Toward Multi-Agent Reinforcement Learning for Distributed Event-Triggered Control

Lukas Kesper LUKAS.KESPER@RWTH-AACHEN.DE
Sebastian Trimpe TRIMPE@DSME.RWTH-AACHEN.DE
Institute for Data Science in Mechanical Engineering, RWTH Aachen University, Germany

Dominik Baumann DOMINIK.BAUMANN@AALTO.FI
Department of Electrical Engineering and Automation, Aalto University, Espoo, Finland, Department of Information Technology, Uppsala University, Sweden

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

Event-triggered communication and control provide high control performance in networked control systems without overloading the communication network. However, most approaches require precise mathematical models of the system dynamics, which may not always be available. Model-free learning of communication and control policies provides an alternative. Nevertheless, existing methods typically consider single-agent settings. This paper proposes a model-free reinforcement learning algorithm that jointly learns resource-aware communication and control policies for distributed multi-agent systems from data. We evaluate the algorithm in a high-dimensional and nonlinear simulation example and discuss promising avenues for further research.

Keywords: Event-Triggered Communication, Hierarchical Reinforcement Learning, Networked Control Systems, Multi-Agent Learning

1. Introduction

In networked multi-agent systems, frequent communication of all agents can overload the network, resulting in longer delays and increased risk of message loss (Hespanha et al., 2007). In response, event-triggered methods have been developed (Heemels et al., 2012; Grüne et al., 2014; Miskowicz, 2016). In event-triggered control (ETC), agents only communicate in case they have something relevant to say. Designs for when agents should communicate and how information from other agents should be incorporated into the individual agent's control law are typically based on accurate dynamics models. However, such models may not be readily available for high-dimensional and nonlinear, e.g., robotic, systems. Moreover, even if they were, jointly optimizing both communication and control strategies may be intractable for complex systems. Nevertheless, as the separation principle does not necessarily hold for ETC (Ramesh et al., 2011), such a joint optimization is required to obtain an optimal overall strategy. Both challenges can be addressed by learning joint communication and control strategies from data through reinforcement learning (RL).

Over the last years, the first algorithms have been developed that learn communication and control strategies from data, see, for instance, Vamvoudakis and Ferraz (2018); Yang et al. (2017);

Zhong et al. (2014). However, most approaches simplify the problem by, e.g., imposing a specific structure for the communication strategy, thereby limiting the possibility of the RL algorithm to learn a jointly optimal policy. A key challenge in learning resource-aware communication and control strategies as compared to learning controllers under periodic communication is the hybrid action space. At each time step, an agent has to make a discrete communication decision, whether or not to communicate, and what control input to apply, which is typically a continuous value. Standard RL approaches struggle with this hybrid action space. A possible remedy is hierarchical RL (Sutton et al., 1999). The ability of hierarchical RL algorithms to learn high-performing communication and control policies for high-dimensional and nonlinear systems has been demonstrated by Funk et al. (2021). Nevertheless, the authors consider a single-agent system: one agent that is connected to a remote controller over a network. While this is undoubtedly a relevant problem setting, the issue of resource savings becomes even more paramount when considering distributed multi-agent systems.

Multi-agent RL (MARL), even without communication constraints, is a challenging research field of its own, Zhang et al. (2021) provide a general overview. Including resource-aware communication as a design goal adds another complexity layer. Thus, only few approaches exist that consider learning resource-aware communication and control strategies for multi-agent systems. The few existing approaches either separate the problem, i.e., learn individual communication and control policy (Demirel et al., 2018; Lima et al., 2022) or have only been evaluated in low-dimensional systems (Shibata et al., 2021, 2022).

Contributions. We tackle the problem of jointly learning communication and control strategies for high-dimensional networked multi-agent systems. In particular, we leverage techniques from hierarchical RL and MARL to learn resource-aware communication and control strategies and demonstrate the effectiveness of the resulting algorithm in a high-dimensional and nonlinear simulation example.

2. Related Work

In the following, we relate our contributions to the literature.

Multi-agent reinforcement learning. Learning control policies from data is a rapidly evolving research area. Various algorithms exist that can learn high-performing control policies without any prior knowledge about the system model (Lillicrap et al., 2015; Duan et al., 2016). However, the majority of works consider single-agent systems. Also in MARL, recent works have shown promising results (Zhang et al., 2021; Yu et al., 2020; Tang et al., 2018; Shu and Tian, 2018; Gupta et al., 2017; Lowe et al., 2017). Nevertheless, all mentioned works, both for single-agent and for multi-agent systems, assume that information can be exchanged at high periodic rates. Such frequent communication may be unsustainable in networked systems where information needs to be transmitted over communication networks. Thus, herein, we propose an algorithm that learns resource-aware communication and control policies for multi-agent systems.

Learning ETC for single agent-systems. For an introduction into and an overview of ETC, we refer the reader to Heemels et al. (2012); Miskowicz (2016). In all works discussed therein, the policy design is based on a system model that is assumed to be known. When this assumption is violated, learning approaches provide an alternative. Learning ETC from data has received increasing attention in recent years (Sedghi et al., 2022). Many existing approaches constrain the problem, for instance, by predefining the structure of the event-trigger (Vamvoudakis and Ferraz, 2018; Zhong et al., 2014; Yang et al., 2017; Sahoo et al., 2016). Restricting the space of communication policies

may lead to suboptimal policies, mainly because the separation principle does not generally hold in ETC (Ramesh et al., 2011). That is, for obtaining optimal communication and control policies, both need to be optimized jointly. Such a joint optimization is, e.g., done by Baumann et al. (2018); Hashimoto et al. (2021); Funk et al. (2021). However, those works, as the ones we discussed before, consider single-agent systems. Herein, we seek to go a step further and target multi-agent systems.

Learning ETC for multi-agent systems. Alternative approaches that aim at learning resource-aware control for multi-agent systems are sparse. Demirel et al. (2018) propose learning a scheduling strategy for a multi-agent control system. They use a step-by-step approach in which they begin with designing optimal control policies for each agent, followed by a central agent learning to adapt to the subsystems and controllers. A similar approach was chosen by Hu et al. (2021), where agents, connected via a network, learn a control policy and then a separate gating policy for the network. Both approaches thus separate the optimization problem which might yield suboptimal policies since the separation principle does not hold. Lima et al. (2022) learn a resource-allocation and control strategy for a multi-agent system assuming both a centralized allocation mechanism and controller, hence, not a distributed setting. To the best of our knowledge, only one alternative approach exists that jointly optimizes communication and control strategies for distributed multi-agent systems (Shibata et al., 2021, 2022). Nevertheless, their approach is, in essence, based on the approach by Baumann et al. (2018). Already Baumann et al. (2018) stated that the approach has troubles scaling to high-dimensional systems. Consequently, also Shibata et al. (2021, 2022) show results only in relatively low-dimensional environments. In this work, we instead leverage the more scalable approach introduced by Funk et al. (2021) based on hierarchical RL to design an algorithm that can learn resource-aware communication and control policies for multi-agent systems and seamlessly scales to high-dimensional environments.

3. Problem Setting

We consider a setting with N agents, where the dynamics of each agent i are given by

$$x_i[k+1] = f(x[k], u_i[k], v_i[k]), \tag{1}$$

with $k \in \mathbb{N}$ the discrete time step, $x_i[k] \in \mathbb{R}^n$ the state of agent i, $x[k] \in \mathbb{R}^{N \cdot n}$ the concatenated states of all agents, $u_i[k] \in \mathbb{R}^m$ the action of agent i, and $v_i[k] \in \mathbb{R}^n$ process noise. The agents are supposed to achieve a common control goal, i.e., apart from potential couplings through the dynamics function f, agents are also coupled through a joint objective. Thus, while each agent can measure its individual state and select a local control input, they also require information from the other agents for optimal decision-making. This information is exchanged via a communication network. The whole setup is shown in Fig. 1 (left).

At each time step, agent i needs to decide whether or not to communicate with other agents and which control input to apply, forming a hybrid action space. Hence, we have two policies: policy μ_i, which decides about sharing information via the network, and policy π_i, which decides which action u_i to take, i.e., which control input to choose.

Networked communication. Whenever agents decide to share data in the network, this information is shared among all agents. Thus, the last broadcasted state $\tilde{x}_i[k]$ of each agent i is

$$\tilde{x}_i[k] = \begin{cases} x_i[k], & \text{if } o_i = 1 \\ \tilde{x}_i[k-1], & \text{if } o_i = 0, \end{cases} \tag{2}$$

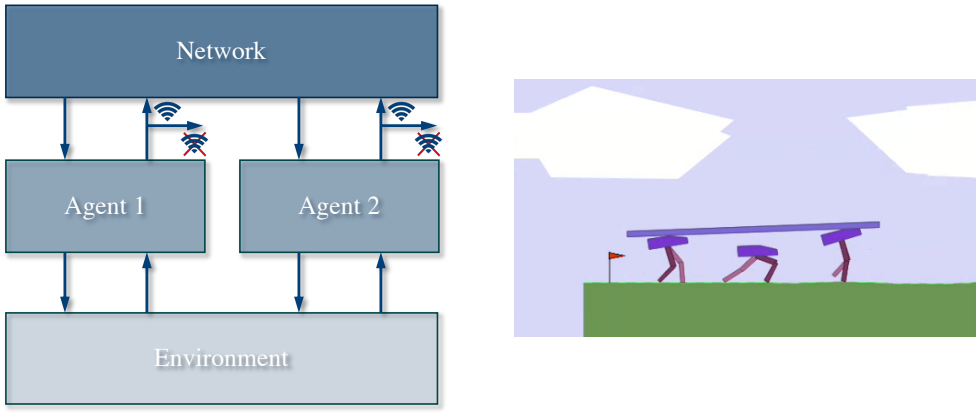

Figure 1: Basic components of the setting and image of the simulation environment. *Left: Agents connected via a network, here for two agents, interacting in an environment. Right: The Multiwalker environment from PettingZoo (Terry et al., 2021) used in our evaluation.*

where $o_i \in \{0, 1\}$ is agent i's communication decision. Implementations of many-to-all communication in wireless control have been demonstrated by Baumann et al. (2021); Mager et al. (2019).

Reward function. The optimization objective for each agent is formalized through the reward function. Unlike standard RL designs, we here explicitly add a reward for saving communication. Thus, we have a standard reward for control performance, r_i^{ctrl}, rewarding both joint and individual control goals of the agents, and a reward $r_i^{\text{comm}} = -o_i \xi$, with $\xi \in \mathbb{R}_{\geq 0}$, which imposes a cost for communication. The overall expected discounted return R_i at time step K is then

$$R_i[K] = \mathbb{E}\left[\sum_{k=1}^{K} \gamma^k (r_i^{\text{ctrl}}[k] + r_i^{\text{comm}})\right] = \mathbb{E}\left[\sum_{k=1}^{K} \gamma^k (r_i^{\text{ctrl}}[k] - o_i[k]\xi)\right], \tag{3}$$

where $\gamma \in (0, 1)$ is a discount factor.

Problem formulation. We aim to develop a framework that lets agents maximize the expected discounted return (3) by optimizing policies μ_i and π_i. To achieve optimal performance, we optimize both policies jointly. Ultimately, this leads to a framework that can learn high-performing and resource-aware communication and control policies for *(i)* linear and nonlinear, *(ii)* low- and high-dimensional cooperative multi-agent environments with direct interaction between agents, but *(iii)* without the need for a dynamics model.

4. Background: Hierarchical Reinforcement Learning

The key to learning event-triggered communication and control policies through RL is handling the hybrid action space, consisting of discrete communication decisions and continuous control tasks. Hierarchical reinforcement learning naturally captures this structure by adding another decision layer above the continuous actions within a Markov decision process (Sutton et al., 1999). This layer consists of discrete options representing primitive decisions with long-lasting effects (Precup, 2000). Relating to our problem setting, we have the option to communicate or not to communicate. Each

option is followed by actions from a policy explicitly related to the option. This policy outputs actions until there are better options to choose from, and, therefore, the option is terminated. Trajectory sampling following this structure can be used to train RL agents based on policy gradient theorems provided by the option-critic architecture (Bacon et al., 2017). Funk et al. (2021) use this architecture to derive an algorithm that learns single-agent event-triggered control. Following Funk et al. (2021), we now outline the fundamentals of the option-critic architecture. To keep the notation uncluttered, we consider single-agent systems in this section and, hence, drop the index i.

To represent the hierarchy, states x and actions u are extended by options o. The architecture uses several policies to parameterize the decision-making, leading from states to the choice of options and actions. Initially, the policy over options $\mu(o \mid x)$ outputs an option, whereupon an associated intra-option policy $\pi_o(u \mid x)$ outputs actions. This continues until a termination function $\beta_o(x)$ ends the option, leading to another sampling of options by the policy over options. In other words, the policy over options $\mu(o \mid x)$ represents the probability of choosing an option, while the termination function $\beta_o(x)$ represents the probability of ending the option. The action choice can then be expressed by a single policy,

$$\pi(u \mid o, x) = (1 - \beta_o(x))\pi_o(u \mid x) + \beta_o(x) \sum_{\tilde{o}} \mu(\tilde{o} \mid x)\pi_{\tilde{o}}(u \mid x). \tag{4}$$

Sampling from this policy yields trajectories from which policy updates can be computed. For computing these updates, the expected return (3), is approximated using a value function. In particular, we use a variant of the Q-function, which in standard RL methods outputs the expected reward when starting in state x, choosing action u, and following the according policy from there on. In the option critic framework, the Q-function also depends on the option, i.e.,

$$Q(o, x) = \int_{\tilde{u}} \pi(\tilde{u} \mid x)\hat{Q}(o, x, \tilde{u}) \, \mathrm{d}\tilde{u}, \tag{5}$$

where $\hat{Q}(o, x, u) = r(o, x, u) + \gamma Q(o', x')$, with x' the successor state of x, and $r(o, x, u)$ the reward in a time step. Using the Q-function, we can define the return to be maximized through learning,

$$R(o, x) = [1 - \beta_o(x)]Q(o, x) + \beta_o(x) \sum_{\tilde{o}} \mu(\tilde{o} \mid x)Q(\tilde{o}, x). \tag{6}$$

Formulating reward and Q-function as functions of state and option allows us to derive gradients to optimize the policies, parametrized by θ_μ, θ_π, and θ_β. For the policy over options, taking the gradient of the reward results in

$$\frac{\partial R(o, x, \theta_\mu, \theta_\beta)}{\partial \theta_\mu} = \beta_o(x, \theta_\beta)\mathbb{E}\left[\frac{\partial}{\partial \theta_\mu} \log(\mu(o, \theta_\mu \mid x))Q(o, x)\right]. \tag{7}$$

Besides, it follows from the Q-function for the intra-option policy that

$$\frac{\partial Q(o, x, \theta_\pi)}{\partial \theta_\pi} = \mathbb{E}\left[\frac{\partial}{\partial \theta_\pi} \log(\pi_o(u, \theta_\pi \mid x))\hat{Q}(o, x, u)\right]. \tag{8}$$

Following Funk et al. (2021), we decide about communication in every single time step. Consequently, we fix $\beta_o(x) \equiv 1$, i.e., we terminate options and evaluate the policy over options in every time step.

We can now compute policy updates based on the simplified gradients in (7) and (8). However, we leverage extensions that provide increased performance to improve learning capabilities.

In particular, we use proximal policy option-critic (PPOC) (Klissarov et al., 2017), which aims to exploit the performance of proximal policy optimization (PPO) Schulman et al. (2017) in the option-critic architecture. The theory behind PPO is based on trust region policy optimization (Schulman et al., 2015a), where the aim is to maximize a surrogate loss function

$$L(\theta) = \mathbb{E}\left[\frac{\pi(u \mid x)}{\pi^{\text{old}}(u \mid x)}\hat{A}^{\pi}\right] \tag{9}$$

with θ parametrizing policy π, and $\hat{A}^{\pi}$ an estimator of the advantage. This advantage is used to evaluate how beneficial an action is in relation to the expected reward in a time step. PPO extends this loss with a clipping function that clips the value of the ratio $\frac{\pi(u|x)}{\pi^{\text{old}}(u|x)}$, with a clipping parameter ϵ. Using the resulting loss function for updates of the intra-option policy, we obtain

$$\frac{\partial L(\theta_\pi)}{\partial(\theta_\pi)} = \mathbb{E}\left[\frac{\partial}{\partial\theta_\pi}\min[\frac{\pi_o(u|x)}{\pi_o^{\text{old}}(u|x)}A^{\pi}(o,x,u); \text{clip}(\frac{\pi_o(u|x)}{\pi_o^{\text{old}}(u|x)}, 1-\epsilon, 1+\epsilon)A^{\pi}(o,x,u)]\right]. \tag{10}$$

Finally, PPOC combines the clipped function (10) with generalized advantage estimation (GAE) (Schulman et al., 2015b) for computing the advantage $A^{\pi}(x,o,u)$ to obtain its policy updates.

Remark 1 *Generally, also other RL approaches that can handle hybrid action spaces, such as by Neunert et al. (2020), could be used. Such approaches have not been extended to ETC yet.*

5. Learning Resource-Aware Communication and Control for Multi-Agent Systems

We now present our algorithm. As discussed previously, we link hierarchical RL and event-triggered communication and control by letting agents share information over a network only if the corresponding option is triggered. However, the information an agent sends does not directly impact its decision about which control input to apply. Thus, we omit implementing intra-option policies that are unique to the option chosen by the policy over options μ_i. That is, each agent has, independent of the current option, only one control policy π_i. Nevertheless, we preserve the general nature of hierarchical RL, since the option choice does have a long-term impact by influencing the behavior of other agents.

In the case of communication, agents share information with all others. Thus, the available information for each agent consists of its own state x_i, as well as the last broadcasted states of the other agents, resulting in an extended state $\hat{x}_i = \{x_i, \tilde{x}_1, \ldots, \tilde{x}_{\text{N}}\}$. An update $\tilde{x}_i$ does not necessarily include all information an agent can send, allowing to send only a subset of information from x_i depending on the environment. The amount of information sent thus becomes an optimization parameter that is relevant for both computing resources and control performance. Nevertheless, receiving state information alone is insufficient for the agents to sample meaningful actions. Agents can only process this information if they are aware of the age of information, i.e., how old the last received update is. Thus, we introduce timers, with τ_i the timer for information sent by agent i,

$$\tau_i[k] = \begin{cases} 0, & \text{if } o_i[k] = 1, \\ \tau_i[k-1] + 1, & \text{else.} \end{cases} \tag{11}$$

Appending timers τ_i to last broadcasted states $\tilde{x}_i$ leads to policies $\mu_i(o_i \mid \hat{x}_i)$ and $\pi_i(u_i \mid \hat{x}_i)$ that enable agents to act within the setting shown in Fig. 1.

Communication might not have an immediate impact on other agents, which potentially destabilizes learning the policy over options. Thus, we also use PPO to update the policy over options, compared to PPOC, which uses native policy gradients. PPO's sample efficiency can help agents find a resource-aware communication strategy. Utilizing the derivation by Funk et al. (2021), we have

$$\frac{\partial L(\theta_{\mu_i})}{\partial(\theta_{\mu_i})} = \frac{\partial}{\partial\theta_{\mu_i}}\mathbb{E}[\min[\frac{\mu_i(o_i|\hat{x}_i)}{\mu_i^{\text{old}}(o_i|\hat{x}_i)}A_i^{\mu}(o_i,\hat{x}_i,x); \text{clip}(\frac{\mu_i(o_i|\hat{x}_i)}{\mu_i^{\text{old}}(o_i|\hat{x}_i)}, 1-\epsilon, 1+\epsilon)A_i^{\mu}(o_i,\hat{x}_i,x)]], \quad (12)$$

where we use a greedy advantage function

$$A_i^{\mu}(o_i,\hat{x}_i,x) = Q_i(o_i,\hat{x}_i,x) - \max_{\tilde{o}_i\in\{0,1\}} Q_i(\tilde{o}_i,\hat{x}_i,x). \quad (13)$$

Besides performance improvements, restricting the updates of the policy over options through the clipping function is also a reasonable measure to limit drastic changes in the policy that could potentially destabilize the learning of other agents.

In (13), we see that Q_i depends on the concatenated state x in addition to the extended state $\hat{x}_i$. This is connected to the destabilizing effect of other agents who change their policy through learning. From the individual agent's perspective, which models other agents as part of the environment, these changes make the environment appear unstable. In our case, this is an even greater challenge as each agent has several policies. To compensate for this perception, we use centralized learning and decentralized execution (CLDE): during learning, we use a critic for each agent i, i.e., the Q_i-function, to compute policy updates. During execution, the policies μ_i and π_i are used to sample trajectories without the critic. By centralizing the critic, learning about the value function can be improved, helping to stabilize the overall learning process (Lyu et al., 2021). Such centralization can be achieved by handing over all relevant state information from other agents j to the critic of an agent i, following the approach by Lowe et al. (2017). We implement this by always passing the true states x_j to the critic of agent i in addition to the last broadcasted states $\hat{x}_j$ to compensate for the lack of information through saving communication.

Remark 2 *CLDE requires all agents to communicate periodically during training. We assume that during training in a laboratory environment, we can provide sufficient bandwidth for periodic communication, but during execution in the real world need to save resources.*

We use deep neural networks (DNNs) to approximate the Q_i-functions, the policy over options μ_i, and each agent's control policy π_i. We follow the DNN structure proposed by Funk et al. (2021). However, we set the standard deviation of the stochastic control policy to a constant value instead of slowly decreasing it. Lowering the stochasticity aims to reduce the dispersion in the behavior of other agents, making it easier for the agents to adapt to each other.

This reduced stochasticity also causes less exploration. Thus, we always consider the learning progress over training epochs and introduce a curriculum that supports the agents in generating a solution. First, we apply noise to the actions u_i. In particular, we use values from a Gaussian distribution with zero mean and a linearly decreasing standard deviation σ. This way, we achieve adjustability and phasing for agent exploration, effectively making exploration a hyperparameter. However, while the agents are still exploring the observation space to improve the output of the control policy, the communication is already penalized. The policy over options typically reacts quickly by reducing communication. This inevitably leads to local optima, as no cooperative action is possible without shared information. Therefore, we leverage entropy regularization, which ensures

that communication is learned only after exploration and, thus, forms the next step in the curriculum. In the beginning, we apply the entropy term $\zeta \log(\mu_i(o_i \mid \hat{x}_i))\mu_i(o_i \mid \hat{x}_i)$ to the updates of the policy over options, limiting updates through the entropy regularization coefficient ζ. After sufficiently exploring the control policy, we lower the value of ζ linearly over epochs, resulting in slowly enabled updates to the policy over options. After reaching the final epoch of the decrease, the coefficient remains at a low value. Typically, this leads to destabilization and decreased rewards since agents receive less information as communication savings increase. As the last step, we, therefore, introduce another episode of noise in the curriculum, whose standard deviation first increases linearly over epochs and then decreases again. This yields further exploration of the policies π_i and μ_i, as well as the Q_i-function. In summary, the curriculum proceeds in three steps: we start by *(I)* adding noise to the actions with decreasing variance, followed by *(II)* enabling policy updates of the policy over options, and *(III)* finally, we apply noise again to stimulate full exploration of the observation space. The entire algorithm is summarized in Alg. 1.

Algorithm 1 Learning algorithm for distributed event-triggered control.

Input: clipping ϵ, curriculum param. ζ, σ
for number of agents **do**
  Initialize $Q_i(o_i, \hat{x}_i, x), \mu_i(o_i|\hat{x}_i), \pi_i(u_i|\hat{x}_i)$
for number of epochs **do**
  for number of agents **do**
    $Q_i \leftarrow Q_i^{\text{old}}$, $\mu_i \leftarrow \mu_i^{\text{old}}$, $\pi_i \leftarrow \pi_i^{\text{old}}$
  Sample (o, x, u)-transitions using μ_i, π_i
  for number of agents **do**
    Calculate $A_i^{\pi}(o_i, \hat{x}_i, x, u)$ using GAE
    for number of optimizer iterations **do**
      for number of options **do**
        Sample batch
        Calculate $A_i^{\mu}(o_i, \hat{x}_i, x)$ from (13)
        Calculate L_i^{μ} from (12)
        Calculate L_i^{π} from (10)
        Calculate L_i^{Q} as Funk et al. (2021)
        $\theta_{\mu_i} \leftarrow \theta_{\mu_i} + \alpha \frac{\partial L_i^{\mu}(\theta_{\mu_i})}{\partial \theta_{\mu_i}}$
        $\theta_{\pi_i} \leftarrow \theta_{\pi_i} + \alpha \frac{\partial L_i^{\pi}(\theta_{\pi_i})}{\partial \theta_{\pi_i}}$
        $\theta_{Q_i} \leftarrow \theta_{Q_i} - \alpha \frac{\partial L_i^{Q}(\theta_{Q_i})}{\partial \theta_{Q_i}}$
  switch (curriculum phase)
    case I: Update σ
    case II: Update ζ
    case III: Update σ

6. Results and Analysis

We demonstrate the effectiveness of our framework in a simulation example that includes a traceable cooperative task in a high-dimensional multi-agent environment with complex physics. For this, we choose the Multiwalker environment from PettingZoo (Terry et al., 2021), shown in Fig. 1 (right). The goal in this environment is for the three agents to cooperatively transport a package forward, without falling or dropping the package.

We slightly modify the environment to fit our problem setting. In particular, we change the agents' observation space. In the native implementation, this information includes all joint angles, speeds, and contact sensors of the individual agent. Besides, agents use LIDAR sensors to scan the environment and obtain information about the position of the agent in front of them. Alongside the LIDAR, agents receive state information about their neighbors in the form of relative positions, as well as their relative position towards the package and the package angle. To make communication necessary, we remove the LIDAR information. Similarly, we remove relative information about neighbors, which we obtain through communication. The state of an individual agent consists then in their own

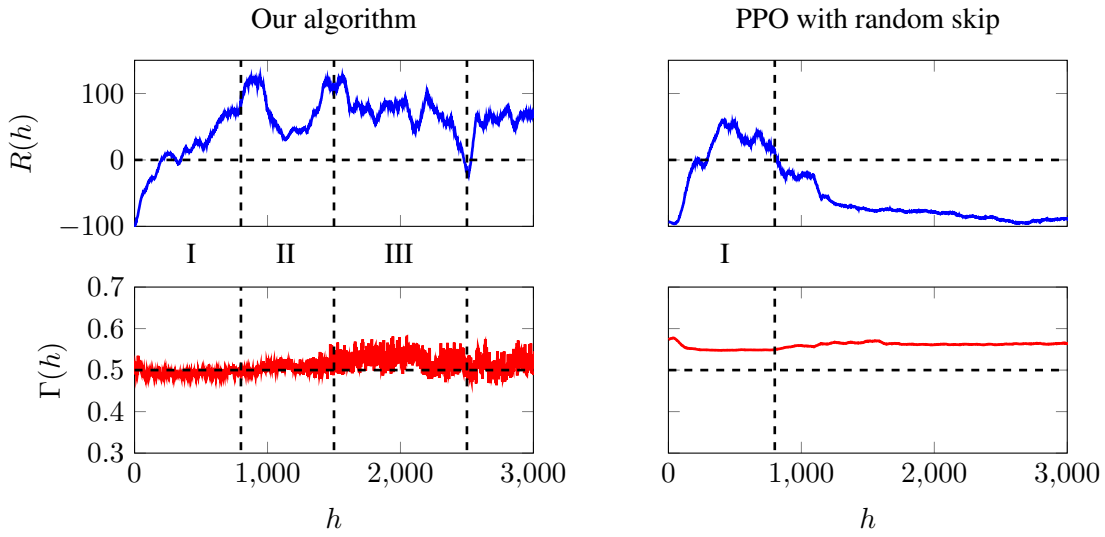

Figure 2: Performance of our algorithm and PPO with random communication skips. *We show the return (top plots) and communication savings (bottom plots). While our algorithm achieves significant communication savings and good performance (left), PPO with random communication skips is unstable and cannot learn a good policy (right). Dashed vertical lines represent the stages of the curriculum.*

joint angles, speeds, contact forces, and information about the package, which results in 17 state variables per agent. We introduce communication by allowing agents to share their absolute position and horizontal and vertical velocity, from which receiving agents can calculate relative positions and velocities. We thus define the last broadcasted state $\tilde{x}_i$ as the relative distances in position and the relative differences in the velocities of the agents, combined with the corresponding timer τ_i. Ultimately, the agents know their own state, relevant relative positions, as well as information about the relative state of the package, all combined in the networked observation $\hat{x}_i$.

We further need to decide about inputs to the policies. As we use CLDE, the Q_i-functions receive all information available to the agent and the relative data of the other agents that have not been communicated. The policy π_i receives the package measurements and the communicated states of other agents to sample meaningful actions. For the policy μ_i, experiments have shown that it is most effective to limit the input to the package information. The idea is to initiate communication based on how the package behaves during a training run. For instance, a deflecting package can be a relevant trigger for sending an update. Thus, we add only the current and the package information from the last update to the policy over options μ_i.

Lastly, we define the reward function. Different from the original environment, we add a communication penalty. Further, to encourage agents to find creative solutions, we do not penalize head deflection and extend the possible torque for the joints by a factor of 1.5. In addition, we introduce a small penalty for speed in the horizontal direction to prevent agents from learning risky gaits that, according to our observations, can lead to the collapse of the entire learning process.

We train the algorithm for ten runs, where one training iteration includes 2048 time steps for each agent, and we compute updates over 3000 epochs. We show the best performing run in Fig. 2 (left).

In particular, we show the return $R(h)$ and communication savings $\Gamma(h)$, calculated in comparison to periodic communication at each time step with an underlying sampling time of 20 ms, of all agents over epochs h, averaged over 50 epochs. We see the effect of the three curriculum stages in the learning curve, marked by dashed vertical lines. Initially, we only explore the control policy, leading to increasing reward. As the agents start to actively reduce communication in the second phase, the reward first drops, but recovers after a while. In the third phase, we again explore the control policy, which stabilizes the learning result. Finally, the agents learn to move the package cooperatively forward while saving around 55 % of communication. The resulting policy is intuitive in that, most of the time, only two agents actively work on transporting the package, reducing the need to coordinate[1].

We further compare our algorithm to native PPO. Kargar and Kyrki (2021) have shown that PPO, in general, can learn good policies for the Multiwalker. We thus train PPO on the Multiwalker and randomly skip communication to also achieve savings around 55 %. While PPO does not make its own decisions about communication, we leave the remaining design unchanged, i.e., we use CLDE, the same DNN structure, and the first stage of the learning curriculum. We then also execute ten training runs and show the best performing policy in Fig. 2 (right). The return PPO obtains is significantly lower than for our algorithm. This is particularly remarkable as PPO does not incur any communication penalties. The PPO agents cannot successfully transport the package forward, demonstrating that the task is non-trivial.

7. Conclusion

This article presents a first step toward automatic learning of joint communication and control policies for high-dimensional and distributed multi-agent systems. We capture the hybrid nature of joint communication and control policies through a hierarchical RL approach and optimize both. The resulting algorithm significantly outperforms a control policy with random communication skips.

While these are promising results, there are various possibilities for further research. The algorithm achieves good performance on a challenging simulation task. However, the training stability may still be improved. When training multiple agents in parallel, around half of the agents end up with a performance comparable to what we have shown in the previous section. Through hyperparameter tuning, a higher success rate may still be attainable. Further, the underlying algorithm and software framework are, in machine learning terms, relatively old. In particular, we use Tensorflow 1 (Abadi et al., 2016) and the baselines (Dhariwal et al., 2017) implementation of PPO (Schulman et al., 2017). By updating to more modern frameworks, increased performance and stability might be possible. The choice for PPO as the underlying RL algorithm was motivated by findings that it performs well in multi-agent settings (Yu et al., 2021). Nevertheless, dedicated MARL algorithms may still provide a further performance increase. Also, adopting specific MARL measures beyond CLDE, such as parameter sharing among agents (Christianos et al., 2021), might yield more stable and successful learning. Lastly, the options framework allows for even greater flexibility. For instance, one could imagine options for sending different kinds of information depending on the state of the environment, or combining our approach with event-triggered control updates.

1. Videos and code are available at https://sites.google.com/view/learning-distributed-etc/.

Acknowledgments

Special thanks go to Niklas Funk, Friedrich Solowjow, and Bernd Frauenknecht for their technical input, as well as to Jonas Reiher, Johanna Menn, and Johannes Berger for their helpful advice. This work is funded in part under the Excellence Strategy of the Federal Government and the Laender (G:(DE-82)EXS-SF-SFDdM035) of Germany. Simulations were performed with computing resources granted by RWTH Aachen University under project thes1218.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. TensorFlow: a system for Large-Scale machine learning. In *USENIX Symposium on Operating Systems Design and Implementation*, pages 265–283, 2016.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI Conference on Artificial Intelligence*, pages 1726–1734, 2017.

Dominik Baumann, Jia-Jie Zhu, Georg Martius, and Sebastian Trimpe. Deep reinforcement learning for event-triggered control. In *IEEE Conference on Decision and Control*, pages 943–950, 2018.

Dominik Baumann, Fabian Mager, Ulf Wetzker, Lothar Thiele, Marco Zimmerling, and Sebastian Trimpe. Wireless control for smart manufacturing: Recent approaches and open challenges. *Proceedings of the IEEE*, 109(4):441–467, 2021.

Filippos Christianos, Georgios Papoudakis, Muhammad A Rahman, and Stefano V Albrecht. Scaling multi-agent reinforcement learning with selective parameter sharing. In *International Conference on Machine Learning*, pages 1989–1998, 2021.

Burak Demirel, Arunselvan Ramaswamy, Daniel E. Quevedo, and Holger Karl. DeepCAS: a deep reinforcement learning algorithm for control-aware scheduling. *IEEE Control Systems Letters*, 2 (4):737–742, 2018.

Prafulla Dhariwal, Christopher Hesse, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. OpenAI baselines. https://github.com/openai/baselines, 2017.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

Niklas Funk, Dominik Baumann, Vincent Berenz, and Sebastian Trimpe. Learning event-triggered control from data through joint optimization. *IFAC Journal of Systems and Control*, 16:100144, 2021.

L Grüne, S Hirche, O Junge, P Koltai, D Lehmann, J Lunze, A Molin, R Sailer, M Sigurani, C Stöcker, and F. Wirth. Event-based control. In Jan Lunze, editor, *Control Theory of Digitally Networked Dynamic Systems*, pages 169–261. Springer, 2014.

Jayesh K. Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 66–83, 2017.

Kazumune Hashimoto, Yuichi Yoshimura, and Toshimitsu Ushio. Learning self-triggered controllers with Gaussian processes. *IEEE Transactions on Cybernetics*, 51(12):6294–6304, 2021.

Wilhelmus P. M. H. Heemels, Karl Henrik Johansson, and Paulo Tabuada. An introduction to event-triggered and self-triggered control. In *IEEE Conference on Decision and Control*, pages 3270–3285, 2012.

Joo P Hespanha, Payam Naghshtabrizi, and Yonggang Xu. A survey of recent results in networked control systems. *Proceedings of the IEEE*, 95(1):138–162, 2007.

Guangzheng Hu, Yuanheng Zhu, Dongbin Zhao, Mengchen Zhao, and Jianye Hao. Event-triggered communication network with limited-bandwidth constraint for multi-agent reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13, 2021.

Eshagh Kargar and Ville Kyrki. MACRPO: Multi-agent cooperative recurrent policy optimization. *arXiv preprint arXiv:2109.00882*, 2021.

Martin Klissarov, Pierre-Luc Bacon, Jean Harb, and Doina Precup. Learnings options end-to-end for continuous action tasks. *arXiv preprint arXiv:1712.00004*, 2017.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Vinicius Lima, Mark Eisen, Konstantinos Gatsis, and Alejandro Ribeiro. Model-free design of control systems over wireless fading channels. *Signal Processing*, 197:108540, 2022.

Ryan Lowe, Y. I. WU, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6382–6393, 2017.

Xueguang Lyu, Yuchen Xiao, Brett Daley, and Christopher Amato. Contrasting centralized and decentralized critics in multi-agent reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, page 844–852, 2021.

Fabian Mager, Dominik Baumann, Romain Jacob, Lothar Thiele, Sebastian Trimpe, and Marco Zimmerling. Feedback control goes wireless: Guaranteed stability over low-power multi-hop networks. In *ACM/IEEE International Conference on Cyber-Physical Systems*, page 97–108, 2019.

Marek Miskowicz. *Event-Based Control and Signal Processing*. CRC Press, 2016.

Michael Neunert, Abbas Abdolmaleki, Markus Wulfmeier, Thomas Lampe, Jost Tobias Springenberg, Roland Hafner, Francesco Romano, Jonas Buchli, Nicolas Heess, and Martin Riedmiller. Continuous-discrete reinforcement learning for hybrid control in robotics. In *Conference on Robot Learning*, pages 735–751, 2020.

Doina Precup. *Temporal Abstraction in Reinforcement Learning*. PhD thesis, University of Massachusetts Amherst, 2000.

Chithrupa Ramesh, Henrik Sandberg, Lei Bao, and Karl Henrik Johansson. On the dual effect in state-based scheduling of networked control systems. In *American Control Conference*, pages 2216–2221, 2011.

Avimanyu Sahoo, Hao Xu, and Sarangapani Jagannathan. Neural network-based event-triggered state feedback control of nonlinear continuous-time systems. *IEEE Transactions on Neural Networks and Learning Systems*, 27(3):497–509, 2016.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Leila Sedghi, Zohaib Ijaz, Md. Noor-A-Rahim, Kritchai Witheephanich, and Dirk Pesch. Machine learning in event-triggered control: Recent advances and open issues. *IEEE Access*, 10:74671–74690, 2022.

Kazuki Shibata, Tomohiko Jimbo, and Takamitsu Matsubara. Deep reinforcement learning of event-triggered communication and control for multi-agent cooperative transport. In *IEEE International Conference on Robotics and Automation*, pages 8671–8677, 2021.

Kazuki Shibata, Tomohiko Jimbo, and Takamitsu Matsubara. Deep reinforcement learning of event-triggered communication and consensus-based control for distributed cooperative transport. *Robotics and Autonomous Systems*, page 104307, 2022.

Tianmin Shu and Yuandong Tian. M^3RL: Mind-aware multi-agent management reinforcement learning. *arXiv preprint arXiv:1810.00147*, 2018.

Richard S Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Hongyao Tang, Jianye Hao, Tangjie Lv, Yingfeng Chen, Zongzhang Zhang, Hangtian Jia, Chunxu Ren, Yan Zheng, Zhaopeng Meng, Changjie Fan, et al. Hierarchical deep multiagent reinforcement learning with temporal abstraction. *arXiv preprint arXiv:1809.09332*, 2018.

J. Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S. Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, Niall Williams, Yashas Lokesh, and Praveen Ravi. PettingZoo: Gym for multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 15032–15043, 2021.

Kyriakos G. Vamvoudakis and Henrique Ferraz. Model-free event-triggered control algorithm for continuous-time linear systems with optimal performance. *Automatica*, 87:412–420, 2018.

Xiong Yang, Haibo He, and Derong Liu. Event-triggered optimal neuro-controller design with reinforcement learning for unknown nonlinear systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(9):1866–1878, 2017.

Chao Yu, Yinzhao Dong, Yangning Li, and Yatong Chen. Distributed multi–agent deep reinforcement learning for cooperative multi–robot pursuit. *The Journal of Engineering*, 2020(13):499–504, 2020.

Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021.

Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. In Kyriakos G. Vamvoudakis, Yan Wan, Frank L. Lewis, and Derya Cansever, editors, *Handbook of Reinforcement Learning and Control*, pages 321–384. Springer, 2021.

Xiangnan Zhong, Zhen Ni, Haibo He, Xin Xu, and Dongbin Zhao. Event-triggered reinforcement learning approach for unknown nonlinear continuous-time system. In *International Joint Conference on Neural Networks*, pages 3677–3684, 2014.