Difference Rewards Policy Gradients

Jacopo Castellini
Dept. of Computer Science
University of Liverpool
Liverpool, United Kingdom
J.Castellini@liverpool.ac.uk

Sam Devlin
Microsoft Research Cambridge
Cambridge, United Kingdom
Sam.Devlin@microsoft.com

Frans A. Oliehoek
Interactive Intelligence Group
Delft University of Technology
Delft, The Netherlands
F.A.Oliehoek@tudelft.nl

Rahul Savani
Dept. of Computer Science
University of Liverpool
Liverpool, United Kingdom
rahul.savani@liverpool.ac.uk

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 Multiagent 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 [36], packet routing in sensor networks [42], and traffic light control [37], can be naturally modelled as *cooperative multi-agent systems* [5]. 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) [4, 16, 21], including recently with *deep reinforcement learning*. Often in these settings agents have to behave in a *decentralized fashion* [23], relying only on local perceptions, due to the prohibitive complexity of a centralized solution or because communication is too expensive [3, 27].

The paradigm of *centralized training with decentralized execution* (CTDE) [21, 28] 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 [29], and multi-agent policy gradient (MAPG) methods have become one of the most popular approaches for the CTDE paradigm [12, 22].

However, one key problem that agents face with CDTE that is not directly tackled by many MAPG methods is *multi-agent credit assignment* [7, 26, 40, 43]. 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* [9, 10, 30, 41] were proposed to tackle this problem: agents learn from a shaped reward that allows them to infer how their actions contributed to the shared reward.

Only one MAPG method has incorporated this idea so far: Counterfactual Multiagent Policy Gradients (COMA) [12] 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 known in many MARL problems.

To overcome these potential difficulties, we take inspiration from [9] and incorporate the *differencing of the reward function* into MAPG. 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 practical implementation, called Dr.ReinforceR, for settings where the reward function is not known upfront. In contrast to [9], 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.

Multi-Agent Reinforcement Learning. Our setting can be formalized as a multi-agent MDP [3, 27] $\mathcal{M} = \langle D, S, \{A^i\}_{i=1}^{|D|}, T, R, \gamma \rangle$, 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 $a = \langle a^1, \dots, a^N \rangle \in \times_{i=1}^{|D|} A^i = A$ denotes the joint action; $T(s'|s,a) : S \times A \times S \rightarrow [0,1]$ is the transition function that determines the probability of ending up in state s' from s under joint action a; $R(s,a) : S \times A \rightarrow \mathbb{R}$ is the shared reward function, and γ is the discount factor.

Agent i selects actions using a stochastic policy $\pi_{\theta^i}(a^i|s) : A^i \times S \rightarrow [0,1]$ with parameters θ^i, with $\theta = \langle \theta^1, \dots, \theta^N \rangle$ and $\pi_\theta = \langle \pi_{\theta^1}, \dots, \pi_{\theta^N} \rangle$ 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 $V^{\pi_\theta}(s_t) = \mathbb{E}\left[\sum_{l=0}^{\infty} \gamma^l r_{t+l} | s_t\right]$ and action-value function $Q^{\pi_\theta}(s_t, a_t) = \mathbb{E}\left[\sum_{l=0}^{\infty} \gamma^l r_{t+l} | s_t, a_t\right]$. At each time step t, the agents try to maximize the value-function $V^{\pi_\theta}(s_t)$.

REINFORCE and Actor-Critic. In single-agent reinforcement learning [19, 33], *policy gradient* methods [34] 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 $V(\theta) = \mathbb{E}_{s_0}\left[V^{\pi_\theta}(s_0)\right]$. The simplest policy gradient method is REINFORCE [39], which executes π_θ for an episode of T steps and then optimizes it with the following updates at each time step t:

$$\theta \leftarrow \theta + \alpha \underbrace{\gamma^t G_t \nabla_\theta \log \pi_\theta(a_t|s_t)}_{g_t},$$

where the return $G_t = \sum_{l=0}^{T} \gamma^l r_{t+l}$ is an unbiased estimate of $V^{\pi_\theta}(s_t)$ computed over the episode. This update rule corresponds to performing stochastic gradient ascent [2] on $V(\theta)$ because the expectation of the cumulative update target $\sum_{t=0}^{T} g_t$ is the gradient of the value-function, $\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} g_t\right] = \nabla_\theta V(\theta)$. Under appropriate choices of step sizes α the method will converge [34].

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 [33].

Actor-critic methods [20, 24] try to overcome this problem 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 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:

$$\theta \leftarrow \theta + \alpha Q_\omega(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t). \tag{2}$$

As for REINFORCE, a baseline $b(s)$ can be subtracted from the critic estimate in (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 [24].

In cooperative MARL, each agent i can individually learn a decentralized policy by using a *distributed policy gradient* [29] update target for π_{θ^i}:

$$\theta^i \leftarrow \theta^i + \alpha \underbrace{\gamma^t G_t \nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i|s_t)}_{g_t^i}, \tag{3}$$

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.

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 [30, 41]. 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* [41] 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 Colby et al. [9] 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 (4) as:

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

Counterfactual Multiagent Policy Gradients (COMA) [12] 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 $Q_\omega(s,a)$, by providing the policy gradients of the agents with a counterfactual advantage function:

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

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 (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 [25], 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 [39] 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:

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

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 [41] as in (5).

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

$$\theta^i \leftarrow \theta^i + \alpha \underbrace{\gamma^t \Delta G_t^i(a_{t:t+T}^i|s_{t:t+T}, a_{t:t+T}^{-i}) \nabla_{\theta^i} \log \pi_{\theta^i}(a_t^i|s_t)}_{g_t^{DR,i}}, \quad (7)$$

where ΔG_t^i is the difference return defined in (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:

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

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 [6].

We can now use the learned R_ψ to compute the difference rewards ΔR_ψ^i using the aristocrat utility [41] as:

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

The second term of the r.h.s. of (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 (6) as the discounted sum of the estimated difference rewards $\Delta R_\psi^i(s_t, a_t)$:

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

3.3 Theoretical Results

Above, we introduced Dr.Reinforce, which intuitively can improve learning by providing individual agents with a better learning signal. Using difference rewards as the agents' learning signals induces a stochastic game $\hat{\mathcal{P}} = \langle D, S, \{A^i\}_{i=1}^{|D|}, T, \{\Delta R^i\}_{i=1}^{|D|}, \rangle$ 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], for these games convergence to an optimal solution is not immediate, and is proven in this section.

Moreover, REINFORCE [39] 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 [14, 33]. We address these concerns in our multi-agent setting 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 (i.e., a multi-agent MDP) 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 [29], which under some technical conditions is known to converge to a local optimum [20, 34].

Lemma 1. *Using difference return ΔG_t^i as the learning signal for policy gradients in* (7) *is equivalent to subtracting an unbiased baseline $b^i(s_{t:t+T}, a_{t:t+T}^{-i})$ from the distributed policy gradients in* (3).

Proof. We start by rewriting ΔG_t^i from (6) as:

$$\Delta G_t^i = \sum_{l=0}^{T} \gamma^l r_{t+l} - \sum_{l=0}^{T} \gamma^l \sum_{b^i \in A^i} \pi_{\theta_t^i}(b^i|s_{t+l}) R(s_{t+l}, \langle b^i, a_{t+l}^{-i}\rangle). \quad (11)$$

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

$$b^i(s_{t:t+T}, a_{t:t+T}^{-i}) = \sum_{l=0}^{T} \gamma^l \sum_{b^i \in A^i} \pi_{\theta_t^i}(b^i|s_{t+l}) \cdot R(s_{t+l}, \langle b^i, a_{t+l}^{-i}\rangle). \quad (12)$$

We can thus rewrite the total expected update target as:

$$\begin{aligned}
\mathbb{E}_{\pi_\theta}\left[g_t^{DR,i}\right] &= \mathbb{E}_{\pi_\theta}\left[\nabla_{\theta^i}\log\pi_{\theta^i}(a_t^i|s_t)\Delta G_t^i\right] \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_{\theta^i}\log\pi_{\theta^i}(a_t^i|s_t)\left(G_t - b^i(s_{t:t+T}, a_{t:t+T}^{-i})\right)\right] \\
&\text{(by definition of } \Delta G_t^i\text{)} \\
&= \mathbb{E}_{\pi_\theta}[\nabla_{\theta^i}\log\pi_{\theta^i}(a_t^i|s_t)G_t \\
&- \nabla_{\theta^i}\log\pi_{\theta^i}(a_t^i|s_t)b^i(s_{t:t+T}, a_{t:t+T}^{-i})] \qquad (13) \\
&\text{(distributing the product)} \\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_{\theta^i}\log\pi_{\theta^i}(a_t^i|s_t)G_t\right] \\
&- \mathbb{E}_{\pi_\theta}\left[\nabla_{\theta^i}\log\pi_{\theta^i}(a_t^i|s_t)b^i(s_{t:t+T}, a_{t:t+T}^{-i})\right] \\
&\text{(by linearity of the expectation)} \\
&= \mathbb{E}_{\pi_\theta}\left[g_t^{pg,i}\right] + \mathbb{E}_{\pi_\theta}\left[g_t^{b,i}\right],
\end{aligned}$$

We have to show that the baseline is unbiased, and so the expected value of its update $\mathbb{E}_{\pi_\theta}\left[g_t^{b,i}\right]$ with respect to the policy π_θ is 0. Let $d^{\pi_\theta}(s)$ be the discounted ergodic state distribution induced by policy π_θ as defined in [33], we have:

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

Therefore, using the baseline in (12) reduces the variance of the updates [14] but does not change their expected value, as it is unbiased and its expected update target $\mathbb{E}_{\pi_\theta}\left[g_t^{b,i}\right] = 0$. □

Corollary 1. *Using the estimated reward network R_ψ to compute the baseline in* (12) *still results in an unbiased baseline.*

Proof. We rewrite ΔG_t^i from (10) as:

$$\Delta G_t^i = \sum_{l=0}^{T}\gamma^l r_{t+l} - \sum_{l=0}^{T}\gamma^l\sum_{b^i\in A^i}\pi_{\theta_t^i}(b^i|s_{t+l})R_\psi(s_{t+l}, \langle b^i, a_{t+l}^{-i}\rangle), \qquad (15)$$

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

$$b_\psi^i(s_{t:t+T}, a_{t:t+T}^{-i}) = \sum_{l=0}^{T}\gamma^l\sum_{b^i\in A^i}\pi_{\theta_t^i}(b^i|s_{t+l})\cdot R_\psi(s_{t+l}, \langle b^i, a_{t+l}^{-i}\rangle).$$

We observe that the derivation of (14) still holds, as it does not explicitly involve the reward network R_ψ in the computations. Therefore, the baseline $b_\psi^i(s_{t:t+T}, a_{t:t+T}^{-i})$ is again unbiased and does not alter the expected value of the updates. □

Theorem 1. *In a multi-agent MDP with shared rewards, given the conditions on function approximation detailed in [34], using Dr.Reinforce update target as in* (7)*, the series of parameters $\{\theta_t = \langle\theta_t^1, \ldots, \theta_t^N\rangle\}_{t=0}^k$ converges in the limit such that the corresponding joint policy π_{θ_t} is a local optimum:*

$$\lim_{k\to\infty}\inf_{\{\theta_t\}_{t=0}^k}||g_t|| = 0 \qquad w.p.\ 1.$$

Proof. To prove convergence, we have to show that:

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

We can rewrite the total expected update target as:

$$\mathbb{E}_{\pi_{\theta_t}}\left[g_t^{DR,i}\right] = \mathbb{E}_{\pi_{\theta_t}}\left[g_t^{pg,i}\right] + \mathbb{E}_{\pi_{\theta_t}}\left[g_t^{b,i}\right]$$

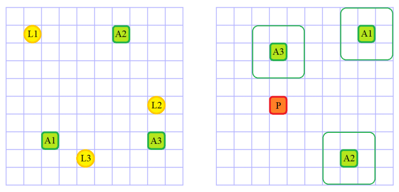
as in (13), and by Lemma 1 we have that $\mathbb{E}_{\pi_{\theta_t}}\left[g_t^{b,i}\right] = 0$. Therefore, the overall expected update $\mathbb{E}_{\pi_{\theta_t}}\left[g_t^{DR,i}\right]$ for agent i reduces to $\mathbb{E}_{\pi_{\theta_t}}\left[g_t^{pg,i}\right]$, that is equal to the distributed policy gradient update target in (3). These updates for all the agents has been proved to be equal to these of a centralized policy gradients agent $\mathbb{E}_{\pi_{\theta_t}}[g_t]$ by Theorem 1 in [29], and therefore converge to a local optimum of $\nabla_{\theta_t}V(\theta_t)$ by Theorem 3 in [34]. □

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 discrete environments with shared reward: the multi-rover domain, an established multi-agent cooperative domain [10], 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 [35].



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

Figure 1: Schematic representation of our two domains. Agents are green, landmarks are yellow, and prey are red.

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. We also compare against a standard A2C algorithm [20] 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 Q-A2C here. Our main comparison is with COMA [12], a state-of-the-art difference rewards method using the Q-function for computing the differences. Finally, we compare against the algorithm proposed by Colby et al. [9], to show the benefit of learning a centralized reward network to estimate the difference rewards in Dr.ReinforceR. We adapted this method to use policy gradients instead of evolutionary algorithms to optimise 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.

The optimal learning rates for the policy networks α_θ, the critics α_ω (set to an higher value than α_θ in order to allow these to learn faster than the policies and provide them with informative gradients, a standard assumption in policy gradient methods [13]) and the reward networks α_ψ for each method have been found through a gridsearch over the same set of standard values for each method independently on both tasks with only $N = 3$ agents, and these have been subsequently used for the same problems with more agents. Table 1 reports the value of the used learning rates α_θ and $\alpha_{\omega/\psi}$ for each compared method on each problem.

Table 1: Value of the learning rates for each method.

	Multi-Rover		Predator-Prey	
Method	α_θ	$\alpha_{\omega/\psi}$	α_θ	$\alpha_{\omega/\psi}$
Dr.Reinforce	$5 \cdot 10^{-4}$		$5 \cdot 10^{-4}$	
Dr.ReinforceR	$5 \cdot 10^{-4}$	$25 \cdot 10^{-3}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-4}$
REINFORCE	$5 \cdot 10^{-4}$		$5 \cdot 10^{-4}$	
Q-A2C	$5 \cdot 10^{-4}$	$25 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$5 \cdot 10^{-3}$
COMA [12]	$5 \cdot 10^{-5}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$5 \cdot 10^{-3}$
Colby [9]	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$5 \cdot 10^{-5}$

All the methods have been trained for the same amount of episodes and all their other hyperparameters are set to the same values and have not been optimized: the reward network R_ψ and the critic network Q_ω for Q-A2C and COMA all have the same structure, that is a single-layer feedforward neural network with 256 hidden units using ReLU as the activation function 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 different runs, and in each episode the starting positions of each agent and the other entities have been randomly reset to avoid the policies to overfit to a specific configuration.

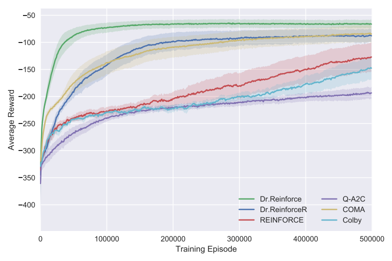
Multi-Rover Domain. In this domain, a team of N agents is placed into a 10×10 gridworld with a set of N landmarks. The aim of the team is to spread over all the landmarks and cover them (which agent covers which landmark is not important): the reward received by the team depends on the distance of each 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. Figure 2 reports the mean performance and 90% confidence interval (shaded area in the plots) across 10 independent runs obtained by the compared methods on a team of increasing size, to investigate scaling to larger multi-agent systems.

While both COMA and Dr.ReinforceR easily learn good policies with 3 agents, with Dr.ReinforceR matching the upper bound given by Dr.Reinforce, when the system gets larger both begin to struggle, achieving sub-optimal performance. Two things are interesting: the first one is that, despite not achieving optimal performance, Dr.ReinforceR always outperforms COMA and the other baselines on any scenario. The simpler learning problem used to provide signals to the agents' policies 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.

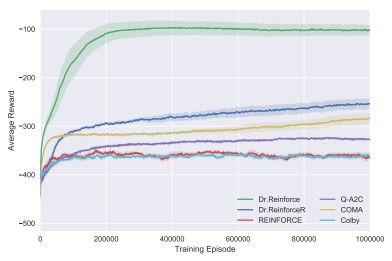
The second observation is that Dr.Reinforce, combining policy gradients with exact difference rewards, always learn high return policies. Given that this represents an upper bound to Dr.ReinforceR performance in case the reward network R_ψ learns a correct approximation, we can hypothesize that the gap in performance are due to a reward network that did not converge properly, and its performance can improve with a better representation of the reward function. 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 encountered during training but not being able to give correct predictions beyond those. It is true however that also difference methods using the action-value function have the same requirements.

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 this even if the gradients signals are accurate enough. We hypothesize that this is what happens in this case, with the policies converged to a sub-optimal solution and not being able to escape from it afterwards, under both Dr.ReinforceR and COMA methods.

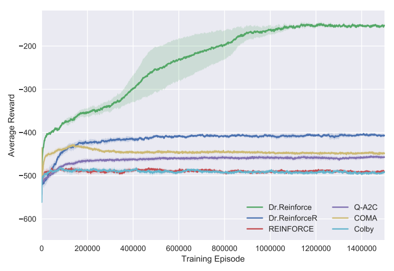
Predator-Prey. In this version of the classical predator-prey problem, a team of N predators has to pursue a single prey for as long



(a) $N = 3$



(b) $N = 5$



(c) $N = 8$

Figure 2: Training curves on the multi-rover domain, showing the mean reward and 90% confidence interval across seeds.

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. Figures 3 shows mean results and 90% confidence interval across 10 independent runs with teams comprising an increasing number of predators.

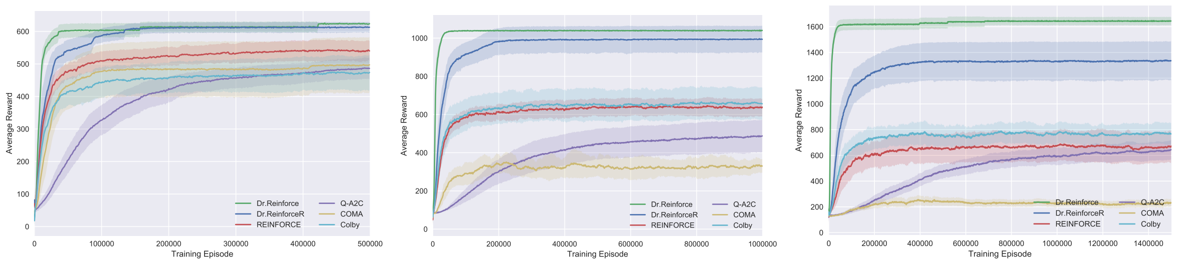
In this environment, Dr.ReinforceR is outperforming all the other compared methods, achieving performance that is equal or close to these of the Dr.Reinforce upper bound. On the other hand, the baselines struggles in learning something useful when more agents are introduced, with COMA being outperformed even by the simple independent learner baseline and Q-A2C when $N = 5$ or $N = 8$. This points out how accurately learning an optimal Q-function may be problematic in many settings, preventing efficient learning of policies when the system comprises more agents. If the Q-function converges to a sub-optimal solution and keeps pushing the agents towards this local optimum, the policies may struggle to escape from it and in turn push the action-value function toward a worst solution. 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, rendering the learning problem even more difficult. From this side, learning the reward function to compute the difference rewards is an easier learning problem, cast as a regression problem not involving bootstrapped estimates or moving target problems, and thus can improve policy gradients performance providing them with better learning signals in achieving high return policies 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 toward a specific region of the true reward function, fostering 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, we analyse the accuracy of the learned representations 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. We call this 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, we also analyse the prediction errors 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, we approximated that via 100 rollouts sampling starting from the state-action sample and following the corresponding learned policies. Figure 4 shows the mean and standard deviation of the prediction error distribution of these networks. We normalized all the prediction errors 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.

These plots give us some insights on the performance reported in Section 4.1: Dr.ReinforceR is maintaining near-optimal performance on the predator-prey environment when the number of agents increase, because the reward network prediction error stays low and scales proportionally to this. Also the variance does not increase, meaning that most of the sampled values are consistently predicted correctly and the network exhibits good generalization



(a) $N = 3$ (b) $N = 5$ (c) $N = 8$

Figure 3: Training curves on the predator-prey problem, showing the mean reward and 90% confidence interval across seeds.

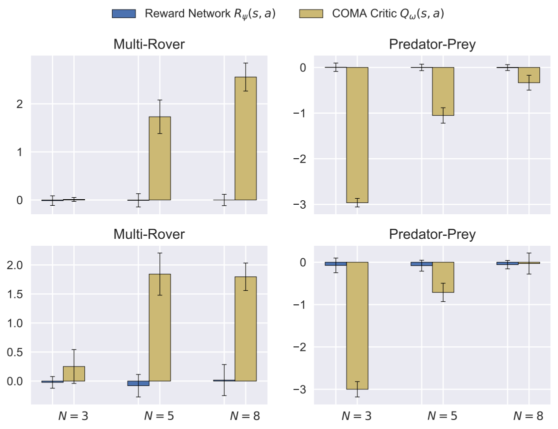


Figure 4: Normalized mean prediction error and standard deviation for Dr.ReinforceR reward network R_ψ and COMA critic Q_ω on the on-policy dataset (first row) and the off-policy dataset (second row), for the two environments.

performances on both datasets. Conversely, for the multi-rover domain both the mean and the standard deviation of the errors get increasingly larger on the off-policy dataset, pointing out how the network predicts correctly some of the samples, but it struggles to generalize properly across the entire reward function. This may be the cause of the sub-optimal performance of Dr.ReinforceR on the multi-rover domain when the number of agents increase. 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. We can observe that the critic network is biased toward overestimating most of the samples for the multi-rover domain, while instead underestimates them for predator-prey, thus resulting in bad estimations of the counterfactual baseline. On the predator-prey environment, it seems that COMA critic quickly overfit to a sub-optimal Q-function with a very low prediction error when the number of agents increase, that is not able to give good signal to the agents' policies and leading them to getting stuck into a local optimum in turn. These results can also explain why COMA is performing worse than Q-A2C in the predator-prey problem: if the critic is not accurate or is representing the value of a poor policy, COMA requirement of more estimations from it to compute the counterfactual baseline only exacerbates this problem and further hinder final performances.

Finally, we investigate the effect of noise on computation of the difference rewards. Generally, accurate reward values for every agent action are 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, we added noise sampled from different processes to reward values for each agent action that are obtained from the environment. We used these to compute the baseline (the second term of the r.h.s. in (5), as this is the only term for which R_ψ is used in (9)), and we compared the resulting difference rewards with the true ones for a generic agent i under a uniform policy $\pi_{\theta^i}(a^i|s) = \frac{1}{|A^i|}$. Figure 5 reports the mean value and variance over 1000 noise samples of a set of sampled state-action pairs from the reward functions of the two investigated problems with $N = 3$ agents.

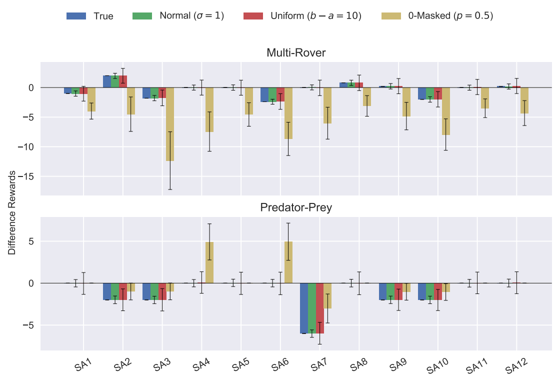


Figure 5: Mean and variance of difference rewards for a set of samples under different noise profiles.

We can observe 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 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 introduce noise, the difference rewards mechanism is resistant against common types of noise and still provides useful signals to policy gradients. Therefore, if we are able to learn a good approximation of the reward, our algorithm learns better and more reliable policies than other policy gradients algorithms, without the difficulties of learning the Q-function.

5 RELATED WORK

Application of reinforcement learning techniques to multi-agent systems has a long and fruitful history [4]. Fundamental works like that of Tan [35] were the first to investigate the applicability of these algorithms in the form of independent learners to cooperative settings, while Claus and Boutilier [8] further analyse 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 Guestrin et al. [15]. Also policy gradients has been widely applied to cooperative settings: Peshkin et al. [29] 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, Foerster et al. [12] present COMA, that efficiently estimates a counterfactual baseline for a team of cooperating homogeneous agents using a centralized critic for discrete problems. Srinivasan et al. [32] take inspiration from game theory and regret minimization to design a family of algorithms based on counterfactual regret minimization for partially observable domains. Zhang and Zavlanos [44] combine actor-critic with a consensus mechanism to solve cooperative problems when communication is available, and also provide convergence proof under certain conditions, while Wang et al. [38] combine 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 Papoudakis et al. [28] and Hernandez-Leal et al. [17].

Another important line of work for us is that on difference rewards [41], that already served as a basis for some existing algorithms like COMA. Tumer and Agogino [36] use difference rewards in learning to control a fleet of air vehicles that has to coordinate on traffic routes. Nguyen et al. [26] propose two difference rewards-based value-functions to improve multi-agent actor-critic in the ℂDec-POMDP setting, while Devlin et al. [11] combine difference rewards and potential-based reward shaping to improve performance and convergence speed. Also, Yliniemi and Tumer [43] apply difference rewards to multi-objective problems, speeding up learning and improving performance. Finally, some works try to improve the standard definition of difference rewards: Proper and Tumer [30] propose 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 Colby et al. [10] and Colby et al. [9] 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. Romoff et al. [31] learn an additional state-reward network to reduce variance when updating the value-function in noisy environments, Chang et al. [7] use Kalman filters in problems with noise coming from different sources to explicitly learn about the reward function and the noise term, while Jaderberg et al. [18] propose UNREAL, that additionally learn to predict rewards as an auxiliary task to improve deep reinforcement learning agent performance. Finally, Castellini et al. [6] learn 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] and [6], 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.

6 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.

REFERENCES

[1] Adrian K. Agogino and Kagan Tumer. 2008. Analyzing and visualizing multiagent rewards in dynamic and stochastic domains. *Autonomous Agents and Multi-Agent Systems* 17 (2008), 320–338.

[2] Léon Bottou. 1998. Online Learning and Stochastic Approximations.

[3] Craig Boutilier. 1996. Planning, Learning and Coordination in Multiagent Decision Processes. In *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge (TARK '96)*. Morgan Kaufmann Publishers Inc., 195—-210.

[4] Lucian Busoniu, Robert Babuska, and Bart De Schutter. 2008. A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38 (2008), 156–172.

[5] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. 2013. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial Informatics* 9, 1 (2013), 427–438.

[6] Jacopo Castellini, Frans A. Oliehoek, Rahul Savani, and Shimon Whiteson. 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 (AAMAS'19)*. International Foundation for Autonomous Agents and Multiagent Systems, 1862–1864.

[7] Yu-Han Chang, Tracey Ho, and Leslie P. Kaelbling. 2003. All Learning is Local: Multi-Agent Learning in Global Reward Games. In *Advances in Neural Information Processing Systems 16*. MIT Press, 807–814.

[8] Caroline Claus and Craig Boutilier. 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 (AAAI'98/IAAI'98)*. American Association for Artificial Intelligence, 746–752.

[9] Mitchell K. Colby, William Curran, Carrie Rebhuhn, and Kagan Tumer. 2014. Approximating Difference Evaluations with Local Knowledge. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'14)*. International Foundation for Autonomous Agents and Multiagent Systems, 1577–1578.

[10] Mitchell K. Colby, William Curran, and Kagan Tumer. 2015. Approximating Difference Evaluations with Local Information. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'15)*. International Foundation for Autonomous Agents and Multiagent Systems, 1659–1660.

[11] Sam Devlin, Logan Yliniemi, Daniel Kudenko, and Kagan Tumer. 2014. Potential-based Difference Rewards for Multiagent Reinforcement Learning. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'14)*. International Foundation for Autonomous Agents and Multiagent Systems, 165–172.

[12] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual Multi-Agent Policy Gradients. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence (AAAI'18)*. AAAI Press, 2974–2982.

[13] Scott Fujimoto, Herke van Hoof, and David Meger. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In *Proceedings of the 36th International Conference on Machine Learning (ICML'18)*. PMLR, 1587–1596.

[14] Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. 2004. Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning. *Journal of Machine Learning Research* 5 (2004), 1471–-1530.

[15] Carlos Guestrin, Michail G. Lagoudakis, and Ronald Parr. 2002. Coordinated Reinforcement Learning. In *Proceedings of the 19th International Conference on Machine Learning (ICML'02)*. Morgan Kaufmann Publishers Inc., 227–234.

[16] Jayesh K. Gupta, Maxim Egorov, and Mykel Kochenderfer. 2017. Cooperative Multi-agent Control Using Deep Reinforcement Learning. *Autonomous Agents and Multi-Agent Systems* (2017), 66–83.

[17] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. 2019. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems* 33 (2019), 750–797.

[18] Max Jaderberg, Volodymyr Mnih, Wojciech M. Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. 2016. Reinforcement Learning with Unsupervised Auxiliary Tasks. *arXiv* abs/1611.05397 (2016).

[19] Leslie P. Kaelbling, Michael L. Littman, and Andrew W. Moore. 1996. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research* 4, 1 (1996), 237–285.

[20] Vijay R. Konda and John N. Tsitsiklis. 2003. On Actor-Critic Algorithms. *SIAM Journal of Control and Optimization* 42, 4 (2003), 1143–1166.

[21] Landon Kraemer and Bikramjit Banerjee. 2016. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing* 190 (2016), 82–94.

[22] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 6379–6390.

[23] Laetitia Matignon, Guillaume J. Laurent, and Nadine Le Fort-Piat. 2012. Independent Reinforcement Learners in Cooperative Markov games: a Survey Regarding Coordination Problems. *Knowledge Engineering Review* 27, 1 (2012), 1–31.

[24] Volodymyr Mnih, Adrià P. Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings 33rd International Conference on Machine Learning (ICML'16)*. PMLR, 1928–1937.

[25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[26] Duc T. Nguyen, Akshat Kumar, and Hoong C. Lau. 2018. Credit Assignment for Collective Multiagent RL with Global Rewards. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 8113–8124.

[27] Frans A. Oliehoek and Christoper Amato. 2016. *A Concise Introduction to Decentralized POMDPs* (1st ed.). Springer Publishing Company, Incorporated.

[28] Georgios Papoudakis, Filippos Christianos, Arrasy Rahman, and Stefano V. Albrecht. 2019. Dealing with Non-Stationarity in Multi-Agent Deep Reinforcement Learning. *arXiv* abs/1906.04737 (2019).

[29] Leonid Peshkin, Kee-Eung Kim, Nicolas Meuleau, and Leslie Pack Kaelbling. 2000. Learning to Cooperate via Policy Search. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI'00)*. Morgan Kaufmann Publishers Inc., 489–-496.

[30] Scott Proper and Kagan Tumer. 2012. Modeling Difference Rewards for Multiagent Learning. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'12)*. International Foundation for Autonomous Agents and Multiagent Systems, 1397–1398.

[31] Joshua Romoff, Peter Henderson, Alexandre Piche, Vincent Francois-Lavet, and Joelle Pineau. 2018. Reward Estimation for Variance Reduction in Deep Reinforcement Learning. In *Proceedings of the 6th International Conference on Learning Representations (ICLR'18)*. 6.

[32] Sriram Srinivasan, Marc Lanctot, Vinicius Zambaldi, Julien Pérolat, Karl Tuyls, Remi Munos, and Michael Bowling. 2018. Actor-critic Policy Optimization in Partially Observable Multiagent Environments. In *Advances in Neural Information Processing Systems 32*. Curran Associates Inc., 3426–3439.

[33] Richard S. Sutton and Andrew G. Barto. 1998. *Introduction to Reinforcement Learning* (1st ed.). MIT Press.

[34] Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 2000. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems 12*. MIT Press, 1057–1063.

[35] Ming Tan. 1993. Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. In *Proceedings of the 10th International Conference on Machine Learning (ICML'93)*. Morgan Kaufmann Publishers Inc., 330–337.

[36] Kagan Tumer and Adrian Agogino. 2007. Distributed Agent-based Air Traffic Flow Management. In *Proceedings of the 6th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'07)*. Association for Computing Machinery, 8.

[37] Elise Van der Pol and Frans A. Oliehoek. 2016. Coordinated Deep Reinforcement Learners for Traffic Light Control. In *NIPS'16 Workshop on Learning, Inference and Control of Multi-Agent Systems*.

[38] Yihan Wang, Beining Han, Tonghan Wang, Heng Dong, and Chongjie Zhang. 2020. Off-Policy Multi-Agent Decomposed Policy Gradients. *arXiv* abs/2007.12322 (2020).

[39] Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8, 3 (1992).

[40] David H. Wolpert and Kagan Tumer. 1999. *An Introduction to COllective INtelligence*. Technical Report. NASA-ARC-IC-99-63, Nasa Ames Research Center.

[41] David H. Wolpert and Kagan Tumer. 2001. Optimal Payoff Functions for Members of Collectives. *Advances in Complex Systems* 4 (2001), 265–280.

[42] Dayon Ye, Minji Zhang, and Yu Yang. 2015. A Multi-Agent Framework for Packet Routing in Wireless Sensor Networks. *Sensors* 15, 5 (2015), 10026–10047.

[43] Logan Yliniemi and Kagan Tumer. 2014. Multi-objective Multiagent Credit Assignment Through Difference Rewards in Reinforcement Learning. In *Asia-Pacific Conference on Simulated Evolution and Learning*. Springer International Publishing, 407–418.

[44] Yan Zhang and Michael M. Zavlanos. 2019. Distributed off-Policy Actor-Critic Reinforcement Learning with Policy Consensus. *arXiv* abs/1903.09255 (2019).