Deep Decentralized Reinforcement Learning for Cooperative Control

Florian Köpf, [⋆] **Samuel Tesfazgi,** [⋆]
Michael Flad and Sören Hohmann

*Institute of Control Systems, Karlsruhe Institute of Technology (KIT),
76131 Karlsruhe, Germany (e-mail: florian.koepf@kit.edu)*

Abstract: In order to collaborate efficiently with unknown partners in cooperative control settings, adaptation of the partners based on online experience is required. The rather general and widely applicable control setting, where each cooperation partner might strive for individual goals while the control laws and objectives of the partners are unknown, entails various challenges such as the non-stationarity of the environment, the multi-agent credit assignment problem, the alter-exploration problem and the coordination problem. We propose new, modular deep decentralized Multi-Agent Reinforcement Learning mechanisms to account for these challenges. Therefore, our method uses a time-dependent prioritization of samples, incorporates a model of the system dynamics and utilizes variable, accountability-driven learning rates and simulated, artificial experiences in order to guide the learning process. The effectiveness of our method is demonstrated by means of a simulated, nonlinear cooperative control task.

Keywords: Reinforcement Learning, Deep Learning, Learning Control, Shared Control, Decentralized Control, Machine Learning, Non-stationary Systems, Nonlinear Control.

1. INTRODUCTION

In numerous control problems including highly-automated driving, robotics and manufacturing plants, several entities (e.g. machines and/or humans) are required to collaborate in order to achieve complex control objectives. Although the cooperating partners' goals usually do not completely contradict each other, the partners might have individual preferences. Suitable partners need to be flexible enough to account for the preferences of each other while representing their interests. We refer to this kind of setting as *Cooperative Control* (Köpf et al., 2018) [1] in order to emphasize that partners need to cooperate with each other and make compromises when conflicts occur. However, this does not necessarily imply that they are facing a so-called *fully cooperative* setting with a single global goal. Instead individual goals for the agents are allowed. Bearing the vision of future human-machine collaboration and plug-and-play machine-machine cooperation in mind, we focus on the case where the partners do not know the others' control laws or objective functions and no explicit communication is used. This *decentralized* setting requires the partners to constantly adapt to each other based on online experience.

Due to its generalization capabilities and major successes in the single-agent Reinforcement Learning (RL) setting, Multi-Agent Reinforcement Learning (MARL) has recently become the focus of increasing attention in order to solve Cooperative Control problems. Compared to single-agent RL, the multi-agent case is inherently more complex as agents directly or indirectly interact with each other and their common environment.

A major challenge occurring here is the *non-stationarity* of the dynamics from the local perspective of each agent which violates the Markov property that is commonly assumed in RL. Besides the severe challenge of non-stationarity, it is in general difficult to deduce to what extent an agent contributed to state transitions and thus the rewards received as each agent is capable of manipulating the environment. This is known as the *multi-agent credit assignment problem* (Chang et al. (2004)). Additionally, the exploration-exploitation trade-off common to RL even worsens in the cooperative case. This is due to other learning agents which might concurrently explore. Matignon et al. (2012) refer to this problem as *alter-exploration.* Furthermore, the *coordination problem* states that successful cooperation requires the agents to coordinate their controls in order to avoid e.g. shadowed equilibria (Matignon et al. (2012)). Finally, in order to cope with the majority of control problems, we require compatibility with continuous state and control spaces and nonlinear systems and do not assume restrictions concerning the structure of the agents' objectives. However, as a system model is usually available in control engineering as a result of model design or an identification process, we desire to incorporate this beneficial knowledge into our method. Due to causality, we assume that the joint control signals of other agents are not instantaneously measurable at run time but are retrospectively measurable or deducible.

1.1 Related Work

In the following, a short overview regarding related work concerning cooperative control will be given and analyzed

[⋆] These authors contributed equally to this work.

[1] Alternatively termed *Mixed Cooperative-Competitive Control* (Lowe et al., 2017).

w.r.t. our problem. One possible approach as proposed by Köpf et al. (2019) is to identify and constantly update the aggregated control law of all other agents. Relying on a model of the system dynamics, this allows a simulation-based optimization of the cooperative control problem. The concept of opponent or partner modeling is also discussed by Lowe et al. (2017) (Section 4.2 therein). When facing a dynamic game setting, another approach to cope with unknown partners in cooperative scenarios is given by the identification of associated cost functionals as done by Köpf et al. (2017) and Inga et al. (2018) and a subsequent optimization. In the human-machine context, this setting is motivated by the assumption that human motion can be modeled by means of optimal control (Scott, 2004).

In contrast to these methods, the following approaches avoid the need to identify the partners' cost functionals or control laws. Among these methods, Adaptive Dynamic Programming in the Cooperative Control setting (Vamvoudakis and Lewis (2011); Köpf et al. (2018)) focuses on efficient adaptation from a control-oriented perspective but has more restricting assumptions regarding reward structures and system dynamics compared to deep RL methods. Thus, the following methods either rely on extensions to *Deterministic Policy Gradient* (DPG) methods (Silver et al. (2014)) or extensions to *Deep Q-Networks* (DQN) (Mnih et al. (2015)).

Among the DPG methods, either all agents need to know the policy parameters of all others (Gupta et al., 2017), explicit opponent modeling is required when facing our problem (cf. (Lowe et al., 2017, Section 4.2)), or all agents share the same critic and a global reward function (Foerster et al., 2018), i.e. the agents are not decentralized and fully cooperative. Furthermore, the DPG based methods suffer from increasing variance in multi-agent domains (cf. Lowe et al. (2017) and Foerster et al. (2018)), which destabilizes the training process particularly with independently learning agents. Concerning the DQN-based methods, they either work in the fully cooperative setting with finite state and control spaces (Foerster et al. (2017); Matignon et al. (2007)), are limited to finite control spaces (Omidshafiei et al., 2017) or finite state and control spaces (Palmer et al., 2018).

1.2 Contributions of This Paper

As none of the deep MARL methods in literature fulfills our control-oriented requirements, we propose a new approach for cooperative control in continuous state and control spaces. Our method does not depend on the explicit identification of the other agents' behavior. Instead, an adapting automation is explored, which is not reliant on the premise of other agents behaving optimally and is expected to facilitate a high degree of generalizability across domains and partners. Compared to recent deep RL methods in the multi-agent domain, we face the challenge of decentralized agents with no knowledge of the partners' control strategies or objectives and no explicit means of communication and present three new, modular mechanisms which explicitly address the associated challenges.

Although the deep MARL methods in Section 1.1 cannot applied be directly to our problem setting, they reveal reoccurring mechanisms which we rely on: First, extensions to the *experience replay memory* (ERM) in order to counteract the difficulty of applying experience replay in non-stationary environments. Second, *variable learning rates* in order to induce coordination and facilitate the use of a temporal dimension in the sampling process. We propose *Temporal Experience Replay* (TER) to account for the non-stationarity of the environment each agent faces. The main idea behind TER is a time-dependent prioritization of samples in the experience replay memory. Furthermore, we introduce the idea of *Imagined Experience Replay* (IER), which benefits from a model of the system dynamics and grounds the training process by means of fictional experiences. IER can be understood as an adaptation of the idea of imagination rollouts (cf. Gu et al. (2016)) to cope with the challenges encountered in multi-agent settings. In addition, in order to address the multi-agent credit assignment problem, we propose a new mechanism of variable learning rates. Our accountability-driven approach termed *impact Q-learning* (IQL) ties the learning rate to the agent's contribution towards the joint control. We further combine IQL and IER to simulate targeted cooperation scenarios in order to exhaust potential coordination between agents. Finally, the mechanisms are made dependent on an exploration rate such that the influence of each distinct concept is varied according to its current utility. This increases their effectiveness and reduces issues connected to alter-exploration.

2. FORMAL PROBLEM DEFINITION AND PREREQUISITES

We now formalize our problem definition and introduce prerequisites on which our proposed mechanisms rely on.

2.1 Formal Problem Definition

Consider a discrete-time system $f : X \times U \to X$ that is controlled by N agents given by

$$x_{k+1} = f(x_k, u_{1,k}, \ldots, u_{N,k}), \tag{1}$$

where $x_k \in X \subseteq \mathbb{R}^n$ denotes the state at time step k, $u_{i,k} \in U_i \subseteq \mathbb{R}$ the control of agent $i \in \mathcal{N} = \{1, \ldots, N\}$ and $U = U_1 \times \cdots \times U_N$ the joint control space. Depending on the current state x_k and controls $u_{i,k}$, each agent $i \in \mathcal{N}$ experiences a reward r_i that results from a reward function $g_i : X \times U \to \mathbb{R}$, i.e.

$$r_{i,k} = g_i(x_k, u_{1,k}, \ldots, u_{N,k}). \tag{2}$$

The goal of each agent is to adapt his control law $\pi_i : X \to U_i$ in order to maximize his value

$$V_i^{\boldsymbol{\pi}}(x_k) = \sum_{k=0}^{\infty} \gamma_i^k r_{i,k} = \sum_{k=0}^{\infty} \gamma_i^k g_i(x_k, \pi_1(x_k), \ldots, \pi_N(x_k)), \tag{3}$$

i.e. the long-term discounted reward under the tuple of control laws $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_N)$, where $\gamma_i \in [0, 1)$ denotes a discount factor. Thus, our deterministic game setting (in contrast to the stochastic game definition in Buşoniu et al. (2010)) is defined by the tuple $G = (X, U_1, \ldots, U_N, f, g_1, \ldots, g_N, \gamma_1, \ldots, \gamma_N)$. Our problem is then formalized as follows.

Problem 1. Given the game G, each agent $i \in \mathcal{N}$ knows the system dynamics f and his own reward function g_i. Furthermore, each agent $i \in \mathcal{N}$ receives his current reward

$r_{i,k}$ at time step k and is able to deduce the *previous* controls $u_{j,k-1}$, $\forall j \in \mathcal{N} \setminus \{i\}$ of other agents but has no access to the current controls $u_{j,k}$, other agents' control laws π_j, their reward functions g_j or actual rewards. In this setting, each agent $i \in \mathcal{N}$ aims at adapting his control law π_i in order to maximize $V_i^{\boldsymbol{\pi}}$ as defined in (3).

2.2 Prerequisites Concerning Deep Q-Networks

Our algorithm is based on DQN. Thus, the fundamental concepts of Q-learning and DQN are introduced in the following. Q-learning (Watkins (1989)) is an iterative algorithm which intends to learn an optimal state-action-value function Q^*. Here,

$$Q^*(x_k, u_k) = \max_{\pi} Q^{\pi}(x_k, u_k) \tag{4}$$

holds, where $Q^{\pi}(x_k, u_k)$ represents the discounted long-term cost, if an agent is in state x_k and applies the control, i.e. action, u_k at time step k and follows the control law π thereafter. The relevance of Q^* becomes clear as the optimal control law maximizing the long-term discounted reward (cf. (3) for $N = 1$) is given by

$$\pi^*(x_k) = \arg\max_{u_k} Q^*(x_k, u_k). \tag{5}$$

The update rule in order to estimate Q^* is given by

$$\begin{aligned} Q(x_k, u_k) \leftarrow\, & Q(x_k, u_k) \\ & + \alpha_k \underbrace{\Big[r_k + \gamma \max_{u} Q(x_{k+1}, u) - Q(x_k, u_k)\Big]}_{\delta_k}, \end{aligned} \tag{6}$$

where δ_k denotes the temporal difference (TD) error and $\alpha_k \in (0, 1]$ a learning rate. The TD error δ_k thus measures the difference between the current Q-function estimate $Q(x_k, u_k)$ and the TD target $r_k + \gamma \max_u Q(x_{k+1}, u)$. The tuple $\chi_k = (x_k, u_k, r_k, x_{k+1})$ is taken from interaction with the environment.

In order to extend Q-learning to continuous state spaces, function approximators such as deep neural networks which parametrize the Q-function have been introduced. Here, the work of Mnih et al. (2015) marked a breakthrough, as the introduction of *Experience Replay* (ER) significantly improved training. The idea is to randomize training samples in order to remove correlation between observed state-transition sequences. Therefore, experience tuples χ_k are stored in an ER memory (ERM) $\mathcal{M}$ at each time step k. A Q-learning update is then performed by sampling (e.g. uniformly at random) from the ERM and minimizing the associated squared TD error δ_k. In order to account for continuous control spaces, Gu et al. (2016) introduced the concept of *Normalized Advantage Functions* (NAF), allowing to deduce an analytical expression in order to solve (5).

3. DECENTRALIZED COOPERATIVE CONTROL METHOD

In order to gain control of the challenges associated with Problem 1, we propose a time-dependent mechanism termed *Temporal Experience Replay* (TER) to account for the non-stationary environment, include known system dynamics by means of *Imagined Experience Replay* (IER) and use variable learning rates with the proposed *Impact Q-Learning* (IQL) in order to induce coordination. As these mechanisms can be applied in a modular fashion, they are separately introduced and then combined in Section 3.4.

3.1 Temporal Experience Replay (TER)

The proposed method of *Temporal Experience Replay* attempts to unify the idea of favoring more recent experiences with the concept of more probable sampling of experiences according to a prioritization factor. Analogue to *Prioritized Experience Replay* (Schaul et al., 2016), we suggest to bias the sampling process. However, instead of utilizing the TD error for the prioritization, we propose to focus towards recent experiences by introducing a *temporal prioritization* τ, which is proportional to the time that has passed since collection k_c of the state-transition:

$$\tau_{k_c}(k) = \exp\big(-|k - k_c|\,\big) + \xi_{\text{temp}}, \tag{7}$$

with the sampling probability $P_{k_c}(k)$ given by

$$P_{k_c}(k) = \frac{\tau_{k_c}(k)}{\sum_l \tau_l(k)}. \tag{8}$$

In (7) the optional offset ξ_{temp} can be used to ensure that experiences are sampled with non-zero probability and the term k denotes the current time step. Hence, to compute (7) and (8) at runtime, the experience tuple has to be extended by the respective current time step, producing the new tuple:

$$\chi_k^* = (x_k, u_k, r_k, x_{k+1}, k). \tag{9}$$

The underlying idea is that agents are more capable of adjusting to ever changing policies of other agents by experiencing recent state-transitions tuples more often than old ones. However, the TER as described by (7) and (8) is impractical, as it leads to two major issues: Firstly, similar to approaches that restrict the memory size itself, the proposed temporal prioritization suffers from biasing the ERM too much towards recent experiences. This can lead to over-fitting of an agent's policy. Secondly, the temporal prioritization increases the computational complexity of the sampling process to a degree that is not feasible in practice. This is due to the computation of the temporal prioritization $\tau_{k_c}(k)$ itself, as it has to be updated for each experience tuple at every time step.

To overcome both of these issues, a two step sampling process is proposed. Initially, a *macro-batch* $\mathcal{B}$ of size B is sampled uniformly at random from the complete experience replay buffer $\mathcal{M}$. Subsequently, a smaller *mini-batch* $\mathcal{T}$ of size t, with $t < B$, is sampled from $\mathcal{B}$ utilizing the temporally prioritized probabilities given in (8). By dividing the sampling process into two manageable parts, both of the above mentioned problems are solved. The macro-batch $\mathcal{B}$ is only of size B, thus, the computational complexity of calculating the temporal priorities $\tau_{k_c}(k)$ is equally reduced to B. Additionally, the initial macro-batch is sampled uniformly at random, which reduces the risk of overemphasizing experiences related to recent episodes.

In order to account for the varying exploration rate ε_k of agents at different stages of the training process, we propose an additional exploration rate dependency of B yielding a time-dependent macro-batch size B_k. TER attempts to induce adaptation to other agents' policy changes.

Therefore, it is most effective when the partners' policies start to converge and are less influenced by exploration noise. Thus, experiences should be sampled uniformly at random during early training (i.e. when $\varepsilon_k \approx 1$), which can be achieved by choosing B_k close to the mini-batch size t, whereas during later training stages, i.e. once $\varepsilon_k \to 0$, B_k should approach the final macro-batch size B. Consequently, we choose

$$B_k = (B - t)(1 - \varepsilon_k) + t. \tag{10}$$

3.2 Imagined Experience Replay (IER)

The above mentioned augmentations to ER attempt to either stabilize the training process in order to make agents less susceptible to changing environment dynamics or bias learning towards recent experiences to enable agents to adapt to changes in the dynamics. In any case it is acknowledged that the other agents' behavior is indissociable from the dynamics of the environment, which is generally a reasonable presumption given independent and decentralized agents. However, due to the assumption that a system model is available, it becomes possible to ground the training process through simulated experiences in which the partners' controls are marginalized leading to stationary environment dynamics. This is the fundamental idea of our second proposed modification to the ER, which is termed *Imagined Experience Replay* (IER).

The concept of IER was inspired by the *imagination roll-outs* developed by Gu et al. (2016), who proposed the idea of accelerating the training process by utilizing a learned model to simulate artificial experiences that were then added to the replay buffer. Differently, IER is used here to simulate experiences, which would not occur under normal circumstances. Specifically, all other agents' controls $\boldsymbol{u}_{-i} = \{u_1, \dots, u_{i-1}, u_{i+1}, \dots, u_N\}$ are set to zero while retaining the agent's control u_i unchanged. Given a regular experience

$$\chi_{i,k} = (x_k, u_{i,k}, r_{i,k}, x_{k+1}) \tag{11}$$

for agent i which occurred at time k, the successor state x_{k+1} and received reward $r_{i,k}$ can be substituted by utilizing the underlying system dynamics f and reward function g_i:[2]

$$\chi_{i,k} = \big(x_k, u_{i,k},\ g_i(x_k, u_{i,k}, \boldsymbol{u}_{-i,k}),\ p(x_k, u_{i,k}, \boldsymbol{u}_{-i,k})\big). \tag{12}$$

Subsequently, an *imagined experience* $\tilde{\chi}$ can be simulated by replacing the other agents' controls $\boldsymbol{u}_{-i}$ by $\mathbf{0}$ yielding the imagined successor state $\tilde{x}_{k+1}$ and reward $\tilde{r}_{i,k}$:

$$\tilde{\chi}_{i,k} = \big(x_k, u_{i,k},\ \underbrace{g_i(x_k, u_{i,k}, \mathbf{0})}_{\tilde{r}_{i,k}},\ \underbrace{p(x_k, u_{i,k}, \mathbf{0})}_{\tilde{x}_{k+1}}\big). \tag{13}$$

In contrast to the imagination roll-out of Gu et al. (2016), the imagined experiences in (13) are not stored to the actual ERM and sampled from there. Instead, an exploration rate dependent probability $\tilde{P}$ is utilized to determine, whether an imagined experience is computed in addition to the sampled, observed experience. Once used for training, the imagined experience is discarded in order to reduce the risk of overemphasizing artificial experiences in which partners are non-existent. During the initial training phase, agents predominantly explore random controls. Hence, it is not possible to infer other agents' policies from observations, and the application of IER, in order to stabilize the training process, is most useful, as experiences are simulated in which solely the agent i interacts with the environment. These imagined experiences, at this stage of training, are essentially observations for which the exploration noise of other agents is not present. Consequently, the probability $\tilde{P}$ of simulating imagined experiences $\tilde{\chi}_{i,k}$ at time step k is proposed to be proportional to the current exploration rate ε_k, i.e. $\tilde{P}(k) \sim \varepsilon_k$.

However, during the later stages of training, when policies start to converge and are less influenced by exploratory controls, the coordination between agents and the adaptation to the partners' policies becomes more important. Upon closer examination it can be seen that by generating artificial experiences in which cooperation between agents is simulated, the IER can potentially be utilized to induce coordination between agents. Opposite to (13), these *imagined coordination experiences* are more useful during later stages of training. Therefore, the respective sampling probability is proposed as $P_{\text{coord}}(k) \sim (1 - \varepsilon_k)$. When generating experiences with the purpose of inducing coordination, it has to be considered that the final algorithm is required to entail mixed cooperative-competitive task types. Thus, it cannot generally be presumed that the agents' respective goals are compatible. Consequently, it is proposed that the IER is utilized to simulate three additional scenarios:

(1) In order to induce coordination, the control u_i of agent i is discarded, causing it to be idle:

$$\chi^{(i)}_{\text{idle}} = \big(x, 0,\ g_i(x, 0, \boldsymbol{u}_{-i}),\ p(x, 0, \boldsymbol{u}_{-i})\big). \tag{14}$$

Therefore, by utilizing $\chi^{(i)}_{\text{idle}}$, agent i can observe how other agents behave by themselves and whether the resulting environment transitions are beneficial.

(2) Here, the first of two cooperation scenarios $\chi^{(i)}_{\text{coop1}}$ is simulated. For this purpose, the agent's controls u_i are set to be equal to the average of all other agents' joint control $\overline{u}_{-i}$:

$$\overline{u}_{-i} = \frac{1}{N-1} \sum_{\forall j \in \mathcal{N} \setminus \{i\}} u_j,$$

resulting in:

$$\chi^{(i)}_{\text{coop1}} = \big(x, \overline{u}_{-i},\ g_i(x, \overline{u}_{-i}, \boldsymbol{u}_{-i}),\ p(x, \overline{u}_{-i}, \boldsymbol{u}_{-i})\big). \tag{15}$$

(3) In the second cooperation scenario $\chi^{(i)}_{\text{coop2}}$, the inverse is generated. Each element of the other agents' joint control $\boldsymbol{u}_{-i}$ is modified to be equal to u_i:

$$\chi^{(i)}_{\text{coop2}} = \big(x, u_i, g_i(x, u_i, \boldsymbol{u}_{-i}), p(x, u_i, \boldsymbol{u}_{-i})\big), \tag{16}$$
$$\text{with } u_j \coloneqq u_i, \forall j \in \{1, \dots, N\}.$$

By imagining these three scenarios, potential coordination possibilities are exhausted. The first one specifically enables agents to evaluate whether being idle leads to an acceptable reward, which in competitive environments is generally discouraged and will be evaluated correspondingly, whereas the second and third evaluate the effect if

[2] For convenience of notation, $g_i(x_k, u_{i,k}, \boldsymbol{u}_{-i,k})$ and $p(x_k, u_{i,k}, \boldsymbol{u}_{-i,k})$ evaluate $g_i(\cdot)$ and $p(\cdot)$ at the state x_k while agent i applies the control $u_{i,k}$ and all other agents controls are denoted by the tuple $\boldsymbol{u}_{-i,k}$.

agent i imitates the average control of the others or if all agents stick to the control of agent i.

3.3 Impact Q-Learning (IQL)

In the previous subsections, different additions to the ERM were proposed, which focused mainly on providing stability to counteract the problem of a non-stationary environment. An additional, often utilized mechanism in MARL are variable learning rates. The hysteretic (Matignon et al. (2007)) and lenient (Palmer et al. (2018)) method are two representatives of optimistic learners, which are generally well suited to induce coordination. However, the core principle of optimistic agents, which reduce their learning rate given negative experiences, is diametrically opposed to the challenge of multi-agent credit assignment. In order to credit agents correctly with respect to the observed outcome, it is necessary for them to not only learn notably from positive experiences but also from negative ones. Furthermore, the concept of tying the learning rate to rewards (Matignon et al. (2007)) is itself flawed to combat credit assignment, as the attention an agent should pay to certain experiences ideally does not depend on the quality of the outcome, but on the contribution of an agent towards the observed outcome.

Therefore, we propose a novel approach for solving the multi-agent credit assignment problem using variable learning rates. We attempt to tie the variable learning rate to the actual contribution of an agent towards the observed state transitions. This is facilitated by the retrospective observation of all agents' controls, as it enables each agent to compare its control $u_{i,k}$ at time step k to the separately remaining joint control $\boldsymbol{u}_{-i,k}$ of all other agents. To this end we introduce a novel quantity, called *impact factor*

$$\lambda_{i,k} = \frac{|u_{i,k}|}{\sum\limits_{j=1}^{N} |u_{j,k}|}, \tag{17}$$

which describes the agent's relative contribution to the joint control, and thus, to experienced state transitions. In order to enable the computation of a meaningful impact factor λ in (17) it is presupposed for IQL that agents share the same control space $U_1 = \ldots = U_N$ and that all agents' controls manipulate the system equally. Subsequently, the update rule (6) for an agent i can be modified to apply different learning rates depending on the agent's impact factor:

$$\delta_k \leftarrow r_k + \gamma \max_u Q_i(x_{k+1}, u) - Q_i(x_k, u_k),$$
$$Q_i(x_k, u_k) \leftarrow \begin{cases} Q_i(x_k, u_k) + \alpha\delta_k & \text{if } 1.0 \geq \lambda_{i,k} > \lambda_{\text{high}} \\ Q_i(x_k, u_k) + \sigma\delta_k & \text{if } \lambda_{\text{high}} \geq \lambda_{i,k} \geq \lambda_{\text{low}} \\ Q_i(x_k, u_k) + \beta\delta_k & \text{if } \lambda_{\text{low}} > \lambda_{i,k} \geq 0, \end{cases} \tag{18}$$

with $0 < \beta < \sigma < \alpha < 1$. In (18), the Q-learning update rule is partitioned into three distinct impact ranges with which it is possible to differentiate whether an agent had a high, medium, or low influence towards a state transition. Hence, the amount an agent learns from an experience is proportional to its respective contribution or impact. When a positive experience is observed, the Q-value estimate is only increased heavily, if the agent can be credited for the event. On the other hand, when a punishment occurs, the agent is mainly discouraged from the corresponding state-action pair, if the agent is at least in part responsible. Particularly this kind of accountability-driven learning behavior is required for agents to overcome the credit assignment challenge.

3.4 Algorithm

Upon closer examination of IER, it can be seen that the concept of simulated experiences specifically for the coordination scenarios in (14), (15), and (16) may reduce the coordination problem's severity, but also produces additional computational effort. Thus, it is advisable to limit these calculations to state-action pairs with high potential for coordination. This can be done by utilizing the computed impact factors. In (18), three intervals with different degrees of an agent's impact were distinguished. When analyzing the ones corresponding to learning rates α and β, it can be seen that the potential for coordination is limited here, because the agent either predominantly contributes towards the state-transition or only has a minor impact. However, for the case of medium learning rates σ, the impact of agents, particularly in the case of only few agents, is distributed more evenly, which in turn increases the need for coordination. In this instance, the simulation of different IER scenarios is most powerful and the trade-off between computational effort and induced coordination most beneficial. Further, two distinct kinds of medium-impact experiences are distinguished. Either the agent's control $u_{i,k}$ and the average of all remaining controls $\overline{u}_{-i,k}$ work in the same direction, or against each other. This can be determined by sampling an experience χ_k and computing a *coordination coefficient* ψ_k as such:

$$\psi_{i,k} = \text{sgn}(\overline{u}_{-i,k} \cdot u_{i,k}). \tag{19}$$

If $\psi_{i,k}$ equals 1, agent i and the others work in the same direction and it is not necessary to simulate the coordination experiences χ_{idle}, χ_{coop1} and χ_{coop2}. Instead, the learning rate σ, which is normally used for $\lambda_{\text{high}} \geq \lambda_{i,k} \geq \lambda_{\text{low}}$ in (18), is substituted by the larger learning rate α. Therefore, agents are induced to emphasize cooperative experiences during the learning process. In the case that agents act in opposing directions, $\psi_{i,k}$ equals -1. Besides the sampled experience χ_k, the artificial experiences χ_{idle}, χ_{coop1} and χ_{coop2} are simulated, and subsequently, the agent is trained on all of them. Here the lowest learning rate β is applied, because the trained on experiences have not actually occurred and are only imagined for coordination purposes. Thus, the instances for which the computational strenuous task of simulating multiple coordination experiences is required, can be reduced greatly and focused to occasions connected to the highest expected learning progress. The resulting algorithm after finally assembling the above described mechanisms is described in Algorithm 1, where $U(a, b)$ denotes a uniform distribution in the intervall $[a, b]$.

4. RESULTS

In this section, the previously described algorithm is trained on a control task. Subsequently, the method's effectiveness is evaluated.

Algorithm 1 Deep impact Q-learning with TER and IER

```
Input: macro-batch size $B$, mini-batch size $t$,
       learning rates $\alpha$, $\sigma$, and $\beta$, ERM size $M$,
       target update frequency $m$, decay rate $\varpi$,
       minimum exploration rate $\varepsilon_{\min}$, maximum
       number of episodes $E_{\max}$ and maximum
       time steps per episode $K_{\max}$
Initialize: $Q(x, u; \theta)$ and $Q(x, u; \hat{\theta})$ with random
            weights $\theta$ and $\hat{\theta}$, ER buffer $\mathcal{M} \leftarrow \varnothing$ with
            size $M$, exploration rate $\varepsilon = 1$
for episode $e = 1, \dots, E_{\max}$ do
    $k = 0$
    while episode not terminated and $k \leq K_{\max}$ do
        With probability $\varepsilon$ select random control $u_k$
        Otherwise select $u_k = \arg\max_u Q(x_k, u; \theta)$
        Execute $u_k$ and observe $r_k$, $x_{k+1}$
        Store tuple $(x_k, u_k, r_k, x_{k+1}, k)$ in $\mathcal{M}$
        Compute $B_k = (B - t)(1 - \varepsilon_k) + t$          (10)
        Sample uniformly at random $\mathcal{B}$ of size $B_k$
        Compute $\tau_{k_c}(k)$ for transition in $\mathcal{B}$          (7)
        Sample $\mathcal{T}$ of size $t \sim P_{k_c}(k) = \tau_{k_c}(k) / \sum_l \tau_l$ (8)
        for each $\chi \in \mathcal{T}$ do
            Extract time of collection $c$ from $\chi$
            Draw random variable $w \sim U(0, 1)$
            if $w < \varepsilon_c$ (exploration rate at time $c$) then
                Compute $\tilde{\chi}_c = \big(x_c, u_c,\ \tilde{r}_c,\ \tilde{x}_{c+1}\big)$
                Set $\tilde{y}_c = \tilde{r}_c + \gamma \max_u Q(\tilde{x}_{c+1}, u; \hat{\theta}\,)$
                Update $\theta$ with learning rate $\beta$ for $\tilde{y}_c$
            end if
            Compute $\lambda_c = |u_{i,c}| / \sum_j |u_{j,c}|$          (17)
            Set $y_c = r_c + \gamma \max_u Q(x_{c+1}, u; \hat{\theta}\,)$
            if $\lambda_c > \lambda_{\text{high}}$ then
                Update $\theta$ with learning rate $\alpha$ for $y_c$
            else if $\lambda_{\text{high}} \geq \lambda_c \leq \lambda_{\text{low}}$ then
                if $\operatorname{sgn}(\overline{u}_{-i,c} \cdot u_{i,c}) \geq 0$ then
                    Update $\theta$ with learning rate $\alpha$ for $y_c$
                else
                    Update $\theta$ with learning rate $\sigma$ for $y_c$
                end if
                if $\operatorname{sgn}(\overline{u}_{-i,c} \cdot u_{i,c}) < 0$ and $\varepsilon_c < w$ then
                    Compute $\chi_{\text{idle},c}$, $\chi_{\text{coop1},c}$, $\chi_{\text{coop2},c}$
                    Set target $y_{\text{idle},c}$, $y_{\text{coop1},c}$, and $y_{\text{coop2},c}$
                    Update $\theta$ with learning rate $\beta$ for
                    $y_{\text{idle},c}$, $y_{\text{coop1},c}$, and $y_{\text{coop2},c}$
                end if
            else
                Update $\theta$ with learning rate $\beta$ for $y_c$
            end if
        end for
        Every $m$ steps, update target network: $\hat{\theta} \leftarrow \theta$
        $k = k + 1$
    end while
    Decay exploration rate: $\varepsilon \leftarrow \max[\varpi \cdot \varepsilon;\ \varepsilon_{\min}]$
end for
```

4.1 Example System and Network Architecture

For the simulated environment, we use a customized two-player-variation of the *OpenAI gym* (Brockman et al. (2016)) cart-pole problem. Here, two agents balance a pole, which is hinged to a movable cart, by concurrently applying forces to the cart's base. The system dynamics are defined by the nonlinear differential equations

$$\ddot{\theta}_k = \frac{g\sin(\theta_k) - \cos(\theta_k)\left[\dfrac{-F_{k,\text{res}} - m_{\text{pole}}\, l\, \dot{\theta}_k^2\, \sin(\theta_k)}{m_{\text{pole}} + m_{\text{cart}}}\right]}{l\left[\dfrac{4}{3} - \dfrac{m_{\text{pole}}\cos^2(\theta_k)}{m_{\text{pole}} + m_{\text{cart}}}\right]}, \tag{20}$$

$$\ddot{s}_k = \frac{F_{k,\text{res}} + m_{\text{pole}}\, l\left[\dot{\theta}_k^2 \sin(\theta_k) - \ddot{\theta}_k\cos(\theta_k)\right]}{m_{\text{pole}} + m_{\text{cart}}}, \tag{21}$$

where $g = -9.8\,\text{m/s}^2$, $m_{\text{pole}} = 0.1\,\text{kg}$, $m_{\text{cart}} = 1.0\,\text{kg}$, $l = 0.5\,\text{m}$ (half-pole length) and $F_{k,\text{res}} \in [-10\,\text{N}, 10\,\text{N}]$ (clipped sum of forces). In (20) and (21), θ_k denotes the angular displacement of the pole from 0 rad, which is defined by the pole standing perfectly upright. The cart's position is defined by s_k with the center being at 0 m and the system state is given by $x_k = \begin{bmatrix} s_k & \dot{s}_k & \theta_k & \dot{\theta}_k \end{bmatrix}^\intercal$. The successor state x_{k+1} according to (1) is calculated using the semi-implicit Euler method with a discrete time step of 0.02 s.

We assume that one agent focuses on balancing the pole upright, while the other agent is rewarded depending on the position of the cart. Thus, the first agent receives a reward $r_{1,k}$ of 1 for each time step k in which the pole angle $\theta_k \in (-0.21\,\text{rad}, 0.21\,\text{rad})$. If the episode is terminated, a reward of -1 is observed. On the contrary, the second agent's reward $r_{2,k}$ solely depends on the current cart position s_k. Specifically, a step-wise reward function is defined as such:

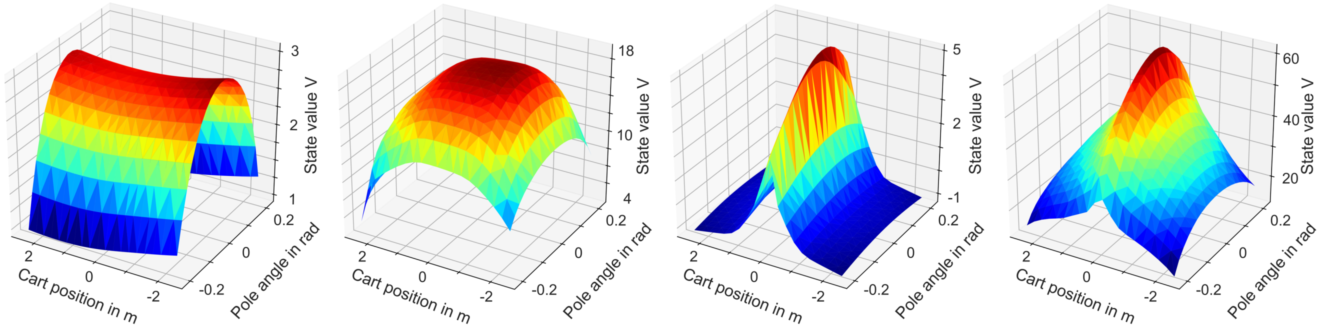
$$r_{2,k} = \begin{cases} +5, & \text{if } |s_k - s^*| < 0.1\,\text{m} \\ +1, & \text{if } 0.1\,\text{m} \leq |s_k - s^*| < 0.5\,\text{m} \\ 0, & \text{if } 0.5\,\text{m} \leq |s_k - s^*| < 2.4\,\text{m} \\ -1, & \text{if episode is terminated,} \end{cases} \tag{22}$$

with the target position denoted by s^*. We choose $s^* = 0\,\text{m}$. Agent 2 receives the highest reward in a small range around the target position, while the received reward is reduced step-wise once a certain boundary distance is exceeded. At the beginning of each of the $E_{\max} = 2000$ training episodes, the cart is initiated uniformly at random with the initial position $s_0 \sim U(-2.3\,\text{m}, 2.3\,\text{m})$ and the initial pole angle $\theta_0 \sim U(-0.085\,\text{rad}, 0.085\,\text{rad})$. An episode is terminated once one of the intervals $s_k \in [-2.4, 2.4]$ m or $\theta_k \in [-0.21, 0.21]$ rad is exceeded or $K_{\max} = 3000$ time steps have passed. We set $\gamma = 0.999$.

In our work, a dueling network architecture with NAFs as introduced by Gu et al. (2016) was used. Additionally, multiple fully connected layers and dropout layers are stacked in front of the dueling network architecture to process observations. A description of the parameters corresponding to the network architecture and the hyperparameters used for training is given in the Appendix A.

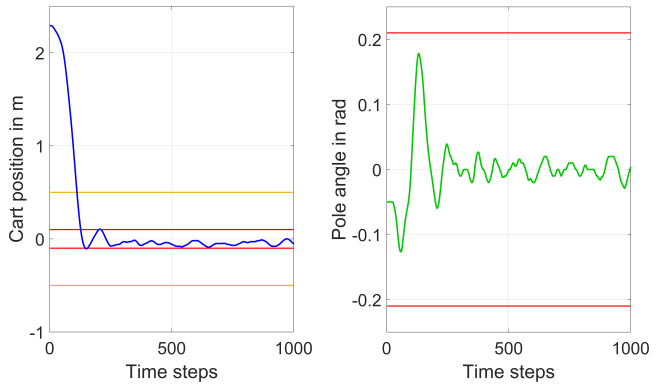
4.2 Simulations

Fig. 1 shows the resulting state-value estimates $V(x)$ of both agents at different training episodes, where each data point is averaged over $\dot{s}_k$ and $\dot{\theta}_k$ for reasons of presentability. After 1000 episodes of training, agent 1 expects the highest return along the pole angle of 0 rad. The lowest state-values are estimated for θ close to the



(a) agent 1, 1000 episodes (b) agent 1, 2000 episodes (c) agent 2, 1000 episodes (d) agent 2, 2000 episodes

Fig. 1. State-value estimates of the agents at different training stages. Each data point is averaged over $\dot{s}_k$ and $\dot{\theta}_k$.



(a) (b)

Fig. 2. Cart-pole position and pole angle for an example initialization using the learned controllers.

terminal pole angles $\theta = \pm 0.21\,\mathrm{rad}$. Analogously, after 1000 episodes, agent 2 evaluates states close to the desired target position $s^* = 0\,\mathrm{m}$ as most beneficial. This maximum state-value drops abruptly with slight deviation from s^*. Fig. 1b and 1d show the agents' estimated state-values after 2000 episodes of training. For both agents, the expected returns have generally increased compared to the previous training stage. Because they successfully learned how to jointly balance the pole without moving the cart outside the boundaries, the available time steps to accumulate rewards is increased. Additionally, both agents learned to appropriately reduce $V(x)$ close to all terminal states.

Example trajectories of the cart position s_k and pole angle θ_k resulting from the trained control law are depicted in Fig. 2. In Fig. 2a, the boundaries for the highest 5 point reward range and the lower 1 point reward range for the seconds agent's position control are depicted in red and orange and the red lines in Fig. 2b mark the terminal conditions. It can be seen that the agents are capable of moving the cart from the initial position to the desired target while holding the pendulum upright.

4.3 Discussion

After 1000 training episodes, the state-value estimations are predominantly dependent on the state dimension associated with the individual preferences as this yields the highest rewards while the cart-pole cannot be successfully controlled yet. However, in later training stages (2000 episodes), the agents develop understanding concerning coordination possibilities (guided by IER) and their relative contribution (thanks to IQL) allowing the agents to move the cart to a desired state without terminating the episode yielding much higher rewards. The decreased steepness of the state-value gradients when comparing $V(x)$ after 1000 and 2000 episodes is a result of the agent's increased control capabilities allowing to transition from a state with low rewards to a state associated with higher rewards.

It is noticeable that without the mechanisms proposed in Section 3, the agents were not able to learn to stabilize the cart-pole at all with the given parametrization. Thanks to IER and IQL and an appropriate focus on recent experiences due to TER, the agents were successful at adapting to each other. The agents also learned to perform complex trajectories, which included deflecting the pole close to the terminal positions and angles (cf. Fig. 2). Thus, it is possible for them to flexibly control the cart-pole even in difficult situations.

5. CONCLUSION

In this paper, new mechanisms have been proposed in order to account for challenges arising in deep Multi-Agent Reinforcement Learning problems with restricted information. Two novel extensions to experience replay were presented. First, TER allows the sampling process to properly reflect the fact that recent experiences carry more information regarding the current control laws of cooperation partners and are thus better suited to counteract the non-stationarity compared to outdated experiences. Second, artificial experiences denoted as IER complement the experience replay memory. In the early training stage, alter-exploration problems are reduced due to simulated transitions in which the agents interact separately with the environment. Later, coordination is induced to exhaust the cooperation potential between agents as adaptation becomes feasible. Finally, these experience replay enhancements are supplemented by a mechanism termed IQL. Here, the relative contribution of the agent towards the observed outcome is accounted for by means of an impact factor which adapts an agent's learning rate. Our algorithm was evaluated on a simulated cart-pole-problem, where two agents successfully learned to cooperate.

REFERENCES

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *Arxiv.org*, 1606.01540.

Buşoniu, L., Babuška, R., and de Schutter, B. (2010). Multi-agent reinforcement learning: An overview. In *Innovations in Multi-Agent Systems and Applications*, 183–221. Springer, Berlin, Heidelberg.

Chang, Y.h., Ho, T., and Kaelbling, L.P. (2004). All learning is local: Multi-agent learning in global reward games. In *Advances in neural information processing systems*, 807–814.

Foerster, J., Nardelli, N., Farquhar, G., Afouras, T., Torr, P.H.S., Kohli, P., and Whiteson, S. (2017). Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of the 34th ICML*, volume 70 of *PMLR*, 1146–1155.

Foerster, J.N., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. (2018). Counterfactual multi-agent policy gradients. In *32nd AAAI Conference on Artificial Intelligence.*

Gu, S., Lillicrap, T., Sutskever, I., and Levine, S. (2016). Continuous deep q-learning with model-based acceleration. In *ICML*, 2829–2838.

Gupta, J.K., Egorov, M., and Kochenderfer, M. (2017). Cooperative multi-agent control using deep reinforcement learning. In *Autonomous Agents and Multiagent Systems*, volume 10642 of *Lecture Notes in Computer Science*, 66–83. Springer Int. Publishing, Cham.

Inga, J., Eitel, M., Flad, M., and Hohmann, S. (2018). Evaluating human behavior in manual and shared control via inverse optimization. In *2018 IEEE International Conference on Systems, Man, and Cybernetics*, 2699–2704.

Köpf, F., Ebbert, S., Flad, M., and Hohmann, S. (2018). Adaptive dynamic programming for cooperative control with incomplete information. In *2018 IEEE International Conference on Systems, Man and Cybernetics.*

Köpf, F., Inga, J., Rothfuß, S., Flad, M., and Hohmann, S. (2017). Inverse reinforcement learning for identification in linear-quadratic dynamic games. *IFAC-PapersOnLine*, 50(1), 14902–14908.

Köpf, F., Nitsch, A., Flad, M., and Hohmann, S. (2019). Partner approximating learners (pal): Simulation-accelerated learning with explicit partner modeling in multi-agent domains. *arXiv e-prints*, arXiv:1909.03868.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O.P., and Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, 6379–6390.

Matignon, L., Laurent, G.J., and Le Fort-Piat, N. (2007). Hysteretic q-learning: An algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 64–69.

Matignon, L., Laurent, G.J., and Le Fort-Piat, N. (2012). Independent reinforcement learners in cooperative markov games: A survey regarding coordination problems. *The Knowledge Engineering Review*, 27(1), 1–31.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.

Omidshafiei, S., Pazis, J., Amato, C., How, J.P., and Vian, J. (2017). Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *Proceedings of the 34th ICML*, volume 70 of *PMLR*, 2681–2690.

Palmer, G., Tuyls, K., Bloembergen, D., and Savani, R. (2018). Lenient multi-agent deep reinforcement learning. In *Proceedings of the 17th Int.Conference on Autonomous Agents and MultiAgent Systems*, 443–451.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2016). Prioritized experience replay. In *Proceedings of the International Conference on Learning Representations.*

Scott, S.H. (2004). Optimal feedback control and the neural basis of volitional motor control. *Nature reviews. Neuroscience*, 5(7), 532–546.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms. *31st ICML.*

Vamvoudakis, K.G. and Lewis, F.L. (2011). Multi-player non-zero-sum games: Online adaptive learning solution of coupled hamilton–jacobi equations. *Automatica*, 47(8), 1556–1569.

Watkins, C.J.C.H. (1989). *Learning from Delayed Rewards.* Ph.D. thesis, King's College, Cambridge, UK.

Appendix A. NETWORK ARCHITECTURE AND TRAINING PARAMETER

Table A.1. Hyperparameters of the network architecture

hyperparameter	value
number of hidden layers	3
neurons per hidden layer	64
dropout probability	0.2
activation f. hidden layer	LeakyReLU, $\alpha = 0.01$
initialization of hidden layer	Xavier uniform, $\sim U(-0.5, 0.5)$
activation f. output layer A/C	linear
initial weights all layers A/C	$\sim U(-1, 1)$
optimizer	Adam, $\beta_1 = 0.9$, $\beta_2 = 0.999$, no gradient clipping, decay, fuzz factor or AMSGrad
error metric	Huber loss
target network update frequency m	4000

Table A.2. Hyperparameters of the algorithm

hyperparameter	value
discount factor γ	0.999
ξ_{temp}	0
ERM size M	1×10^5
macro-batch size B	256
mini-batch size t	80
α learning rate	5×10^{-4}
σ learning rate	2×10^{-4}
β learning rate	5×10^{-5}
λ_{high}	0.8
λ_{low}	0.2
exploration	$\varepsilon_{\text{min}} = 0.01$, decay rate $\varpi = 0.999$