Neural Computing and Applications
https://doi.org/10.1007/s00521-022-07960-5

S.I.: ADAPTIVE AND LEARNING AGENTS 2021

Difference rewards policy gradients

Jacopo Castellini[1] · Sam Devlin[2] · Frans A. Oliehoek[3] · Rahul Savani[1]

Received: 17 November 2021 / Accepted: 17 October 2022

Abstract
Policy gradient methods have become one of the most popular classes of algorithms for multi-agent reinforcement learning. A key challenge, however, that is not addressed by many of these methods is multi-agent credit assignment: assessing an agent's contribution to the overall performance, which is crucial for learning good policies. We propose a novel algorithm called Dr.Reinforce that explicitly tackles this by combining difference rewards with policy gradients to allow for learning decentralized policies when the reward function is known. By differencing the reward function directly, Dr.Reinforce avoids difficulties associated with learning the Q-function as done by counterfactual multi-agent policy gradients (COMA), a state-of-the-art difference rewards method. For applications where the reward function is unknown, we show the effectiveness of a version of Dr.Reinforce that learns an additional reward network that is used to estimate the difference rewards.

Keywords Multi-agent reinforcement learning · Policy gradients · Difference rewards · Multi-agent credit assignment · Reward learning

1 Introduction

Many real-world problems, like air traffic management [49], packet routing in sensor networks [55] and traffic light control [40], can be naturally modelled as *cooperative multi-agent systems* [6]. Here multiple agents must learn to work together to achieve a common goal. Such problems have commonly been approached with *multi-agent reinforcement learning* (MARL) [5, 22, 28], including recently with *deep reinforcement learning*. Often in these settings agents have to behave in a *decentralized fashion* [30], relying only on local perceptions, due to the prohibitive complexity of a centralized solution or because communication is too expensive [4, 37].[1]

The paradigm of *centralized training with decentralized execution* (CTDE) [28, 38] deals with this: agents use global information during training, but then only rely on local sensing during execution. In such settings, policy gradient methods are amongst the few methods with convergence guarantees [39], and multi-agent policy gradient (MAPG) methods have become one of the most popular approaches for the CTDE paradigm [18, 29].

However, one key problem that agents face with CTDE that is not directly tackled by many MAPG methods is *multi-agent credit assignment* [9, 35, 53, 56]. With a shared reward signal, an agent cannot readily tell how its own actions affect the overall performance. This can lead to sub-optimal policies even with just a few agents. *Difference rewards* [12, 13, 41, 54] were proposed to tackle this problem: agents learn from a shaped reward that allows

✉ Jacopo Castellini
J.Castellini@liverpool.ac.uk

Sam Devlin
Sam.Devlin@microsoft.com

Frans A. Oliehoek
F.A.Oliehoek@tudelft.nl

Rahul Savani
rahul.savani@liverpool.ac.uk

[1] Department of Computer Science, University of Liverpool, Liverpool, UK

[2] Microsoft Research Cambridge, Cambridge, UK

[3] Interactive Intelligence Group, Delft University of Technology, Delft, The Netherlands

[1] This preliminary version of this work appeared in the Autonomous and Learning Agents 2021 workshop, where it won the Best Paper Award.

Published online: 11 November 2022

them to infer how their actions contributed to the shared reward.

Only one MAPG method has incorporated this idea so far: counterfactual multi-agent policy gradients (COMA) [18] is a state-of-the-art algorithm that does the differencing with a learned action-value function $Q_\omega(s,a)$. However, there are potential disadvantages to this approach: learning the Q-function is a difficult problem due to compounding factors of bootstrapping, the moving target problem (as target values used in the update rule change over time) and Q's dependence on the joint actions. This makes the approach difficult to apply with more than a few agents. Moreover, COMA is not exploiting knowledge about the reward function, even though this might be available in many MARL problems.

To overcome these potential difficulties, we take inspiration from [12] and incorporate the *differencing of the reward function* into MAPG. Extending the work in [8] with additional results and analysis, we propose *difference rewards REINFORCE* (Dr.Reinforce), a new MARL algorithm that combines decentralized policies learned with policy gradients with difference rewards that are used to provide gradients with information on each agent's individual contribution to overall performance. Additionally, we provide a version, called Dr.ReinforceR, for settings where the reward function is not known upfront. In contrast to [12], Dr.ReinforceR exploits the CTDE paradigm and learns a centralized reward network to estimate difference rewards. Although the dimensionality of the reward function is the same as the Q-function, and similarly depends on joint actions, learning the reward function is a simple regression problem. It does not suffer from the moving target problem, which allows for faster training and improved performance. Our empirical results show that our approaches can significantly outperform other MAPG methods, particularly with more agents.

2 Background

Here we introduce some notions about multi-agent systems and policy gradients used to understand the remainder of this work.

2.1 Multi-agent reinforcement learning

Our setting can be formalized as a multi-agent Markov decision process (MMDP) [4] $\mathcal{M} = \langle D, S, \{A^i\}_{i=1}^{|D|}, T, R, \gamma \rangle$, where $D = \{1, \ldots, N\}$ is the set of agents; $s \in S$ is the state; $a^i \in A^i$ is the action taken by agent i and $a = \langle a^1, \ldots, a^N \rangle \in \times_{i=1}^{|D|} A^i = A$ denotes the joint action; $T(s'|a,s) : S \times A \times S \to [0,1]$ is the transition function that determines the probability of ending up in state s' from s under joint action a; $R(s,a) : S \times A \to \mathbb{R}$ is the shared reward function and γ is the discount factor.

Agent i selects actions using a stochastic policy $\pi_{\theta^i}(a^i|s) : S \times A^i \to [0,1]$ with parameters θ^i, with $\theta = \langle \theta^1, \ldots, \theta^N \rangle$ and $\pi_\theta = \langle \pi_{\theta^1}, \ldots, \pi_{\theta^N} \rangle$ denoting the joint parameters and policy, respectively. With r_t denoting the reward at time t, and expectations taken over sequences of executions, the policy π_θ induces the value functions $V^{\pi_\theta}(s_t) = \mathbb{E}_{\pi_\theta}\left[\sum_{l=0}^{\infty} \gamma^l r_{t+l} | s_t\right]$ and action-value function $Q^{\pi_\theta}(s_t, a_t) = \mathbb{E}_{\pi_\theta}\left[\sum_{l=0}^{\infty} \gamma^l r_{t+l} | s_t, a_t\right]$. At each time step t, the agents try to maximize the value function $V^{\pi_\theta}(s_t)$.

2.2 Reinforce and actor-critic

In single-agent reinforcement learning [26, 46], *policy gradient* methods (PG) [47] aims to maximize the expected value function $V^{\pi_\theta}(s_t)$ by directly optimizing the policy parameters θ. These methods perform gradient ascent in the direction that maximizes the expected parametrized value function $V(\theta) = \mathbb{E}_{s_0}[V^{\pi_\theta}(s_0)]$. The simplest policy gradient method is REINFORCE [52], which is a Monte Carlo algorithm, executing the current policy π_θ for an entire episode of T steps and then optimizing it with the following update:

$$\theta \leftarrow \theta + \alpha \underbrace{\sum_{t=0}^{T-1} \gamma^t G_t \nabla_\theta \log \pi_\theta(a_t|s_t)}_{\hat{g}},$$

where the return $G_t = \sum_{l=0}^{T-t-1} \gamma^l r_{t+l}$ is an unbiased estimate of $V^{\pi_\theta}(s_t)$ computed over the episode. This update rule corresponds to performing stochastic gradient ascent [3] on $V(\theta)$ because the expectation of the update target is the gradient of the value function, $\mathbb{E}_{\pi_\theta}[\hat{g}] = \nabla_\theta V(\theta)$. Under appropriate choices of step sizes α the method will converge [47].

REINFORCE suffers from the high variance of the sampled returns because of the stochasticity of environment and agent policy itself, and thus converges slowly. To reduce such variance, a suitable baseline $b(s)$ can be subtracted from the return G_t [46].

Another possibility to overcome such problem are *actor-critic* methods [27, 32] that try to do so by learning an additional component called the critic. The critic is parametrized by ω and represents either the value or action-value function. It is learned along with the policy π_θ to minimize the on-policy *temporal difference (TD) error* at

each time step t, which for a critic that represents the Q-function is:

$$\delta_t = r_t + \gamma Q_\omega(s_{t+1}, a_{t+1}) - Q_\omega(s_t, a_t). \tag{1}$$

The policy is then optimized using the estimates given by the critic:

$$\theta \leftarrow \theta + \alpha \sum_{t=0}^{T-1} Q_\omega(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t). \tag{2}$$

As for REINFORCE, a baseline $b(s)$ can be subtracted from the critic estimate in Eq. (2) to further reduce variance. If $b(s) = V(s)$, then $A(s,a) = Q_\omega(s,a) - V(s)$ is called the *advantage function* and is a used in many actor-critic methods [32].

In cooperative MARL, each agent i can individually learn a decentralized policy by using the *distributed policy gradient* [39] update target for π_{θ^i}:

$$\theta^i \leftarrow \theta^i + \alpha \underbrace{\sum_{t=0}^{T-1} \gamma^t G_t \nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i|s_t)}_{\hat{g}^i}, \tag{3}$$

where a^i is this agent's action and G_t is the return computed with the shared reward and is identical for all agents.

2.3 Difference rewards

In settings where the reward signal is shared, agents cannot easily determine their individual contribution to the reward, a problem known as multi-agent credit assignment. It can be tackled with difference rewards [41, 54]. Instead of using the shared reward $R(s, a)$, agents compute a shaped reward:

$$\Delta R^i(a^i|s, a^{-i}) = R(s,a) - R(s, \langle a^{-i}, c^i \rangle), \tag{4}$$

where a^{-i} is the joint action all agents except i and c^i is a *default action* for agent i used to replace a^i. This way, an agent can assess its own contribution, and therefore, each action that improves ΔR^i also improves the global reward $R(s, a)$ [1]. This, however, requires access to the complete reward function or the use of a resettable simulator to estimate $R(s, \langle a^{-i}, c^i \rangle)$. Moreover, the choice of the default action can be problematic. The *aristocrat utility* [54] avoids this choice by marginalizing out an agent by computing its expected contribution to the reward given its current policy π_{θ^i}:

$$\Delta R^i(a^i|s, a^{-i}) = R(s,a) - \mathbb{E}_{b^i \sim \pi_{\theta^i}} \left[R(s, \langle a^{-i}, b^i \rangle) \right]. \tag{5}$$

The work of [12] learns a local approximation of the reward function $R_{\psi^i}(s, a^i)$ for each agent i and uses it to compute the difference rewards of Eq. (4), by fixing a default action c^i, as:

$$\Delta R^i_{\psi^i}(a^i|s) = R(s,a) - R_{\psi^i}(s, c^i).$$

Counterfactual multi-agent policy gradients (COMA) [18] is a state-of-the-art deep MAPG algorithm that adapts difference rewards and aristocrat utility to use the Q-function, approximated by a centralized critic $Q_\omega(s,a)$ learned under the CTDE paradigm (as the algorithm is designed for general partially observable multi-agent domains [37], where agents cannot access the environment state s_t), by providing the policy gradients of the agents with a counterfactual advantage function:

$$A^i(s,a) = Q_\omega(s,a) - \sum_{c^i \in A^i} \pi_{\theta^i}(c^i|h_t^i) Q_\omega(s, \langle a^{-i}, c^i \rangle).$$

3 Difference rewards policy gradients

COMA learns a centralized action-value function critic $Q_\omega(s,a)$ to do the differencing and drive agents' policy gradients. However, learning such a critic using the TD error in Eq. (1) presents a series of challenges that may dramatically hinder final performance if they are not carefully tackled. The Q-value updates rely on bootstrapping that can lead to inaccurate updates. Moreover, the target values for these updates are constantly changing because the other estimates used to compute them are also updated, leading to a moving target problem. This is exacerbated when function approximation is used, as these estimates can be indirectly modified by the updates of other Q-values. Target networks are used to try and tackle this problem [31], but these require careful tuning of additional parameters and may slow down convergence with more agents.

Our proposed algorithm, named Dr.Reinforce, combines the REINFORCE [52] policy gradient method with a difference rewards mechanism to deal with credit assignment in cooperative multi-agent systems, thus avoiding the need of learning a critic.

3.1 Dr.Reinforce

If the reward function $R(s, a)$ is known, we can directly use difference rewards with policy gradients. We define the *difference return* ΔG_t^i for agent i as the discounted sum of

the difference rewards $\Delta R^i(a_t^i|s_t, a_t^{-i})$ from time step t onward as:

$$\Delta G_t^i(a_{t:T}^i|s_{t:T}, a_{t:T}^{-i}) \triangleq \sum_{l=0}^{T-t-1} \gamma^l \Delta R^i(a_{t+l}^i|s_{t+l}, a_{t+l}^{-i}), \tag{6}$$

where T is the length of the sampled trajectory and $\Delta R^i(a_t^i|s_t, a_t^{-i})$ is the difference rewards for agent i, computed using the aristocrat utility [54] as in Eq. (5). Please note that the subscript $t:T$ in our notation is a shorthand used to identify the sequence of values of given quantity from time step t up to (but not including) time step T.

To learn the decentralized policies π_θ, we follow a modified version of the distributed policy gradients in Eq. (3) that uses our difference return, optimizing each policy by using the update target:

$$\theta^i \leftarrow \theta^i + \alpha \underbrace{\sum_{t=0}^{T-1} \gamma^t \Delta G_t^i(a_{t:T}^i|s_{t:T}, a_{t:T}^{-i}) \nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i|s_t)}_{g^{\mathrm{DR},i}}, \tag{7}$$

where ΔG_t^i is the difference return defined in Eq. (6). This way, each policy is guided by an update that takes into account its individual contribution to the shared reward, and an agent thus takes into account the real value of its own actions. We expect this signal to drive the policies towards regions in which individual contributions are higher, and thus also the shared reward, since a sequence of actions improving ΔG_t^i also improves the global return [1].

3.2 Online reward estimation

In many settings, complete access to the reward function to compute the difference rewards is not available. Thus, we propose Dr.ReinforceR, which is similar to Dr.Reinforce but additionally learns a *centralized reward network* R_ψ, with parameters ψ, that is used to estimate the value $R(s, \langle a^i, a^{-i} \rangle)$ for every local action $a^i \in A^i$ for agent i. Following the CTDE paradigm, this centralized network is only used during training to provide policies with learning signals and is not needed during execution, when only the decentralized policies are used. The reward network receives as input the environment state s_t and the joint action of the agents a_t at time t, and is trained to reproduce the corresponding reward value $r_t \sim R(s_t, a_t)$ by minimizing a standard MSE regression loss:

$$\mathcal{L}_t(\psi) = \frac{1}{2}\big(r_t - R_\psi(s_t, a_t)\big)^2. \tag{8}$$

Although the dimensionality of the function $R(s, a)$ that we are learning with the reward network is the same as that of $Q(s, a)$ learned by the COMA critic, growing exponentially with the number of agents as both depend of the joint action $a \in A = \times_{i=1}^{|D|} A^i$, learning R_ψ is a regression problem that does not involve bootstrapping or moving targets, thus avoiding many of the problems faced with an action-value function critic. Moreover, alternative representations of the reward function can be used to further improve learning speed and accuracy, e.g. by using factorizations [7].

We can now use the learned R_ψ to compute the difference rewards ΔR_ψ^i using the aristocrat utility [54] as:

$$\Delta R_\psi^i(a_t^i|s_t, a_t^{-i}) \triangleq r_t - \sum_{c^i \in A^i} \pi_{\theta^i}(c^i|s_t) R_\psi(s_t, \langle c^i, a_t^{-i} \rangle). \tag{9}$$

The second term of the r.h.s. of Eq. (9) can be estimated with a number of network evaluations that is linear in the size of the local action set A^i, as the actions of the other agents a_t^{-i} remains fixed, avoiding an exponential cost.

We now redefine the difference return ΔG_t^i from Eq. (6) as the discounted sum of the estimated difference rewards $\Delta R_\psi^i(a_{t+l}^i|s_{t+l}, a_{t+l}^{-i})$:

$$\Delta G_t^i(a_{t:T}^i|s_{t:T}, a_{t:T}^{-i}) \triangleq \sum_{l=0}^{T-t-1} \gamma^l \Delta R_\psi^i(a_{t+l}^i|s_{t+l}, a_{t+l}^{-i}). \tag{10}$$

3.3 Theoretical results

REINFORCE [52] suffers from high variance of gradients estimates because of sample estimation of the return. This can be accentuated in the multi-agent setting. Using an unbiased baseline is crucial to reducing this variance and improving learning [20, 46]. Here we address these concerns by showing that using difference rewards in policy gradient methods corresponds to subtracting an unbiased baseline from the policy gradient of each individual agent. Since this unbiased baseline does not alter the expected value of the update targets, applying difference rewards policy gradients to a common-reward MARL problem turns out to be same in expectation as using distributed policy gradients update targets. Such gradients' updates have been shown to be equivalent to those of a joint gradient [39], which under some technical conditions is known to converge to a local optimum [27, 47].

Lemma 1 *In a MMDP, using difference return* $\Delta G_t^i(a_{t:T}^i|s_{t:T}, a_{t:T}^{-i})$ *as the learning signal for policy gradients in Eq. (7) is equivalent to subtracting an unbiased baseline* $B^i(s_{t:T}, a_{t:T}^{-i})$ *from the distributed policy gradients in Eq. (3).*

Proof We start by rewriting $\Delta G_t^i(a_{t:T}^i|s_{t:T}, a_{t:T}^{-i})$ from Eq. (6) as:

$$\begin{aligned}\Delta G_t^i(a_{t:T}^i|s_{t:T}, a_{t:T}^{-i}) &= \sum_{l=0}^{T-t-1} \gamma^l r_{t+l} \\ &- \sum_{l=0}^{T-t-1} \gamma^l \sum_{c^i \in A^i} \pi_{\theta^i}(c^i|h_{t+l}^i) R(s_{t+l}, \langle c^i, a_{t+l}^{-i}\rangle).\end{aligned} \tag{11}$$

Note that the first term on the r.h.s. of Eq. (11) is the return G_t used in Eq. (3). We then define the second term on the r.h.s. of Eq. (11) as the baseline $B^i(s_{t:T}, a_{t:T}^{-i})$:

$$B^i(s_{t:T}, a_{t:T}^{-i}) = \sum_{l=0}^{T-t-1} \gamma^l \sum_{c^i \in A^i} \pi_{\theta^i}(c^i|s_{t+l}) \cdot R(s_{t+l}, \langle c^i, a_{t+l}^{-i}\rangle). \tag{12}$$

We can thus rewrite the total expected update target for agent i as:

$$\begin{aligned}\mathbb{E}_{\pi_\theta}\left[g^{\mathrm{DR},i}\right] &= \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} \left(\nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i|s_t)\right) \Delta G_t^i(a_{t:T}^i|s_{t:T}, a_{t:T}^{-i})\right] \\ &= \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} \left(\nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i|s_t)\right) \left(G_t - B^i(s_{t:T}, a_{t:T}^{-i})\right)\right] \\ &\quad \text{(by definition of } \Delta G_t^i\text{)} \\ &= \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} \left(\nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i|s_t)\right) G_t \right. \\ &\quad \left. - \left(\nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i|s_t)\right) B^i(s_{t:T}, a_{t:T}^{-i})\right] \\ &\quad \text{(distributing the product)} \\ &= \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} \left(\nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i|s_t)\right) G_t\right] \\ &\quad - \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} \left(\nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i|s_t)\right) B^i(s_{t:T}, a_{t:T}^{-i})\right] \\ &\quad \text{(by linearity of the expectation)} \\ &= \mathbb{E}_{\pi_\theta}\left[g^{\mathrm{PG},i}\right] + \mathbb{E}_{\pi_\theta}\left[g^{B,i}\right].\end{aligned} \tag{13}$$

We have to show that the baseline is unbiased, and so, the expected value of its update $\mathbb{E}_{\pi_\theta}[\hat{g}^{B,i}]$ with respect to the policy π_θ is 0. Let

$$P_t^{\pi_\theta}(s_t) = \sum_{s_{t-1} \in S} P_{t-1}^{\pi_\theta}(s_{t-1}) \sum_{a_{t-1} \in A} \pi_\theta(a_{t-1}|s_{t-1})\, T(s_t|a_{t-1}, s_{t-1})$$

be the probability of the state at time step t to be s_t under the joint policy π_θ (with $P_0^{\pi_\theta}(s_0) = \rho(s_0)$ and ρ is the initial state distribution), we have:

$$\begin{aligned}\mathbb{E}_{\pi_\theta}\left[\hat{g}^{B,i}\right] &\triangleq - \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} \left(\nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i|s_t)\right) B^i(s_{t:T}, a_{t:T}^{-i})\right] \\ &= -\sum_{t=0}^{T-1} \sum_{s_t \in S} P_t^{\pi_\theta}(s_t) \sum_{a_t^{-i} \in A^{-i}} \pi_{\theta^{-i}}(a_t^{-i}|s_t) \sum_{a_t^i \in A^i} \pi_{\theta^i}(a_t^i|s_t) \\ &\quad \left(\nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i|s_t)\right) \sum_{s_{t+1:T}, a_{t+1:T}} \\ &\quad \prod_{l=1}^{T-t-1} T(s_{t+l}|a_{t+l-1}, s_{t+l-1}) \cdot \\ &\quad \pi_\theta(a_{t+l}|s_{t+l})\, B^i(s_{t:T}, a_{t:T}^{-i}) \\ &\quad \text{(by expanding the expectation)} \\ &= -\sum_{t=0}^{T-1} \sum_{s_t \in S} P_t^{\pi_\theta}(s_t) \sum_{a_t^{-i} \in A^{-i}} \pi_{\theta^{-i}}(a_t^{-i}|s_t) \sum_{a_t^i \in A^i} \\ &\quad \left(\nabla_{\theta^i} \pi_{\theta^i}(a_t^i|s_t)\right) \\ &\quad \sum_{s_{t+1:T}, a_{t+1:T}} \prod_{l=1}^{T-t-1} T(s_{t+l}|a_{t+l-1}, s_{t+l-1}) \cdot \\ &\quad \pi_\theta(a_{t+l}|s_{t+l})\, B^i(s_{t:T}, a_{t:T}^{-i}) \\ &\quad \text{(by applying the inverse log trick)} \\ &= -\sum_{t=0}^{T-1} \sum_{s_t \in S} P_t^{\pi_\theta}(s_t) \sum_{a_t^{-i} \in A^{-i}} \pi_{\theta^{-i}}(a_t^{-i}|s_t) \\ &\quad \left(\nabla_{\theta^i} \sum_{a_t^i \in A^i} \pi_{\theta^i}(a_t^i|s_t)\right) \\ &\quad \sum_{s_{t+1:T}, a_{t+1:T}} \prod_{l=1}^{T-t-1} T(s_{t+l}|a_{t+l-1}, s_{t+l-1}) \cdot \\ &\quad \pi_\theta(a_{t+l}|s_{t+l})\, B^i(s_{t:T}, a_{t:T}^{-i}) \\ &\quad \text{(by moving the gradient outside the policy sum)} \\ &= -\sum_{t=0}^{T-1} \sum_{s_t \in S} P_t^{\pi_\theta}(s_t) \sum_{a_t^{-i} \in A^{-i}} \pi_{\theta^{-i}}(a_t^{-i}|s_t)\, \nabla_{\theta^i} 1 \\ &\quad \sum_{s_{t+1:T}, a_{t+1:T}} \prod_{l=1}^{T-t-1} T(s_{t+l}|a_{t+l-1}, s_{t+l-1}) \cdot \\ &\quad \pi_\theta(a_{t+l}|s_{t+l})\, B^i(s_{t:T}, a_{t:T}^{-i}) \\ &\quad \text{(policy probabilities sum up to 1)} \\ &= 0.\end{aligned} \tag{14}$$

Therefore, using the baseline in Eq. (12) reduces the variance of the updates [20] but does not change their expected value, as it is unbiased and its expected update target $\mathbb{E}_{\pi_\theta}[\hat{g}^{B,i}] = 0$. □

Corollary *Using the estimated reward network* R_ψ *to compute the baseline in Eq. (12) still results in an unbiased baseline.*

Proof We rewrite $\Delta G_t^i(a_{t:T}^i|s_{t:T}, a_{t:T}^{-i})$ from Eq. (10) as:

$$\begin{aligned}\Delta G_t^i(a_{t:T}^i|s_{t:T}, a_{t:T}^{-i}) &= \sum_{l=0}^{T-t-1} \gamma^l r_{t+l} \\ &- \sum_{l=0}^{T-t-1} \gamma^l \sum_{c^i \in A^i} \pi_{\theta^i}(c^i|s_{t+l}) R_\psi(s_{t+l}, \langle c^i, a_{t+l}^{-i}\rangle),\end{aligned} \tag{15}$$

for which we define the second term on the r.h.s. of Eq. (15) as the baseline $B_\psi^i(s_{t:T}, a_{t:T}^{-i})$:

$$B^i_\psi(s_{t:T}, a^{-i}_{t:T}) = \sum_{l=0}^{T-t-1} \gamma^l \sum_{c^i \in A^i} \pi_{\theta^i}(c^i|s_{t+l}) \cdot R_\psi(s_{t+l}, \langle c^i, a^{-i}_{t+l}\rangle).$$

We observe that the derivation of Eq. (14) still holds, as it is not altered by the use of the reward network R_ψ rather than the true reward function $R(s, a)$. Therefore, the baseline $B^i_\psi(s_{t:T}, a^{-i}_{t:T})$ is again unbiased and does not alter the expected value of the updates. □

Theorem 1 *In a MMDP with shared rewards, given the conditions on function approximation detailed in* [47], *using Dr.Reinforce update target as in Eq.* (7), *the series of parameters* $\{\theta_t = \langle \theta^1_t, \ldots, \theta^N_t \rangle\}^k_{t=0}$ *converges in the limit such that the corresponding joint policy* π_{θ_t} *is a local optimum*:

$$\lim_{k\to\infty} \inf_{\{\theta_t\}^k_{t=0}} ||\hat{g}^{\mathrm{DR}}|| = 0 \qquad w.p.\ 1.$$

Proof To prove convergence, we have to show that:

$$\mathbb{E}_{\pi_{\theta_t}}\left[\hat{g}^{\mathrm{DR}}\right] = \mathbb{E}_{\pi_{\theta_t}}\left[\sum_{i=0}^{N} \hat{g}^{\mathrm{DR},i}\right] = \nabla_{\theta_t} V(\theta_t).$$

We can rewrite the total expected update target as:

$$\mathbb{E}_{\pi_{\theta_t}}\left[\hat{g}^{\mathrm{DR},i}\right] = \mathbb{E}_{\pi_{\theta_t}}\left[\hat{g}^{\mathrm{PG},i}\right] + \mathbb{E}_{\pi_{\theta_t}}\left[\hat{g}^{B,i}\right]$$

as in Eq. (13), and by Lemma 1, we have that $\mathbb{E}_{\pi_{\theta_t}}[\hat{g}^{B,i}] = 0$. Therefore, the overall expected update $\mathbb{E}_{\pi_{\theta_t}}[\hat{g}^{\mathrm{DR},i}]$ for agent i reduces to $\mathbb{E}_{\pi_{\theta_t}}[\hat{g}^{\mathrm{PG},i}]$ that is equal to the distributed policy gradient update target in Eq. (3). These updates for all the agents has been proved to be equal to these of a centralized policy gradients agent $\mathbb{E}_{\pi_{\theta_t}}[\hat{g}^{\mathrm{PG}}]$ by Theorem 1 in [39] and therefore converge to a local optimum of $\nabla_{\theta_t} V(\theta_t)$ by Theorem 3 in [47]. □

4 Experiments

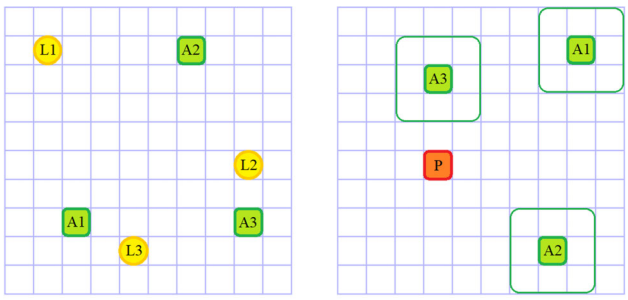
We are interested in investigating the following questions:

1. How does Dr.Reinforce compare to existing approaches?
2. How does the use of a learned reward network R_ψ instead of a known reward function affect performance?
3. Is learning the Q-function (as in COMA) more difficult than learning the reward function $R(s, a)$ (as in Dr.ReinforceR)?

To investigate these questions, we tested our methods on two gridworld environments with shared reward: the multi-rover domain, an established multi-agent cooperative domain [13], in which agents have to spread across a series of landmarks, and a variant of the classical predator–prey problem with a randomly moving prey [48].

4.1 Comparison to baselines

We compare to a range of other policy gradient methods: independent learners using REINFORCE to assess the benefits of using a difference rewards mechanism, labelled PG. We also compare against a standard actor-critic algorithm [27] with decentralized actors and a centralized action-value function critic to show that our improvements are not only due to the centralized information provided to the agents during training, denoted as Central$Q$ here. Our main comparison is with COMA [18], a state-of-the-art difference rewards method using the Q-function for computing the differences. Finally, we compare against the algorithm proposed in [12], to show the benefit of learning a centralized reward network to estimate the difference rewards in Dr.ReinforceR. Indeed, this algorithm learns an individual approximation of the reward function $R_{\psi^i}(s, a^i)$ for each agent i and uses this in estimating the difference rewards as in Eq. 4 to learn the agents' policies. We adapted this method to use policy gradients instead of evolutionary algorithms to optimize the policies to not conflate the comparisons with the choice of a policy optimizer where possible, and only focus on the effect of using difference rewards during learning. Additionally, the multi-agent A* (MAA*) exact planning algorithm [36, 37] has been applied to the smaller instances of the two problems with only $N = 3$ agents, as an upper bound for assessing the overall performance of the investigated learning algorithms. Because of the exponentially many joint actions to expand at each state, it has not been possible to apply such an algorithm to larger instances.



(a) Multi-Rover (b) Predator-Prey

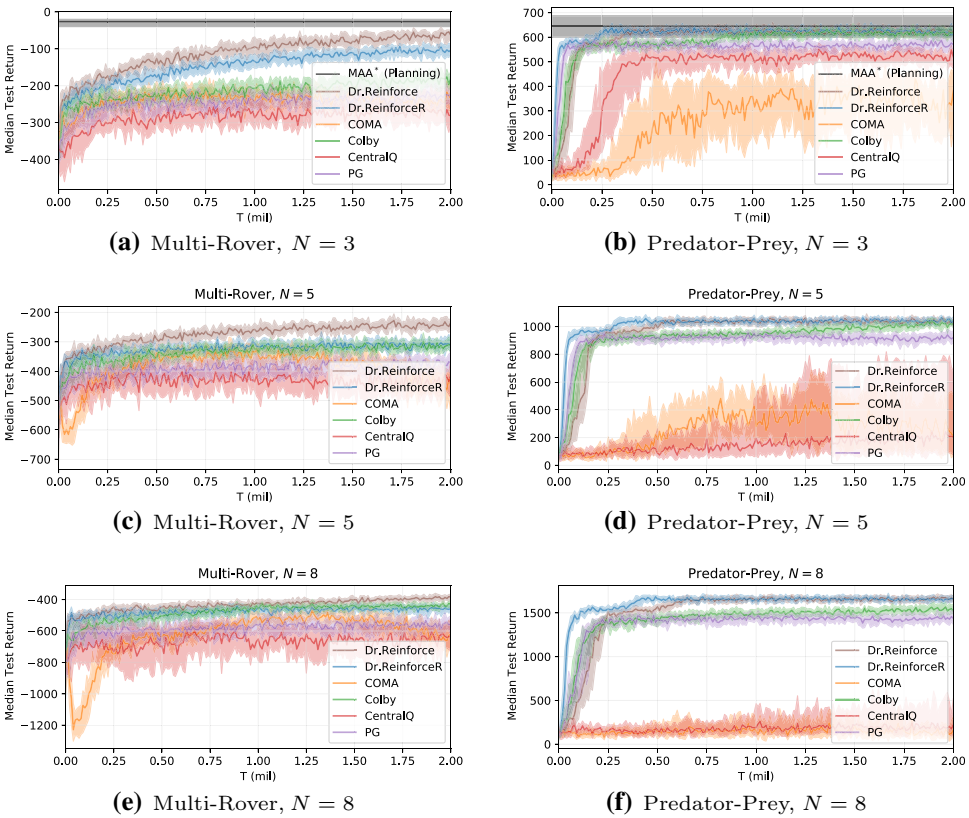
Fig. 1 Schematic representation of the two gridworld domains. Agents are green, landmarks are yellow, and the prey is red

4.1.1 Multi-rover domain

In this domain, a team of N agents is placed into a 10×10 gridworld with a set of N landmarks. The aim of the team is to spread over all the landmarks and cover them (which agent covers which landmark is not important): the reward received by the team depends on the distance of each landmark to its closest agent, and a penalty if two agents collide (reach the same cell simultaneously) during their movements is also applied. Each agent observes its relative position with respect to all the other agents and landmarks and can move in the four cardinal directions or stand still. Fig. 2 (left) reports the median performance and $25-75\%$ percentiles (shaded area in the plots) across 30 independent runs obtained by the compared methods on a team of increasing size, to investigate scaling to larger multi-agent systems.

It can be observed that both Dr.Reinforce and Dr.ReinforceR are always outperforming all of the other compared baselines on this domain. Also, Dr.ReinforceR is generally matching the upper bound given by Dr.Reinforce (that represents a limit case when the centralized reward network R_ψ has perfectly converged to the true reward function). However, the wide gap between these two algorithms and the other baselines when $N = 3$ reduces when more agents are introduced in the system, possibly pointing out that also these methods start to struggle in achieving optimal and coordinated behaviours on larger instances of this domain. When more agents are present, the gridworld becomes quite crowded: an explanation for this loss in performance is that the difference rewards signal pushes each agent towards the landmark that is furthest from all of the agents, thus contributing the most to the negative reward value, in an attempt to mitigate this problem, but letting another landmark increase its negative contribution in turn. Coordination is key to efficiently solve this domain, and achieving such coordination may be difficult in larger settings.

Moreover, even if the reward network learns a good representation, the synergy between this and the agents' policies has to be carefully considered: the reward network has to converge properly before the policies got stuck into a local optimum, or it could be the case that these will not be able to escape it even if the gradients signals are then accurate enough. However, the simpler learning problem used to provide signals to the agents' policies, as opposed to the very complex learning of the action-value function critic used by COMA, proves effective in speeding up learning and achieve higher returns, even in difficult settings with many agents where all the other policy gradient methods seem to fail as well. Computing the difference rewards requires very accurate reward estimates, so if the reward network do not exhibit appropriate generalization capabilities it may end up overfitting on the reward values



(a) Multi-Rover, $N = 3$ **(b)** Predator-Prey, $N = 3$

(c) Multi-Rover, $N = 5$ **(d)** Predator-Prey, $N = 5$

(e) Multi-Rover, $N = 8$ **(f)** Predator-Prey, $N = 8$

Fig. 2 Training curves on the multi-rover domain (left) and the predator–prey problem (right), showing the median reward and $25-75\%$ percentiles across seeds

encountered during training but not being able to give correct predictions beyond those. It is true, however, that also difference rewards methods using the action-value function like COMA have the same requirements.

4.1.2 Predator–prey

In this version of the classical predator–prey problem, a team of N predators has to pursue a single prey for as long as possible in a 10×10 gridworld. Each predator has got a range of sight of one cell in each direction from its current position: if the prey is into this range, the whole team receives a positive reward bonus; otherwise, they do not receive any reward. Each agent observes its relative position with respect to the other agents and the prey itself and can move in the four cardinal directions or stand still. The prey selects actions uniformly at random from the same set of actions available to the agents. Figs. 2 (right) shows median results and $25-75\%$ percentiles across 30 independent runs with teams comprising an increasing number of predators.

Also in this environment, Dr.ReinforceR is outperforming all the other compared methods, achieving performance that is equal or close to these of the Dr.Reinforce upper bound (of which the former is an approximated version). On the one hand, some of the other baselines are also performing well: PG and Colby are almost performing on-par with the two above algorithms, even on larger instances of the problem. This is probably due to the less strict coordination requirements of the predator–prey problem compared to the previous multi-rover domain: each agent is independently contributing towards the common goal and thus simply needs to optimize its own behaviour by learning how to reach and stay on the prey in order to improve global performances.

On the other hand, COMA is performing extremely poorly, being outperformed even by the simple Central$Q$ (that has slowly learned something in the simpler case with $N = 3$). This points out how accurately learning an optimal Q-function may be problematic in many settings, even more so on a sparse setting such as this, in which the agents are only perceiving rewards if some of them are effectively on the prey. If the Q-function converges to a sub-optimal solution and keeps pushing the agents towards a local optimum, the policies may struggle to escape from it afterwards and in turn push the action-value function towards a worst approximation. Moreover, to compute the counterfactual baseline in COMA, estimates of Q-values need to be accurate even on state-action pairs that the policies do not visit often, further exacerbating this problem. From this side, learning the reward function to compute the difference rewards is an easier learning problem, cast as a regression task and not involving bootstrapped estimates or a moving target, and thus can improve policy gradient performance providing them with better learning signals in achieving high return behaviours with no further drawback.

4.2 Analysis

The results of the proposed experiments show the benefits of learning the reward function over the more complex Q-function, leading to faster policy training and improved final performances, but also that this is not always an easy task and it can present issues on its own that can hinder the learning of an optimal joint policy. Indeed, although not suffering from the moving target problem and no bootstrapping is involved, learning the reward function online together with the policies of the agents can lead to biases of the learned function due to the agents behaviours. These biases could push the training samples towards a specific region of the true reward function, hindering the generalization capacity of the learned reward network and in turn leading to worst learning signal for the policies themselves, that can get stuck into a sub-optimal region. Similarly, this problem can appear when a centralized action-value critic is used to drive the policy gradients.

To investigate the claimed benefits of learning the reward function rather the Q-function, let now analyse the accuracy of the learned representations on the two proposed gridworld domains by sampling a set of different trajectories from the execution of the corresponding policies and comparing the predicted values from the reward network $R_\psi(s,a)$ of Dr.ReinforceR and the $Q_\omega(s,a)$ critic from COMA to the real ground-truth values of the reward function and the Q-function, respectively. This has been called the *on-policy dataset*, representing how correctly can the reward network and the critic represent the values of state-action pairs encountered during their training phase. Moreover, both Dr.ReinforceR and COMA rely on a difference rewards mechanism and thus need to estimate values for state-action pairs that are only encountered infrequently (or not at all) in order to compute correct values to drive the policy gradients. To investigate the generalization performances of the learned representations, let also analyse the prediction error on a *off-policy dataset*, by sampling uniformly across the entire action-state space $S \times A$ and again comparing the predicted values from the learned reward function $R_\psi(s,a)$ of Dr.ReinforceR and the $Q_\omega(s,a)$ critic from COMA to their corresponding ground-truth values. Please note that, not knowing the true Q-function for the proposed problems to compare against, these have been approximated that via 100 rollouts sampled starting from the current state-action sample and following the corresponding learned policies afterwards.

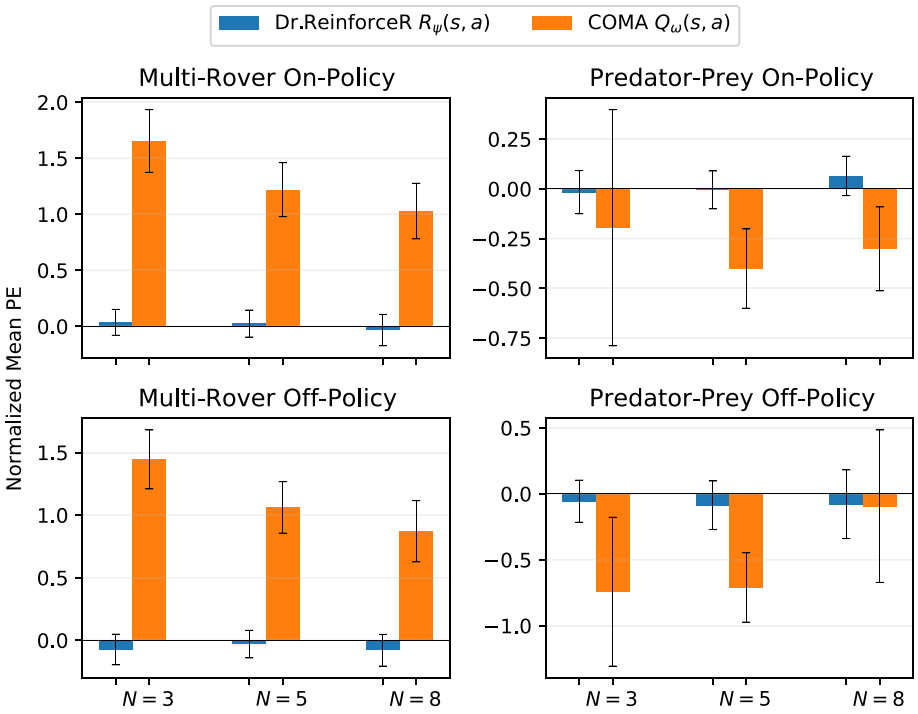


Fig. 3 Normalized mean prediction error and standard deviation for Dr.ReinforceR reward network R_ψ and COMA critic Q_ω on the on-policy dataset (first row) and the off-policy dataset (second row), for the two environments

Fig. 3 shows the mean and standard deviation of the prediction error (PE) distribution of these networks. All the prediction errors have been normalized by the value of $r_{\max} - r_{\min}$ (respectively, $q_{\max} - q_{\min}$ for COMA critic) for each environment and number of agents individually, so that the resulting values are comparable across the two different methodologies and across different settings. It is to note that, although normalized, the errors may be higher than the normalization range itself, and thus exceed the value of 1 (as it is the case with the errors of COMA critic on the multi-rover domain).

These plots give us some insights on the performance reported in Sect. 4.1. Dr.ReinforceR is in general achieving improved performances with respect to the compared baselines, and the low prediction error of its reward network on the two problems may be an explanation for this: with correct value estimates, the learning signals provided to the policy gradients are better in turn, and thus lead to higher-return behaviours. Also the variance is low, meaning that most of the sampled values are consistently predicted correctly and the network exhibits good generalization performances across the increasing number of agents on both datasets. This generalization capacity of the learned approximation also explains why Dr.ReinforceR is in general matching the Dr.Reinforce upper bound: the difference rewards mechanism requires multiple predictions to compute the agents' signals and, if these are not accurate enough, the resulting values may be completely wrong and push the agents towards sub-optimal policies in turn.

The prediction errors for COMA action-value critic instead are higher, especially on the multi-rover domain, where the errors do not scale so gracefully in the number of agents even on the on-policy dataset. It can be observed that the critic network is biased towards overestimating most of the samples for the multi-rover domain, while instead underestimates them for predator–prey (especially more so on the off-policy dataset, where non-encountered state-action pairs may be sampled), thus resulting in bad estimations of the counterfactual baseline. On the predator–prey environment, it seems that COMA critic quickly overfits to the Q-function of a sub-optimal joint policy, resulting in a very low prediction error on the off-policy dataset when the number of agents increases (and most of the samples indeed lead to no rewards trajectories), that does not seem able to give good signals to the agents' policies and leads them to get stuck into this poor local optimum in turn. These results can also explain why COMA is performing worse than Central$Q$ on this domain: if the critic is not accurate or is representing the value of a poor policy (as it can be hypothesized for the above results), COMA requirement of more estimations from it in order to compute the counterfactual baseline only

exacerbates this problem and further hinders the final performance.

Finally, the effect of noise on computation of the difference rewards are investigated. Generally, an accurate reward value for every agent's action is needed to compute correct difference rewards. The reward network R_ψ is an approximation of the true reward function $R(s, a)$ and can therefore give noisy estimates that could dramatically affect the resulting computation. To investigate this, noise sampled from different processes is added to the reward values of the agent's different actions that are obtained from the environment. These are used to compute the baseline (the second term of the r.h.s. in Eq. 5, as this is the only term for which R_ψ is used in Eq. 9), and the resulting difference rewards are compared with the true ones for a generic agent i under a uniform policy $\pi_{\theta^i}(a^i|s) = \frac{1}{|A^i|}$. Fig. 4 reports the mean value and variance over 1000 noise samples of a set of sampled state-action (SA) pairs from the reward function of the two investigated domains with $N = 3$ agents.

It can be observed how different noise processes differently affect the resulting difference rewards. For example, in both environments, the difference rewards mechanism is quite resistant against noise from a normal or a uniform distribution. This is probably due to the symmetricity of these noises that tends to cancel out with each other. However, a masking kind of noise, under which some of the reward values are replaced with a value of 0 with a certain probability, seems to be more detrimental for difference rewards evaluation: cancelling out some of the reward values definitely changes the computation and gives wrong estimates. This is worse in the multi-rover domain, in which the reward function is dense, while for the predator–prey environment and its sparse reward function it seems to be less harming.

These two observations together help explain why Dr.ReinforceR outperforms COMA on the two proposed environments: learning the reward function $R(s, a)$ is easier than learning the Q-function and, although function approximation introduces noise, the difference rewards mechanism is resistant against common types of noise and still provides useful signals to policy gradients. Therefore, if one is able to learn a good approximation of the reward, the proposed algorithm learns better and more reliable policies than other policy gradient algorithms, without the difficulties of learning the Q-function.

5 Partial observability

Full observability of the environment as in MMDPs is a desirable property, but in many real-world situations [40, 43, 55] such a strong assumption is often unrealistic. The complexity of the environment itself or the limited sensing or communication capabilities available are usually transforming such problems into a partially observable ones from the perspective of the agents. In these, the agents cannot directly observe the state of the environment, but instead are provided with a local and possibly noisy observation that represents only a limited amount of information about the underlying environment state itself.

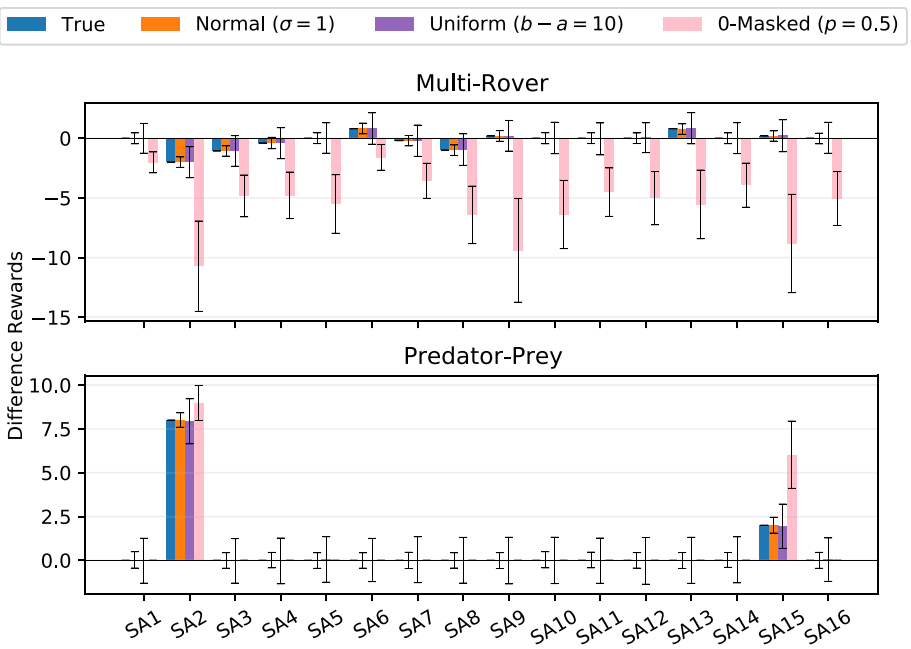


Fig. 4 Mean and variance of difference rewards for a set of samples under different noise profiles

Formally, such settings can be modelled as a decentralized partially observable Markov decision process (Dec-POMDP) [37] $\mathcal{M} = \langle \mathrm{D}, \mathrm{S}, \{\mathrm{A}^i\}_{i=1}^{|\mathrm{D}|}, \mathrm{T}, \mathrm{R}, \{O^i\}_{i=1}^{|\mathrm{D}|}, Z, \gamma \rangle$, where $\mathrm{D}, \mathrm{S}, \mathrm{A}^i, \mathrm{T}, \mathrm{R}$ and γ are the same as in a MMDP. As mentioned above, agents are provided with a local observation $o^i \in O^i$, such that $o = \langle o^1, \ldots, o^N \rangle \in \times_{i=1}^{|\mathrm{D}|} O^i = O$ is called a joint observation and $o \sim Z(s)$, where $Z : S \rightarrow O$ is the observation function. With such limitations, each agent has to keep track of its own action-observation history $h_t^i = (o_0^1, a_0^i, o_1^i, a_1^i, \ldots, o_{t-1}^i, a_{t-1}^i, o_t^i) \in (O^i \times A^i)^* \times O^i = \mathcal{H}^i$ up to the current time step t to try and assess the underlying state of the environment, and use this to condition its policy and draw its decisions. A joint history at time step t can also be defined as $h_t = (o_0, a_0, o_1, a_1, \ldots, o_{t-1}, a_{t-1}, o_t) \in (O \times A)^* \times O = \mathcal{H}$.

Policy gradients algorithms can easily be adapted to work under partial observability by simply replacing the environment state s used by the agents policies π_{θ^i} with the corresponding agent's local action-observation history h_t^i. The distributed policy gradients in Eq. (3) thus becomes:

$$\theta^i \leftarrow \theta^i + \alpha \underbrace{\sum_{t=0}^{T-1} \gamma^t G_t \nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i | h_t^i)}_{g^i}. \tag{16}$$

5.1 Method

Similarly, it is straightforward to also adapt Dr.Reinforce to work in Dec-POMDPs by simply adjusting the policy terms that appear in Eq. (6) and Eq. (7) to condition on the agents' local action-observation histories h_t^i. The difference return ΔG_t^i in thus defined as:

$$\Delta G_t^i(a_{t:T}^i | s_{t:T}, a_{t:T}^{-i}, h_{t:T}^i) \triangleq \sum_{l=0}^{T-t-1} \gamma^l \Delta R^i(a_{t+l}^i | s_{t+l}, a_{t+l}^{-i}, h_{t+l}^i), \tag{17}$$

while the decentralized policies are learned by using the update target:

$$\theta^i \leftarrow \theta^i + \alpha \underbrace{\sum_{t=0}^{T-1} \gamma^t \Delta G_t^i(a_{t:T}^i | s_{t:T}, a_{t:T}^{-i}, h_{t:T}^i) \nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i | h_t^i)}_{g^{\mathrm{DR},i}}. \tag{18}$$

When complete access to the reward function is not available, a modified version of Dr.ReinforceR can be applied. The centralized reward network R_ψ, by following the CTDE paradigm, can still be learned in the same way as in Eq. (8) and condition on the environment state $s \in S$, as it is not required during execution. It is enough to adapt Eq. (9) as done before, thus obtaining:

$$\Delta R_\psi^i(a_t^i | s_t, a_t^{-i}, h_t^i) \triangleq r_t - \sum_{c^i \in A^i} \pi_{\theta^i}(c^i | h_t^i) R_\psi(s_t, \langle c^i, a_t^{-i} \rangle), \tag{19}$$

and consequently adjust Eq. (10) as:

$$\Delta G_t^i(a_{t:T}^i | s_{t:T}, a_{t:T}^{-i}, h_{t:T}^i) \triangleq \sum_{l=0}^{T-t-1} \gamma^l \Delta R_\psi^i(a_{t+l}^i | s_{t+l}, a_{t+l}^{-i}, h_t^i). \tag{20}$$

5.2 Theoretical results

Above, we adapted Dr.Reinforce, which intuitively can improve learning by providing individual agents with a better learning signal, to partially observable settings. In these, using difference rewards as the agents' learning signals induces a partially observable stochastic game [23, 37] $\hat{\mathcal{P}} = \langle \mathrm{D}, \mathrm{S}, \{\mathrm{A}^i\}_{i=1}^{|\mathrm{D}|}, \mathrm{T}, \{\Delta \mathrm{R}^i\}_{i=1}^{|\mathrm{D}|}, \{O^i\}_{i=1}^{|\mathrm{D}|}, Z \rangle$ in which the cooperating agents do not receive the same reward after each time step. Even though difference rewards are aligned with the true reward values [1, 34], for these games convergence to an optimal solution is not immediate.

When agents are required to base their decisions on their local action-observation history h_t^i, the same result on an unbiased baseline derived in Sect. 3.3 for the fully observable case does not hold anymore. Generally speaking, this is due to the Monte Carlo nature of the difference return ΔG_t^i that requires future quantities in order to compute the value of the baseline. The local histories for the episode time steps (used to compute the aristocrat utility values in the r.h.s. of Eq. (17)) are now strictly depending on the actions selected at the previous time steps, and thus break this independence of the baseline from the current action selection.

Observation In a Dec-POMDP setting, using difference return $\Delta G_t^i(a_{t:T}^i | s_{t:T}, a_{t:T}^{-i}, h_{t:T}^i)$ as the learning signal for policy gradients in Eq. (18) is in general not equivalent to subtracting an unbiased baseline $B^i(s_{t:T}, a_{t:T}^{-i}, h_{t:T}^i)$ from the distributed policy gradients in Eq. (3).

Proof We start by rewriting $\Delta G_t^i(a_{t:T}^i | s_{t:T}, a_{t:T}^{-i}, h_{t:T}^i)$ from Eq. (17) as:

$$\Delta G_t^i(a_{t:T}^i|s_{t:T}, a_{t:T}^{-i}, h_{t:T}^i) = \sum_{l=0}^{T-t-1} \gamma^l r_{t+l} - \sum_{l=0}^{T-t-1} \gamma^l \sum_{c^i \in A^i} \pi_{\theta^i}(c^i|h_{t+l}^i) R(s_{t+l}, \langle c^i, a_{t+l}^{-i} \rangle). \tag{21}$$

Note that the first term on the r.h.s. of Eq. (21) is the return G_t used in Eq. (3). We then define the second term on the r.h.s. of Eq. (21) as the baseline $B^i(s_{t:T}, a_{t:T}^{-i}, h_{t:T}^i)$:

$$B^i(s_{t:T}, a_{t:T}^{-i}, h_{t:T}^i) = \sum_{l=0}^{T-t-1} \gamma^l \sum_{c^i \in A^i} \pi_{\theta^i}(c^i|h_{t+l}^i) \cdot R(s_{t+l}, \langle c^i, a_{t+l}^{-i} \rangle). \tag{22}$$

We can thus rewrite the total expected update target for agent i as:

$$\begin{aligned}
\mathbb{E}_{\pi_\theta}\left[\hat{g}^{\mathrm{DR},i}\right] &= \mathbb{E}_{\pi_\theta}\Bigg[\sum_{t=0}^{T-1}\big(\nabla_{\theta^i} \log \pi_{\theta^i} \\
&\quad (a_t^i|h_t^i)\big)\Delta G_t^i(a_{t:T}^i|s_{t:T}, a_{t:T}^{-i}, h_{t:T}^i)\Bigg] \\
&= \mathbb{E}_{\pi_\theta}\Bigg[\sum_{t=0}^{T-1}\big(\nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i|h_t^i)\big)(G_t \\
&\quad -B^i(s_{t:T}, a_{t:T}^{-i}, h_{t:T}^i))\Bigg] \\
&\quad \text{(by definition of } \Delta G_t^i\text{)} \\
&= \mathbb{E}_{\pi_\theta}\Bigg[\sum_{t=0}^{T-1}\big(\nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i|h_t^i)\big)G_t \\
&\quad -\big(\nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i|h_t^i)\big)B^i(s_{t:T}, a_{t:T}^{-i}, h_{t:T}^i)\Bigg] \\
&\quad \text{(distributing the product)} \\
&= \mathbb{E}_{\pi_\theta}\Bigg[\sum_{t=0}^{T-1}\big(\nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i|h_t^i)\big)G_t\Bigg] \\
&\quad - \mathbb{E}_{\pi_\theta}\Bigg[\sum_{t=0}^{T-1}\big(\nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i|h_t^i)\big)B^i(s_{t:T}, a_{t:T}^{-i}, h_{t:T}^i)\Bigg] \\
&\quad \text{(by linearity of the expectation)} \\
&= \mathbb{E}_{\pi_\theta}\left[\hat{g}^{\mathrm{PG},i}\right] + \mathbb{E}_{\pi_\theta}\left[\hat{g}^{B,i}\right].
\end{aligned}$$

In order to show that the baseline is unbiased the expected value of its update $\mathbb{E}_{\pi_\theta}[\hat{g}^{B,i}]$ with respect to the policy π_θ should be 0. Let

$$P^{\pi_\theta}(h_t) = P^{\pi_\theta}(h_{t-1}) \cdot \pi_\theta(a_{t-1}|h_{t-1}) \sum_{s_t \in S} P_t^{\pi_\theta}(s_t) \cdot Z(o_t, s_t)$$

(with $P^{\pi_\theta}(h_0) = \sum_{s_0 \in S} Z(o_0|s_0)\rho(s_0)$ and $\rho(s_0)$ the initial state distribution) be the joint action-observation history distribution. Let also define the *complete system history* $\hat{h}_t = \langle h_t, a_t, s_{0:t} \rangle \in \hat{\mathcal{H}}_t$, so that $P^{\pi_\theta}(\hat{h}_t) = P^{\pi_\theta}(h_t) \cdot \pi_\theta(a_t|h_t) \cdot \prod_{l=0}^{t} P_l^{\pi_\theta}(s_l)$, we have:

$$\begin{aligned}
\mathbb{E}_{\pi_\theta}\left[\hat{g}^{B,i}\right] &\triangleq -\mathbb{E}_{\pi_\theta}\Bigg[\sum_{t=0}^{T-1}\big(\nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i|h_t^i)\big)B^i(s_{t:T}, a_{t:T}^{-i}, h_{t:T}^i)\Bigg] \\
&= -\sum_{t=0}^{T-1}\sum_{\hat{h}_t \in \hat{\mathcal{H}}_t} P^{\pi_\theta}(\hat{h}_t)\big(\nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i|h_t^i)\big) \\
&\quad \sum_{\hat{h}_T \in \hat{\mathcal{H}}_T} P^{\pi_\theta}(\hat{h}_T|\hat{h}_t)\, B^i(s_{t:T}, a_{t:T}^{-i}, h_{t:T}^i) \\
&\quad \text{(by expanding the expectation)} \\
&= -\sum_{t=0}^{T-1}\sum_{h_t \in \mathcal{H}_t} P^{\pi_\theta}(h_t) \sum_{a_t^{-i} \in A^{-i}} \pi_{\theta^{-i}}(a_t^{-i}|h_t^{-i}) \\
&\quad \sum_{a_t^i \in A^i}\big(\nabla_{\theta^i} \pi_{\theta^i}(a_t^i|h_t^i)\big) \sum_{\hat{h}_T \in \hat{\mathcal{H}}_T} P^{\pi_\theta}(\hat{h}_T|\hat{h}_t)\, B^i(s_{t:T}, a_{t:T}^{-i}, h_{t:T}^i) \\
&\quad \text{(by applying the inverse log trick)} \\
&\neq -\sum_{t=0}^{T-1}\sum_{h_t \in \mathcal{H}_t} P^{\pi_\theta}(h_t) \sum_{a_t^{-i} \in A^{-i}} \pi_{\theta^{-i}}(a_t^{-i}|h_t^{-i}) \\
&\quad \left(\nabla_{\theta^i} \sum_{a_t^i \in A^i} \pi_{\theta^i}(a_t^i|h_t^i)\right) \\
&\quad \sum_{\hat{h}_T \in \hat{\mathcal{H}}_T} P^{\pi_\theta}(\hat{h}_T|\hat{h}_t)\, B^i(s_{t:T}, a_{t:T}^{-i}, h_{t:T}^i)) \\
&\quad \text{(by moving the gradient outside the policy sum)}
\end{aligned}$$

We cannot move the gradient outside of the sum now (as done in Eq. (14)), because of the baseline B^i depending on the policy parameters via the agent action a_t^i included in the histories $h_{t+1:T}^i$. The sum over the policy term is therefore a weighted summation over different baseline values, and these in general do not sum up to 0, and thus, the baseline is in general not unbiased (although problems for which the summation is 0 in any case may exist, and in these special cases the baseline is still unbiased). □

The result in the above Lemma shows that using the baseline in Eq. (22) alter the expected value of the overall gradient, as the baseline $B^i(s_{t:T}, a_{t:T}^{-i}, h_{t:T}^i)$ is not unbiased, and thus, the policy gradients are not guaranteed to converge to the same solutions of the distributed policy gradients [39].

5.3 StarCraftII experiments

Although there is no theoretical guarantee on the convergence of our proposed method under partial observability, it might still work well in practice. Therefore, we investigate the application of our method on the StarCraftII multi-

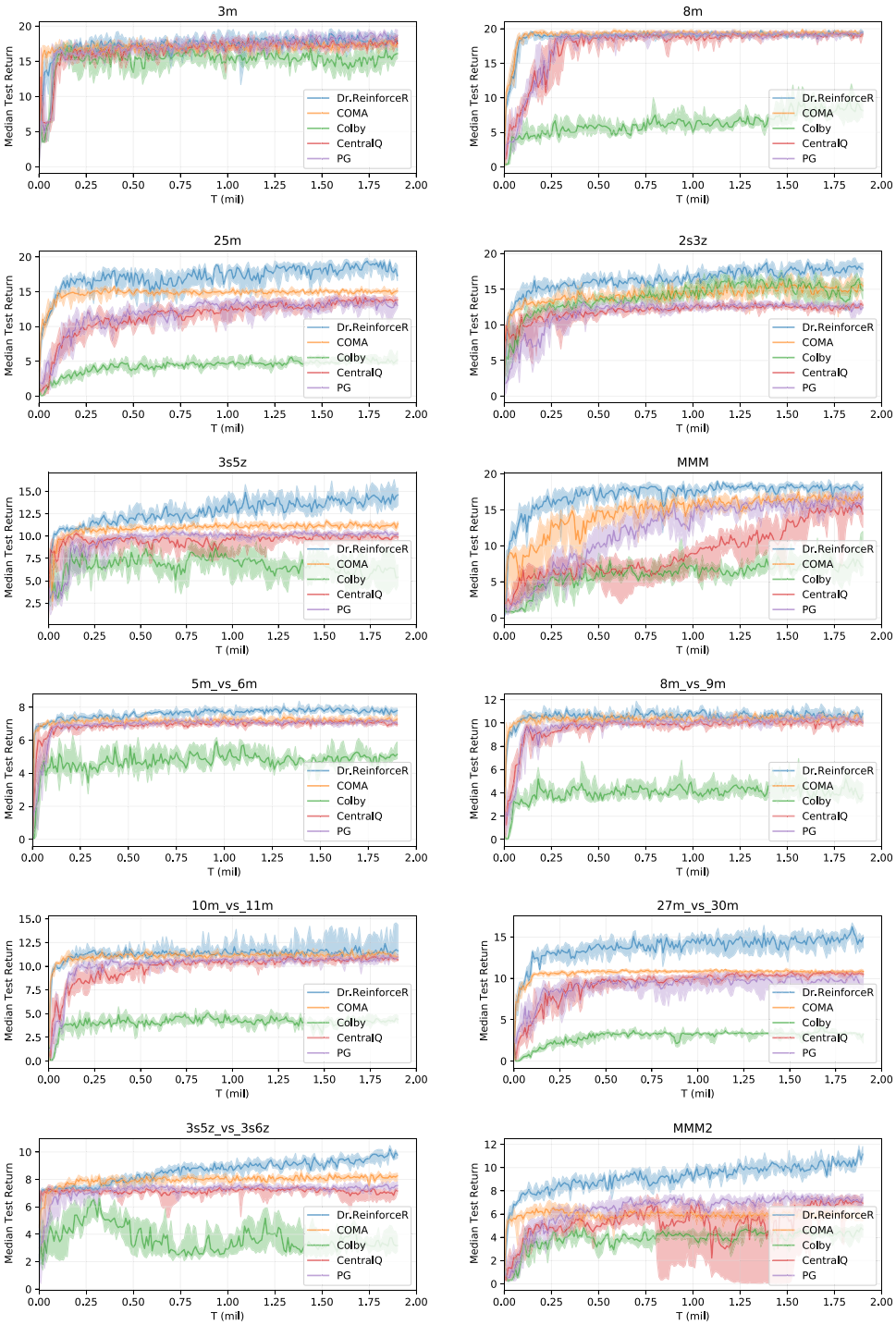
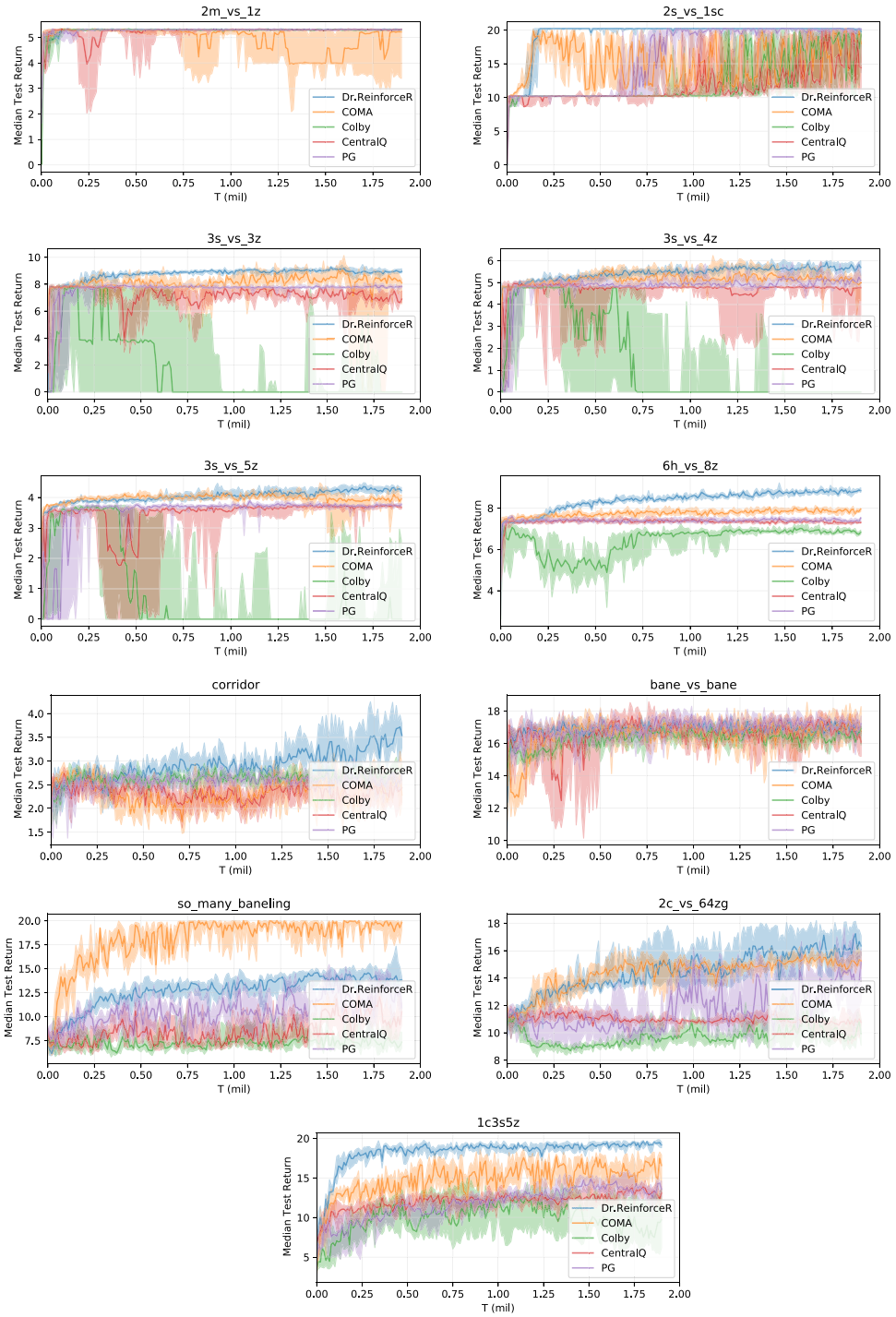


Fig. 5 Training curves on the entire set of SMAC maps, showing the median return and $25-75\%$ percentiles across seeds

agent challenge (SMAC) [43], a very complex, partially observable environment that provides a wide set of different maps, each with a different number and different types of units that has to fight against an opposing team controlled by the game AI, to show good empirical performances. As with the current game back end [50], it has not been possible to obtain all the reward values for the possible agents actions, we have not been able to apply Dr.Reinforce here. Figure 5 shows median return and $25-75\%$ percentiles across 10 independent runs on the whole set of available maps, with the difficulty level of the opponent team set to Very Hard.

Fig. 5 continued



In this setting, Dr.ReinforceR is almost never underperforming with respect to all the other baselines, with significant improvements over COMA on heterogeneous maps like `3s5z`, `1c3s5z` or `MMM`. This shows how learning the Q-function may be difficult in complex setting, while the reward network is easier to learn and in turn produces better policies. Also, it is worth mentioning that the severe partial observability of this setting is well addressed in practice by our use of the CTDE paradigm, with the reward network conditioned on the true state s: these results show how advantageous it is to resort to centralized training of the reward network over a local

approximation as in the algorithm from [12]. In particular, the good performance on the 25m map, involving a large number of agents, shows again the better scalability of the proposed centralized reward network with respect to a centralized Q-function critic, where the effects of bootstrapping and the moving target problem become even more severe when the number of agents grows larger.

A noticeable exception is represented by the so_many_baneling map, where COMA is achieving good results, where neither Dr.ReinforceR and all the other baselines are outperformed. An hypothesis for this is that the difference return ΔG_t^i is driving each agent into performing the more rewarding actions at each step (for example, hit an opponent if possible), but in the long run this strategy is not a winning one on this particular map, and thus, the agents never experience the high reward for winning and are thus never able to change their learned behaviours. Reasoning on the more complex Q-function here could be helpful to drive the policies towards a winning situation at the cost of performing actions that seem sub-optimal at the current step. In Appendix D, we also report the median win rate obtained by the investigated algorithms. From these, we can observe that, even when Dr.ReinforceR is capable of learning high return policies, these may not be sufficient to also achieve a significant win rate in some scenarios (for example, on more challenging maps with asymmetrical teams, like 6h_vs_6h or MMM2, although the gap in achieved median returns with respect to all the other baselines is very significant).

6 Related work

Application of reinforcement learning techniques to multi-agent systems has a long and fruitful history [5]. Fundamental works like [48] were the first to investigate the applicability of these algorithms in the form of independent learners to cooperative settings, while [11] further analyses the dynamics of their learning process depending on their consideration of the others. Specific algorithms to improve performance by learning the value of cooperation and coordination has been proposed, like in [21]. Also policy gradients has been widely applied to cooperative settings: [39] first proved convergence of distributed policy gradients to the same solution obtained by a centralized agent. Closer to our approach are recent works of policy gradients with deep reinforcement learning: for example, [18] presents COMA that efficiently estimates a counterfactual baseline for a team of cooperating homogeneous agents using a centralized critic for discrete problems. [44] takes inspiration from game theory and regret minimization to design a family of algorithms based on counterfactual regret minimization for partially observable domains. [57] combines actor-critic with a consensus mechanism to solve cooperative problems when communication is available, and also provide convergence proof under certain conditions, while [51] combines value decomposition with a counterfactual baseline in the actor-critic framework. All the above algorithms use the action-value function in order to compute the counterfactuals that can be difficult to learn because of bootstrapping target problems. Our method on the other hand learns the reward function to approximate the difference rewards that do not suffer from these problems. For a more extensive review on recent deep reinforcement learning algorithms for cooperative multi-agent systems see [24, 38].

Another important line of work for us is that on difference rewards [54] that already served as a basis for some existing algorithms like COMA. [49] uses difference rewards in learning to control a fleet of air vehicles that has to coordinate on traffic routes. [35] proposes two difference rewards-based value functions to improve multi-agent actor-critic in the $\mathbb{C}$Dec-POMDP setting, while [16] combines difference rewards and dynamic potential-based reward shaping [14, 15] to improve performance and convergence speed. Also, [56] applies difference rewards to multi-objective problems, speeding up learning and improving performance. Finally, some works try to improve the standard definition of difference rewards: [41] proposes to approximate difference rewards using tabular linear functions when it is not possible to access the value of the reward for the default action through a simulator, while [12, 13] both propose to approximate the difference rewards by using only local information. With the exception of the latter, the aforementioned works all uses value based algorithms to learn, while our method resorts to a policy gradients algorithm that recently showed great premise in multi-agent learning contexts.

Finally, the idea of learning the reward function has also received some attention, especially in the single-agent setting. [42] learns an additional state-reward network to reduce variance when updating the value function in noisy environments, [9] uses Kalman filters in problems with noise coming from different sources to explicitly learn about the reward function and the noise term, while [25] proposes UNREAL that additionally learn to predict rewards as an auxiliary task to improve deep reinforcement learning agent performance. Finally, [7] learns a factored reward representation for multi-agent cooperative one-shot games. While these works learn the reward function, these

are mainly limited to the single-agent setting (with the exceptions of [7, 9], which analyse different aspects from our and can be considered orthogonal and used in conjunction with our work) and do not use it to approximate the difference rewards.

7 Discussion and future work

Despite the good empirical results obtained by Dr.ReinforceR in the experiments detailed above, Lemma 5.2 clearly shows that the combination of difference rewards and policy gradients in a partially observable setting has in general no theoretical guarantees of convergence, as the baseline that is subtracted from the distributed policy gradients is not unbiased. This means that experimental performance could be unstable or arbitrarily bad.

Here we try and identify possible alternatives to our investigated formulation that are capable of restoring the theoretical convergence guarantees. This could be ensured by replacing the current baseline $B^i(s_{t:T}, a^{-i}_{t:T}, h^i_{t:T})$ in Eq. (22) with a new $\tilde{B}^i(s_{t:T}, a^{-i}_{t:T}, h^i_t)$ that does not depend on the currently selected action a^i_t via the local histories $h^i_{t+1:T}$. We identified a couple of possible solutions that are not, however, investigated in the current paper:

1. Replace the current agent policy $\pi_{\theta^i}(a^i_t|h^i_t)$ with a fixed policy $\mu(a^i_t)$ (a type of difference rewards also proposed in [54]):

$$\tilde{B}^i(s_{t:T}, a^{-i}_{t:T}) = \sum_{l=0}^{T-t-1} \gamma^l \sum_{c^i \in A^i} \mu(c^i) \cdot R(s_{t+l}, \langle c^i, a^{-i}_{t+l} \rangle).$$

   This idea, however, would require to fix beforehand a policy $\mu(a^i_t)$ to use, a choice similar to that of the default action [41, 54] in Eq. (4).
2. Use the current agent policy $\pi_{\theta^i}(a^i_t|h^i_t)$, but do not condition on the local histories for the episode time steps $h^i_{t+1:T}$, but only on the current local history h^i_t:

$$\tilde{B}^i(s_{t:T}, a^{-i}_{t:T}, h^i_t) = \sum_{l=0}^{T-t-1} \gamma^l \sum_{c^i \in A^i} \pi_{\theta^i}(c^i|h^i_t) \cdot R(s_{t+l}, \langle c^i, a^{-i}_{t+l} \rangle).$$

3. Use a potential-based reward shaping mechanism. These are known to retain policy invariance in single-agent reinforcement learning, both under full observability [34] as well as partial one [17], while in multi-agent systems converge to the same set of Nash Equilibria of the policies learned with the shared reward alone [14, 15], while improve learning performance. In general, a potential-based reward shaping mechanism provides the agents with a shaped reward $\hat{r}$:

$$\hat{r} \triangleq r_t + \underbrace{F(s_t, s_{t+1})}_{\tilde{B}^i},$$

   where $F(s_t, s_{t+1}) = \gamma\phi(s_{t+1}) - \phi(s_t)$ and $\phi(s)$ is a suitable function that provides additional information on the state s, so that $F(s_t, s_{t+1})$ is unbiased in expectation with respect to the policy gradients, and thus keep the convergence guarantees. A particular form of potential-based reward shaping, which combines its benefit with those of difference rewards, is Counterfactual as Potential [16], in which the potential-based reward shaping function is:

$$\phi(s) = R(s^{-i}),$$

   and $R(s^{-i})$ is a reward term that marginalizes out the presence of agent i. It is to note that in general such term needs to be provided by the environment itself via the use of a simulator (as with difference rewards), with our learned reward network that issue could be overcome.

Another crucial aspect of Dr.ReinforceR is that it resorts to the CTDE framework [28, 38] to learn its centralized reward network. Although CTDE is a widely used and accepted methodology [18, 29], it indeed restricts the training procedure to be carried out offline and in a separate step from the agents execution. There are settings, however, in which being able to retain decentralized execution while being able to learn during real interactions with the environment may be required. In such cases, it may be appropriate to replace the centralized reward network R_ψ with a set of individual reward networks $R_{\psi^i}(s, a^i)$ (or $R_{\psi^i}(h^i_t, a^i)$ when learning in a Dec-POMDP), one for each agent i, to approximate the difference rewards computation. These local networks are learning the expected value of the reward for each agent when performing a certain action in a given situation, independently of what the others are doing

$$R_{\psi^i}(s, a^i) \approx \mathbb{E}_{\pi_{\theta^{-i}}}\left[R_\psi(s, \langle a^i, a^{-i} \rangle)\right].$$

This additional approximation is suitable to break the dependence from the CTDE paradigm, although it may introduce approximation error in the local reward terms via the expectation over the other agents policies (while the centralized reward network R_ψ is in principle capable of perfectly approximate the reward function $R(s, a)$ and thus provide the policy gradients with perfect difference rewards values).

8 Conclusions

In cooperative multi-agent systems agents face the problem of figuring out how they are contributing to the overall performance of the team in which only a shared reward signal is available. Previous methods like COMA, a state-of-the-art difference rewards algorithm, used the action-value function to compute an individual signal for each agent to drive policy gradients. However, learning a centralized Q-function is problematic due to inherent factors like bootstrapping or the dependence on the joint action.

We proposed Dr.Reinforce, a novel algorithm that tackles multi-agent credit assignment by combining policy gradients and differencing of the reward function. When the true reward function is known, our method outperforms all compared baselines on two benchmark multi-agent cooperative environments with a shared reward signal, and scales much better with the number of agents, a crucial capability for real cooperative multi-agent scenarios.

Additionally, for settings in which such reward function is not known, we additionally proposed Dr.ReinforceR that learns a centralized reward network used for estimating the difference rewards. Although the reward function has got the same dimensionality of the Q-function used by COMA, its learning is easier as no bootstrapping or moving target is involved. Although learning a reward network capable of appropriately generalizing across the state-action space may be challenging and have pitfalls, we showed how Dr.ReinforceR is able to outperform COMA, a state-of-the-art difference rewards algorithm, and achieve higher performance.

Therefore, exploring how to improve the representational capabilities of the reward network to allow it to better generalize to unseen situations and to be applicable to more complex scenarios is an interesting future direction that could further push the performance of these methods.

Appendix A: Hyperparameters and training

For our implementation, we relied on and expanded the `pymarl` [43] framework, as already providing many useful tools and the official implementation of COMA to compare against. The policy networks are either feedforward networks for the two gridworld problems or GRU [10] to deal with partial observability on SMAC, and both use parameter sharing across agents [22] to reduce training time, while the critics and reward networks use feedforward networks instead.

On each of the three problems independently, the optimal values for policy learning rate α_θ and the critic or reward network one $\alpha_{\omega/\psi}$ [19] have been found for each method through a gridsearch over a common set of standard values. We used the setting with $N = 3$ agents for the two gridworld environments and the map `2s3z` on SMAC, and the values obtained this way have been subsequently used for the other instances of the same problem, respectively. Table 1 reports the value of the used learning rates α_θ and $\alpha_{\omega/\psi}$ for each compared method on each problem.

COMA [18] and Central$Q$ critics have been trained using the TD(λ) [45, 46] variant presented in [18]. For these, the optimal value for the parameter λ with the learning rates already found by the gridsearches has also been assessed following the same procedure detailed above, resulting in the values in Table 2:

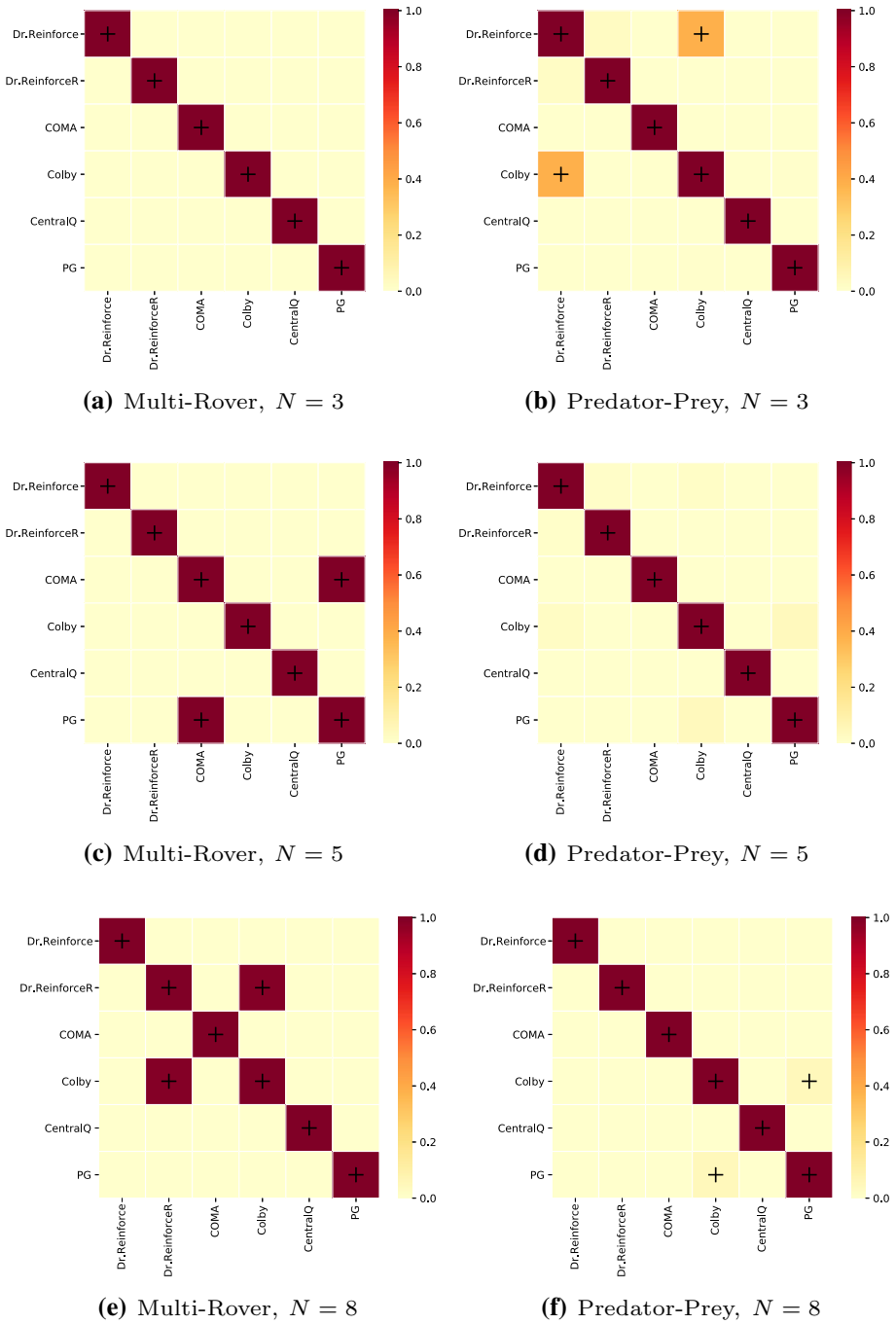
All the methods have been trained for the same amount of steps and all their other hyperparameters are set to the corresponding default values provided by the `pymarl` framework, without being optimized: the reward network R_ψ and the critic network Q_ω for Central$Q$ and COMA all

Table 2 Value of λ for each method

Method	Multi-rover	Predator–prey	SMAC
COMA	0.4	0.8	0.8
Central$Q$	0.2	0.8	0.8

Table 1 Value of the learning rates for each method

	Multi-rover		Predator–prey		SMAC	
Method	α_θ	$\alpha_{\omega/\psi}$	α_θ	$\alpha_{\omega/\psi}$	α_θ	$\alpha_{\omega/\psi}$
Dr.Reinforce	$25 \cdot 10^{-4}$	N.A.	$25 \cdot 10^{-4}$	N.A.	N.A.	N.A.
Dr.ReinforceR	$25 \cdot 10^{-4}$	$25 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$25 \cdot 10^{-4}$	$25 \cdot 10^{-4}$	$25 \cdot 10^{-4}$
COMA	$1 \cdot 10^{-2}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-2}$	$5 \cdot 10^{-4}$	$25 \cdot 10^{-4}$	$5 \cdot 10^{-4}$
[12]	$5 \cdot 10^{-3}$	$25 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-2}$	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$
Central$Q$	$5 \cdot 10^{-4}$	$25 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$5 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$
PG	$5 \cdot 10^{-4}$	N.A.	$5 \cdot 10^{-4}$	N.A.	$5 \cdot 10^{-4}$	N.A.



(a) Multi-Rover, $N = 3$ **(b)** Predator-Prey, $N = 3$

(c) Multi-Rover, $N = 5$ **(d)** Predator-Prey, $N = 5$

(e) Multi-Rover, $N = 8$ **(f)** Predator-Prey, $N = 8$

Fig. 6 Results of the *t*-test for different methods' pairs, corrected using the Bonferroni correction term, on each problem instance

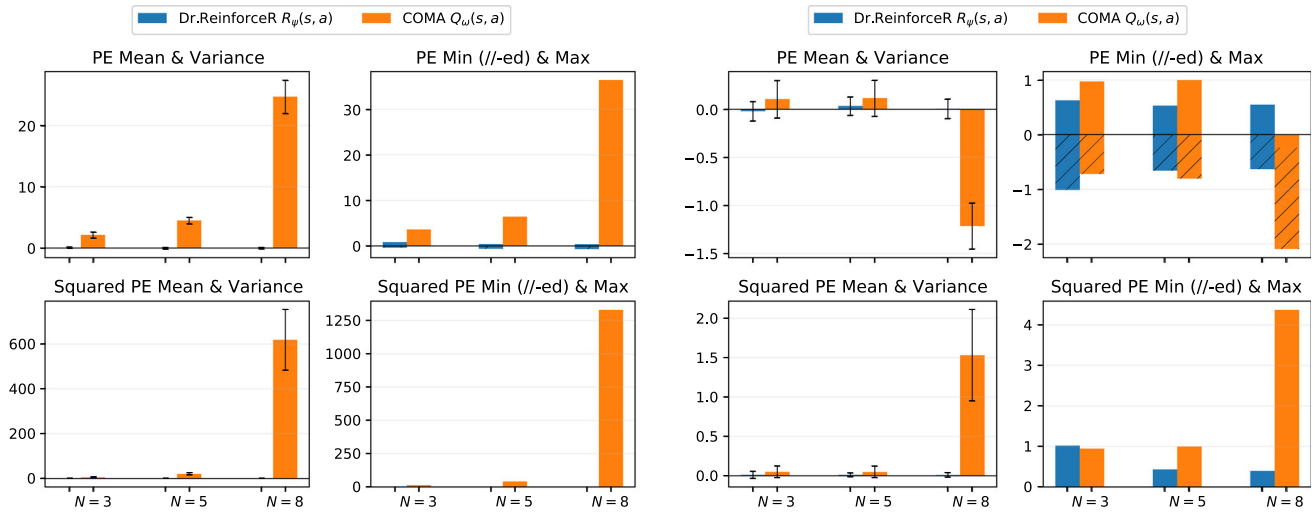
have the same structure, which is a two-layer feedforward neural network with 128 hidden units using the ReLU activation function [33] before the final linear layer, as the size of the functions these have to represent is analogous. Every experiment has been repeated 10 times with different random seeds to assess variance across multiple runs, and in each episode, the initial configuration has been randomly reset to avoid the policies to overfit.

Appendix B: Statistical significance tests

To assess the statistical significance of the proposed results, we computed a *t*-test on each algorithms' pair. The tested null hypothesis is that the samples (the return obtained by the different methods) are taken from the same distribution, meaning that any difference in the corresponding plotted lines are solely due to statistical noise rather than on the different capabilities of the algorithms. The test has been corrected with the Bonferroni correction term [2] to account for the possible errors across the different pairings. Test results are reported in Fig. 6, where a + symbol on a given cell means that the test value for a given algorithms' pair $p > 0.05$. It is to note that results on the diagonal are obtained pairing a method with itself, and thus are clearly statistically correlated.

Appendix C: Additional analysis plots

See Figs. 7, 8 and 9



(a) Multi-Rover **(b)** Predator-Prey

Fig. 7 Distribution statistics for Dr.ReinforceR reward network R_ψ and COMA critic Q_ω on the on-policy dataset, normalized by the value of $r_{\max} - r_{\min}$ (respectively, $q_{\max} - q_{\min}$ for COMA critic), for the two environments

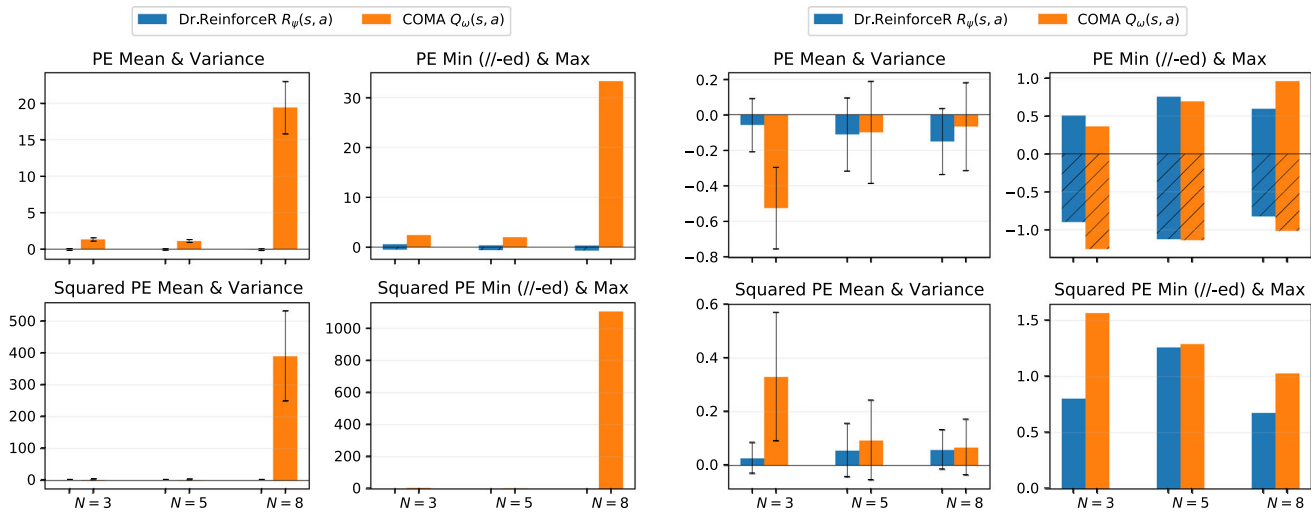


Fig. 8 Distribution statistics for Dr.ReinforceR reward network R_ψ and COMA critic Q_ω on the off-policy dataset, normalized by the value of $r_{\max} - r_{\min}$ (respectively, $q_{\max} - q_{\min}$ for COMA critic), for the two environments

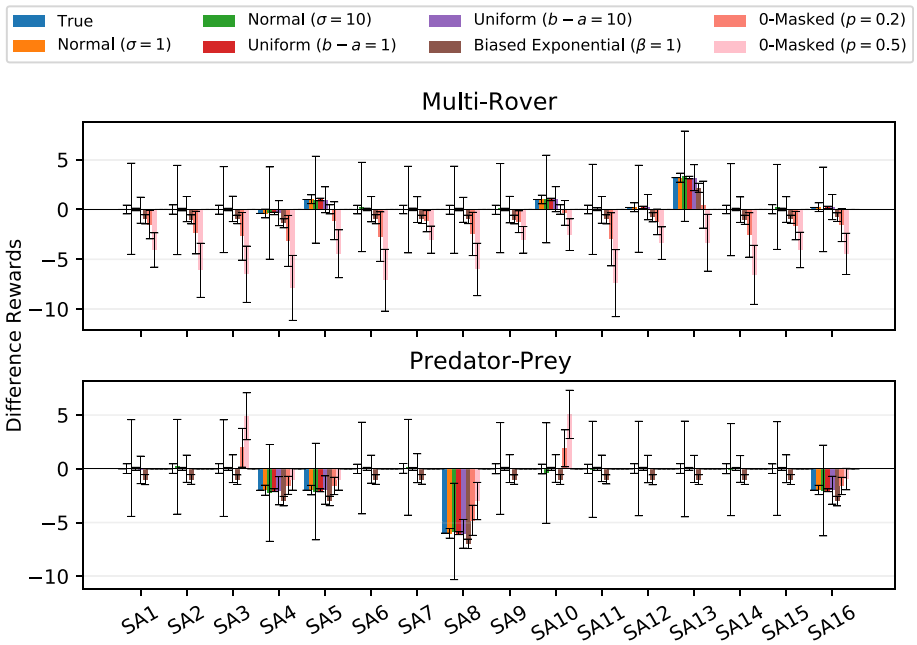


Fig. 9 Mean and variance of difference rewards for a set of samples under different noise profiles

Appendix D: Additional SMAC plots

See Fig. 10.

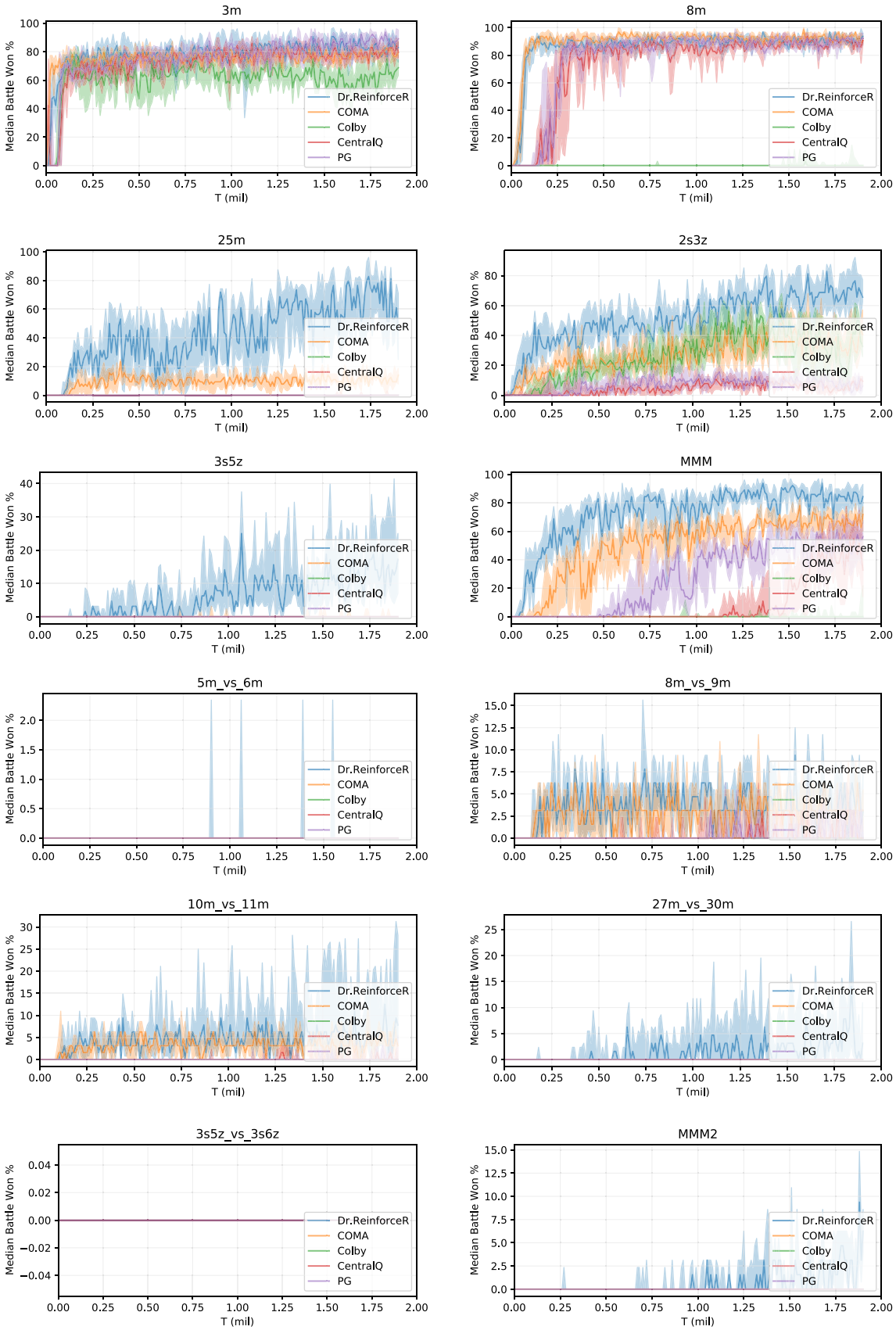


Fig. 10 Training curves on the entire set of SMAC maps, showing the median victory rate and $25-75\%$ percentiles across seeds

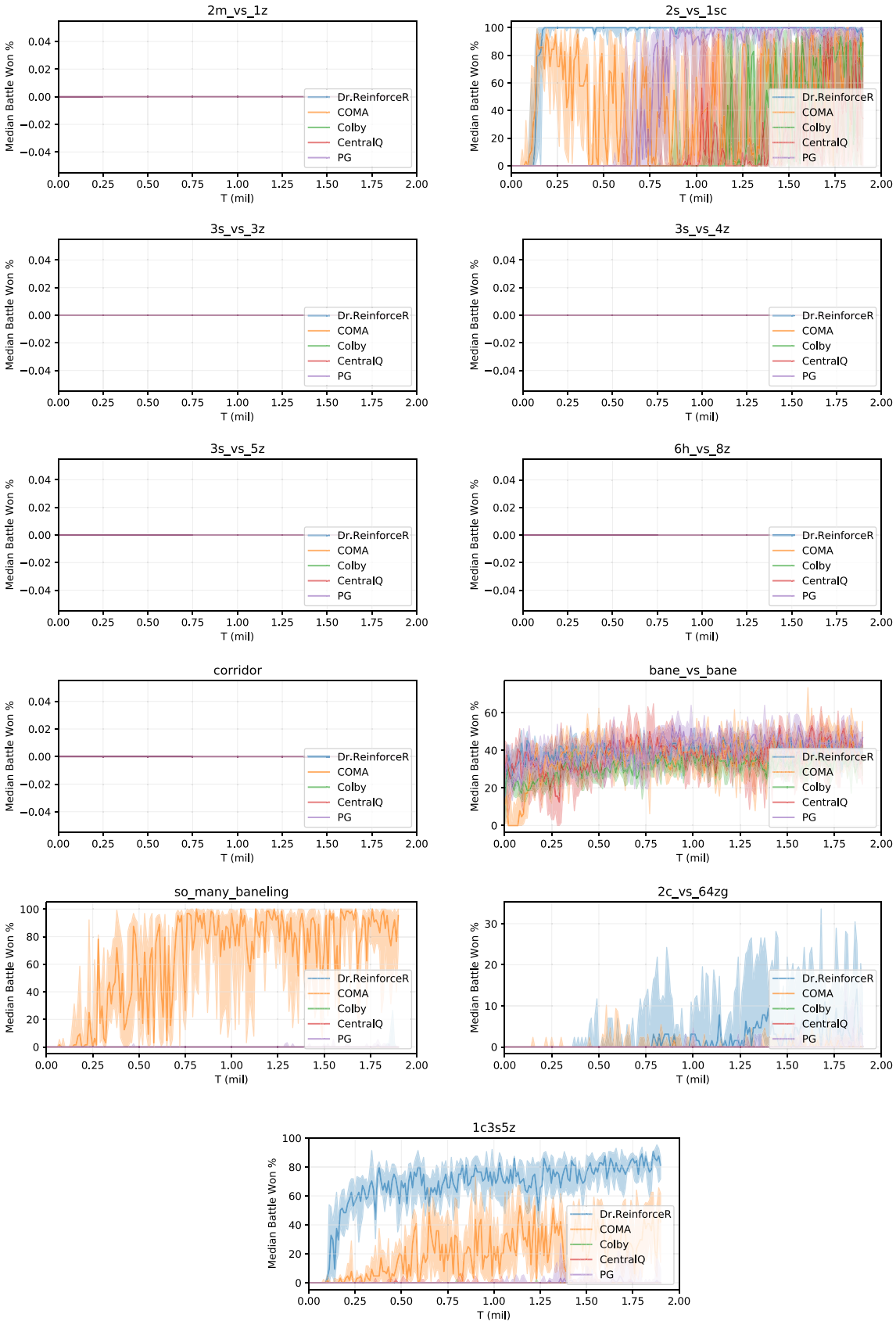


Fig. 10 continued

Acknowledgements This work was supported by an Azure for Research computing grant. F.A.O. is funded by EPSRC First Grant EP/R001227/1.  This project received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 758824 – INFLUENCE).

Declarations

Conflict of Interest The authors have no conflict of interest to declare that are relevant to the content of this article.

References

1. Agogino AK, Tumer K (2008) Analyzing and visualizing multiagent rewards in dynamic and stochastic domains. Auton Agent Multi-Agent Syst 17:320–338
2. Bonferroni CE (1936) Teoria statistica delle classi e calcolo delle probabilità. Pubbl del R Ist Super di Sci Econ e Commer di Firenze 8:3–62
3. Bottou L (1998) Online learning and stochastic approximations
4. Boutilier C (1996) Planning, learning and coordination in multiagent decision processes. In: Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge. Morgan Kaufmann Publishers Inc., TARK '96, pp. 195–210
5. Busoniu L, Babuska R, De Schutter B (2008) A comprehensive survey of multiagent reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics. Part C Appl Rev 38:156–172
6. Cao Y, Yu W, Ren W et al (2013) An overview of recent progress in the study of distributed multi-agent coordination. IEEE Trans Industr Inf 9(1):427–438
7. Castellini J, Oliehoek FA, Savani R, et al (2019) The representational capacity of action-value networks for multi-agent reinforcement learning. In: Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems. International Foundation for Autonomous Agents and Multiagent Systems, AAMAS'19, pp 1862–1864
8. Castellini J, Devlin S, Oliehoek FA, et al (2021) Difference rewards policy gradients. In: proceedings of the 20th international conference on autonomous agents and multiagent systems. international foundation for autonomous agents and multiagent systems, AAMAS'21, pp 1475–1477
9. Chang YH, Ho T, Kaelbling LP (2003) All learning is local: multi-agent learning in global reward games. In: advances in neural information processing systems 16. NIPS'03, MIT Press, p 807–814
10. Chung J, Gulcehre C, Cho KH, et al (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. In: NIPS'14 workshop on deep learning and representation learning. NIPS'14
11. Claus C, Boutilier C (1998) The dynamics of reinforcement learning in cooperative multiagent systems. In: Proceedings of the 15th/10th AAAI conference on artificial intelligence/innovative applications of artificial intelligence. american association for artificial intelligence, AAAI'98/IAAI'98, pp 746–752
12. Colby MK, Curran W, Rebhuhn C, et al (2014) Approximating difference evaluations with local knowledge. In: Proceedings of the 13th international conference on autonomous agents and multiagent systems. international foundation for autonomous agents and multiagent systems, AAMAS'14, pp 1577–1578
13. Colby MK, Curran W, Tumer K (2015) Approximating difference evaluations with local information. In: Proceedings of the 14th international conference on autonomous agents and multiagent systems. international foundation for autonomous agents and multiagent systems, AAMAS'15, pp 1659–1660
14. Devlin S, Kudenko D (2011) Theoretical considerations of potential-based reward shaping for multi-agent systems. In: AAMAS. international foundation for autonomous agents and multiagent systems, pp 225–232
15. Devlin S, Kudenko D (2012) Dynamic potential-based reward shaping. In: Proceedings of the 11th international conference on autonomous agents and multiagent systems. international foundation for autonomous agents and multiagent systems, AAMAS'12, pp 433–440
16. Devlin S, Yliniemi L, Kudenko D, et al (2014) Potential-based difference rewards for multiagent reinforcement learning. In: Proceedings of the 13th international conference on autonomous agents and multiagent systems. international foundation for autonomous agents and multiagent systems, AAMAS'14, pp 165–172
17. Eck A, Soh LK, Devlin S et al (2015) Potential-based reward shaping for finite horizon online pomdp planning. Auton Agent Multi-Agent Syst 30:403–445
18. Foerster JN, Farquhar G, Afouras T, et al (2018) Counterfactual multi-agent policy gradients. In: Proceedings of the 32th AAAI conference on artificial intelligence. AAAI Press, AAAI'18, pp 2974–2982
19. Fujimoto S, van Hoof H, Meger D (2018) Addressing function approximation error in actor-critic methods. In: Proceedings of the 36th international conference on machine learning. PMLR, ICML'18, pp 1587–1596
20. Greensmith E, Bartlett PL, Baxter J (2004) Variance reduction techniques for gradient estimates in reinforcement learning. J Mach Learn Res 5:1471–1530
21. Guestrin C, Lagoudakis MG, Parr R (2002) Coordinated reinforcement learning. In: Proceedings of the 19th international conference on machine learning. morgan kaufmann publishers Inc., ICML'02, pp 227–234
22. Gupta JK, Egorov M, Kochenderfer MJ (2017) Cooperative multi-agent control using deep reinforcement learning. autonomous agents and multi-agent systems. Springer, Cham pp 66–83 pp 66-83
23. Hansen EA, Bernstein DS, Zilberstein S (2004) Dynamic programming for partially observable stochastic games. In: Proceedings of the 19th AAAI conference on artifical intelligence. AAAI Press, AAAI'04, pp 709–715
24. Hernandez-Leal P, Kartal B, Taylor ME (2019) A survey and critique of multiagent deep reinforcement learning. Auton Agent Multi-Agent Syst 33:750–797
25. Jaderberg M, Mnih V, Czarnecki WM, et al (2016) Reinforcement learning with unsupervised auxiliary tasks. arXiv abs/1611.05397

26. Kaelbling LP, Littman ML, Moore AW (1996) Reinforcement learning: A survey. J Artif Intell Res 4(1):237–285
27. Konda VR, Tsitsiklis JN (2003) On actor-critic algorithms. SIAM J Control Optim 42(4):1143–1166
28. Kraemer L, Banerjee B (2016) Multi-agent reinforcement learning as a rehearsal for decentralized planning. Neurocomputing 190:82–94
29. Lowe R, Wu Y, Tamar A, et al (2017) Multi-agent actor-critic for mixed cooperative-competitive environments. In: Advances in neural information processing systems 30. NIPS'17, curran associates, Inc., p 6379–6390
30. Matignon L, Laurent GJ, Le Fort-Piat N (2012) Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. Knowl Eng Rev 27(1):1–31
31. Mnih V, Kavukcuoglu K, Silver D et al (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
32. Mnih V, Badia AP, Mirza M, et al (2016) Asynchronous methods for deep reinforcement learning. In: Proceedings 33rd international conference on machine learning. PMLR, ICML'16, pp 1928–1937
33. Nair V, Hinton GE (2010) Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on international conference on machine learning. omnipress, ICML'10, pp 807–814
34. Ng AY, Harada D, Russell S (1999) Policy invariance under reward transformations: Theory and application to reward shaping. In: Proceedings of the 16th international conference on machine learning. Morgan Kaufmann, ICML'99, pp 278–287
35. Nguyen DT, Kumar A, Lau HC (2018) Credit assignment for collective multiagent rl with global rewards. In: Advances in neural information processing systems 32. NIPS'18, curran associates, Inc., p 8113–8124
36. Nissim R, Brafman RI (2012) Multi-agent a* for parallel and distributed systems. In: Proceedings of the 11th international conference on autonomous agents and multiagent systems. international foundation for autonomous agents and multiagent systems, AAMAS'12, pp 1265–1266
37. Oliehoek FA, Amato C (2016) A concise introduction to decentralized POMDPs, 1st edn. Springer Publishing Company, Incorporated
38. Papoudakis G, Christianos F, Rahman A, et al (2019) Dealing with non-stationarity in multi-agent deep reinforcement learning. arXiv abs/1906.04737
39. Peshkin L, Kim KE, Meuleau N, et al (2000) Learning to cooperate via policy search. In: Proceedings of the 16th conference on uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., UAI'00, pp 489—496
40. Van der Pol E, Oliehoek FA (2016) Coordinated deep reinforcement learners for traffic light control. In: NIPS'16 workshop on learning, inference and control of multi-agent systems. NIPS'16
41. Proper S, Tumer K (2012) Modeling difference rewards for multiagent learning. In: Proceedings of the 11th international conference on autonomous agents and multiagent systems. international foundation for autonomous agents and multiagent systems, AAMAS'12, pp 1397–1398
42. Romoff J, Henderson P, Piche A, et al (2018) Reward estimation for variance reduction in deep reinforcement learning. In: Proceedings of the 6th international conference on learning representations, ICLR'18
43. Samvelyan M, Rashid T, Schröeder de Witt C, et al (2019) The starcraft multi-agent challenge. arXiv abs/1902.04043
44. Srinivasan S, Lanctot M, Zambaldi V, et al (2018) Actor-critic policy optimization in partially observable multiagent environments. In: Advances in neural information processing systems 32. NIPS'18, Curran Associates Inc., p 3426–3439
45. Sutton RS (1988) Learning to predict by the methods of temporal differences. Mach Learn 3(1):9–44
46. Sutton RS, Barto AG (1998) Introduction to reinforcement learning, 1st edn. MIT Press
47. Sutton RS, McAllester DA, Singh SP, et al (2000) Policy gradient methods for reinforcement learning with function approximation. In: Advances in neural information processing systems 12. NIPS'00, MIT Press, p 1057–1063
48. Tan M (1993) Multi-agent reinforcement learning: Independent vs. cooperative agents. In: Proceedings of the 10th international conference on machine learning. Morgan Kaufmann Publishers Inc., ICML'93, pp 330–337
49. Tumer K, Agogino A (2007) Distributed agent-based air traffic flow management. In: Proceedings of the 6th international conference on autonomous agents and multiagent systems. association for computing machinery, AAMAS'07
50. Vinyals O, Ewalds T, Bartunov S, et al (2017) StarCraft II: A new challenge for reinforcement learning. arXiv abs/1708.04782
51. Wang Y, Han B, Wang T, et al (2020) Off-policy multi-agent decomposed policy gradients. arXiv abs/2007.12322
52. Williams RJ (1992) Simple statistical gradient-gollowing algorithms for connectionist reinforcement learning. Mach Learn 8(3):229–56
53. Wolpert DH, Tumer K (1999) An introduction to collective intelligence. Tech. rep., NASA-ARC-IC-99-63, Nasa Ames Research Center
54. Wolpert DH, Tumer K (2001) Optimal payoff functions for members of collectives. Adv Complex Syst 4:265–280
55. Ye D, Zhang M, Yang Y (2015) A multi-agent framework for packet routing in wireless sensor networks. Sensors 15(5):10026–47
56. Yliniemi L, Tumer K (2014) Multi-objective multiagent credit assignment through difference rewards in reinforcement learning. In: Asia-Pacific conference on simulated evolution and learning. Springer International Publishing, pp 407–418
57. Zhang Y, Zavlanos MM (2019) Distributed off-policy actor-critic reinforcement learning with policy consensus. arXiv abs/1903.09255