Robust Event-Driven Interactions in Cooperative Multi-agent Learning

Jarne Ornia, Daniel; Mazo, Manuel

DOI
10.1007/978-3-031-15839-1_16

Publication date
2022

Document Version
Final published version

Published in
Formal Modeling and Analysis of Timed Systems

Citation (APA)
Jarne Ornia, D., & Mazo, M. (2022). Robust Event-Driven Interactions in Cooperative Multi-agent Learning. In S. Bogomolov, & D. Parker (Eds.), *Formal Modeling and Analysis of Timed Systems: 20th International Conference, FORMATS 2022, Warsaw, Poland, September 13–15, 2022, Proceedings* (pp. 281-297). (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 13465 LNCS). Springer. https://doi.org/10.1007/978-3-031-15839-1_16

Robust Event-Driven Interactions in Cooperative Multi-agent Learning

Daniel Jarne Ornia(✉) and Manuel Mazo Jr.

Delft Center for Systems and Control, Delft University of Technology, 2628 CD Delft, The Netherlands
d.jarneornia@tudelft.nl

Abstract. We present an approach to safely reduce the communication required between agents in a Multi-Agent Reinforcement Learning system by exploiting the inherent robustness of the underlying Markov Decision Process. We compute robustness certificate functions (off-line), that give agents a conservative indication of how far their state measurements can deviate before they need to update other agents in the system with new measurements. This results in fully distributed decision functions, enabling agents to decide when it is necessary to communicate state variables. We derive bounds on the optimality of the resulting systems in terms of the discounted sum of rewards obtained, and show these bounds are a function of the design parameters. Additionally, we extend the results for the case where the robustness surrogate functions are learned from data, and present experimental results demonstrating a significant reduction in communication events between agents.

Keywords: Multi-Agent Systems · Event-Triggered Communication · Reinforcement Learning

1 Introduction

In the last two decades we have seen a surge of learning-based techniques applied to the field of multi agent game theory, enabling the solution of larger and more complex problems, both model based and model free [3,13,24]. Lately, with the wide adoption of Deep Learning techniques for compact representations of value functions and policies in model-free problems [17,23,34], the field of Multi-Agent Reinforcement Learning (MARL) has seen an explosion in the applications of such algorithms to solve real-world problems [20]. However, this has naturally led to a trend where both the amount of data handled in such data driven approaches and the complexity of the targeted problems grow exponentially. In a MARL setting where communication between agents is required, this may inevitably lead to restrictive requirements in the frequency and reliability of the communication to and from each agents (as it was already pointed out in [25]).

This work was partially supported by the ERC Starting Grant SENTIENT #755953.

S. Bogomolov and D. Parker (Eds.): FORMATS 2022, LNCS 13465, pp. 281–297, 2022.
https://doi.org/10.1007/978-3-031-15839-1_16

The effect of asynchronous communication in dynamic programming problems was studied already in [2]. In particular, one of the first examples of how communication affects learning and policy performance in MARL is found in [32], where the author investigates the impact of agents sharing different combinations of state variable subsets or Q values. After that, there have been multiple examples of work studying different types of communication in MARL and what problems arise from it [1,16,28,30]. In this line, in [37] actor coordination minimization is addressed and in [10,15] authors allow agents to choose a communication action and receive a reward when this improves the policies of other agents. In [6] multi agent policy gradient methods are proposed with convergence guarantees where agents communicate gradients based on some trigger conditions, and in [19] agents are allowed to communicate a simplified form of the parameters that determine their value function.

We focus particularly in a *centralised training - decentralised execution*, where agents must communicate state measurements to other agents in order to execute the distributed policies. Such a problem represents most real applications of MARL systems: It is convenient to train such systems in a simulator, in order to centrally learn all agents' value functions and policies. But if the policies are to be executed in a live (real) setting, agents will have access to different sets of state variables that need to be communicated with each-other. In this case, having non-reliable communication leads to severe disruptions in the robustness of the distributed policies' performance. The authors in [18] demonstrated experimentally how very small adversarial disruptions in state variable communications leads to a collapse of the performance of general collaborative MARL systems. In this regard, [7] proposes learning an "adviser" model to fall back on when agents have too much uncertainty in their state measurements, and more recently in [14] the authors enable agents to run simulated copies of the environment to compensate for a disruption in the communication of state variables, and in [36] agents are trained using adversarial algorithms to achieve more robust policies. This lack of robustness in communicative multi-agent learning presents difficulties when trying to design efficient systems where the goal is to communicate less often.

With this goal in mind, we can look into event triggered control (ETC) as a strategy to reduce communication [22,31] in a networked system by trading off communication for robustness against state measurement deviations. This has been applied before in linear multi-agent systems [8] and non-linear systems [9,38]. In [33] and [26] ideas on how to use ETC on model-free linear and non-linear systems were explored. Additionally in other learning problems such as [29], where authors show how event triggered rules can be applied to learn model parameters more efficiently, and in [12] by applying a similar principle to demonstrate how ETC can be used to compute stochastic gradient descent steps in a decentralised event-based manner.

1.1 Main Contribution

We consider in this work a general cooperative MARL scenario where agents have learned distributed policies that must be executed in an on-line scenario, and that depend on other agent's measurements. We propose a constructive approach to synthesise communication strategies that minimise the amount of communication required and guarantee a minimum performance of the MARL system in terms of cumulative reward when compared to an optimal policy. We construct so-called *robustness surrogate* functions, which quantify the robustness of the system to disturbances in agent state variable measurements, allowing for less communication in more robust state regions. Additionally, we consider the case where these surrogate functions are learned through the *scenario approach* [4,5], and show how the guarantees are adapted for learned approximated functions.

2 Preliminaries

2.1 Notation

We use calligraphic letters for sets and regular letters for functions $f : \mathbb{R}^m \to \mathbb{R}^n$. We say a function $f : \mathbb{R}_+ \to \mathbb{R}_+$ is $f \in \mathcal{K}_\infty$ if it is continuous, monotonically increasing and $f(0) = 0$, $\lim_{a\to\infty} f(a) = \infty$. We use $\mathcal{F}$ as the σ-algebra of events in a probability space, and P as a probability measure $P : \mathcal{F} \to [0, 1]$. We use $E[\cdot]$ and $\mathrm{Var}[\cdot]$ for the expected value and the variance of a random variable. We use $\|\cdot\|_\infty$ as the sup-norm, $|\cdot|$ as the absolute value or the cardinality, and $\langle v, u\rangle$ as the inner product between two vectors. We say a random process X_n converges to a random variable X *almost surely* (a.s.) as $t \to \infty$ if it does so with probability one for any event $\omega \in \mathcal{F}$. For a conditional expectation, we write $E[X|Y] \equiv E_Y[X]$.

2.2 MDPs and Multi-agent MDPs

We first present the single agent MDP formulation.

Definition 1 *[Markov Decision Process]. A Markov Decision Process (MDP) is a tuple* $(\mathcal{X}, \mathcal{U}, P, r)$ *where* $\mathcal{X}$ *is a set of states,* $\mathcal{U}$ *is a set of actions,* $P : \mathcal{U} \to \mathbb{P}^{|\mathcal{X}|\times|\mathcal{X}|}$ *is a probability measure of the transitions between states and* $r : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to \mathbb{R}$ *is a reward function, such that* $r(\mathbf{x}, u, \mathbf{x}')$ *is the reward obtained when action* u *is performed at state* $\mathbf{x}$ *and leads to state* $\mathbf{x}'$.

In general, an agent has a policy $\pi : \mathcal{X} \to \mathbb{P}(\mathcal{U})$, that maps the states to a probability vector determining the chance of executing each action. We can extend the MDP framework to the case where multiple agents take simultaneous actions on an MDP. For the state transition probabilities we write in-distinctively $P_{\mathbf{x}\mathbf{x}'}(u) \equiv P(\mathbf{x}, \mathbf{x}', u)$, and for the reward obtained in two consecutive states $\mathbf{x}_t, \mathbf{x}_{t+1}$ we will write $r_t \equiv r(\mathbf{x}_t, u_t, \mathbf{x}_{t+1})$.

Definition 2 *[Collaborative Multi-Agent MDP]. A Collaborative Multi-Agent Markov Decision Process (c-MMDP) is a tuple* $(\mathcal{N}, \mathcal{X}, \mathcal{U}^n, P, r)$ *where* $\mathcal{N}$ *is a set of* n *agents,* $\mathcal{X}$ *is a cartesian product of* metric *state spaces* $\mathcal{X} = \prod_{i\in\mathcal{N}} \mathcal{X}_i$, $\mathcal{U}^n = \prod_{i\in\mathcal{N}} \mathcal{U}_i$ *is a joint set of actions,* $P : \mathcal{U}^n \to \mathbb{P}^{\mathcal{X}\times\mathcal{X}}$ *is a probability measure of the transitions between states and* $r : \mathcal{X} \times \mathcal{U}^n \times \mathcal{X} \to \mathbb{R}$ *is a reward function.*

Assumption 1. *We assume that each agent* i *has access to a set* $\mathcal{X}_i \subset \mathcal{X}$*, such that the observed state for agent* i *is* $\mathbf{x}(i) \in \mathcal{X}_i$*. That is, the global state at time* t *is* $\mathbf{x}_t = (\mathbf{x}_t(1)\ \mathbf{x}_t(2)\ ...\ \mathbf{x}_t(n))^T$*. Furthermore, we assume that the space* $\mathcal{X}$ *accepts a sup-norm* $\|\cdot\|_\infty$*.*

In the c-MMDP case, we can use $U \in \mathcal{U}^n$ to represent a specific joint action $U := \{U(1), U(2), ..., U(n)\}$, and $\mathit{\Pi} := \{\pi_1, \pi_2, ..., \pi_n\}$ to represent the joint policies of all agents such that $\mathit{\Pi} : \mathcal{X} \to \mathcal{U}^n$. We assume in this work that agents have a common reward function, determined by the joint action. That is, even if agents do not have knowledge of the actions performed by others, the reward they observe still depends on everyone's joint action. Additionally, we assume in the c-MMDP framework that the control of the agents is fully distributed, with each agent having its own (deterministic) policy π_i that maps the global state to the individual action, i.e. $\pi_i : \mathcal{X} \to \mathcal{U}_i$. We define the optimal policy in an MDP as the policy π^* that maximises the expected discounted reward sum $E[\sum_{t=1}^{\infty} \gamma^t r_t \,|\pi, \mathbf{x}_0]\ \forall \mathbf{x}_0 \in \mathcal{X}$ over an infinite horizon, for a given discount $\gamma \in (0, 1)$. The optimal joint (or *centralised*) policy in a c-MMDP is the joint policy $\mathit{\Pi}^*$ that maximises the discounted reward sum in the "centrally controlled" MDP, and this policy can be decomposed in a set of agent-specific optimal policies $\mathit{\Pi}^* = \{\pi_1^*, \pi_2^*, ..., \pi_n^*\}$.

Remark 1. Assumption 1 is satisfied in most MARL problems where the underlying MDP represents some real physical system (*e.g.* robots interacting in a space, autonomous vehicles sharing roads, dynamical systems where the state variables are metric...). In the case where $\mathcal{X}$ is an abstract discrete set, we can still assign a trivial bijective map $I : \mathcal{X} \to \mathbb{N}$ and compute distances on the mapped states $\|\mathbf{x}_1 - \mathbf{x}_2\|_\infty \equiv \|I(\mathbf{x}_1) - I(\mathbf{x}_2)\|_\infty$. However, we may expect the methods proposed in this work to have worse results when the states are artificially numbered, since the map I may have no relation with the transition probabilities (we come back to this further in the work).

2.3 Value Functions and Q-Learning in c-MMDPs

Consider a c-MMDP, and let a value function under a joint policy $\mathit{\Pi}$, $V^{\mathit{\Pi}} : \mathcal{X} \to \mathbb{R}$ be $V^{\mathit{\Pi}}(\mathbf{x}) = \sum_{\mathbf{x}'} P_{\mathbf{x}\mathbf{x}'}(\mathit{\Pi}(\mathbf{x}))(r(\mathbf{x}, \mathit{\Pi}(\mathbf{x}), \mathbf{x}') + \gamma V^{\mathit{\Pi}}(\mathbf{x}'))$. There exists an optimal value function V^* for a centralised controller in a c-MMDP that solves the Bellman equation:

$$V^*(\mathbf{x}) := \max_U \sum_{\mathbf{x}'} P_{\mathbf{x}\mathbf{x}'}(U)(r(\mathbf{x}, U, \mathbf{x}') + \gamma V^*(\mathbf{x}')).$$

Now consider so-called Q functions $Q : \mathcal{X} \times \mathcal{U}^n \to \mathbb{R}$ on the centrally controlled c-MMDP [35], such that the optimal Q function satisfies

$$Q^*(\mathbf{x}, U) := \sum_{\mathbf{x}'} P_{\mathbf{x}\mathbf{x}'}(U)(r(\mathbf{x}, U, \mathbf{x}') + \gamma \max_{U'} Q^*(\mathbf{x}', U')),$$

and the optimal centralised policy is given by $U^* := \pi^*(\mathbf{x}) = \text{argmax}_U\, Q^*(\mathbf{x}, U)$. Additionally, $\max_U Q^*(\mathbf{x}, U) = V^*(\mathbf{x}) = E[\sum_{t=1}^{\infty} \gamma^t r(\mathbf{x}, \Pi^*(\mathbf{x}), \mathbf{x}') \,|\mathbf{x}_0]$.

3 Information Sharing Between Collaborative Agents: Problem Formulation

Consider now the case of a c-MMDP where each agent has learned a distributed policy $\pi_i : \mathcal{X} \to \mathcal{U}_i$. We are interested in the scenario where the state variable $\mathbf{x}_t \in \mathcal{X}$ at time t is composed by a set of joint observations from all agents, and these observations need to be communicated to other agents for them to compute their policies.

Assumption 2. *Agents have a set of optimal policies Π^* available for executing on-line and a global optimal function Q^* (learned as a result of e.g. a multi-agent actor critic algorithm [20]).*

Consider the case where at a given time t, a subset of agents $\hat{\mathcal{N}}_t \subseteq \mathcal{N}$ does not share their state measurements with other agents. Let t_i be the last time agent i transmitted its measurement. We define $\hat{\mathbf{x}}_t \in \mathcal{X}$ as

$$\hat{\mathbf{x}}_t := (\mathbf{x}_{t_1}(1), \mathbf{x}_{t_2}(2), ..., \mathbf{x}_{t_n}(n))\,. \tag{1}$$

That is, $\hat{\mathbf{x}}_t$ is the last known state corresponding to the collection of agent states last transmitted, at time t. Then, the problem considered in this work is as follows.

Problem 1. *Consider a c-MMDP with a set of optimal shared state policies Π^*. Synthesise strategies that minimise the communication events between agents and construct distributed policies $\hat{\Pi}$ that keep the expected reward within some bounds of the optimal rewards, these bounds being a function of design parameters.*

4 Efficient Communication Strategies

To solve the problem of minimizing communication, we can first consider a scenario where agents can request state measurements from other agents. Consider a c-MMDP where agents have optimal policies Π^*. If agents are allowed to request state observations from other agents at their discretion, a possible algorithm to reduce the communication when agents execute their optimal policies is to use sets of neighbouring states $\mathcal{D} : \mathcal{X} \to 2^{\mathcal{X}}$ such that $\mathcal{D}(\mathbf{x}) = \{\mathbf{x}' : \|\mathbf{x} - \mathbf{x}'\| \leq d\}$ for some maximum distance d. Agents could compute such sets for each point

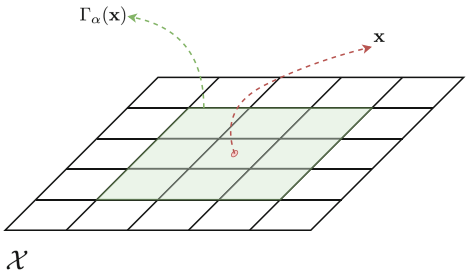

Fig. 1. Robustness surrogate representation.

in space, and request information from others only if the optimal action changes for any state $\mathbf{x}' \in \mathcal{D}(\mathbf{x})$. This approach, however, is not practical on a large scale multi agent system. First, it requires agents to request information, which could already be considered a communication event and therefore not always desirable. Additionally, computing the sets "on the fly" has a complexity of $\mathcal{O}(|\mathcal{X}|^2)$ in the worst case, and it has to be executed at every time-step by all agents. We therefore propose an approach to reduce communication in a MARL system where agents do not need to request information, but instead send messages (or not) to other agents based on some triggering rule.

4.1 Event-Driven Interactions

To construct an efficient communication strategy based on a distributed triggering approach, let us first define a few useful concepts. In order to allow agents to decide when is it necessary to transmit their own state measurements, we define the *robustness indicator* $\Gamma : \mathcal{X} \to \mathbb{R}_{\geq 0}$ as follows.

Definition 3. *For a c-MMDP with optimal global* $Q^* : \mathcal{X} \times \mathcal{U}^n \to \mathbb{R}$*, we define the* robustness surrogate $\Gamma_\alpha : \mathcal{X} \to \mathbb{R}_{\geq 0}$ *with sensitivity parameter* $\alpha \in \mathbb{R}_{\geq 0}$ *as:*

$$\Gamma_\alpha(\mathbf{x}) := \max\{d \,|\, \forall \mathbf{x}' : \|\mathbf{x}' - \mathbf{x}\|_\infty \leq d \Rightarrow \\ \Rightarrow Q^*(\mathbf{x}', \Pi^*(\mathbf{x})) \geq V^*(\mathbf{x}') - \alpha\}.$$

The function Γ_α gives a maximum distance (in the sup-norm) such that for any state $\mathbf{x}'$ which is Γ_α close to $\mathbf{x}$ guarantees the action $\Pi^*(\mathbf{x})$ has a Q value which is α close to the optimal value in $\mathbf{x}'$. A representation can be seen in Fig. 1. Computing the function Γ_α in practice may be challenging, and we cover this in detail in following sections.

Proposition 1. *Consider a c-MMDP communicating and acting according to Algorithm 1. Let* $\hat{\mathbf{x}}_t^i$ *be the last known joint state stored by agent* i *at time* t*, and* $\mathbf{x}_t$ *be the true state at time* t*. Then, it holds:*

$$\hat{\mathbf{x}}_t^i = \hat{\mathbf{x}}_t \;\; \forall i \in \mathcal{N}, \tag{2}$$

$$\|\hat{\mathbf{x}}_t - \mathbf{x}_t\|_\infty \leq \Gamma_\alpha(\hat{\mathbf{x}}_t) \;\; \forall t. \tag{3}$$

Algorithm 1. Self-Triggered state sharing

Initialise $\mathcal{N}$ agents at $\mathbf{x}_0$;
Initialise last-known state vector $\hat{\mathbf{x}}_0 = \mathbf{x}_0, \ \ i \in \mathcal{N}$
$t = 0$,
while $t < t_{max}$ **do**
  for $i \in \mathcal{N}$ **do**
    if $\|\mathbf{x}_t(i) - \hat{\mathbf{x}}_{t-1}(i)\|_\infty > \Gamma_\alpha(\hat{\mathbf{x}}_{t-1})$ **then**
      $\hat{\mathbf{x}}_t(i) \leftarrow \mathbf{x}_t(i)$
      Send updated $\hat{\mathbf{x}}_t(i)$ to all $\mathcal{N}_{-i}$;
    end if
    Execute action $\hat{U}_i^* = \pi_i^*(\hat{\mathbf{x}})$;
  end for
  $t++$;
end while

Proof. Properties (2) and (3) hold by construction. First, all agents update their own $\hat{\mathbf{x}}_t^i$ based on the received communication, therefore all have the same last-known state. Second, whenever the condition $\|\mathbf{x}_t(i) - \hat{\mathbf{x}}_{t-1}(i)\|_\infty > \Gamma_\alpha(\hat{\mathbf{x}})$ is violated, agent i transmits the new state measurement to others, and $\hat{\mathbf{x}}_t$ is updated. Therefore $\|\mathbf{x}_t - \hat{\mathbf{x}}_t\|_\infty > \Gamma_\alpha(\hat{\mathbf{x}}_t)$ holds for all times. □

Now let us use $\hat{r}_t = r(\mathbf{x}_t, \mathit{\Pi}^*(\hat{\mathbf{x}}_t), \mathbf{x}_{t+1})$ as the reward obtained when using the delayed state $\hat{\mathbf{x}}_t$ as input for the optimal policies. We then present the following result.

Theorem 1. *Consider a c-MMDP and let agents apply Algorithm 1 to update the delayed state vector $\hat{\mathbf{x}}_t$. Then it holds $\forall \mathbf{x}_0 \in \mathcal{X}$:*

$$E_{\mathbf{x}_0}[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{x}_t, \mathit{\Pi}^*(\hat{\mathbf{x}}_t), \mathbf{x}_{t+1})] \geq V^*(\mathbf{x}_0) - \alpha \frac{\gamma}{1-\gamma}.$$

Proof. From Proposition 1, $\|\mathbf{x}_t - \hat{\mathbf{x}}_t\|_\infty \leq \Gamma_\alpha(\hat{\mathbf{x}}_t) \, \forall t$, and recalling the expression for the optimal Q values:

$$Q^*(\mathbf{x}_t, \mathit{\Pi}^*(\hat{\mathbf{x}}_t)) = E_{\mathbf{x}_t}[\hat{r}_t + \gamma V^*(\mathbf{x}_{t+1})] \geq V^*(\mathbf{x}_t) - \alpha. \tag{4}$$

Now let $\hat{V}(\mathbf{x}_0) := E_{\mathbf{x}_0}[\sum_{t=0}^{\infty} \gamma^t \hat{r}_t]$ be the value of the policy obtained from executing the actions $\mathit{\Pi}^*(\hat{\mathbf{x}}_t)$. Then:

$$\begin{aligned} E_{\mathbf{x}_0}[\sum_{t=0}^{\infty} \gamma^t \hat{r}_t] &= E_{\mathbf{x}_0}[\hat{r}_0 + \gamma V^{\hat{\mathit{\Pi}}}(\mathbf{x}_1)] \\ &= E_{\mathbf{x}_0}[\hat{r}_0 + \gamma \hat{V}(\mathbf{x}_1) + \gamma V^*(\mathbf{x}_1) - \gamma V^*(\mathbf{x}_1)] \\ &= E_{\mathbf{x}_0}[\hat{r}_0 + \gamma V^*(\mathbf{x}_1)] + \gamma E_{\mathbf{x}_0}[\hat{V}(\mathbf{x}_1) - V^*(\mathbf{x}_1)]. \end{aligned} \tag{5}$$

Then, substituting (4) in (5):

$$E_{\mathbf{x}_0}[\sum_{t=0}^{\infty} \gamma^t \hat{r}_t] \geq V^*(\mathbf{x}_0) - \alpha + \gamma E_{\mathbf{x}_0}[\hat{V}(\mathbf{x}_1) - V^*(\mathbf{x}_1)]. \tag{6}$$

Now, observe we can apply the same principle as in (5) for the last term in (6),

$$\begin{aligned}
&\hat{V}(\mathbf{x}_1) - V^*(\mathbf{x}_1) = E_{\mathbf{x}_1}[\hat{r}_1 + \gamma\hat{V}(\mathbf{x}_2)] - V^*(\mathbf{x}_1) \\
&= Q^*(\mathbf{x}_1, \Pi^*(\hat{\mathbf{x}}_1)) + \gamma E_{\mathbf{x}_1}[\hat{V}(\mathbf{x}_2) - V^*(\mathbf{x}_2)] - V^*(\mathbf{x}_1) \\
&\geq V^*(\mathbf{x}_1) - \alpha - V^*(\mathbf{x}_1) + \gamma E_{\mathbf{x}_1}[\hat{V}(\mathbf{x}_2) - V^*(\mathbf{x}_2)] \\
&= -\alpha + \gamma E_{\mathbf{x}_1}[\hat{V}(\mathbf{x}_2) - V^*(\mathbf{x}_2)].
\end{aligned} \tag{7}$$

Substituting (7) in (6):

$$E_{\mathbf{x}_0}[\sum_{t=0}^{\infty}\gamma^t\hat{r}_t] \geq V^*(\mathbf{x}_0) - \alpha - \gamma\alpha + \gamma^2 E_{\mathbf{x}_0}[E_{\mathbf{x}_1}[\hat{V}(\mathbf{x}_2) - V^*(\mathbf{x}_2)]]. \tag{8}$$

Now it is clear that, applying (7) recursively:

$$\begin{aligned}
&E_{\mathbf{x}_0}[\sum_{t=0}^{\infty}\gamma^t\hat{r}_t] \geq V^*(\mathbf{x}_0) - \alpha \\
&- \gamma\alpha + \gamma^2 E_{\mathbf{x}_0}[E_{\mathbf{x}_1}[\hat{V}(\mathbf{x}_2) - V^*(\mathbf{x}_2)]] \geq V^*(\mathbf{x}_0) - \alpha\sum_{k=0}^{\infty}\gamma^k.
\end{aligned} \tag{9}$$

Substituting $\sum_{k=0}^{\infty}\gamma^k = \frac{\gamma}{1-\gamma}$ in (9):

$$E_{\mathbf{x}_0}[\sum_{t=0}^{\infty}\gamma^t\hat{r}_t] \geq V^*(\mathbf{x}_0) - \alpha\frac{\gamma}{1-\gamma}.$$

□

5 Robustness Surrogate and Its Computation

The computation of the robustness surrogate Γ_α may not be straight forward. When the state-space of the c-MMDP is metric, we can construct sets of neighbouring states for a given $\mathbf{x}$. Algorithm 2 produces an exact computation of the robustness surrogate Γ_α for a given c-MMDP and point $\mathbf{x}$. Observe, in the worst case, Algorithm 2 has a complexity of $O(|\mathcal{X}|)$ to compute the function $\Gamma_\alpha(\mathbf{x})$ for a single point $\mathbf{x}$. If this needs to be computed across the entire state-space, it explodes to an operation of worst case complexity $O(|\mathcal{X}|^2)$. In order to compute such functions more efficiently while retaining probabilistic guarantees, we can make use of the Scenario Approach for regression problems [5].

5.1 Learning the Robustness Surrogate with the Scenario Approach

The data driven computation of the function Γ_α can be proposed in the terms of the following optimization program. Assume we only have access to a uniformly sampled set $\mathcal{X}_S \subset \mathcal{X}$ of size $|\mathcal{X}_S| = S$. Let $\hat{\Gamma}_\alpha^\theta$ be an approximation of the

Algorithm 2. Computation of Robustness Indicator

Initialise $\mathbf{x}$.
Initialise $d = 1$.
Done = *False*
while Not Done **do**
  Compute Set $\mathcal{X}^d := \{\mathbf{x}' : \|\mathbf{x} - \mathbf{x}'\| = d\}$;
  if $\exists \mathbf{x}' \in \mathcal{X}^d : Q^*(\mathbf{x}', \Pi^*(\mathbf{x})) \leq V^*(\mathbf{x}') - \alpha$ **then**
    Done = *True*
  else $d++$
  end if
end while
$\Gamma_\alpha(\mathbf{x}) = d - 1$

real robustness surrogate parametrised by θ. To apply the scenario approach optimization, we need $\hat{\Gamma}_\alpha^\theta$ to be convex with respect to θ. For this we can use a Support Vector Regression (SVR) model, and embed the state vector in a higher dimensional space trough a feature non-linear map $\phi(\cdot)$ such that $\phi(\mathbf{x})$ is a feature vector, and we use the kernel $k(\mathbf{x}_1, \mathbf{x}_2) = \langle \phi(\mathbf{x}_1), \phi(\mathbf{x}_2) \rangle$. Let us consider sampled pairs $\{(\mathbf{x}_s, y_s)\}_S$, with $y_s = \Gamma_\alpha(\mathbf{x}_s)$ computed through Algorithm 2. Then, we propose solving the following optimization problem with parameters $\tau, \rho > 0$:

$$\begin{aligned} \min_{\substack{\theta \in \mathcal{X}, \kappa \geq 0, b \in \mathbb{R}, \\ \xi_i \geq 0, i=1,2,\ldots,S}} & \left(\kappa + \tau \|\theta\|^2\right) + \rho \sum_{i=1}^{S} \xi_i, \\ s.t. \quad & |y_i - k(\theta, \mathbf{x}_i) - b| - \kappa \leq \xi_i, \quad i = 1, \ldots, S. \end{aligned} \tag{10}$$

The solution to the optimization problem (10) yields a trade-off between how many points are outside the *prediction tube* $|y - k(\theta^*, \mathbf{x}_i) - b^*| < \kappa^*$ and how large the tube is (the value of κ^*). Additionally, the parameter ρ enables us to tune how much we want to penalise sample points being out of the prediction tube. Now take $(\theta^*, \kappa^*, b^*, \xi_i^*)$ as the solution to the optimization problem (10). Then, the learned robustness surrogate function will be:

$$\hat{\Gamma}_\alpha^{\theta^*} := k(\theta^*, \mathbf{x}_i) + b^*.$$

From Theorem 3 [5], it then holds for a sample of points $\mathcal{X}_S$ and a number of outliers $s^* := |\{(\mathbf{x}, y) \in \mathcal{X}_S : |y - k(\theta^*, \mathbf{x}) - b^*| > \kappa^*\}|$:

$$\overset{S}{\Pr}\left\{\underline{\epsilon}(s^*) \leq \Pr\left\{\mathbf{x} : \left|\Gamma_\alpha(\mathbf{x}) - \hat{\Gamma}_\alpha^{\theta^*}(\mathbf{x})\right| > \kappa^*\right\} \leq \bar{\epsilon}(s^*)\right\} \geq 1 - \beta \tag{11}$$

where $\underline{\epsilon}(s^*) := \max\{0, 1 - \bar{t}(s^*)\}$, $\bar{\epsilon}(s^*) := 1 - \underline{t}(s^*)$, and $\bar{t}(s^*), \underline{t}(s^*)$ are the solutions to the polynomial

$$\binom{S}{s^*} t^{S-s^*} - \frac{\beta}{2S} \sum_{i=k}^{S-1} \binom{i}{s^*} t^{i-k} - \frac{\beta}{6S} \sum_{i=S+1}^{4S} \binom{i}{s^*} t^{i-s^*} = 0.$$

Now observe, in our case we would like $\Gamma_\alpha(\mathbf{x}) \geq \hat{\Gamma}_\alpha^{\theta^*}(\mathbf{x})$ to make sure we are never over-estimating the robustness values. Then, with probability larger than $1-\beta$:

$$\begin{aligned}\bar{\epsilon}(s^*) &\geq \Pr\left\{\mathbf{x} : \left|\Gamma_\alpha(\mathbf{x}) - \hat{\Gamma}_\alpha^{\theta^*}(\mathbf{x})\right| > \kappa^*\right\} \geq \Pr\left\{\mathbf{x} : \Gamma_\alpha(\mathbf{x}) - \hat{\Gamma}_\alpha^{\theta^*}(\mathbf{x}) < -\kappa^*\right\} \\ &= \Pr\left\{\mathbf{x} : \Gamma_\alpha(\mathbf{x}) < \hat{\Gamma}_\alpha^{\theta^*}(\mathbf{x}) - \kappa^*\right\}.\end{aligned} \tag{12}$$

Therefore, taking $\|\mathbf{x}_t(i) - \hat{\mathbf{x}}_{t-1}(i)\|_\infty > \hat{\Gamma}_\alpha^{\theta^*}(\hat{\mathbf{x}}_{t-1}) - \kappa^*$ as the condition to transmit state measurements for each agent, we know that the probability of using an over-estimation of the true value $\Gamma_\alpha(\mathbf{x}_t)$ is at most $\bar{\epsilon}(s^*)$ with confidence $1-\beta$.

Then, let $\{\hat{U}_t\}$ be the sequence of joint actions taken by the system. The probability of U_t violating the condition $Q^*(\mathbf{x}_t, U_t) \geq V^*(\mathbf{x}_t) - \alpha$ for any $\mathbf{x}_t \in \mathcal{X}$ is at most $\bar{\epsilon}(s^*)$. Then, we can extend the results from Theorem 1 for the case where we use a SVR approximation as a robustness surrogate. Define the worst possible suboptimality gap $\iota := \max_{\mathbf{x},U} |V^*(\mathbf{x}) - Q^*(\mathbf{x}, U)|$.

Corollary 1. *Let $\hat{\Gamma}_\alpha^{\theta^*}$ obtained from (10) from collection of samples $\mathcal{X}_S$. Then, a c-MMDP communicating according to Algorithm 1 using as trigger condition $\|\mathbf{x}_t(i) - \hat{\mathbf{x}}_{t-1}(i)\|_\infty > \hat{\Gamma}_\alpha^{\theta^*}(\hat{\mathbf{x}}_{t-1}) - \kappa^*$ yields, with probability higher than $1-\beta$:*

$$E_{\mathbf{x}_0}[\sum_{t=0}^{\infty} \gamma^t \hat{r}_t] \geq V^*(\mathbf{x}_0) - \delta,$$

with $\delta := (\alpha + \bar{\epsilon}(s^*)(\iota - \alpha))\frac{\gamma}{1-\gamma}$.

Proof. Take expression (12), and consider the action sequence executed by the c-MMDP to be $\{\hat{U}_t\}_{t=0}^{\infty}$. We can bound the total expectation of the sum of rewards by considering $\{\hat{U}_t\}_{t=0}^{\infty}$ to be a sequence of random variables that produce $Q^*(\mathbf{x}_t, \hat{U}_t) \geq V^*(\mathbf{x}_t) - \alpha$ with probability $1 - \bar{\epsilon}(s^*)$, and $Q^*(\mathbf{x}_t, \hat{U}_t) \geq V^*(\mathbf{x}_t) - \iota$ with probability $\bar{\epsilon}(s^*)$. Then,

$$\begin{aligned}&E_{\mathbf{x}_0}[\sum_{t=0}^{\infty} \gamma^t \hat{r}_t] = E[E_{\mathbf{x}_0}[\sum_{t=0}^{\infty} \gamma^t \hat{r}_t | \{\hat{U}_t\}]] = E[E_{\mathbf{x}_0}[\hat{r}_0 + \gamma \hat{V}(\mathbf{x}_1) | \{\hat{U}_t\}]] \\ &= E[E_{\mathbf{x}_0}[\hat{r}_0 + \gamma V^*(\mathbf{x}_1) | \{\hat{U}_t\}] + \gamma E_{\mathbf{x}_0}[\hat{V}(\mathbf{x}_1) - V^*(\mathbf{x}_1) | \{\hat{U}_t\}]].\end{aligned} \tag{13}$$

Observe now, for the first term in (13):

$$\begin{aligned}&E[E_{\mathbf{x}_0}[\hat{r}_0 + \gamma V^*(\mathbf{x}_1) | \{\hat{U}_t\}]] \geq (1 - \bar{\epsilon}(s^*))(V^*(\mathbf{x}_0) - \alpha) \\ &+ \bar{\epsilon}(s^*)(V^*(\mathbf{x}_0) - \iota) = V^*(\mathbf{x}_0) - \alpha - \bar{\epsilon}(s^*)(\iota - \alpha).\end{aligned} \tag{14}$$

Take the second term in (13), and $\forall \mathbf{x}_1 \in \mathcal{X}$ given actions $\{\hat{U}_t\}$ it holds:

$$\begin{aligned}&E[\hat{V}(\mathbf{x}_1) - V^*(\mathbf{x}_1) | \{\hat{U}_t\}] \geq E[\hat{r}_1 + \gamma V^*(\mathbf{x}_2) + \gamma(\hat{V}(\mathbf{x}_2)) - V^*(\mathbf{x}_2)) \\ &- V^*(\mathbf{x}_1) | \{\hat{U}_t\}] \geq -\alpha - \bar{\epsilon}(s^*)(\iota - \alpha) + \gamma E[\hat{V}(\mathbf{x}_2) - V^*(\mathbf{x}_2)].\end{aligned}$$

Therefore, we can write

$$\begin{aligned}&\gamma E[E_{\mathbf{x}_0}[\hat{V}(\mathbf{x}_1) - V^*(\mathbf{x}_1)|\{\hat{U}_t\}]] = \gamma E_{\mathbf{x}_0}[E[\hat{V}(\mathbf{x}_1) - V^*(\mathbf{x}_1)|\{\hat{U}_t\}]] \\ &\geq \gamma \left(-\alpha - \bar{\epsilon}\,(s^*)\,(\iota - \alpha) + \gamma E_{\mathbf{x}_0}[E[\hat{V}(\mathbf{x}_2) - V^*(\mathbf{x}_2)]]\right)\end{aligned} \tag{15}$$

At last, substituting (14) and (15) in (13):

$$E_{\mathbf{x}_0}[\sum_{t=0}^{\infty} \gamma^t \hat{r}_t] \geq V^*(\mathbf{x}_0) - (\alpha + \bar{\epsilon}\,(s^*)\,(\iota - \alpha))\frac{\gamma}{1-\gamma}. \tag{16}$$

□

We can interpret the results of Corollary 1 in the following way. When using the exact function Γ_α, the sequence of actions produced ensures that, at all times, an action is picked such that the expected sum of rewards is always larger than some bound close to the optimal. When using the approximated $\hat{\Gamma}_\alpha^{\theta^*}$, however, we obtain from the scenario approach a maximum probability of a real point not satisfying the design condition: $\|\mathbf{x} - \mathbf{x}'\| \leq \hat{\Gamma}_\alpha^{\theta^*} - \kappa^* \wedge Q^*(\mathbf{x}', \Pi^*(\mathbf{x})) < V^*(\mathbf{x}') - \alpha$. When this happens during the execution of the c-MMDP policies it means that the agents are using delayed state information for which they do not have guarantees of performance, and the one-step-ahead value function can deviate by the worst sub-optimality gap ι.

6 Experiments

We set out now to quantify experimentally the impact of Algorithm 1 on the performance and communication requirements of a benchmark c-MMDP system. First of all, it is worth mentioning that the comparison of the proposed algorithm with existing work is not possible since, to the best of our knowledge, no previous work has dealt with the problem of reducing communication when executing learned policies in a c-MMDP system. For this reason, the results are presented such that the performance of different scenarios (in terms of different Γ_α functions) is compared with the performance of an optimal policy with continuous communication.

6.1 Benchmark: Collaborative Particle-Tag

We evaluate the proposed solution in a typical particle tag problem (or predator-prey) [20]. We consider a simple form of the problem with 2 predators and 1 prey. The environment is a 10×10 arena with discrete states, and the predators have the actions $\mathcal{U}_i = \{\text{up}, \text{down}, \text{left}, \text{right}, \text{wait}\}$ available at each time step and can only move one position at a time. The environment has no obstacles, and the prey can move to any of the 8 adjacent states after each time step. The predators get a reward of 1 when, being in adjacent states to the prey, *both* choose to move into the prey's position (tagging the prey). They get a reward of -1 when they move

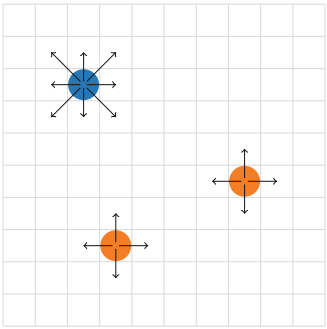

Fig. 2. Particle Tag, Predators in orange, Prey in blue. (Color figure online)

into the same position (colliding), and a reward of 0 in all other situations. A representation of the environment is presented in Fig. 2. The global state is then a vector $\mathbf{x}_t \in \{0, 1, 2, ..., 9\}^6$, concatenating the x, y position of both predators and prey. For the communication problem, we assume each agent is only able to measure its own position and the prey's. Therefore, in order to use a joint state based policy $\pi_i : \{0, 1, 2, ..., 9\}^6 \to \mathcal{U}$, at each time-step predators are required to send its own position measurement to each other.

6.2 Computation of Robustness Surrogates

With the described framework, we first compute the optimal Q^* function using a fully cooperative vanilla Q-learning algorithm [3], by considering the joint state and action space, such that $Q : \{0, 1, 2, ..., 9\}^6 \times \mathcal{U}^2 \to \mathbb{R}$. The function was computed using $\gamma = 0.97$. We then take the joint optimal policy as $\Pi^*(\mathbf{x}) = \text{argmax}_U Q^*(\mathbf{x}, U)$, and load in each predator the corresponding projection π_i^*. To evaluate the trade-off between expected rewards and communication events, we compute the function $\hat{\Gamma}_\alpha$ by solving an SVR problem as described in (10) for different values of sensitivity α. Then, the triggering condition for agents to communicate their measurements is $\|\mathbf{x}_t(i) - \hat{\mathbf{x}}_{t-1}(i)\|_\infty > \hat{\Gamma}_\alpha^{\theta^*}(\mathbf{x}) - \kappa^*$.

The hyper-parameters for the learning of the SVR models are picked through heuristics, using a sample of size $S = 10^4$ to obtain reasonable values of mis-predicted samples s^* and regression mean-squared error scores. Note that $S = \frac{1}{100}|\mathcal{X}|$. To estimate the values $\bar{\epsilon}(s^*)$, a coefficient of $\beta = 10^{-3}$ was taken, and the values were computed using the code in [11]. For more details on the computation of ρ-SVR models (or μ–SVR) [27] see the project code[1].

Figure 3 shows a representation of the obtained SVR models for different values of α, plotted over a 2D embedding of a subset of state points using a t-SNE [21] embedding. It can be seen how for larger α values, more "robust" regions appear, and with higher values of Γ_α. This illustrates how, when increasing the sensitivity, the obtained approximated $\hat{\Gamma}_\alpha^{\theta^*}$ take higher values almost everywhere in the state space, and form clusters of highly robust points.

[1] https://github.com/danieljarne/Event-Driven-MARL.

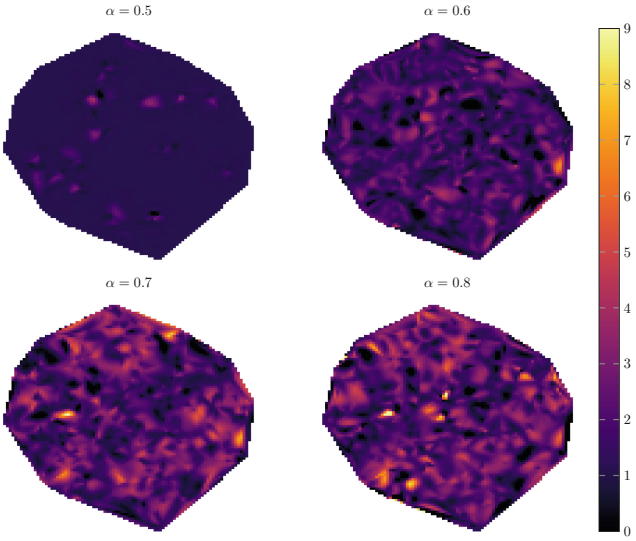

Fig. 3. Obtained $\hat{\Gamma}_{\alpha}^{\theta^*}$ models on a 2D embedding.

6.3 Results

The results are presented in Table 1. We simulated in each case 1000 independent runs of 100 particle tag games, and computed the cumulative reward, number of communication events and average length of games. For the experiments, $\hat{E}[\cdot]$ is the expected value approximation (mean) of the cumulative reward over the 1000 trajectories, and in every entry we indicate the standard deviation of the samples (after $\pm$). We use $\mathcal{T}_\alpha$ as the generated trajectories (games) for a corresponding parameter α, $h(\mathcal{T}_\alpha)$ as the total sum of communication events per game for a collection of games, and $\overline{g} := \sum_{g \in \mathcal{T}_\alpha} \frac{|g|}{|\mathcal{T}_\alpha|}$ as the average length of a game measured over the collected $\mathcal{T}_\alpha$. For the obtained Q^* function, the worst optimality gap is computed to be $\iota = 1.57$.

Table 1. Simulation results

α	$\hat{E}[\sum_{t=0}^{\infty} \gamma^t r]$	$\overline{g}$	$h(\mathcal{T}_\alpha)$	$\frac{h(\mathcal{T}_\alpha)}{\overline{g}}$	$\overline{\epsilon}(s^*)$	δ
0	**2.72** $\pm$ 0.50	**10.72** $\pm$ 0.44	**21.49** $\pm$ 0.89	**2.00**	-	-
0.4	**2.72** $\pm$ 0.53	**10.77** $\pm$ 0.44	**19.13** $\pm$ 0.87	**1.78**	0.079	16.33
0.5	**1.62** $\pm$ 0.92	**12.45** $\pm$ 0.58	**16.75** $\pm$ 0.85	**1.35**	0.148	22.39
0.6	**0.99** $\pm$ 1.09	**13.71** $\pm$ 0.71	**14.49** $\pm$ 0.87	**1.06**	0.205	26.61
0.7	**0.93** $\pm$ 1.08	**13.74** $\pm$ 0.69	**14.09** $\pm$ 0.85	**1.03**	0.117	26.85
0.8	**0.74** $\pm$ 1.10	**14.65** $\pm$ 0.80	**14.11** $\pm$ 0.87	**0.96**	0.075	28.10
0.9	**0.64** $\pm$ 1.08	**14.82** $\pm$ 0.83	**14.33** $\pm$ 0.88	**0.96**	0.097	31.69

Let us remark the difference between the 4th and 5th column in Table 1. The metric $h(\mathcal{T}_\alpha)$ is a direct measure of amount of messages for a given value of α. However, note that we are simulating a fixed number of games, and the average number of steps per game increases with α: the lack of communication causes the agents to take longer to solve the game. For this reason we add the metric $h(\mathcal{T}_\alpha)/\overline{g}$, which is a measure of total amount of messages sent versus amount of simulation steps for a fixed α (i.e. total amount of steps where a message *could* be sent). Broadly speaking, $h(\mathcal{T}_\alpha)$ compares raw amount of information shared to solve a fixed amount of games, and $h(\mathcal{T}_\alpha)/\overline{g}$ compares amount of messages per time-step (information transmission rate). Note at last that there are two collaborative players in the game, therefore a continuous communication scheme would yield $h(\mathcal{T}_\alpha)/\overline{g} = 2$.

From the experimental results we can get an qualitative image of the trade-off between communication and performance. Larger α values yield a decrease in expected cumulative reward, and a decrease in state measurements shared between agents. Note finally that in the given c-MMDP problem, the minimum reward every time step is $\min r(\mathbf{x}_t, U, \mathbf{x}_{t+1}) = -1$, therefore a lower bound for the cumulative reward is $E[\sum_{t=0}^{\infty} \gamma^t r] \geq -1\frac{\gamma}{1-\gamma} = -32.333$. Then, the performance (even for the case with $\alpha = 0.9$ remains relatively close to the optimum computed with continuous communication.

At last, let us comment on the general trend observed regarding the values of α. Recall the bound obtained in Corollary 1, and observe that for $\alpha = 0.5 \Rightarrow \delta = 22.39$. On average, $\hat{E}[V^*(\mathbf{x}_0)] \approx 2.72$ when initialising $\mathbf{x}_0$ at random (as seen on Table 1). This yields a quite conservative bound of $\hat{E}[V^*(\mathbf{x}_0)] - \delta = -19.67$ on the expected sum of rewards, while the communication events are reduced by around 22% due to the conservative computation of Γ_α. One first source of conservativeness is in Algorithm 1. When computing the exact value $\Gamma_\alpha(\mathbf{x}) = d$, it requires every point $\mathbf{x}' : \|\mathbf{x} - \mathbf{x}'\|_\infty \leq d$ to satisfy the condition in Definition 3. The number of states to be checked grows exponentially with d, and many of those states may not even be reachable from $\mathbf{x}$ by following the MDP transitions. Therefore we are effectively introducing conservativeness in cases where probably, for many points $\mathbf{x}$, we could obtain much larger values $\Gamma_\alpha(\mathbf{x})$ if we could check the transitions in the MDP. Another source of conservativeness comes from the SVR learning process and in particular, the values of κ^*. Since the states are discretised, $\|\mathbf{x}_t(i) - \hat{\mathbf{x}}_{t-1}(i)\|_\infty \in \{0, 1, 2, 3, ..., 10\}$. Therefore, the triggering condition is effectively constrained to $\|\mathbf{x}_t(i) - \hat{\mathbf{x}}_{t-1}(i)\|_\infty > \lfloor \hat{\Gamma}_\alpha^{\theta^*}(\mathbf{x}) - \kappa^* \rfloor$, which makes it very prone to under-estimate even further the true values of $\Gamma_\alpha(\mathbf{x})$. Additionally, for most SVR models we obtained predictions $\hat{\Gamma}_\alpha^{\theta^*}(\mathbf{x}) - \kappa^*$ that are extremely close to the real value, so small deviations in κ^* can have a significant impact in the number of communications that are triggered "unnecessarily".

7 Discussion

We have presented an approach to reduce the communication required in a collaborative reinforcement learning system when executing optimal policies in real

time, while guaranteeing the discounted sum of rewards to stay within some bounds that can be adjusted through the parameters α and $\overline{\epsilon}(s^*)$ (this last one indirectly controlled by the learning of data driven approximations $\hat{\Gamma}_{\alpha}^{\theta^*}$). The guarantees were first derived for the case where we have access to *exact* robustness surrogates Γ_{α}, and extended to allow for surrogate functions learned through a *scenario approach* based SVR optimization. In the proposed experiments for a 2-player particle tag game the total communication was reduced between 10%–44% and the communication rate by 12%–52%, while keeping the expected reward sum $\hat{E}[\sum_{t=0}^{\infty}\gamma^t r] \in [0.68, 2.76]$. The conclusion to draw from this is that, in general, agents are able to solve the c-MMDP problem with delayed state information introduced through robustness surrogate triggers, resulting in a significant reduction in communication requirements. When extending these methods to larger number of agents, the main bottleneck is the computation of Γ_{α}. For many c-MMDPs the SVR representation may still be an efficient approach to compute them, but otherwise more efficient approaches (perhaps decentralised) could be devised.

The computation of the values $\Gamma_{\alpha}(\mathbf{x})$ and the learning of the SVR models for $\hat{\Gamma}_{\alpha}^{\theta^*}(\mathbf{x})$ introduced significant conservativeness with respect to the theoretical bounds. A possible improvement for future work could be to compute the true values $\Gamma_{\alpha}(\mathbf{x})$ through a Monte-Carlo based approach by sampling MDP trajectories. This would yield a much more accurate representation of how "far" agents can deviate without communicating, and the guarantees could be modified to include the possibility that the values $\Gamma_{\alpha}(\mathbf{x})$ are correct up to a certain probability. At last, we can come back now to the statements in Remark 1. It is now evident how having a certain physical structure in the MDP (*i.e.* transition probabilities being larger for states closer in space) would help mitigate the conservativeness. An MDP with large transition jumps with respect to the sup-norm will result in more conservative and less meaningful robustness surrogates.

Other problems that branch out of this work are the implications of learning robustness surrogate functions. These functions could be used to modify the agent policies, to sacrifice performance in favour of robustness versus communication faults or attacks. Finally, it would be insightful to compare the approaches presented in this work with ideas in the line of [10], where we can incorporate the communication as a binary action (to communicate or not) into the learning algorithm, to optimise simultaneously with the sum of rewards.

Acknowledgements. The authors want to thank Gabriel Gleizer, Giannis Delimpaltadakis and Andrea Peruffo for the useful and insightful discussions related to this work.

References

1. Ackley, D.H., Littman, M.L.: Altruism in the evolution of communication. In: Artificial Life IV, Cambridge, MA, pp. 40–48 (1994)
2. Bertsekas, D.: Distributed dynamic programming. IEEE Trans. Autom. Control **27**(3), 610–616 (1982)

3. Busoniu, L., Babuska, R., De Schutter, B.: A comprehensive survey of multiagent reinforcement learning. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **38**(2), 156–172 (2008)
4. Calafiore, G.C., Campi, M.C.: The scenario approach to robust control design. IEEE Trans. Autom. Control **51**(5), 742–753 (2006)
5. Campi, M.C., Garatti, S.: Scenario optimization with relaxation: a new tool for design and application to machine learning problems. In: 2020 59th IEEE Conference on Decision and Control (CDC), pp. 2463–2468. IEEE (2020)
6. Chen, T., Zhang, K., Giannakis, G.B., Basar, T.: Communication-efficient distributed reinforcement learning. arXiv preprint arXiv:1812.03239 (2018)
7. Da Silva, F.L., Hernandez-Leal, P., Kartal, B., Taylor, M.E.: Uncertainty-aware action advising for deep reinforcement learning agents. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 5792–5799 (2020)
8. Dimarogonas, D.V., Frazzoli, E., Johansson, K.H.: Distributed event-triggered control for multi-agent systems. IEEE Trans. Autom. Control **57**(5), 1291–1297 (2012). https://doi.org/10.1109/TAC.2011.2174666
9. Elvis Tsang, K.F., Johansson, K.H.: Distributed event-triggered learning-based control for nonlinear multi-agent systems. In: 2021 60th IEEE Conference on Decision and Control (CDC), pp. 3399–3405 (2021). https://doi.org/10.1109/CDC45484.2021.9683215
10. Foerster, J.N., Assael, Y.M., De Freitas, N., Whiteson, S.: Learning to communicate with deep multi-agent reinforcement learning. arXiv preprint arXiv:1605.06676 (2016)
11. Garatti, S., Campi, M.C.: Risk and complexity in scenario optimization. Math. Program. **191**, 243–279 (2019). https://doi.org/10.1007/s10107-019-01446-4
12. George, J., Gurram, P.: Distributed stochastic gradient descent with event-triggered communication. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 7169–7178 (2020)
13. Hu, J., Wellman, M.P., et al.: Multiagent reinforcement learning: theoretical framework and an algorithm. In: ICML, vol. 98, pp. 242–250. Citeseer (1998)
14. Karabag, M.O., Neary, C., Topcu, U.: Planning not to talk: multiagent systems that are robust to communication loss (2022)
15. Kim, D., et al.: Learning to schedule communication in multi-agent reinforcement learning. arXiv preprint arXiv:1902.01554 (2019)
16. Kok, J.R., Vlassis, N.: Sparse cooperative Q-learning. In: Proceedings of the Twenty-First International Conference on Machine Learning, p. 61 (2004)
17. Lillicrap, T.P., et al.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
18. Lin, J., Dzeparoska, K., Zhang, S.Q., Leon-Garcia, A., Papernot, N.: On the robustness of cooperative multi-agent reinforcement learning. In: 2020 IEEE Security and Privacy Workshops (SPW), pp. 62–68. IEEE (2020)
19. Lin, Y., et al.: A communication-efficient multi-agent actor-critic algorithm for distributed reinforcement learning. In: 2019 IEEE 58th Conference on Decision and Control (CDC), pp. 5562–5567. IEEE (2019)
20. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments. arXiv preprint arXiv:1706.02275 (2017)
21. Van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**(11), 2579–2605 (2008)

22. Mazo, M., Tabuada, P.: On event-triggered and self-triggered control over sensor/actuator networks. In: 2008 47th IEEE Conference on Decision and Control, pp. 435–440. IEEE (2008)
23. Mnih, V., et al.: Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
24. Nowé, A., Vrancx, P., De Hauwere, Y.M.: Game theory and multi-agent reinforcement learning. In: Wiering, M., van Otterlo, M. (eds.) Reinforcement Learning. ALO, vol. 12, pp. 441–470. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27645-3_14
25. Panait, L., Luke, S.: Cooperative multi-agent learning: the state of the art. Auton. Agent. Multi-Agent Syst. **11**(3), 387–434 (2005). https://doi.org/10.1007/s10458-005-2631-2
26. Sahoo, A., Xu, H., Jagannathan, S.: Neural network-based event-triggered state feedback control of nonlinear continuous-time systems. IEEE Trans. Neural Netw. Learn. Syst. **27**(3), 497–509 (2015)
27. Schölkopf, B., Bartlett, P., Smola, A., Williamson, R.C.: Shrinking the tube: a new support vector regression algorithm. In: Advances in Neural Information Processing Systems, vol. 11 (1998)
28. Sen, S., Sekaran, M., Hale, J., et al.: Learning to coordinate without sharing information. In: AAAI, vol. 94, pp. 426–431 (1994)
29. Solowjow, F., Trimpe, S.: Event-triggered learning. Automatica **117**, 109009 (2020)
30. Szer, D., Charpillet, F.: Improving coordination with communication in multi-agent reinforcement learning. In: 16th IEEE International Conference on Tools with Artificial Intelligence, pp. 436–440. IEEE (2004)
31. Tabuada, P.: Event-triggered real-time scheduling of stabilizing control tasks. IEEE Trans. Autom. Control **52**(9), 1680–1685 (2007)
32. Tan, M.: Multi-agent reinforcement learning: independent vs. cooperative agents. In: Proceedings of the Tenth International Conference on Machine Learning, pp. 330–337 (1993)
33. Vamvoudakis, K.G., Ferraz, H.: Model-free event-triggered control algorithm for continuous-time linear systems with optimal performance. Automatica **87**, 412–420 (2018)
34. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double Q-learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30 (2016)
35. Watkins, C.J., Dayan, P.: Q-learning. Mach. Learn. **8**(3–4), 279–292 (1992). https://doi.org/10.1007/BF00992698
36. Xue, W., Qiu, W., An, B., Rabinovich, Z., Obraztsova, S., Yeo, C.K.: Mis-spoke or mis-lead: achieving robustness in multi-agent communicative reinforcement learning. arXiv preprint arXiv:2108.03803 (2021)
37. Zhang, C., Lesser, V.: Coordinating multi-agent reinforcement learning with limited communication. In: Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems, pp. 1101–1108 (2013)
38. Zhong, X., Ni, Z., He, H., Xu, X., Zhao, D.: Event-triggered reinforcement learning approach for unknown nonlinear continuous-time system. In: 2014 International Joint Conference on Neural Networks (IJCNN), pp. 3677–3684. IEEE (2014)