University of Trento, Italy

REVIEWED BY
Laura Ferrarotti,
Bruno Kessler Foundation (FBK), Italy
Eiji Uchibe,
Advanced Telecommunications Research Institute International (ATR), Japan

*CORRESPONDENCE
Volker Gabler,
v.gabler@tum.de

RECEIVED 26 May 2023
ACCEPTED 07 February 2024
PUBLISHED 16 April 2024

CITATION
Gabler V and Wollherr D (2024), Decentralized multi-agent reinforcement learning based on best-response policies.
Front. Robot. AI 11:1229026.
doi: 10.3389/frobt.2024.1229026

Decentralized multi-agent reinforcement learning based on best-response policies

Volker Gabler* and Dirk Wollherr

Chair of Automatic Control Engineering, TUM School of Computation, Information and Technology, Technical University of Munich, Munich, Germany

Introduction: Multi-agent systems are an interdisciplinary research field that describes the concept of multiple decisive individuals interacting with a usually partially observable environment. Given the recent advances in single-agent reinforcement learning, multi-agent reinforcement learning (RL) has gained tremendous interest in recent years. Most research studies apply a fully centralized learning scheme to ease the transfer from the single-agent domain to multi-agent systems.

Methods: In contrast, we claim that a decentralized learning scheme is preferable for applications in real-world scenarios as this allows deploying a learning algorithm on an individual robot rather than deploying the algorithm to a complete fleet of robots. Therefore, this article outlines a novel actor–critic (AC) approach tailored to cooperative MARL problems in sparsely rewarded domains. Our approach decouples the MARL problem into a set of distributed agents that model the other agents as responsive entities. In particular, we propose using two separate critics per agent to distinguish between the joint task reward and agent-based costs as commonly applied within multi-robot planning. On one hand, the agent-based critic intends to decrease agent-specific costs. On the other hand, each agent intends to optimize the joint team reward based on the joint task critic. As this critic still depends on the joint action of all agents, we outline two suitable behavior models based on Stackelberg games: a game against nature and a dyadic game against each agent. Following these behavior models, our algorithm allows fully decentralized execution and training.

Results and Discussion: We evaluate our presented method using the proposed behavior models within a sparsely rewarded simulated multi-agent environment. Although our approach already outperforms the state-of-the-art learners, we conclude this article by outlining possible extensions of our algorithm that future research may build upon.

KEYWORDS

multi-agent reinforcement learning, game theory, deep learning, artificial intelligence, actor–critic algorithm, multi-agent, Stackelberg, decentralized learning schemes, reinforcement leaning

1 Introduction

Based on recent advances in robotics research over the last few decades, automated robotic systems have been established in everyday life, even beyond industrial applications. Nonetheless, it remains tedious and challenging to impart new tasks to robots, especially if

the environment is stochastic and hard to model. In this context, applied machine learning (ML), specifically reinforcement learning (RL), is a promising research field that aims to continuously improve robotic performance from collected task trial samples. In particular, the core motivation is to equip robots with the ability to explore and learn unknown tasks simultaneously without relying on an accurate model of the environment or the task. Building upon this, the concept of MARL has raised interest in improving scalability by executing tasks by a fleet of robots rather than a single autonomous unit. In order to exploit results from single-agent RL, a common paradigm in MARL is *centralized learning with decentralized execution*. Nonetheless, it is desirable to handle each robot as an independent individual, such that the learning phase of a MARL algorithm also scales well. In contrast to simulated environments, where access to other agents' policies and observation is realistic, this assumption is overly restrictive for real robot systems and adds additional constraints to heterogeneous robot fleets.

Therefore, the contribution of this article is a novel AC method for cooperative MARL problems in sparsely rewarded environments. Our MARL algorithm allows fully decoupling learning among the agents while achieving comparable performance to current state-of-the-art MARL approaches. This approach uses the best-response policies of other agents conditioned on each agent's policy output. This leverages the need to access the exact agent policies during centralized learning to achieve a fully decentralized learning scheme. This decision-theoretic principle stems from Stackelberg equilibria from game theory and is tailored to non-zero-sum games in the scope of this article.

Our proposed method incorporates another multi-robot planning and game theory concept by explicitly differentiating between *interactive* task rewards and agent-specific costs, i.e., *native* costs. In other words, each agent estimates the performance of the joint policy with regard to the current task to be learned but also a cost critic that evaluates agent-specific costs.

In the remainder of this article, we briefly summarize the state-of-the-art algorithms of MARL in Section 2, followed by a summary of the technical foundations of this article and the technical problem in Section 3. The core concept of our proposed framework is outlined in Section 4. In order to evaluate the presented method, we present the collected results of our method compared against state-of-the-art MARL algorithms in Section 5. Based on these results, we discuss our algorithm and explicitly sketch conceptual modifications of our approach in Section 6. Eventually, we conclude this article in Section 7.

2 Related work

Even though early applications of RL in robotic systems have shown promising results (Ng et al., 2004; Kolter and Ng, 2009), it was the success of outperforming humans in computer games via deep RL (Mnih et al., 2015; Silver et al., 2016; Vinyals et al., 2019) without suffering from catastrophic interference (McCloskey and Cohen, 1989) problems that has opened the door for RL applications within complex, real-world environments. Given the computational power of modern graphics processing units (GPUs), policy gradients such as the stochastic policy gradient from Sutton et al. (1999a) or the deterministic policy gradient (PG) from Silver et al. (2014) have been realized via function approximators such as neural networks (NNs). A famous example is given as the deep deterministic policy gradient (DPG) by Lillicrap et al. (2016). DDPG has shown that deep RL can also be applied on continuous action spaces such that the applicability of RL within robotic systems has been boosted drastically ever since. Even though further PG methods have been developed in order to improve the variance sensitivity issue, such as trust region policy optimization (Schulman et al., 2015), proximal policy optimization (Schulman et al., 2017), or maximum *a posteriori* policy optimization (Song et al., 2020), the majority of algorithms relies on an AC architecture, where an additional critic reduces the variance drastically, such as the soft actor–critic (SAC) (Haarnoja et al., 2018). As an intense outline of advances in single-agent RL is beyond the scope of this article, we encourage the interested reader to available literature survey papers (Kaelbling et al., 1996; Kober et al., 2013; Arulkumaran et al., 2017). Building upon the results from single-agent RL, MARL has gained great interest over the last decades and is thus outlined separately in the following.

2.1 Multi-agent reinforcement learning

In addition to solving complex Markov decision process (MDP) problems, the decentralized extension of the Markov game (MG) has gained attention in the context of MARL (van der Wal, 1980; Littman, 1994). The naive approach of extending Q-learning to a set of $N_{\mathfrak{A}}$ independent learners (Tan, 1993) works well for small-scale problems or selective applications. Similar to deep RL, initial results on MARL have been found on discrete action sets, such as the differentiable inter-agent learning by Foerster et al. (2016) or the explicit communication learning by Havrylov and Titov (2017); Mordatch and Abbeel (2017). In general, however, independent learners violate the Markov assumption (Laurent et al., 2011).

Multi-agent deep deterministic policy gradient (DDPG) is an extension of DDPG to MARL (Lowe et al., 2017), which also applies an AC architecture. During training, a centralized critic uses additional information about the other agents' states and actions to approximate the $\mathfrak{Q}$-function. Given this centralized critic, each agent updates a policy that is only conditioned on the local observations of each agent. Thus, the actor only relies on local observations during execution. MADDPG has achieved very robust results in simulated benchmark environments (Mordatch and Abbeel, 2017) for cooperative and competitive scenarios. Various extensions to the multi-agent deep deterministic policy gradient (MADDPG) have been proposed. Li et al. (2019) introduced an extension to MADDPG that uses the minimax concept of game theory to make decisions under uncertainty. The idea is to take the best action in the worst possible case.

As pointed out by Ackermann et al. (2019), the overestimation bias is also present in MARL. Some initial works have proposed to bridge concepts from the single-agent domain (van Hasselt, 2010) to MARL (Sun et al., 2020). Thus, SAC has been adjusted to the multi-agent domain by Wei et al. (2018), for which further extensions have been outlined, e.g., Zhang et al. (2020) proposed a Lyapunov-based penalty term to the policy update to stabilize the policy gradient. As centralized learning inherently suffers from poor scaling, Iqbal

and Sha (2019) introduced attention mechanisms in the multi-actor-attention-critic (MAAC). In order to cope with large-scale MARL, Sheikh and Bölöni (2020) explicitly differentiated between local and global reward metrics that each agent obtains from the environment.

In contrast to single-agent systems, the critic also suffers from the non-stationarity of the policies of other agents. This initiated the research on explicitly modeling the learning behavior of other agents, such as Foerster et al. (2018). Alternatively, Tian et al. (2019) proposed to model the MARL problem as an inference problem, i.e., to estimate the most likely action of the other agents and respond with the best response (br). Jaques et al. (2019) reversed this idea by applying counterfactual reasoning and thus incorporating the mutual influence among agents into the reward of each agent. They outlined a decentralized version of their algorithm, which applies behavioral cloning similarly to the decentralized version of MADDPG.

As a complete survey of MARL is beyond the scope of this article, we refer to Zhang et al. (2021), Hernandez-Leal et al. (2020), Yang and Wang (2020), Hernandez-Leal et al. (2019), and Nguyen et al. (2020) for a more detailed literature review. In order to illustrate the relevance of MARL from an application-driven perspective, there exists a variety of recent examples, such as logistics (Tang et al., 2021), the Internet of Things (Wu et al., 2021), or motion-planning for robots (He et al., 2021).

3 Preliminaries

As the methods presented in this article build upon various findings from the literature, we provide an insight into these methods. We begin with sketching the notation used in this article, followed by an initial example in the form of introducing MGs.

3.1 Notation

In order to outline the notation for the remainder of this article, we use $\mathfrak{p}$ as an arbitrary placeholder variable. We denote $\mathfrak{p}^{(i)}$ as a variable explicitly assigned to agent i, while $(\mathfrak{p})_{i\in N_{\mathfrak{A}}}$ denotes the *joint* team analog of the said variable for all agents. This is most commonly denoted as $\underline{\mathfrak{p}}$ for brevity, while $\underline{\mathfrak{p}}^{(-i)}$ denotes all elements of $(\mathfrak{p})_{i\in N_{\mathfrak{A}}}$ except $\mathfrak{p}^i$. Furthermore, we denote the vector as $\mathfrak{p}$ and matrices as $\mathfrak{p}$, while $1^{\mathfrak{p}}$, $1^{\mathfrak{p}\times\mathfrak{p}}$, $0^{\mathfrak{p}}$, and $0^{\mathfrak{p}\times\mathfrak{p}}$ denote identity- and zero-vectors/matrices. In general, we denote time-variant variables as $\mathfrak{p}_t$, while a temporal successor $\mathfrak{p}_{t+1}$ is denoted as $'$ for brevity. In the context of stochastic variables, we denote probability density functions (PDFs) as $\mathbb{P}[\mathfrak{p}]$ and conditionally dependent PDFs as $\mathbb{P}[\mathfrak{p}_1|\mathfrak{p}_2]$. Similarly, $\mathbb{E}_{\mathfrak{p}_1\sim\rho(\mathfrak{p}_2)}[]$ and $\mathbb{V}\mathrm{ar}_{\mathfrak{p}_1\sim\rho(\mathfrak{p}_2)}[]$ symbolize the expectation and variance of the random variable $\mathfrak{p}_1$ that follows a probability distribution function $\rho()$, which depends on $\mathfrak{p}_2$. Eventually, we denote hierarchical systems by denoting layer k as ${}^{\{k\}}\mathfrak{p}$.

3.2 Markov game

An MG is an extension of an MDP to the multi-agent domain, which is fully described by the tuple $(\underline{\mathfrak{A}},\underline{\mathcal{S}},\underline{\mathcal{A}},\mathcal{T},\underline{\mathcal{R}},\gamma)$, where $N_{\mathfrak{A}}$ agents $\underline{\mathfrak{A}}=\left(\mathfrak{A}^{(1)},\mathfrak{A}^{(2)},\ldots,\mathfrak{A}^{(N_{\mathfrak{A}})}\right)=(\mathfrak{A})_{i\in N_{\mathfrak{A}}}$ interact with each other in a stochastic environment (Shapley, 1952; Shapley, 1953), as shown in Figure 1. The state $\underline{\mathrm{s}}=\left(\mathrm{s}^{(1)},\mathrm{s}^{(2)},\ldots,\mathrm{s}^{(N_{\mathfrak{A}})}\right)\in\underline{\mathcal{S}}$ of the environment with state space $\underline{\mathcal{S}}$ is perceived as the individual state observations $\mathrm{s}^{(i)}$ for each agent. Due to the Markov property, the dynamics of an MG is given by each individual choosing an action $\mathrm{a}^{(i)}\in\mathcal{A}^{(i)}\subset\underline{\mathcal{A}}$ out of an agent-specific action space $\mathcal{A}^{(i)}$, thus forming a joint action $\underline{\mathrm{a}}$ that transitions $\underline{\mathrm{s}}$ to $\underline{\mathrm{s}}'$ according to a transition probability function $\mathcal{T}:=\mathbb{P}[\underline{\mathrm{s}}'|\underline{\mathrm{s}},\underline{\mathrm{a}}]$, and $\mathbb{P}[\underline{\mathrm{s}}'|\underline{\mathrm{s}},\underline{\mathrm{a}}]$ is the conditional probability for $\underline{\mathrm{s}}'$, given $\underline{\mathrm{s}}$ and $\underline{\mathrm{a}}$. The individual reward functions $\underline{\mathcal{R}}=(\mathcal{R}^{(i)}:\mathcal{S}^{(i)}\times\mathcal{A}^{(i)}\times\underline{\mathcal{A}}^{(-i)}\times\mathcal{S}^{(i)}\to\mathbb{R})_{i\in N_{\mathfrak{A}}}$ map a transition from $\underline{\mathrm{s}}$ to $\underline{\mathrm{s}}'$, given $\underline{\mathrm{a}}$, to a numeric value for each agent $\mathfrak{A}^{(i)}$, which is denoted as $\mathrm{r}^{(i)}:=\mathcal{R}^{(i)}(\mathrm{s}^{(i)},\mathrm{a}^{(i)},\underline{\mathrm{a}}^{(-i)},\mathrm{s}^{(i)\prime})$. Given this, each agent $\mathfrak{A}^{(i)}$ follows the stochastic behavior policy $\mathrm{a}^{(i)}\sim\pi^{(i)}(\mathrm{s}^{(i)})$ that intends to maximize the objective for each agent

$$\begin{aligned}\mathcal{J}^{(i)}:&=\sum_{t=0}^{\infty}\gamma^{\mathrm{t}}\int_{\underline{\mathcal{A}}}\underline{\pi}\left(\underline{\mathrm{a}}_{\mathrm{t}}|\underline{\mathrm{s}}_{\mathrm{t}}\right)\int_{\underline{\mathcal{S}}}\mathcal{T}\left(\underline{\mathrm{s}}_{\mathrm{t}+1}|\underline{\mathrm{s}}_{\mathrm{t}},\underline{\mathrm{a}}_{\mathrm{t}}\right)\mathrm{r}^{(i)}d\underline{\mathrm{s}}_{\mathrm{t}+1}d\underline{\mathrm{a}}_{\mathrm{t}}\\&=\sum_{t=0}^{\infty}\gamma^{\mathrm{t}}\mathbb{E}_{\underline{\mathrm{a}}^{(-i)}\sim\underline{\pi}^{(-i)}\left(\mathrm{s}_{\mathrm{t}}^{(i)}\right)}\\&\quad\times\left[\int_{\underline{\mathcal{A}}}\pi\left(\mathrm{a}_{\mathrm{t}}^{(i)}|\mathrm{s}_{\mathrm{t}}^{(i)}\right)\mathbb{P}[\underline{\mathrm{a}}^{(-i)}]\int_{\underline{\mathcal{S}}}\mathcal{T}\left(\underline{\mathrm{s}}_{\mathrm{t}+1}|\underline{\mathrm{s}}_{\mathrm{t}},\underline{\mathrm{a}}_{\mathrm{t}}\right)\mathrm{r}^{(i)}d\underline{\mathrm{s}}_{\mathrm{t}+1}d\underline{\mathrm{a}}_{\mathrm{t}}\right]\end{aligned},\quad(1)$$

where the hyperparameter $\gamma\in(0,1]$ is a temporal decay weight that scales short-term versus long-term impact. In order to solve Equation 1, the state-value function,

$$\mathrm{V}_{\underline{\pi}}^{(i)}(\underline{\mathrm{s}})=\sum_{t=0}^{\infty}\mathbb{E}_{\underline{\mathrm{a}}_{\mathrm{t}}\sim\rho(\underline{\pi}),\left(\underline{\mathrm{s}}_{\mathrm{t}},\underline{\mathrm{s}}_{\mathrm{t}+1}\right)\sim\rho(\mathcal{T},\underline{\pi})}\left[\gamma^{\mathrm{t}}\mathrm{r}^{(i)}|\underline{\mathrm{s}}_0\right],\quad(2)$$

the state-action value function,

$$\mathrm{Q}_{\underline{\pi}}^{(i)}(\underline{\mathrm{s}},\underline{\mathrm{a}})=\mathrm{r}^{(i)}+\gamma\mathbb{E}_{\underline{\mathrm{s}}'\sim\rho(\mathcal{T},\underline{\pi})}\left[\mathrm{V}_{\underline{\pi}}^{(i)}(\underline{\mathrm{s}}')\right],\quad(3)$$

and the advantage function,

$$\mathrm{A}_{\underline{\pi}}^{(i)}(\underline{\mathrm{s}},\underline{\mathrm{a}})=\mathrm{Q}_{\underline{\pi}}^{(i)}(\underline{\mathrm{s}},\underline{\mathrm{a}})-\mathrm{V}_{\underline{\pi}}^{(i)}(\underline{\mathrm{s}})\quad(4)$$

have been introduced as the multi-agent version of the Bellman backup operator for MDPs (Bellman, 1957). Given that the agents follow a fixed and optimal policy $\underline{\pi}^*$, the dynamic programming problem eventually solves Equation 1 as the global optimum of the MG, as shown by Littman (1994). Given the optimal $\mathrm{Q}_{\underline{\pi}^*}$ function, the optimal policies for each agent can be obtained as the following:

$$\pi^{(i)}(\underline{\mathrm{s}})^*\leftarrow\arg\max_{\pi^{(i)}}\mathrm{Q}_{\underline{\pi}^*}^{(i)}\left(\underline{\mathrm{s}},\underline{\mathrm{a}}^{(-i)},\mathrm{a}\right)\Big|_{\mathrm{a}\leftarrow\pi^{(i)}(\underline{\mathrm{s}})}.\quad(5)$$

As solving Equation 5 requires each agent to follow an optimal policy, the definition of a best-response policy is of importance in MGs.

Definition 3.1: Best response policy

Given a joint policy $\underline{\pi}^{(-i)}$ for the neighboring agents of agent $\mathfrak{A}^{(i)}$, a policy ${}^{\mathrm{br}}\pi^{(i)}$ is called a br to $\underline{\pi}^{(-i)}$ if and only if

$$\mathcal{J}^{(i)}\left({}^{\mathrm{br}}\mathrm{a}^{(i)}\leftarrow{}^{\mathrm{br}}\pi^{(i)}|\underline{\pi}^{(-i)}\right)\geq\mathcal{J}^{(i)}\left(\mathrm{a}^{(i)}\neq{}^{\mathrm{br}}\mathrm{a}^{(i)}|\underline{\pi}^{(-i)}\right),$$

i.e., the agent $\mathfrak{A}^{(i)}$ cannot improve the individual payoff return $\mathcal{J}^{(i)}$ by deviating from ${}^{\mathrm{br}}\pi^{(i)}$ (Shoham and Leyton-Brown, 2008).

Within an MG, the optimal policy requires that the policies of the individual agents are the br to the policies of the surrounding agents, leading to the definition of a Nash equilibrium (NE).

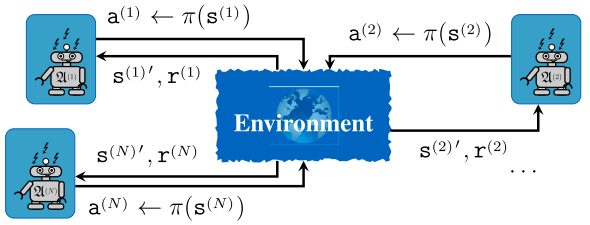

FIGURE 1
Sketch of a general MARL problem, where $N_{\mathfrak{A}}$ agents interact with each other in an unknown environment. Each agent has access to the individual state observation $\mathfrak{s}^{(i)}$, which can be mapped to an action $\mathfrak{a}^{(i)}$ via the policy $\pi^{(i)}$ in such a manner that the expected individual return $\mathfrak{r}^{(i)}$ is maximized.

Definition 3.2: Nash equilibrium

According to Nash (1950), a policy ${}^{\mathrm{NE}}\underline{\pi} := \left({}^{\mathrm{NE}}\pi\right)_{i\in N_{\mathfrak{A}}}$ is a NE if and only if each agent following ${}^{\mathrm{NE}}\pi^{i} \in {}^{\mathrm{NE}}\underline{\pi}$ results in each policy being a br policy, according to Definition 3.1. Replacing the objectives $\mathcal{J}^{(i)}$ by the state-action value $\mathfrak{Q}^{(i)}$, this requires

$$\begin{aligned} \mathfrak{Q}^{(i)}_{{}^{\mathrm{NE}}\pi^{(i)},{}^{\mathrm{NE}}\underline{\pi}^{(-i)}} &\geq \mathfrak{Q}^{(i)}_{\tilde{\pi}^{(i)},{}^{\mathrm{NE}}\underline{\pi}^{(-i)}} \\ \mathfrak{Q}^{(i)}_{{}^{\mathrm{NE}}\pi^{(i)},{}^{\mathrm{NE}}\underline{\pi}^{(-i)}} &\geq \mathfrak{Q}^{(i)}_{{}^{\mathrm{NE}}\pi^{(i)},\tilde{\underline{\pi}}^{(-i)}}, \\ \text{with } \tilde{\pi} \neq {}^{\mathrm{NE}}\pi,\ \forall \mathfrak{A}^{(i)} \in \underline{\mathfrak{A}}, \forall \underline{\mathfrak{s}} \in \underline{\mathcal{S}} & \end{aligned}$$

to hold on the global state space $\underline{\mathcal{S}}$.

Nonetheless, in real-world problems, neither $\underline{\pi}^{*}$ nor the value functions are known. In addition, the environment is characterized by multiple learners, whose policies and, thus, actions vary over time and cannot directly be controlled by an individual agent in an MG. This results in the problem formulation of this article.

3.3 Multi-agent reinforcement learning problem

Given a set of agents $(\mathfrak{A})_{i\in N_{\mathfrak{A}}}$ that try to optimize their individual accumulated discounted reward, according to Section 3.2, an optimal policy for each agent has to be found, which fulfills the following:

- $(\pi \leftarrow \arg\max \mathfrak{Q}_{\pi})_{i\in N_{\mathfrak{A}}}$ according to Equation 5.
- The joint action $\underline{\mathfrak{a}} = (\mathfrak{a} \leftarrow \pi^{*})_{i\in N_{\mathfrak{A}}}$ is an NE of the MG, according to Definition 3.2.

We will continue with a short overview of RL methods that have been established as the current state-of-the-art methods within single-agent RL and MARL.

3.4 Policy gradient methods

Obtaining an optimal policy π_{Π}, parameterized by II, has been tackled by generating PGs (Sutton et al., 1999b) that estimate the stochastic gradient over II of a policy, and the policy-loss function is defined as the following:

$$\nabla_{\Pi}\mathcal{J}(\pi_{\Pi}) = \mathbb{E}_{\mathfrak{s}\sim\pi(\mathfrak{s})}\left[\sum_{\mathfrak{t}=0}^{\infty} \nabla_{\Pi} \log \pi_{\Pi}(\mathfrak{a}_{\mathfrak{t}}|\mathfrak{s}_{\mathfrak{t}})\ \chi_{\mathfrak{t}}\right], \tag{6}$$

where $\chi_{\mathfrak{t}}$ may, for example, be the single agent version of Equation 3 or Equation 4, i.e., $\mathfrak{Q}_{\pi}$ or $\mathfrak{A}_{\pi}$. If one can obtain the gradient $\nabla_{\mathfrak{a}}\chi_{\mathfrak{t}}$ directly, i.e., the action space is continuous and the environment is stationary, it is also possible to obtain the deterministic policy gradient (DPG) from Equation 6 as the following:

$$\nabla_{\Pi}\mathcal{J}(\pi_{\Pi}) = \mathbb{E}_{\mathfrak{s}\sim\mathcal{D}}\left[\nabla_{\Pi}\pi_{\Pi}(\mathfrak{a}|\mathfrak{s})\ \nabla_{\mathfrak{a}}\chi|_{\mathfrak{a}\leftarrow\pi(\mathfrak{s})}\right], \tag{7}$$

where the expectation is approximated by drawing samples from an experience replay buffer $\mathcal{D}$ that contains observed environment transitions. Exemplary DDPG uses $\chi_{\mathfrak{t}} := \mathfrak{Q}_{\pi}$ in order to obtain the gradient of the state-action value in Equation 7. As it can be seen in Equations 6 and 7, PGs and DPGs are generally highly sensitive to the variance of $\chi_{\mathfrak{t}}$. As a consequence, AC methods have been outlined that add a policy evaluation metric to the policy update of PG methods.

3.5 Actor–critic methods

As the accumulated reward does generally suffer from high variance over repeated episodes, AC algorithms simultaneously estimate $\mathfrak{A}_{\pi}$ or $\mathfrak{Q}_{\pi}$ alongside the PGs in Equation 6. The deep Q-network presented by Mnih et al. (2015) uses NNs as function approximators, thus approximating $\mathfrak{Q}_{\pi}$ by $\mathfrak{Q}_{\Theta}$ and ${}^{\dagger}\mathfrak{Q}_{\Theta}$, parametrized by Θ, where ${}^{\dagger}\mathfrak{Q}$ denotes the *target-net* of $\mathfrak{Q}$. These two function approximators are then used to learn $\mathfrak{Q}_{\pi}$ via off-policy temporal-difference learning, which is obtained via iteratively minimizing the loss function as follows:

$$\begin{aligned} \mathcal{L}_{\mathfrak{Q}}(\Theta) :=& \mathbb{E}_{(\mathfrak{s},\mathfrak{a},\mathfrak{r},\mathfrak{s}')\sim\mathcal{D}}\left[\frac{1}{2}\left(\mathfrak{p} - \mathfrak{Q}_{\Theta}(\mathfrak{s},\mathfrak{a})\right)^{2}\right] \\ \text{with } \mathfrak{p} =& \mathfrak{r}(\mathfrak{s},\mathfrak{a},\mathfrak{s}') + \gamma(1-\mathfrak{d})\,{}^{\dagger}\mathfrak{V}_{\Theta}(\mathfrak{s}') \\ {}^{\dagger}\mathfrak{V}_{\Theta}(\mathfrak{s}') =& \mathbb{E}_{\mathfrak{a}'\leftarrow\pi(\mathfrak{s}')}\left[{}^{\dagger}\mathfrak{Q}_{\Theta}(\mathfrak{s}',\mathfrak{a}')\right] \end{aligned}, \tag{8}$$

where $\mathcal{D}$ is again a replay buffer that stores experienced transitions from the environment during the exploration process. Each sample contains the state $\mathfrak{s}$, action $\mathfrak{a}$, next state $\mathfrak{s}'$, the experienced reward $\mathfrak{r}$, and the termination flag $\mathfrak{d}$. The term $(1-\mathfrak{d})$ thus ignores the value of the successor state in the Bellman backup operator in Equation 3 at the terminal states.

The SAC (Haarnoja et al., 2018) is an extension of the general AC that approximates the solution of Equation 1 via a maximum entropy objective by introducing a soft-value function, thus replacing Equation 2 by the following:

$$\mathfrak{V}_{\pi}(\mathfrak{s}) := \mathbb{E}_{(\mathfrak{s}_{\mathfrak{t}},\mathfrak{a}_{\mathfrak{t}},\mathfrak{s}_{\mathfrak{t}+1})\sim\mathcal{D}}\left[\sum_{\mathfrak{t}=0}^{\infty}\gamma^{\mathfrak{t}}\left(\mathfrak{r}(\mathfrak{s}_{\mathfrak{t}},\mathfrak{a}_{\mathfrak{t}},\mathfrak{s}_{\mathfrak{t}+1}) + \alpha\mathbb{H}(\pi(\cdot|\mathfrak{s}_{\mathfrak{t}}))\right)\right]_{\mathfrak{s}_0=\mathfrak{s}}, \tag{9}$$

In Eq. 9 $\mathbb{H}(\cdot)$ denotes the policy entropy at a given state and α is a temperature parameter that weighs the impact of the entropy against the environment reward. In contrast to Equation 2, this objective explicitly encourages exploration in regions of high rewards, thus decreasing the chance of converging to local minima. Furthermore,

two function approximators are used for the critic as in the twin-delayed deep deterministic policy gradient (TD3) such that the target value function in Equation 8 is obtained as the following:

$$^{\dagger}\mathrm{V}_{\Theta}(\mathrm{s}') = \mathbb{E}_{\mathrm{a}'\sim\pi(\mathrm{s}')}\left[\min_{j=1,2} {}^{\dagger}\mathrm{Q}_{\Theta,j}(\mathrm{s}',\mathrm{a}') - \alpha \log \pi_{\Pi}(\mathrm{a}'|\mathrm{s}')\right], \tag{10}$$

In Eq. 10 a is obtained from $\pi(\mathrm{s}')$, whereas s' is drawn from $\mathcal{D}$. In contrast to this, the actual policy loss is obtained by applying the *reparameterization trick* as follows:

$$\mathcal{L}_{\pi}^{\mathrm{SAC}}(\Pi) := \mathbb{E}_{\mathrm{s}\sim\mathcal{D}}\left[\min_{j=1,2} {}^{\dagger}\mathrm{Q}_{\Theta,j}\underbrace{(\mathrm{s}, f_{\Pi}(\mathrm{s},\zeta))}_{\chi} - \alpha \log \pi_{\Pi}(f_{\Pi}(\mathrm{s},\zeta)|\mathrm{s})\right], \tag{11}$$

which computes a deterministic function $f_{\Pi}(\mathrm{s},\zeta)$ that depends on the state s, policy parameters Π, and independent noise vector ζ drawn from a fixed distribution, e.g., mean free Gaussian noise. In contrast to DDPG, this parameterized policy is also squashed via a tanh function to the bounds of the action space, thus resulting in valid samples that can be used to generate a stochastic policy for the stochastic policy gradient update step.

3.6 Multi-agent actor–critic algorithms

The methods mentioned above have been recently extended to the multi-agent domain. The MADDPG extends AC with DDPG by proposing the schematic representation of decentralized execution in combination with centralized learning. As such, each $\mathfrak{A}^{(i)}$ learns an individual (deterministic) policy $\pi := \mathcal{S}^{(i)} \times \mathcal{A}^{(i)} \mapsto [0,1]$ while setting $\chi_{\mathrm{t}} := \mathrm{Q}^{(i)}(\underline{\mathrm{s}},\underline{\mathrm{a}})$ in Equation 7, which has access to the observations $\mathrm{s}^{(i)}$, actions $\mathrm{a}^{(i)}$, and policies of all agents such that Equation 8 can be directly applied to the multi-agent domain. This requires having access to all policies during learning in order to calculate the target values of Equation 8. Similar approaches have been proposed by MAAC and counterfactual multi-agent (COMA)-PG, which additionally incorporate a baseline value function for the policy update and thus use the multi-agent advantage function

$$\chi := \mathrm{A}^{(i)}(\underline{\mathrm{s}},\underline{\mathrm{a}}) = \mathrm{Q}^{(i)}\left(\underline{\mathrm{s}},\mathrm{a}^{(i)},\underline{\mathrm{a}}^{(-i)}\right) - \mathrm{V}_{b}\left(\underline{\mathrm{s}},\underline{\mathrm{a}}^{(-i)}\right) \tag{12}$$

for their policy loss declarations. The baseline $\mathrm{V}_{b}(\underline{\mathrm{s}},\underline{\mathrm{a}}^{(-i)})$ estimates the value of a current state and the opponents' current actions such that optimizing Equation 12 leads to a best-response action of the agent $\mathfrak{A}^{(i)}$, according to Definition 3.1. Although COMA substitutes Equation 12 into Equation 6, MAAC uses SAC, thus inserting Equation 12 into Equation 11. Furthermore, MAAC improves the centralized critic by adding an attention mechanism that explicitly learns which parts of the observations have an actual impact on the values of the critic.

4 Technical approach

A key challenge for real-world applications in multi-agent systems is the ability to handle decentralized decisions asynchronously. Although most MARL approaches allow decentralized execution, they still rely on centralized learning (Lowe et al., 2017). This imposes various constraints on the overall multi-agent system, e.g., the necessity to have access to all observations of all agents during learning. Furthermore, real robot systems are often commanded by a task planner or similar, where each robot is assigned a dedicated sub-task. Similarly, the upper layer could be realized via a hierarchical reinforcement learning (HRL) learner that learns to allocate tasks to each agent in a team-optimal manner. To visualize the necessity of our algorithm, a two-layered hierarchical decision framework for a multi-agent system is shown in Figure 2. The upper layer could either stem from a sub-task allocator that assigns tasks to each agent or a multi-agent HRL algorithm. As can be seen from this figure, centralized learning would not only require synchronous updates along all agents and layers, but it also would require knowing the current output of the upper layer of each agent. In order to leverage this constraint, we propose a novel decentralized MARL concept that builds upon the concept of best-response policies and separates joint rewards from internal agent objectives.

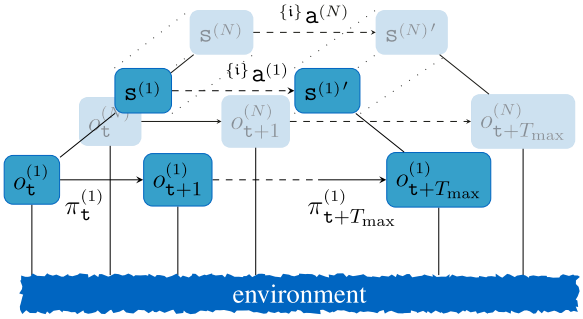

FIGURE 2
Exemplary step of a two-level hierarchical MARL step where each low-level step represents an interaction with the environment from Figure 1. For brevity, only selective nodes and edges are labeled. The upper layer acts synchronously, such that the observed transition would qualify for centralized learning for all layers, which is emphasized via the dashed lines for the upper layer.

4.1 Decentralized MARL based on Stackelberg equilibria

In order to achieve a decentralized model for MARL problems, a previous work has evaluated the application of predicting the br policy to the inferred action of an opponent (Tian et al., 2019) or assuming overly restrictive access to the environment feedback of other agents. The latter is always fulfilled for centralized learning. In order to decouple the decentralized learning procedure, we propose a similar idea to that of Tian et al. (2019) and instead reformulate their inference-based policy by modeling the br policy of other agents. In detail, we apply the concept of Stackelberg equilibria. The Stackelberg equilibrium evaluates the br of an agent if the opponent has unveiled the current actions. Therefore, each agent regresses not only a policy $\pi^{(i)}{}_{\Pi} := \mathcal{S}^{(i)} \mapsto \mathcal{A}^{(i)}$, parameterized by Π, that intends to optimize the player-individual agent-objective but also a br policy $\underline{\pi}_{\mathrm{br}\ \Xi}^{(-i)} := \mathcal{S}^{(i)} \times \mathcal{A}^{(i)} \mapsto \underline{\mathcal{A}}^{(-i)}$– parameterized by Ξ – that represents the reactions of the other agent(s) at each step.

In addition to regressing the br policy of the other agent, we further claim that it is beneficial to distinguish between joint task

rewards and individual or native cost terms. In general, we assume that the individual reward for a cooperative MARL problem is given in the following form:

$$\mathcal{J}^{(i)}\left(\underline{\mathrm{s}},\underline{\pi},\underline{\mathrm{s}}'\right)=\sum_{t=0}^{\infty}\gamma^{\mathrm{t}}\mathbb{E}\left[\mathrm{r}^{(i)}\left(\underline{\mathrm{s}}_{\mathrm{t}},\underline{\pi},\underline{\mathrm{s}}_{\mathrm{t+1}}\right)-\hat{\mathrm{c}}_{\mathrm{nat}}{}^{(i)}\left(\underline{\mathrm{s}}_{\mathrm{t}},\pi^{(i)},\underline{\mathrm{s}}_{\mathrm{t+1}}\right)\right]\quad\in\mathbb{R},\tag{13}$$

Thus, the individual reward in Eq. 13 consists of a joint or cooperative task reward that depends on the joint action or policy, as well as an interactive cost component that only affects each player. Although some existing work assumes to directly have access to local and global rewards, i.e., to obtain $\mathrm{r}\left(\underline{\mathrm{s}},\underline{\pi},\underline{\mathrm{s}}'\right)$ and $\hat{\mathrm{c}}_{\mathrm{nat}}{}^{(i)}$ directly (Sheikh and Bölöni, 2020), we propose a model that only has access to the agent reward and the averaged joint task reward of all agents tailored toward sparsely rewarded environments. Thus, the cost of the agents needs to be estimated from this joint reward at each transition. Thus, we apply the following:

$$\begin{aligned}\hat{\mathrm{c}}_{\mathrm{nat}}{}^{(i)}\left(\mathrm{s}^{(i)},\underline{\mathrm{a}},\mathrm{s}^{(i)\prime}\right)&\approx\min\left(\frac{1}{N_{\mathfrak{A}}}\sum_{j=1}^{N_{\mathfrak{A}}}\mathrm{r}^{(j)}\left(\underline{\mathrm{s}}^{(j)},\underline{\mathrm{a}},\underline{\mathrm{s}}^{(j)\prime}\right)-\mathrm{r}^{(i)}\left(\mathrm{s}^{(i)},\underline{\mathrm{a}},\mathrm{s}^{(i)\prime}\right),0\right)\\&\approx\min\left(\frac{1}{N_{\mathfrak{A}}}\sum_{j=1}^{N_{\mathfrak{A}}}\mathrm{r}^{(j)}-\mathrm{r}^{(i)},0\right)\quad\text{where}\quad\underline{\mathrm{r}}\sim\mathcal{D}^{(i)},\end{aligned}\tag{14}$$

i.e., we keep the individual agent reward as the joint task reward, which solely depends on the observation of each agent. In addition, we propose to regress a non-negative auxiliary cost term that contains information about *local* interaction penalties for each agent. Exploiting the rare occurrence of costs within sparsely rewarded environments, we estimate the step cost for each agent by the difference of the average rewards of all agents compared to the individual agent reward. Having collected empirical data within a replay buffer $\mathcal{D}^{(i)}$ for each agent, the numeric cost is approximated as a non-negative difference of the average reward values of all agents without the necessity of explicitly accessing the observations of each agent. Finally, our br-AC approach approximates the following:

- the (interactive) task critic $\mathrm{Q}_{\mathrm{int}}^{(i)}:\mathcal{S}^{(i)}\times\underline{\mathcal{A}}\mapsto\mathbb{R}$, which intends to maximize the accumulated task reward.
- the (native) agent critic $\mathrm{Q}_{\mathrm{nat}}^{(i)}:\mathcal{S}^{(i)}\times\mathcal{A}^{(i)}\mapsto\mathbb{R}$, which intends to minimize the agent-specific costs.

Therefore, the final goal of each agent is to maximize the interactive task critic, i.e., optimize the accumulated team reward, while minimizing the agent-specific cost penalties from the native agent critic. This concept is similar to the idea of combining RL rewards while also minimizing a myopic objective by means of numeric optimization *cf.* (Englert and Toussaint, 2016). The agent policies and critics can then be regressed by means of existing AC methods, such as SAC or TD3. In contrast to the default methods, the policies need to optimize the joint task critic and the native agent critic simultaneously. Therefore, the difference between the two critics provides the final critic that is used for the policy gradient of the current actor. Eventually, the br policy needs to be updated as well. In contrast to the agent policy, the native critic is independent of the br policy and can thus be neglected for the update of the br policies. As we emphasize on cooperative MARL, the br policies intend to optimize the joint task-critic as well. Thus, the br policy is found by obtaining the gradient of the joint task critic with regard to the policy of the other agents after applying the current agent policy. Denoting the cost estimation from Equation 14 as GetCost, a single update step for agent i is sketched in Algorithm 1. The dedicated critic losses are denoted as CriticLoss and CostCriticUpdate in Algorithm 1. The native cost critic is regressed by calculating

Decentral update step for agent i:
- $\left(\mathrm{s}^{(i)},\mathrm{a}^{(i)},\underline{\mathrm{a}}^{(-i)},\underline{\mathrm{r}},\mathrm{d},\mathrm{s}'^{(i)}\right)\sim\mathcal{D}^{(i)}$ ▷ sample agent experience batch
- /* update (interactive) task critic $\mathrm{Q}_{\mathrm{int}}^{(i)}:=\mathcal{S}^{(i)}\times\underline{\mathcal{A}}\mapsto\mathbb{R}$, cf. (8) or (10) */
- $\mathrm{a}^{(i)\prime}\leftarrow{}^{\dagger}\pi\left(\mathrm{s}^{(i)\prime}\right)$ ▷ get next action
- $\underline{\mathrm{a}}^{(-i)\prime}\leftarrow{}^{\dagger}\underline{\pi}_{\mathrm{br}}^{(-i)}\left(\mathrm{s}^{(i)\prime},\mathrm{a}^{(i)\prime}\right)$ ▷ get best-response to next action
- $\mathscr{L}_{\mathrm{Q}_{\mathrm{int}}^{(i)}}\leftarrow\mathrm{CriticLoss}\left(\mathrm{s}^{(i)},\underline{\mathrm{a}},\mathrm{r}^{(i)},\mathrm{d},\mathrm{s}^{(i)\prime},\underline{\mathrm{a}}'\right)$
- /* update (native) agent critic $\mathrm{Q}_{\mathrm{nat}}^{(i)}:=\mathcal{S}^{(i)}\times\mathcal{A}^{(i)}\mapsto\mathbb{R}$, cf. (8) or (10) */
- $\hat{\mathrm{c}}_{\mathrm{nat}}{}^{(i)}\leftarrow\mathrm{GetCost}\left(\underline{\mathrm{r}}\right)$ ▷ estimate step-cost
- $\mathscr{L}_{\mathrm{Q}_{\mathrm{nat}}^{(i)}}\leftarrow\mathrm{CostCriticUpdate}\left(\mathrm{s}^{(i)},\mathrm{a}^{(i)},\hat{\mathrm{c}}_{\mathrm{nat}}{}^{(i)},\mathrm{d},\mathrm{s}^{(i)\prime},\mathrm{a}^{(i)\prime}\right)$
- /* update agent policy ($\pi^{(i)}:=\mathcal{S}^{(i)}\mapsto\mathcal{A}^{(i)}$) from critics, cf. (6), (7), (11) */
- $\mathscr{J}\left(\pi^{(i)}{}_{\Pi}\right)=\mathbb{E}_{\left(\mathrm{s}^{(i)},\underline{\mathrm{a}}^{(-i)}\right)}\left[\left(\nabla_{\mathrm{a}^{(i)}}\mathrm{Q}_{\mathrm{int}}^{(i)}\left(\mathrm{s}^{(i)},\underline{\mathrm{a}}\right)-\nabla_{\mathrm{a}^{(i)}}\mathrm{Q}_{\mathrm{nat}}^{(i)}\left(\mathrm{s}^{(i)},\mathrm{a}^{(i)}\right)\right)\Big|_{\mathrm{a}^{(i)}\leftarrow\pi^{(i)}{}_{\Pi}\left(\mathrm{s}^{(i)}\right)}\right]$
- /* update br-policy ($\underline{\pi}_{\mathrm{br}}^{(-i)}:=\mathcal{S}^{(i)}\times\mathcal{A}^{(i)}\mapsto\underline{\mathcal{A}}^{(-i)}$) from task-critic, cf. (7) */
- $\mathscr{J}\left(\underline{\pi}_{\mathrm{br}}^{(-i)}{}_{\Xi}\right)=\mathbb{E}_{\left(\mathrm{s}^{(i)},\mathrm{a}^{(i)}\right)}\left[\nabla_{\underline{\mathrm{a}}^{(-i)}}\mathrm{Q}_{\mathrm{int}}^{(i)}\left(\mathrm{s}^{(i)},\underline{\mathrm{a}}\right)\Big|_{\underline{\mathrm{a}}^{(-i)}\leftarrow\underline{\pi}_{\mathrm{br}}^{(-i)}{}_{\Xi}\left(\mathrm{s}^{(i)},\mathrm{a}^{(i)}\right)}\right]$

Algorithm 1. Decentralized br policy-based MARL update step for agent *i*. Due to the decentralized learning, the update step can be run in a fully parallelized procedure. For brevity, the exploration is omitted from this algorithm skeleton.

$$\begin{aligned}\mathrm{p}&=\hat{\mathrm{c}}_{\mathrm{nat}}{}^{(i)}+\gamma\left(1-\mathrm{d}\right)\left(\max_{j=1,2}{}^{\dagger}\mathrm{Q}_{\mathrm{nat},j}^{(i)}\left(\mathrm{s}^{(i)},\mathrm{a}^{(i)\prime}\right)\right),\\\mathrm{a}^{(i)\prime}&\leftarrow{}^{\dagger}\pi^{(i)}{}_{\Pi}\left(\mathrm{s}^{(i)\prime}\right)\end{aligned}\tag{15}$$

in Equation 8. Nonetheless, it has to be noted that Equation 15 uses the maximum critic value to account for the overestimation bias as the cost critic intends to minimize the accumulated native costs. We explicitly do not apply SAC for the cost critic as exploration should be emphasized on the task to be learned rather than exploring accumulated costs. In order to regress the joint task critic, a similar approach to existing AC approaches is followed. Querying $\mathrm{a}^{(i)\prime}$ identically to Equation 15, we set

$$\begin{aligned}\mathrm{p}&=\mathrm{r}^{(i)}\left(\mathrm{s}^{(i)},\underline{\mathrm{a}},\mathrm{s}^{(i)\prime}\right)+\gamma\left(1-\mathrm{d}\right)\left(\min_{j=1,2}{}^{\dagger}\mathrm{Q}_{\mathrm{int},j}^{(i)}\left(\mathrm{s}^{(i)\prime},\underline{\mathrm{a}}'\right)+\mathrm{p}_{\mathrm{SAC}}\right)\\\mathrm{a}^{(-i)\prime}&\leftarrow{}^{\dagger}\underline{\pi}_{\mathrm{br}}^{(-i)}{}_{\Xi}\left(\mathrm{s}^{(i)\prime},\mathrm{a}^{(i)\prime}\right)\\\mathrm{p}_{\mathrm{SAC}}&=\begin{cases}-\alpha\log\pi^{(i)}\left(\mathrm{a}^{(i)\prime}|\mathrm{s}^{(i)\prime}\right) & \text{for an SAC model}\\0 & \text{else}\end{cases},\end{aligned}\tag{16}$$

in Equation 8.

For brevity, Algorithm 1 only sketches the update, i.e., learning, step of our proposed multi-agent reinforcement learning (MARL) approach. During the exploration phase, each agent samples from their own policy and stores the actions and rewards of the other agents. In other words, each agent stores a tuple of $\left(\mathrm{s}^{(i)},\underline{\mathrm{a}},\underline{\mathrm{r}},\mathrm{d},\mathrm{s}^{(i)\prime},\right)$ in the replay buffer $\mathcal{D}^{(i)}$ until the end of an episode is reached or the task is completed. After collecting new data for a fixed number of episodes, Algorithm 1 is run for a fixed number of episodes.

Eventually, we distinguish between two interaction schemes in order to model $\pi^{(-i)}{}_{\Pi,\mathrm{br}}$. First, the other agents can be modeled as an unknown black-box system, usually denoted as a *game-against-nature* within game theory. Thus, a single policy is tracked as follows:

$$\underline{\pi}_{\mathrm{br}}^{(-i)}{}_{\Xi}:=\mathcal{S}^{(i)}\times\mathcal{A}^{(i)}\mapsto\left(\underline{\mathcal{A}}\right)_{j\in-i},\tag{17}$$

which models an interaction with the current agent and the *responsive* nature. Our second approach uses a dyadic interaction scheme and models the br policy of each agent to the current agent individually via Eq. 18.

$$\underline{\pi}_{\mathrm{br}\ \Xi}^{(-i)} \leftarrow \left(\pi_{\mathrm{br}\,\Xi}^{(j)} : = \mathcal{S}^{(i)} \times \mathcal{A}^{(i)} \mapsto \mathcal{A}^{(j)}\right)_{j \in -i}. \tag{18}$$

This requires to regress $N_{\mathfrak{A}} - 1$ opponent policies rather than a single policy in Equation 17. Both policies leverage the effect of a mutual interaction among the other agents to diminish combinatorial explosion.

5 Results

In this section, we evaluate the performance of our decentralized br policy MARL framework within the simulation environment from Appendix 1. Within this environment, we evaluated our algorithm against state-of-the-art algorithms within MARL, namely, MADDPG and multi-agent soft actor–critic (MASAC).

Given our adjusted multi-agent particle environment (MPE) as outlined in Appendix 1, we ran a decentralized version of TD3 (Ackermann et al., 2019) and the multi-agent version of Haarnoja (2018) for the joint critic in our algorithm. Given the dyadic and game-against-nature variants, we use the following notations:

- The state-of-the-art algorithms are directly denoted as commonly known in the literature, i.e., MADDPG and MASAC.
- Our extension of TD3 is denoted as br-TD3-dyad/nature.
- Our extension of SAC is denoted as br-SAC-dyad/nature.

As stated above, our main emphasis is set on improving the performance in sparsely rewarded environments. Furthermore, we explicitly tailor our approach to continuous action spaces in cooperative sections. Therefore, we applied the cooperative collection task, according to the parameterization in Appendix 2.

For brevity, we present the evaluation performance of the individual algorithms based on the average rewards of all agents in Figure 3. In here, the term *evaluation* refers to the agents greedily following their current policies than drawing samples from them. The collected results show a static version, i.e., using fixed goal locations, in the lower figure and a non-static version in the upper figure. In this environment, three agents update their policies over 5,000 exploration episodes. The evaluation is only run every 10th episode, so the number of evaluation steps is lower than the actual explorations. In addition, the averaged reward per evaluation run is logged, which, in return, strongly depends on a randomly sampled starting state of the agent and the goal locations in the non-static environment. As a result, the collected data encounter high noise, which is reduced by smoothing the collected reward values using a Savitzky–Golay filter (Savitzky and Golay, 1964) and the implementation by Virtanen et al. (2020).

As it can be seen, our algorithms outperform the current state-of-the-art algorithms not only in terms of the final performance but also in terms of convergence speed for both scenarios. Unsurprisingly, our method performs best in static environments, requiring reaching static goal locations. In these scenarios, there is a direct relation between the agent states and the actual value functions, which leads to an improved learning speed.

For a closer evaluation of our presented algorithms, the per-agent rewards metrics are listed in Table 1. Furthermore, the number of total successful trials per algorithm during exploration and evaluation is listed. Exploration is not only run distinctly more often but also contains double the amount of steps per run. As a consequence, the number of successful exploration runs is distinctly higher compared to the evaluation numbers.

Nonetheless, the collected numbers underline that our presented method outperforms current state-of-the-art methods distinctly, not only in terms of averaged accumulated rewards, as shown in Figure 3, but also for each individual agent involved. The performance increase becomes evident on comparing the success rates of the algorithms, where MASAC failed completely to find a successful policy.

Comparing the overall results, the TD3 agents outperformed not only the state-of-the-art methods but also our SAC variants. Furthermore, the dyadic setup resulted in improved performance for all evaluation metrics compared to the game-against-nature schematic representation. This confirms our initial statement that it is preferable to handle interactions individually rather than regressing interaction schemes fully from a NN.

Regarding the standard deviations of our proposed methods, it also becomes evident that our methods suffer from higher variance in the accumulated rewards. Even though this may seem like a disadvantage of our approach compared to the existing algorithms, it has to be noted that the obtained reward from a successful episode usually distinctly differs from the reward obtained from an unsuccessful episode. Therefore, the increased variance is not subject to our method but a consequence of the increased success rates, as highlighted in Table 1, where our algorithm achieves distinctly higher success rates than the existing AC methods.

In summary, it can be stated that our presented algorithm outperforms the existing methods within our simulated environments even though they are run fully decentralized.

6 Discussion and technical extensions

Given our presented algorithm, we conclude this article with an outline of possible extensions in order to further improve the overall performance.

6.1 Applying best-response policies on competitive environments

A current drawback of our algorithm is the restriction to competitive domains. If the agents have access to all reward values during learning, an additional critic for the objective of the other agent can be added to the presented algorithm. This results in applying the gradient step for the br policy not only over the joint task critic for the current agent but also the agent-specific agent critic. If this metric is applied, applying the dyadic interaction scheme from above is strongly recommended as our algorithm is restricted to optimizing the average reward over all agents otherwise.

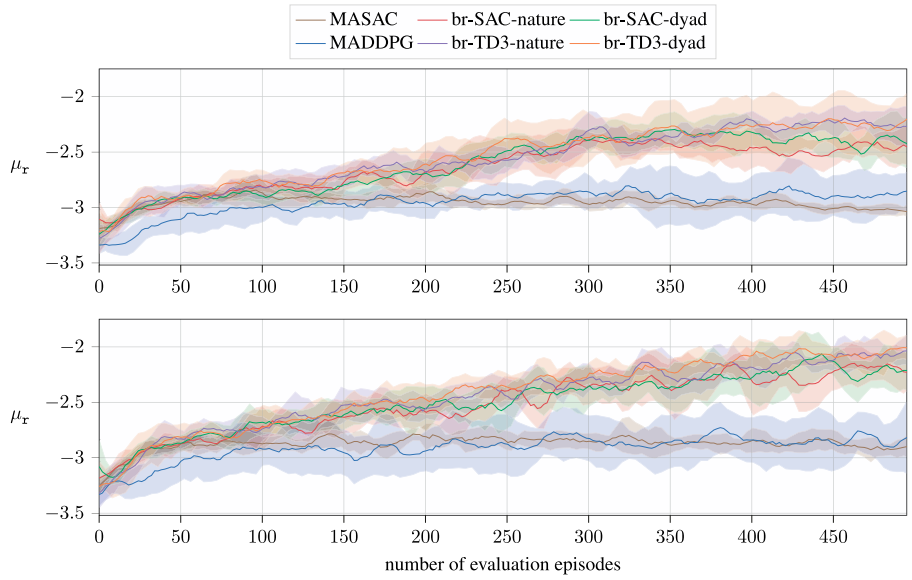

FIGURE 3
Results of the decentralized br-based algorithms for the cooperative collection task using sparse rewards. The figures present averaged rewards of all agents over eight learning runs per algorithm and environment. The shaded areas highlight a CI of 70%. The upper figure shows the performance of the collection task with static goal locations, whereas the environment on the bottom samples new goal locations upon every reset. The *x*-axis denotes the evaluation steps, which are run after 10 exploration episodes to evaluate the current performance.

TABLE 1 Detailed performance metrics for evaluated environments. The results of the static environment are listed on the bottom. The values show the averaged results with the optional standard deviation appended by ±. The terms dyadic and nature are abbreviated by their first letter for brevity. Similarly, the number of successful trials of the exploration and evaluation runs are denoted as d_{ex} and d_{ev}.

	Static	MADDPG	MASAC	br-TD3-d	br-TD3-n	br-SAC-d	br-SAC-n
$\mathrm{r}^{(1)}$		−2.92 ± 0.38	−2.95 ± 0.24	**−2.54 ± 0.48**	−2.55 ± 0.47	−2.6 ± 0.43	−2.64 ± 0.42
$\mathrm{r}^{(2)}$		−2.94 ± 0.4	−2.96 ± 0.23	**−2.54 ± 0.48**	−2.57 ± 0.47	−2.6 ± 0.43	−2.64 ± 0.43
$\mathrm{r}^{(3)}$		−2.99 ± 0.44	−2.95 ± 0.23	**−2.54 ± 0.48**	−2.55 ± 0.46	−2.61 ± 0.44	−2.64 ± 0.43
d_{ev}		3	0	**35**	31	12	10
d_{ex}		86	62	**1121**	901	256	246
$\mathrm{r}^{(1)}$	✓	−2.89 ± 0.46	−2.87 ± 0.3	**−2.4 ± 0.51**	−2.41 ± 0.51	−2.5 ± 0.52	−2.51 ± 0.53
$\mathrm{r}^{(2)}$	✓	−2.97 ± 0.5	−2.87 ± 0.3	**−2.42 ± 0.51**	−2.44 ± 0.52	−2.5 ± 0.52	−2.51 ± 0.53
$\mathrm{r}^{(3)}$	✓	−2.88 ± 0.44	−2.87 ± 0.3	**−2.41 ± 0.51**	−2.44 ± 0.52	−2.5 ± 0.52	−2.52 ± 0.53
d_{ev}	✓	7	0	50	**52**	31	41
d_{ex}	✓	289	68	**1391**	1284	554	481

The best performing values are highlighted in bold.

Another extension is given by modeling non-cooperative agent(s). In order to model this procedure, it is best to condition non-cooperative agents on the joint team policy of all cooperative agents, thus leading to the conditional interaction policy as follows:

$$\underline{\pi}(\underline{\mathrm{s}}) \approx \underline{\pi}^{(-i,-i)}\left(\mathrm{s}^{(i)}|\underline{\mathrm{a}}^{(-i,i)}, \mathrm{a}^{(i)}\right)\underline{\pi}^{(-i,i)}\left(\mathrm{s}^{(i)}|\mathrm{a}^{(i)}\right)\underline{\pi}^{(i)}\left(\mathrm{s}^{(i)}\right), \tag{19}$$

In Eq. 19 the cooperative policy is denoted as $\underline{\pi}^{(-i,i)}\left(\mathrm{s}^{(i)}|\mathrm{a}^{(i)}\right)$ and the non-cooperative policy is denoted as $\underline{\pi}^{(-i,-i)}\left(\mathrm{s}^{(i)}|\underline{\mathrm{a}}^{(-i,i)}, \mathrm{a}^{(i)}\right)$. Alternatively, on-policy-based approaches, such as proximal policy optimization (PPO) or trust region policy optimization, are worth an investigation to model the behavior of other agents. Here, a direct approach is given by conditioning the policy estimate on the average over all estimators. A more promising approach would be given by averaging over all agent advantages and, thus, applying a gradient step. This bears the potential of stabilizing the estimated opponent models and, thus, the

overall task-critic updates, which eventually increases the likelihood of converging to the team-optimal policy. Furthermore, there is great potential on incorporating the findings of Jaques et al. (2019) and, thus, not only regressing the agents' policies but also directly conditioning the reward of each agent on the influencing reward.

6.2 Improving convergence behavior by partially centralized learning

The presented method fully decouples learning by learning opponent models without applying centralized learning schemes. This is endangered to leading to divergent agent behavior and thus converging to suboptimal team-behavior. Therefore, our current method could be further enhanced by introducing centralized learning without adding restrictive full observability assumptions. Rather than sharing the full observations, the individual opponent policy predictions can be shared during learning such that the policy gradient can be conditioned on the Kullback–Leibler (KL)-divergence of the predicted opponent policies.

$$\mathtt{a}^{(j)} \leftarrow {\pi^{(j)}}_{\Pi,\mathrm{br}}\left(\mathtt{s}^{(i)}, \mathtt{a}^{(i)}\right)$$
$$\mathcal{J}\left(\pi^{(j)}_{\Pi,\mathrm{br}}\right) = \mathbb{E}_{\mathtt{s}^{(i)},\mathtt{a}^{(i)}}\left[\nabla_{\mathtt{a}^{(-i)}} \mathtt{Q}^{(i)}_{\mathrm{int}}\left(\mathtt{s}^{(i)}, \underline{\mathtt{a}}\right)\Big|_{\mathtt{a}^{(j)}} - \frac{1}{N_{\mathfrak{A}}-2}\sum_{\substack{k=1\\k\neq j\\k\neq i}}^{N_{\mathfrak{A}}} D_{\mathrm{KL}}\left(\pi^{(j)}||\pi^{(k)}\right)\right]. \tag{20}$$

Eventually, a baseline policy, e.g., obtained from behavioral cloning, can be substituted into Equation 20 to further stabilize the gradients.

6.3 Multi-robot hierarchical actor critic

Recalling the motivation of our algorithm in Section 4, a major advantage of our algorithm is the possibility of applying asynchronous actions. As this directly allows extending AC for MARL to HRL, we outline a conceptual extension of our approach tailored to RL tasks with sparse rewards. This extension relies on a collection of assumptions, which is also assumed in Levy et al. (2019):

- There exists an agent-specific state space $\mathcal{X}^{(i)} \in \mathcal{S}^{(i)}$, and $\mathtt{x}^{(i)} \in \mathtt{s}^{(i)}$ always holds[1].
- There exist deterministic mapping functions $\mathcal{F}_g := \mathcal{X}^{(i)} \times {}^{\{\mathfrak{p}\}}\mathcal{A}^{(i)} \mapsto {}^{\{\mathfrak{p}-1\}}\mathcal{G}^{(i)} \in \mathcal{X}^{(i)}$ and $\mathcal{F}_g^{-1} := \mathcal{X}^{(i)} \times {}^{\{\mathfrak{p}-1\}}\mathcal{G}^{(i)} \mapsto {}^{\{\mathfrak{p}\}}\mathcal{A}^{(i)}$, which map the actions of the upper layer to the goal space of the lower layer and *vice versa* for each agent.
- There exists a deterministic evaluation metric ${}^{\{\mathfrak{p}\}}\mathcal{J} := {}^{\{\mathfrak{p}\}}\mathcal{G} \times \mathcal{X}^{(i)} \mapsto [0,1]$ that evaluates the achievement of a goal ${}^{\{\mathfrak{p}\}}g^{(i)}$, given the current agent state $\mathcal{X}^{(i)}$.

1 Assumption of $\mathtt{x}^{(i)} \in \mathtt{s}^{(i)}$ emphasizes that we do not expect full state observability and that the intrinsic observations do not provide additional knowledge to the robotic agents.

This differs from the original assumption by Levy et al. (2019) because we propose to explicitly distinguish between the internal agent state $\mathtt{x}^{(i)}$ and the full environment observation $\mathtt{s}^{(i)}$.

We claim that within multi-agent HRL, it is specifically beneficial to distinguish between internal and external observations. Therefore, we propose to use structured observations as follows:

$$\mathtt{s}^{(i)} := \left(\mathtt{x}, \mathtt{y}_{\mathfrak{e}}, \mathtt{y}_{(-i)}\right)_{i\in N_{\mathfrak{A}}}, \tag{21}$$

In Eq. 21 $\mathtt{x}^{(i)}$ reflects the internal state of an agent, e.g., current position, velocity, etc., and environmental observations $\mathtt{y}^{(i)}_{\mathfrak{e}}$ from $\mathfrak{A}^{(i)}$'s perspective, e.g., images or laser range data, as well as observations of the other agents $\mathtt{y}^{(i)}_{(-i)}$.

Given this representation, we propose a two-layered hierarchy where the upper layer proposes sub-goals to the lower layer agents. This lower level is denoted as the *environment-layer* or ${}^{\{\mathfrak{e}\}}\mathfrak{p}$ in the following, while the upper layer is denoted as the *team-coordination* layer or ${}^{\{\mathfrak{i}\}}\mathfrak{p}$. On the lowest layer, our decentralized MARL approach based on br policies from Section 4.1 can be applied using a dyadic interaction scheme.

Therefore, the individual components per agent are given as follows:

- A joint task critic $\mathtt{Q}^{(i)}_{\mathrm{int}}\left(\mathtt{s}^{(i)}, \underline{\mathtt{a}}\right)$.
- A native hierarchical critic $\mathtt{Q}^{(i)}_{\mathrm{nat}}\left(\mathtt{x}^{(i)}, \mathtt{y}^{(i)}_{\mathfrak{e}}, g^{(i)}\right)$.
- A goal-conditioned action policy for the current agent $\pi^{(i)}\left(\mathtt{s}^{(i)}, g^{(i)}\right)$.
- The dyadic br policies $\left(\pi^{(j)}\left(\left(\mathtt{x}, \mathtt{y}_{\mathfrak{e}}, \mathtt{y}_{(-i)}, \mathtt{a}\right)_{i\in N_{\mathfrak{A}}}\right)\right)_{j\in -i}$.

As the individual agents are provided with a sub-task that is to be reached by the dedicated agents alone, the hierarchical native critic preferably drops the dependency on the observation of other agents. In other words, this native hierarchical critic evaluates the hierarchically imposed rewards instead of estimating the current step-cost from the deviation with regard to the average reward. As a result, the update step of the lower layer follows Algorithm 1 but replaces the difference in line 9 by an average over the two critics. Furthermore, the native critic not only evaluates the environmental task success $\mathtt{d}$ but also evaluates if the update step has accomplished the current sub-goal. As the rest remains identical to Algorithm 1 and Equations 15 and 16, we omit repeating the same equations.

In contrast, the upper layer only tracks a single critic as infeasible sub-goals result in unpredictable task performance. Unfortunately, the agents do not have access to the goal mapping of other agents such that it is impossible to directly impose their policies or higher-level actions in the critic within decentralized settings. Furthermore, the upper layer usually suffers from asynchronous decisions, which would require adding the decision epochs to the critic's state to allow sampling from the experience buffer. Therefore, we propose to apply an observation oracle instead of a br policy,

$${}^{\{\mathfrak{i}\}}\pi^{(j)}_{\mathrm{br}} := \left(\mathcal{X} \times \mathcal{Y}_{\mathrm{env}} \times \mathcal{Y}^{(j)} \times {}^{\{\mathfrak{i}\}}\mathtt{a}\right)_{i\in N_{\mathfrak{A}}} \mapsto \mathcal{Y}^{(j)}, \tag{22}$$

i.e., instead of predicting the agent action on the upper layer, the next observation is predicted. In case a (partially) centralized learning

scheme is applied, this observation oracle can also be replaced with the following:

$$^{\{\mathrm{i}\}}\pi_{\mathrm{br}}^{(j)} := \left(\mathcal{X} \times \mathcal{Y}_{\mathrm{env}} \times \mathcal{X}^{(j)} \times {}^{\{\mathrm{i}\}}\mathrm{a}\right)_{i \in N_{\mathfrak{A}}} \mapsto \mathcal{X}^{(j)}, \tag{23}$$

thus predicting the next internal state of the agent j. These opponent models from Eqs 22, 23 allow using data from an experience buffer independent of the higher-level policies or decision epochs during execution. As a result, the (interactive) task critic of the upper layers are regressed from Eqs 24 or 25:

$$^{\{\mathrm{i}\}}\mathrm{Q}_{\mathrm{int}}^{(i)} := \mathcal{S}^{(i)} \times \mathcal{A}^{(i)} \times \left(\mathcal{Y}_{(j)}^{(i)}\right)_{j \in -i} \mapsto \mathbb{R} \tag{24}$$

$$^{\{\mathrm{i}\}}\mathrm{Q}_{\mathrm{int}}^{(i)} := \mathrm{s} \times \mathrm{a}^{(i)} \times \left(\mathcal{X}_{(j)}^{(i)}\right)_{j \in -i} \mapsto \mathbb{R} \tag{25}$$

Although the lower layer is updated similarly to Algorithm 1, the lower-layer critic update is given by calculating the following:

$$\begin{aligned} \mathrm{p} = & \frac{1}{^{\{\mathrm{i}\}}T_{\max}} \sum_{n=1}^{^{\{\mathrm{i}\}}T_{\max}} \gamma^{^{\{\mathrm{i}\}}T_{\max}-n} \mathrm{r}^{(i)}\left(\mathrm{s}_{t+n}, \underline{\mathrm{a}}_{t+n}, \mathrm{s}_{t+n+1}\right) \\ & + (1-\mathrm{d}) \begin{cases} 0 & \text{if } \mathcal{J}\left(\mathrm{x}^{(i)\prime}, \mathcal{F}_g\left(\mathrm{x}^{(i)}, {}^{\{\mathrm{i}\}}\mathrm{a}^{(i)}\right)\right) \mapsto \top \\ -(T_{\max}-n) & \text{if } \exists \quad n{:}\mathcal{J}\left({}_{t+n+1}\mathrm{x}^{(i)}, \mathcal{F}_g\left(\mathrm{x}^{(i)}, {}^{\{\mathrm{i}\}}\mathrm{a}^{(i)}\right)\right) \mapsto \top \\ -T_{\max} & \text{else} \end{cases} \\ & + \gamma(1-\mathrm{d})\left(\min_{k=1,2} {}^{\{\mathrm{i}\}}\mathrm{Q}_{\mathrm{int},k}{}^{(i)}\left(\mathrm{s}^{(i)\prime}, {}^{\{\mathrm{i}\}}\mathrm{a}^{(i)\prime}, \left(\mathrm{y}_{(j)}^{(i)\prime}\right)_{j \in -i}\right)\right), \end{aligned} \tag{26}$$

where $^{\{\mathrm{i}\}}T_{\max}$ denotes the number of maximum sub-steps for a hierarchical transition and $\mathrm{s}' := \mathrm{s}_{t+^{\{\mathrm{i}\}}T_{\max}}$ represents the observation of agent i after a hierarchical update step. The actions and observations for the target critic in Equation 26 are obtained via the proposed policies from Eq. 27

$$\begin{aligned} & {}^{\{\mathrm{i}\}}\mathrm{a}^{(i)\prime} \leftarrow {}^{\{\mathrm{i}\}\natural}\pi_{\Pi}^{(i)}\left(\mathrm{s}^{(i)\prime}\right) \\ & \left(\mathrm{y}' \leftarrow {}^{\{\mathrm{i}\}\natural}\pi_{\mathrm{br}\,\Xi}^{(j)}\left(\mathrm{s}^{(i)\prime}, {}^{\{\mathrm{i}\}}\mathrm{a}^{(i)\prime}\right)\right)_{j \in -i} \end{aligned}. \tag{27}$$

The first term in Equation 26 averages the environmental reward, while the second adds the hierarchical penalty term depending on whether the lower layer could achieve the current action or the respective sub-goal. Eventually, the value function is approximated via querying the current higher-level policy and predicting the observations of the other agents.

7 Conclusion

Within this article, we have proposed a novel MARL framework that allows decentralized learning while differentiating between agent-based native costs and joint task rewards. In order to regress the team-optimal policy, each agent not only updates their own policy but also instead models the team-optimal response by estimating the br policy of the other agents. We propose to employ concepts from game theory, namely, either applying dyadic br policies for each agent pair or representing the collection of other agents as a game against nature. Even though our method relies on estimates of the agent-based costs, it outperforms recent state-of-the-art methods in terms of convergence speed within sparsely rewarded environments.

Given the promising results collected in simulation, this article provides a variety of extensions, which bear great potential for future research. First, an extension to competitive domains, i.e., zero-sum games, has been sketched by minimizing instead of maximizing the agent task critic when updating the other agents' policies. Second, we outlined that sharing the predicted br policies can improve the convergence. Eventually, we sketched an extension of our method to a hierarchical MARL algorithm as this may allow bootstrapping performance during learning.

Using such a hierarchical MARL algorithm combined with structured environment observations bears the additional advantage of explicitly incorporating available model knowledge. This allows leveraging the concept of full end-to-end learning and instead combines MARL with optimization-based techniques, which remains a rarely covered field of research.

Data availability statement

The original contributions presented in the study are publicly available. This data can be found here: https://gitlab.com/vg_tum/multi-agentgym.git; https://gitlab.com/vg_tum/mahac_rl.git.

Author contributions

VG proposed, implemented, and outlined the methods presented in the article, performed the experiments, and evaluated the collected evidence. VG and DW verified the approach. All authors contributed to the article and approved the submitted version.

Funding

The research leading to the results presented in this work received funding from the Horizon 2020 research and innovation programme under grant agreement No. 820742 of the project "HR-Recycler—Hybrid Human–Robot RECYcling plant for electriCal and eLEctRonic equipment."

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Ackermann, J., Gabler, V., Osa, T., and Sugiyama, M. (2019). Reducing overestimation bias in multi-agent domains using double centralized critics. CoRR abs/1910.01465 Available at: https://arxiv.org/abs/1910.01465.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. (2017). Deep reinforcement learning: a brief survey. *IEEE Signal Process. Mag.* 34, 26–38. doi:10.1109/MSP.2017.2743240

Bellman, R. (1957). A markovian decision process. *J. Math. Mech.* 6, 679–684. doi:10.1512/iumj.1957.6.56038

Englert, P., and Toussaint, M. (2016). "Combined optimization and reinforcement learning for manipulation skills," in *Robotics: science and systems (RSS)*. Editors D. Hsu, N. M. Amato, S. Berman, and S. A. Jacobs doi:10.15607/RSS.2016.XII.033

Foerster, J. N., Assael, Y. M., de Freitas, N., and Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. CoRR abs/1605.06676 Available at: https://arxiv.org/abs/1605.06676.

Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. (2018). "Counterfactual multi-agent policy gradients," in *AAAI conference on artificial intelligence*. Editors S. A. McIlraith, and K. Q. Weinberger (New Orleans, LA: AAAI Press), 2974–2982. Available at: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17193.

Haarnoja, T. (2018). *Acquiring diverse robot skills via maximum entropy deep reinforcement learning*. Berkeley, USA: University of California. Ph.D. thesis.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., et al. (2018). *Soft actor-critic algorithms and applications*. CoRR abs/1812.05905 https://arxiv.org/abs/1812.05905.

Havrylov, S., and Titov, I. (2017). Emergence of language with multi-agent games: learning to communicate with sequences of symbols. CoRR abs/1705.11192 Available at: https://arxiv.org/abs/1705.11192.

He, Z., Dong, L., Song, C., and Sun, C. (2021). Multi-agent soft actor-critic based hybrid motion planner for mobile robots. CoRR abs/2112.06594 Available at: https://arxiv.org/abs/2112.06594.

Hernandez-Leal, P., Kartal, B., and Taylor, M. E. (2019). A survey and critique of multiagent deep reinforcement learning. *Auton. Agents Multi Agent Syst.* 33, 750–797. doi:10.1007/s10458-019-09421-1

Hernandez-Leal, P., Kartal, B., and Taylor, M. E. (2020). A very condensed survey and critique of multiagent deep reinforcement learning. *AAMAS*. Editor A. E. F. Seghrouchni, G. Sukthankar, B. An, and N. Yorke-Smith (International Foundation for Autonomous Agents and Multiagent Systems), 2146–2148. Available at: https://dl.acm.org/doi/10.5555/3398761.3399105.

Iqbal, S., and Sha, F. (2019). "Actor-attention-critic for multi-agent reinforcement learning," in *International conference on machine learning (ICML), (PMLR), vol. 97 of proceedings of machine learning research*. Editors K. Chaudhuri, and R. Salakhutdinov, 2961–2970.

Jaques, N., Lazaridou, A., Hughes, E., Gülçehre, Ç., Ortega, P. A., Strouse, D., et al. (2019). "Social influence as intrinsic motivation for multi-agent deep reinforcement learning," in *Proceedings of the 36th international conference on machine learning, ICML 2019, 9-15 june 2019, long beach, California, USA vol. 97 of proceedings of machine learning research*. Editors K. Chaudhuri, and R. Salakhutdinov (PMLR, 3040–3049.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: a survey. *J. Artif. Intell. Res.* 4, 237–285. doi:10.1613/jair.301

Kingma, D. P., and Ba, J. (2015). "Adam: a method for stochastic optimization," in *International conference on learning representations (ICLR)*. Editors Y. Bengio, and Y. LeCun

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: a survey. *J. Artif. Intell. Res.* 32, 1238–1274. doi:10.1177/0278364913495721

Kolter, J. Z., and Ng, A. Y. (2009). "Policy search via the signed derivative," in *Robotics: science and systems (RSS)*. doi:10.15607/R_C_RSS.2009.V.027

Laurent, G. J., Matignon, L., and Fort-Piat, N. L. (2011). The world of independent learners is not markovian. *Int. J. Knowl. Based Intell. Eng. Syst.* 15, 55–64. doi:10.3233/KES-2010-0206

Levy, A., Konidaris, G., Jr., and Saenko, K. (2019). "Learning multi-level hierarchies with hindsight," in International Conference on Learning Representations (ICLR), Louisiana, United States, May 6 - May 9, 2019.

Li, S., Wu, Y., Cui, X., Dong, H., Fang, F., and Russell, S. J. (2019). "Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient," in AAAI Conference on Artificial Intelligence, Honolulu, Hawaii, USA, January 27 – February 1, 2019, 4213–4220.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2016). "Continuous control with deep reinforcement learning," in International Conference on Learning Representations (ICLR), Puerto, Rico, May 2-4, 2016. Available at: http://arxiv.org/abs/1509.02971.

Littman, M. L. (1994). "Markov games as a framework for multi-agent reinforcement learning," in International Conference on Machine Learning (ICML), New York, New York, USA, 20-22 June 2016, 157–163. doi:10.1016/b978-1-55860-335-6.50027-1

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. (2017). "Multi-agent actor-critic for mixed cooperative-competitive environments," in Annual Conference on Neural Information Processing Systems (NeurIPS), Long Beach, CA, USA, December 4-9, 2017, 6379–6390. Available at: http://papers.nips.cc/paper/7217-multi-agent-actor-critic-for-mixed-cooperative-competitive-environments.

McCloskey, M., and Cohen, N. J. (1989). "Catastrophic interference in connectionist networks: the sequential learning problem," in *Psychology of learning and motivation vol. 24* (Amsterdam, Netherlands: Elsevier), 109–165.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature* 518, 529–533. doi:10.1038/nature14236

Mordatch, I., and Abbeel, P. (2017). Emergence of grounded compositional language in multi-agent populations. CoRR abs/1703.04908 Available at: https://arxiv.org/abs/1703.04908.

Nash, J. (1950). Equilibrium points in N-person games. *Proc. Natl. Acad. Sci.* 36, 48–49. doi:10.1073/pnas.36.1.48

Ng, A. Y., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., et al. (2004). "Autonomous inverted helicopter flight via reinforcement learning," in International Symposium on Experimental Robotics (ISER), Singapore, June 18–21, 2004. January 2006, 363–372. doi:10.1007/11552246_35

Nguyen, T. T., Nguyen, N. D., and Nahavandi, S. (2020). Deep reinforcement learning for multiagent systems: a review of challenges, solutions, and applications. *IEEE Trans. Cybern.* 50, 3826–3839. doi:10.1109/TCYB.2020.2977374

Savitzky, A., and Golay, M. J. (1964). Smoothing and differentiation of data by simplified least squares procedures. *Anal. Chem.* 36, 1627–1639. doi:10.1021/ac60214a047

Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., and Moritz, P. (2015). "Trust region policy optimization," in *International conference on machine learning (ICML) vol. 37 of JMLR workshop and conference proceedings*. Editors F. R. Bach, and D. M. Blei, 1889–1897. JMLR.org.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. CoRR abs/1707.06347 Available at: https://arxiv.org/abs/1707.06347.

Shapley, L. S. (1952). *A value for N-person games*. Santa Monica, CA: RAND Corporation. doi:10.7249/P0295

Shapley, L. S. (1953). Stochastic games. *Proc. Natl. Acad. Sci.* 39, 1095–1100. doi:10.1073/pnas.39.10.1953

Sheikh, H. U., and Bölöni, L. (2020). "Multi-agent reinforcement learning for problems with combined individual and team reward," in IEEE nternational Joint Conference on Neural Networks (IJCNN) (IEEE), Glasgow, United Kingdom, July 19-24, 2020, 1–8. doi:10.1109/IJCNN48605.2020.9206879

Shoham, Y., and Leyton-Brown, K. (2008). *Multiagent systems: algorithmic, game-theoretic, and logical foundations*. New York, NY, USA: Cambridge University Press.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 484–489. doi:10.1038/nature16961

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. A. (2014). "Deterministic policy gradient algorithms," in International Conference on Machine Learning (ICML), Beijing, China, June 21–June 26, 2014, 387–395. Available at: http://proceedings.mlr.press/v32/silver14.html.

Song, H. F., Abdolmaleki, A., Springenberg, J. T., Clark, A., Soyer, H., Rae, J. W., et al. (2020). "V-MPO: on-policy maximum *a posteriori* policy optimization for discrete and continuous control," in International Conference on Learning Representations (ICLR) (OpenReview.net), Ababa, Ethiopia, April 26-30, 2020.

Sun, Y., Lai, J., Cao, L., Chen, X., Xu, Z., and Xu, Y. (2020). A novel multi-agent parallel-critic network architecture for cooperative-competitive reinforcement learning. *IEEE Access* 8, 135605–135616. doi:10.1109/ACCESS.2020.3011670

Sutton, R. S., McAllester, D. A., Singh, S., and Mansour, Y. (1999a). "Policy gradient methods for reinforcement learning with function approximation," in *Annual conference on neural information processing systems (NeurIPS)*. Editors S. A. Solla, T. K. Leen, and K. Müller (Massachusetts, United States: The MIT Press), 1057–1063.

Sutton, R. S., Precup, D., and Singh, S. (1999b). Between mdps and semi-mdps: a framework for temporal abstraction in reinforcement learning. *Artif. Intell.* 112, 181–211. doi:10.1016/S0004-3702(99)00052-1

Tan, M. (1993). "Multi-agent reinforcement learning: independent versus cooperative agents," in *International conference on machine learning (ICML)*.

Editor P. E. Utgoff (Massachusetts, United States: Morgan Kaufmann), 330–337. doi:10.1016/b978-1-55860-307-3.50049-6

Tang, H., Wang, A., Xue, F., Yang, J., and Cao, Y. (2021). A novel hierarchical soft actor-critic algorithm for multi-logistics robots task allocation. *IEEE Access* 9, 42568–42582. doi:10.1109/ACCESS.2021.3062457

Tian, Z., Wen, Y., Gong, Z., Punakkath, F., Zou, S., and Wang, J. (2019). "A regularized opponent model with maximum entropy objective," in *Proceedings of the twenty-eighth international joint conference on artificial intelligence, IJCAI 2019, Macao, China, august 10-16, 2019*. Editor S. Kraus, 602–608. ijcai.org. doi:10.24963/ijcai.2019/85

van der Wal, J. (1980). *Stochastic dynamic programming*. Amsterdam, Netherlands: Methematisch Centrum. Ph.D. thesis.

van Hasselt, H. (2010). "Double q-learning," in *Annual conference on neural information processing systems (NeurIPS)*. Editors J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta (New York, United States: Curran Associates, Inc.), 2613–2621.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., et al. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 350–354. doi:10.1038/s41586-019-1724-z

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., et al. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat. Methods* 17, 261–272. doi:10.1038/s41592-019-0686-2

Wei, E., Wicke, D., Freelan, D., and Luke, S. (2018). "Multiagent soft q-learning," in AAAI Spring Symposia (AAAI Press), California, USA, March 26-28, 2018.

Wu, X., Li, X., Li, J., Ching, P. C., Leung, V. C. M., and Poor, H. V. (2021). Caching transient content for iot sensing: multi-agent soft actor-critic. *IEEE Trans. Commun.* 69, 5886–5901. doi:10.1109/TCOMM.2021.3086535

Yang, Y., and Wang, J. (2020). An overview of multi-agent reinforcement learning from game theoretical perspective. CoRR abs/2011.00583 Available at: https://arxiv.org/abs/2011.00583.

Zhang, K., Yang, Z., and Başar, T. (2021). Multi-agent reinforcement learning: a selective overview of theories and algorithms. *Handb. Reinf. Learn. Control*, 321–384. doi:10.1007/978-3-030-60990-0_12

Zhang, Q., Dong, H., and Pan, W. (2020). "Lyapunov-based reinforcement learning for decentralized multi-agent control," in *International conference on distributed artificial intelligence (DAI) vol. 12547 of lecture notes in computer science*. Editors M. E. Taylor, Y. Yu, E. Elkind, and Y. Gao (Berlin, Germany: Springer), 55–68. doi:10.1007/978-3-030-64096-5_5

Appendix 1

The proposed algorithm has been evaluated on the MPE that has been extended from previous work (Lowe et al., 2017; Mordatch and Abbeel, 2017) to fit the scope of this article. The source code of the benchmark scenarios[2] and the presented work[3] can be found online, while the hyper-parameters and further implementation details leading to the results are listed in Appendix 2.

In order to meet the assumptions stated in Section 4, the original simulation environment has been adjusted such that the agents are able to differentiate between internal, external agent-related, and external environment-based observations. Thus, we introduce structured observations for our adjusted version of the MPE. Furthermore, the goal-mapping and evaluation metrics stated in Section 4 have been handcrafted and embedded into the dedicated environments similar to the original work of Levy et al. (2019). As claimed in Section 4, our approach tackles cooperative multi-robot RL tasks, such that only environments with pure continuous action spaces have been tested. Before outlining the experimental findings collected, we highlight the adjustments added to the default gym environment and the MPE.

A.1 Structured observations in the multi-agent particle environment

In order to ease the implementation of the presented extensions in Section 6, our adjusted MPE directly allows obtaining the observation of each agent as $\mathrm{s}^{(i)} := \left(\mathrm{x}^{(i)}, \mathrm{y}_{\mathfrak{e}}^{(i)}, \mathrm{y}_{(-i)}^{(i)}\right)$. The MPE is characterized by $N_{\mathfrak{A}}$ agents moving in a xy-planar surface by applying a force on their body center. Thus, each agent is implemented as a point-mass, where the internal state and action are defined as

$$\mathrm{x}^{(i)} := \begin{bmatrix} x^{(i)} \\ y^{(i)} \\ \dot{x}^{(i)} \\ \dot{y}^{(i)} \end{bmatrix} \quad \mathrm{a}^{(i)} := \begin{bmatrix} f_x^{(i)} \\ f_y^{(i)} \end{bmatrix}, \tag{28}$$

where the action is a planar force[4] actuated on the individual point-masses, which then follows the linear point-mass dynamics

$$\mathrm{x}^{(i)\prime} := \begin{bmatrix} 1 & 0 & \mathrm{d}t & 0 \\ 0 & 1 & 0 & \mathrm{d}t \\ 0 & 0 & \mathfrak{v} & 0 \\ 0 & 0 & 0 & \mathfrak{v} \end{bmatrix} \begin{bmatrix} x^{(i)} \\ y^{(i)} \\ \dot{x}^{(i)} \\ \dot{y}^{(i)} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{\mathfrak{m}} & 0 \\ 0 & \frac{1}{\mathfrak{m}} \end{bmatrix} \mathrm{a}^{(i)}, \tag{29}$$

2 Source code: https://gitlab.com/vg_tum/multi-agent-gym.git

3 Source code: https://gitlab.com/vg_tum/mahac_rl.git

4 Various implementations available online realize the action input as the difference of two positive force terms as this eases the comparison to discrete action spaces, where the result equals learning an optimal bang–bang controller. As our framework explicitly highlights continuous applications, we kept this implementation for the comparison to the state-of-the-art methods but used the interfaces from Eqs 28, 29 for our method.

TABLE A1 Hyper-parameters for the experimental evaluation and all evaluated algorithms.

Parameter	Value
Batch size	1,024
Polyak value for the target net update	0.95
Decay parameter γ	0.95
Exploration episode length	200
Evaluation episode length	100
Number of episodes	5,000
Number of update steps	5
Update rate	every fifth episode
Episode buffer size	1,000
Number of agents $N_{\mathfrak{A}}$	3
Polynomial filter order Savitzky and Golay (1964)	3
Filter window size Savitzky and Golay (1964)	31

TABLE A2 Environment parameters for the MPE and *cooperative collection* task.

Parameter	Value
$\mathrm{d}t$	0.02 s
$N_{\mathfrak{A}}$	3
Goal threshold $\zeta_{g,\mathrm{MPE}}$ in Eq. 30	0.02 m
Collision cost	1
$\mathfrak{v}$	0.9
$\mathfrak{m}$	1kg
$\boldsymbol{v}_{\max}$	1.0 m/s
Number of collision objects	0

using the mass of the entity $\mathfrak{m}$ and a damping term $\mathfrak{v} \in [0,1]$ in free space. Each particle is set to a static size, i.e., a diameter of 0.05 m, which is used for collision checking with other agents or the environment. In case a particle collides with an object or an agent, a simple point-mass collision is applied. Even though the actual observation highly depends on the actual scenario or task to be solved, all our implementations contain the internal agent state in the observation of the agents.

As claimed in Section 4, our approach tackles cooperative multi-robot RL tasks in sparsely rewarded environments, such that only environments with pure continuous action spaces have been tested. In addition to *cooperative navigation*, we evaluated our approaches

TABLE A3 Hyper-parameters for each algorithm. In case an entry is left blank, the algorithm does not have this hyper-parameter. The critic-parameters for our br-based approaches have been chosen identically for the interaction critic and native critic. Similarly, the parameters for the br policies are identical as the actor policy.

Parameter	MADDPG	MASAC	br-TD3	br-SAC
Size of (hidden) critic layers	(64, 64)	(64, 64)	(64, 64)	(64, 64)
Size of (hidden) policy layers	(64, 64)	(64, 64)	(64, 64)	(64, 64)
Learning rate critic ι_{Q}	0.01	0.01	0.01	0.01
Learning rate policy ι_{π}	0.01	0.01	0.01	0.01
Polyak value υ	0.01	0.01	0.01	0.01
α_0		0.2		0.2

on the *cooperative collection* task, in which $N_{\mathfrak{A}}$ agents are expected to reach $N_{\mathfrak{A}}$ goal locations. The reward signal is provided sparsely, which is expressed as follows:

$$\mathrm{r}^{(i)} \leftarrow \sum_{k=0}^{N_{\mathfrak{A}}} \begin{cases} 0 & \text{if } \texttt{visited}\left(g_k\right) \vee \exists j : \| \mathrm{x}^{(j)} - g_k \|_2 \leq \zeta_{\mathrm{g,MPE}} \\ -1 & \text{else.} \end{cases} \tag{30}$$

In addition, each agent is penalized with a direct cost value of −1 every time a collision with the environment or another agent is encountered. For both environments, the observations of other agents $\mathrm{y}_{\mathfrak{e}}^{(i)}$ contain a two dimensional distance vector of the agents' center position, while the observations of the environment $\mathrm{y}_{(-i)}^{(i)}$ contain two-dimensional distance vectors to the goal locations and the objects in the environment.

Appendix 2

The hyper-parameters for the learning procedure that is applied for all algorithms identically are listed in Table A1.

Similarly, the (physical) parameters for the MPE environment are listed in Table A2.

Eventually, the hyper-parameters for the individual algorithms are listed in Table A3. Our br-based policies used identical parameters for the dyadic and game-against-nature scheme. Therefore, only one column per algorithm is provided in the table below. We used Adam (Kingma and Ba, 2015) for the stochastic gradient descent (SGD)-optimization for all algorithms.

The experimental evidence was collected on two distributed computers with the following hardware components:

- **OS-Kernel**:
  - (Ubuntu) Linux-5.13.0–44
  - (Ubuntu) Linux-4.15.0–187
- **Processor**:
  - Intel(R) Core(TM) i3-7100 CPU @ 3.90 GHz
  - AMD Ryzen Threadripper 2990WX 32-Core
- **Python-version**: 3.9.7
- **GPU-acceleration**: disabled

Nomenclature

Acronyms

AC	actor–critic
br	best response
CI	confidence interval
COMA	counterfactual multi-agent
DDPG	deep deterministic policy gradient
DPG	deterministic policy gradient
DQN	deep Q-network
GPU	graphics processing unit
HRL	hierarchical reinforcement learning
KL	Kullback–Leibler
MAAC	multi-actor-attention-critic
MADDPG	multi-agent deep deterministic policy gradient
MARL	multi-agent reinforcement learning
MASAC	multi-agent soft actor–critic
MDP	Markov decision process
MG	Markov game
ML	machine learning
MPE	multi-agent particle environment
MPO	maximum *a posteriori*policy optimization
NE	Nash equilibrium
NN	neural network
PDF	probability density function
PG	policy gradient
PPO	proximal policy optimization
RL	reinforcement learning
SAC	soft actor–critic
SGD	stochastic gradient descent
TD3	twin-delayed deep deterministic policy gradient
TRPO	trust region policy optimization

Notation

$\mathfrak{p}$	placeholder variable in notation
$\top$	Boolean *true*
$:=$	equal by definition
${}^{\{k\}}\mathfrak{p}$	hierarchical layer indexing
$1^{\mathfrak{p}\times\mathfrak{p}}$	identity matrix of dimension $\mathfrak{p}\times\mathfrak{p}$
$0^{\mathfrak{p}\times\mathfrak{p}}$	zero matrix of dimension $\mathfrak{p}\times\mathfrak{p}$
$\mathfrak{P}$	matrix in $\mathbb{R}^{m\times n}$
$\mathbb{R}$	rational numbers
$\boldsymbol{\zeta}$	threshold value; indexing defines actual meaning
$\mathtt{t}$	current time or temporal indexing variable
$T_{\max}$	maximum run time (continuous) or the number of time steps (discrete) in $\mathbb{R}^1$
$\mathfrak{p}$	vector in $\mathbb{R}^n$
$1^{\mathfrak{p}}$	identity vector of dimension $\mathfrak{p}$
$0^{\mathfrak{p}}$	zero vector of dimension $\mathfrak{p}$

List of Symbols

$\mathtt{a}$	action of the agent i
$\boldsymbol{\pi}$	action assignment policy that maps a state to an action $\mathtt{a}$
$\mathtt{d}$	binary done flag, symbolizing the end of a task
$\mathtt{r}$	single step reward return
$\pi_{\text{br}}^{(j)}$	dyadic best-response policy of the agent $j$to the action of the agent i
$\underline{\pi}_{\text{br}}^{(-i)}$	dyadic best-response policy of agent *nature*to the action of agent i
$\boldsymbol{x}$	Cartesian x-coordinate in $\mathbb{R}^1$
$\boldsymbol{y}$	Cartesian y-coordinate in $\mathbb{R}^1$
$\mathfrak{v}$	damping constant
$\mathcal{D}$	data buffer containing experiences usable for RL.
$\boldsymbol{\alpha}$	entropy temperature parameter for SAC
$\mathfrak{e}$	environment layer index
$\boldsymbol{f}$	force magnitude or scalar force component in $\mathbb{R}^1$
$\boldsymbol{\Theta}$	function approximation parameterization for a critic
$\boldsymbol{\Pi}$	function approximation parameterization for a policy π
$\boldsymbol{\Xi}$	function approximation parameterization for $\pi_{\text{br}}^{(j)}$or $\underline{\pi}_{\text{br}}^{(-i)}$
$\mathfrak{i}$	interaction layer index
$\boldsymbol{D}_{\text{KL}}$	Kullback–Leibler-divergence of two PDFs
$\boldsymbol{\iota}$	learning rate for SGD
$\mathfrak{m}$	mass of a physical entity
$\boldsymbol{\mu}$	mean of a PDF.
$N_{\mathfrak{A}}$	number of agents
$\boldsymbol{\nu}$	polyak-averaging weight, e.g., used to update the target network
$\mathfrak{A}$	player or (artificial) agent
$\mathtt{s}$	state value in a general state space
$\mathtt{x}$	fully observable state
$\mathtt{y}$	hidden state
$\boldsymbol{\gamma}$	temporal discount factor
$\boldsymbol{v}$	translational velocity in SE (3)

List of Operators

$\mathtt{A}$	advantage function as the difference of $\mathtt{Q}$and $\mathtt{V}$in $\mathbb{R}^1$
$\mathcal{T}$	black box success function in $\mathbb{R}^1$

$\mathcal{R}$	black box reward function in $\mathbb{R}^1$
ρ	PDF over a random variable
$\mathbb{H}$	entropy of a random variable/PDF in $\mathbb{R}^1$
$\mathbb{E}$	expectation of a random variable/distribution
χ	general critic function in $\mathbb{R}^1$, e.g., $\mathtt{V}$, $\mathtt{A}$,or $\mathtt{Q}$
$\mathcal{F}_g$	goal mapping function; maps state/observation to a goal $g \in \mathcal{G} \in \mathbb{R}^n$
∇	gradient over a function or vector in $\mathbb{R}^n$or matrix in $\mathbb{R}^{n\times n}$
$\mathcal{L}$	loss function in $\mathbb{R}^1$
$\mathcal{J}$	general objective function for an optimization problem in $\mathbb{R}^1$
$\mathbb{P}$	probability of a random variable/distribution
$\mathtt{Q}$	Q-function in $\mathbb{R}^1$, also known as state-action value function
f_Π	deterministic function to apply the *reparameterization trick*, *cf.*SAC
$\mathcal{T}$	stochastic transition function
$\mathtt{V}$	value function in $\mathbb{R}^1$
$\mathbb{V}\mathbf{ar}$	variance of a random variable/distribution $\mathbb{V}\mathrm{ar} \in \mathbb{R}^1$