Abstract—Consider a collaborative task carried out by
two autonomous agents that can communicate over a noisy
channel. Each agent is only aware of its own state, while
the accomplishment of the task depends on the value of
the joint state of both agents. As an example, both agents must
simultaneously reach a certain location of the environment,
while only being aware of their own positions. Assuming the
presence of feedback in the form of a common reward to the
agents, a conventional approach would apply separately:
(i) an off-the-shelf coding and decoding scheme in order
to enhance the reliability of the communication of the
state of one agent to the other; and (ii) a standard
multiagent reinforcement learning strategy to learn how
to act in the resulting environment. In this work, it is
argued that the performance of the collaborative task can
be improved if the agents learn how to jointly communicate
and act. In particular, numerical results for a baseline grid
world example demonstrate that the jointly learned policy
carries out compression and unequal error protection by
leveraging information about the action policy.

I. INTRODUCTION

Consider the rendezvous problem illustrated in Fig.
1 and Fig. 2. Two agents, e.g., members of a SWAT
team, need to arrive at the goal point in a grid world
at precisely the same time, while starting from arbitrary
positions. Each agent only knows its own position but
is allowed to communicate with the other agent over a
noisy channel. This set-up is an example of cooperative
multiple agent problems in which each agent has par-
tial information about the environment [1], [2]. In this
scenario, communication and coordination are essential
in order to achieve the common goal [3]–[5], and it is
not optimal to design the communication and control
strategies separately [5], [6].

Assuming the presence of a delayed and sparse com-
mon feedback signal that encodes the team reward,
cooperative multiagent problems can be formulated in
the framework of multiagent reinforcement learning.
As attested by the references [1], [2], [7] mentioned
above, as well by [8], [9], this is a well-studied and
active field of research. To overview some more recent
contributions, paper [10] presents simulation results for a
distributed tabular Q-learning scheme with instantaneous
communication. Deep learning approximation methods
are applied in [11] for Q-learning and in [12] for actor-
critic methods. In [13], a method is proposed that keeps
a centralized critic in the form of a Q-function during
the learning phase and uses a counter-factual approach
to carry out credit assignment for the policy gradients.

The works mentioned above assume a noiseless com-
munication channel between agents or use noise as a
form of regularization [9]. In contrast, in this paper, we
consider the problems of simultaneously learning how
to communicate on a noisy channel and how to act,
creating a bridge between the emerging literature on ma-
chine learning for communications [14] and multiagent
reinforcement learning. A closely related work is [15],
in which, however, the focus is on the joint optimization
of scheduling and actions in a multiagent system.

For a sequential two-agent reinforcement learning
problem, we formulate distributed Q-learning algorithms
that learn simultaneously what to communicate over the
inter-agent noisy channel and which actions to take in
the environment. As a numerical example, we consider
models with sequential or simultaneous communications
and actions. For the rendezvous problem illustrated in
Fig. 2, we provide numerical performance comparisons
between the proposed multiagent reinforcement learning
scheme and a conventional method based on the sepa-
rate optimization of action and communication policies.
While the proposed scheme jointly learns how to act
and communicate, the baseline conventional method ap-
pies separately an off-the-shelf channel coding scheme
for communication of an agent’s state and multiagent
reinforcement learning to adapt the action policies. The advantages of the jointly optimized policy are seen to result from data compression and unequal error protection mechanisms that are tailored by each agent to the action policy.

II. Problem Set-up

As illustrated in Fig. 1 and Fig. 2, we consider a cooperative multiagent system comprising of two agents that communicate over noisy channels. The system operates in discrete time, with agents taking actions and communicating in each time step \( t = 1, 2, \ldots \). While the approach can be applied to any multiagent system, in order to fix the ideas, we focus here on the rendezvous problem illustrated in Fig. 2. The two agents operate on an \( n \times n \) grid world and aim at arriving at the same time at the goal point on the grid. The position of each agent \( i \in \{1, 2\} \) on the grid determines its environment state \( s^e_i \in S^e = [n] \times [n], \) where \([n] = \{1, 2, \ldots, n\}\). Each agent \( i \) communicates its environment state \( s^c_i \in S^c \) to the other agent. Agents communicate over interference-free channels using binary signaling, and the channels between the two agents are independent Binary Symmetric Channels (BSCs), such that the received signal is given as

\[
s^c_{j,t+1} = a^c_i \oplus a^c_{j,t},
\]

where the XOR operation \( \oplus \) is applied element-wise, and \( a^c_{j,t} \) has independent identically distributed (i.i.d.) Bernoulli entries with bit flipping probability \( q \leq 0.5 \). Extensions to other channels are conceptually straightforward.

We first consider a scenario with simultaneous communication and actions. Accordingly, agent \( i \) follows a policy \( \pi_i \) that maps the observations \( s_t = \langle s^e_t, s^c_t \rangle \) of the agent into its actions \( a_t = \langle a^e_i, a^c_i \rangle \). The policy is generally stochastic, and we write it as the conditional probability \( \pi_i(a_t|s_t) \) of taking action \( a_t \) while in state \( s_t \). We assume the joint policy \( \pi \) to select both the environment action \( a^e_i \) and transmitted signal \( a^c_i \) based on the overall state \( s_t \). The overall joint policy \( \pi \) is given by the product \( \pi = \pi_1 \times \pi_2 \). It is noted that the assumed memoryless stationary policies are sub-optimal under partial individual observability of environment state \( s_t \). A model that assumes communications and actions will be covered in Sec. III. B.

At each time \( t \), given states \( \langle s_1, s_2 \rangle \) and actions \( \langle a_1, a_2 \rangle \), both agents receive a single team reward

\[
r_t = \begin{cases} 
R_1, & \text{if } s^e_i \neq s^e_j \in S^e_t \\
R_2, & \text{if } s^e_i = s^e_j \in S^e_t, \\
0, & \text{otherwise,}
\end{cases}
\]

where \( R_1 < R_2 \). Accordingly, when only one agent arrives at the target point \( s^e_t \), a smaller reward \( R_1 \) is obtained at the end of the episode, while the larger reward \( R_2 \) is attained when both agents visit the goal.
encourages the exploration of the state-action tuples that have been experienced fewer times.

The update of the action value function based on the available observations at time $t$ follows the off-policy Q-learning algorithm, i.e., [16]

$$Q_i^t(s_{i,t}, s_{e,t}^e, a_{i,t}^e, a_{e,t}^e) \leftarrow (1 - \alpha)Q_i^t(s_{i,t}, s_{e,t}^e, a_{i,t}^e, a_{e,t}^e) + \alpha\gamma\left(r_t + \max_{a_i^e,a_{e}^e}Q_i^t(s_{i,t+1}, s_{e,t+1}^e, a_{i,t+1}^e, a_{e,t+1}^e)\right),$$

where $\alpha > 0$ is a learning rate parameter. The full algorithm is detailed in Algorithm 1. At the end of the training process, policy $\pi_i(a_i^e, a_e^e|s_i^e, s_e^e)$ can be obtained by

$$\pi_i(a_i^e, a_e^e|s_i^e, s_e^e) = \delta\left((a_i^e, a_e^e) - \arg\max_{a_i^e,a_e^e}Q_i(s_i^e, s_e^e, a_i^e, a_e^e)\right),$$

where $\delta(\cdot)$ is the Dirac delta function.

As a baseline, we also consider a conventional scheme that optimizes communications and actions separately. For communication, each agent $i$ sends its environment state $s_i^e$ to the other agent by using a channel code for the given noisy channel. Note that compression is not possible, since all states are a priori equally likely. Agent $j$ obtains an estimate $\hat{s}_i^e$ of the environment state of $i$ by using a channel decoder based on the received signal. This estimate is used as if it were the correct position of the other agent to define the environment state-action value function $Q_j^e(\hat{s}_j^e, \hat{s}_i^e, a_j^e)$. This function is updated using Q-learning and the UCB policy in a manner similar to Algorithm 1.

**Algorithm 1** Learned Simultaneous Communications and Actions

1. **Input:** $\gamma$, $\alpha$, and $c$
2. **Initialize** all-zero Q-table $Q_i(s_i^e, s_e^e, a_i^e, a_e^e)$ and table $N_i(s_i^e, s_e^e, a_i^e, a_e^e)$, for $i = 1, 2$
3. **for** each episode $m = 1 : M$
4. Randomly initialize $(s_{i,1,t}^e, s_{2,t}^e) \notin S_T$
5. Randomly initialize $(s_{i,1,t}^e, s_{2,t}^e)\notin S_T$
6. set $t_m = 1$
7. **while** $(s_{i,1,t}, s_{2,t}^e) \notin S_T$
8. Jointly select $a_{i,t}^e = a_{i,t}^e \in A_i^c$ and $a_{e,t}^e = a_{e,t}^e \in A_i^c$ by solving (5), for $i = 1, 2$
9. Increment $N_i(s_i^e, s_e^e, a_i^e, a_e^e)$, for $i = 1, 2$
10. Obtain message $s_i^e, s_{e,t+1}^e$, for $i = 1, 2$
11. Obtain reward $r_t$ and move to $s_i^e, s_{e,t+1}^e$, for $i = 1, 2$
12. **for** $i = 1, 2$
13. Update $Q_i(s_i^e, s_{e,t}^e, a_i^e, a_e^e)$ by following (6)
14. end
15. $t_m = t_m + 1$
16. end
17. Compute $\sum_{t=1}^{t_m-1} \gamma^t r_t$ for the $m$th episode
18. end
19. **Output:** $\pi_i(a_i^e, a_e^e|s_i^e, s_e^e)$ by following (7) for $i = 1, 2$
B. Sequential Communications and Actions

In the problem formulation considered so far, agents select both environment and communication actions simultaneously, which inherently leads to a delay in the inter-agent communication. In fact, information embedded by agent $i$ in its communication action $a_{i,t}^c$ cannot be used earlier than in time step $t+1$ by the other agent $j \neq i$. As we have seen in Sec. II, this model can be formalized using the standard Markov Decision Process formulation. We now study an alternative set-up, in which communication is followed by the selection of an environment action at each time instant $t$. A similar model was considered in [17].

To elaborate, at each time $t$, each agent $i$ first observes its environment state $s_{i,t}^e$ and then selects a communication action $a_{i,t}^c$ by following a policy $\pi_{i,t}^c(s_{i,t}^e|a_{i,t}^c)$. In the second phase of time step $t$, agent $i$ receives the communication message $s_{i,t}^e$ over the channel. Finally, agent $i$ selects its environment action $a_{i,t}^e$ by following its policy $\pi_{i,t}^e(a_{i,t}^e|s_{i,t}^e, s_{i,t}^c)$. Both conventional communication and jointly learned communication schemes can be easily adapted to this communication model. We provide details for learned communication in Appendix.

IV. RESULTS AND DISCUSSIONS

In this section, we provide numerical results for the rendezvous problem described in Sec. II. As in Fig. 2, the grid world is of size $4 \times 4$, i.e. $n = 4$, and it contains one goal point at the right-top position. Environment states are numbered row-wise starting from the left-bottom as shown in Fig. 2(a). All the algorithms are run for 50 independent episodes. For each agent $i$ the initial state $s_{i,t=1}^e \notin S_n^e$ in each episode is drawn uniformly from all non-terminal states.

We compare the conventional communication and the learned communication schemes reviewed in the previous section. Conventional communication transmits the position of an agent on the grid as the 4-bit binary version of the indices in Fig. 2(a) after encoding via a binary cyclic $(B,4)$ code, with the generator polynomial $1 + X^2 + X^3$. The received message is then decoded by syndrome decoding.

In order to reduce the dimensions of the state-action space for learned simultaneous communications and actions, in our experiments, we use disjoint policies $\pi_i^e$ and $\pi_i^c$ to separately select environment and communication actions. Accordingly, given the received communication signal $s_{i,t}^e$ and the local environment state $s_{i,t}^e$, each agent $i$ selects its environment actions $a_{i,t}^e$ by following a policy $\pi_{i,t}^e(s_{i,t}^e|a_{i,t}^c)$ based on a state-action value function $Q_{i,t}^e(s_{i,t}^e, a_{i,t}^c)$, while it chooses its communication action $a_{i,t}^c$ by following a second policy $\pi_{i,t}^c$, based on a state-action function $Q_{i,t}^c(s_{i,t}^e, a_{i,t}^c)$.

Figure 3. Average return for conventional communication and learned communication when $B = 7$ with simultaneous actions and communications.

The performance of each scheme is evaluated in terms of the discounted return in (4), averaged over all epochs and smoothed using a moving average filter of memory equal to 4,000 episodes. The rewards in (2) are selected as $R_1 = 1$ and $R_2 = 3$, while the discount factor is $\gamma = 0.9$. A constant learning rate $\alpha = 0.15$ is applied, and the exploration rate $c$ of the UCB policy is selected from the set $\{0.3, 0.4, 0.5\}$ such that it maximizes the average return at the end of the episodes in an epoch.

We first investigate the impact of the channel noise by considering different values of the bit flip probability $q$ for simultaneous communications and actions. In Fig. 3 it is observed that conventional communication performs well at the low bit flipping rate of $q = 0.05$, but at higher rates of $q$ learned communication outperforms conventional communication after a sufficiently large number of episodes. Importantly, for $q = 0.2$, the performance of conventional communication degrades through episodes due to the accumulation of noise in the observations, while learned communication is seen to be robust against channel noise.

In Fig. 4, we consider the case of sequential communications and actions. In this case, for $q = 0$ conventional communication is optimal [1]. In contrast, for noisy channels, learned communication provides significant gain, e.g., 20% for $q = 0.15$ and $q = 0.20$.

We now discuss the reasons that underlie the performance advantages of learned communication. We start by analyzing the capability of learned communication to compress the environment state information before transmission. To obtain quantitative insights, we measure the mutual information $I(s_{i,t}^e; a_{i,t}^c)$ between the environment state $s_{i,t}^e$ and the communication action $a_{i,t}^c$ of an agent $i$ as obtained under the policy learned after 20,000 episodes for $q = 0, 0.05, 0.1, 0.15, 0.2$ [18]. Fig. 5 plots the mutual information as a function of the bit flipping probability $q$ for learned communication. For conventional communication scheme the communication
We can observe that compression is achieved by assigning same message to different locations. In this regard, it is interesting to note the interplay with the learned action policy: groups of states are clustered together if states have similar distance from the goal point, such as \( \{ (4, 3), (3, 4) \} \) and \( \{ (4, 2), (2, 4) \} \); or if they are very far from the goal point such as \( \{ (1, 2), (1, 3) \} \). Furthermore, it is seen that the Hamming distance of the selected messages depends on how critical it is to distinguish between the corresponding states. This is because it is important for an agent to realize whether the other agent is close to the terminal point.

V. CONCLUSIONS

In this paper, we have studied the problem of decentralized control of agents that communicate over a noisy channel. The results demonstrate that jointly learning communication and action policies can significantly outperform methods based on standard channel coding schemes and on the separation between the communication and control policies. We have illustrated that the underlying reason for the improvement in performance is the learned ability of the agents to carry out data compression and unequal error protection as a function of the action policies.

VI. ACKNOWLEDGEMENTS

The work of Arsham Mostaani, Symeon Chatzinotas and Bjorn Ottersten is supported by European Research Council (ERC) advanced grant 2022 (Grant agreement ID: 742648). Arsham Mostaani and Osvaldo Simeone have received funding from the ERC under the European Union’s Horizon 2020 Research and Innovation Program (Grant Agreement No. 725731).
Details of the sequential communications-actions policy can be found in Algorithm 2 below. At the end of the training process, policy $\pi_t^c(a_t^c|s_t^c, s_t^e)$ can be obtained by

$$
\pi_t^c(a_t^c|s_t^c, s_t^e) = \delta \left( a_t^c - \argmax_{a_t^c} Q_t(s_t^c, s_t^e, a_t^c) \right),
$$

(8)

and policy $\pi_t^e(a_t^e|s_t^e)$ by

$$
\pi_t^e(a_t^e|s_t^e) = \delta \left( a_t^e - \argmax_{a_t^e} Q_t(s_t^e, a_t^e) \right).
$$

(9)

**Algorithm 2** Learned Sequential Communications and Actions

1: Input parameters: $\gamma$, $\alpha$, and $c$
2: Initialize all-zero Q-tables $Q_t^c(s_t^c, s_t^e, a_t^c)$, $Q_t^e(s_t^e, a_t^e)$ and tables $N_t^c(s_t^c, s_t^e, a_t^c)$, $N_t^e(s_t^e, a_t^e)$, for $i = 1, 2$
3: for each episode $m = 1 : M$ do
4: Randomly initialize $(s_{1,t}^c, s_{1,t}^e) \notin S_T^c$
5: Randomly initialize $(s_{1,t}^c, s_{1,t}^e) \notin S_T^e$
6: $t = 1$
7: while $(s_{1,t}^c, s_{1,t}^e) \notin S_T^c$ or $(s_{1,t}^c, s_{1,t}^e) \notin S_T^e$ do
8: for $i = 1, 2$ do
9: Select $a_{1,i}^c = A_i^c$, by following UCB policy
10: $a_{1,i}^c = \arg\max Q_t^c(s_{1,t}^c, a_{1,i}^c) + c \sqrt{\frac{\ln \sum_{m=1}^{m-1} t_k}{N_t^c(s_{1,t}^c, s_{1,t}^e, a_{1,i}^c)}}$
11: end
12: Obtain message $s_{1,t}^e$, for $i = 1, 2$, from channel
13: if $t \geq 2$ then
14: for $i = 1, 2$ do
15: $Q_t^c(s_{i,t-1}^c, s_{i,t-1}^e, a_{i,t-1}^c) \leftarrow (1 - \alpha) Q_t^c(s_{i,t-1}^c, s_{i,t-1}^e, a_{i,t-1}^c) + \alpha \gamma (r_{i,t} + \max Q_t^c(s_{i,t}^c, s_{i,t}^e, a_{i,t}^c))$
16: end
17: if $s_{1,t}^c \in S_T^c$ or $s_{1,t}^e \in S_T^e$ then break
18: end
19: for $i = 1, 2$ do
20: Select $a_{1,i}^c = A_i^c$ by following UCB policy
21: $a_{1,i}^c = \arg\max Q_t^c(s_{1,t}^c, a_{1,i}^c) + c \sqrt{\frac{\ln \sum_{m=1}^{m-1} t_k}{N_t^c(s_{1,t}^c, s_{1,t}^e, a_{1,i}^c)}}$
22: end
23: Obtain reward $r_t$ and move to the next environment
24: state $s_{i,t+1}^c$, for $i = 1, 2$
25: $Q_t^c(s_{i,t+1}^c, s_{i,t}^e, a_{i,t}^c) \leftarrow (1 - \alpha) Q_t^c(s_{i,t+1}^c, s_{i,t}^e, a_{i,t}^c) + \alpha \gamma (r_{i,t} + \max Q_t^c(s_{i,t+1}^c, s_{i,t}^e, a_{i,t}^c))$, for $i = 1, 2$
26: $t = t + 1$
27: end
28: Compute $\sum_{t=1}^{T_m-1} \gamma^t r_t$ for the $m$th episode
29: end
30: Output: $\pi_t^c(a_t^c|s_t^c, s_t^e)$ by following (8) for $i = 1, 2$
31: $\pi_t^e(a_t^e|s_t^e)$ by following (9) for $i = 1, 2$.