The Emergence of Adversarial Communication in Multi-Agent Reinforcement Learning

Jan Blumenkamp
Department of Computer Science
and Technology
University of Cambridge
United Kingdom
jb2270@cam.ac.uk

Amanda Prorok
Department of Computer Science
and Technology
University of Cambridge
United Kingdom
asp45@cam.ac.uk

Abstract: Many real-world problems require the coordination of multiple autonomous agents. Recent work has shown the promise of Graph Neural Networks (GNNs) to learn explicit communication strategies that enable complex multi-agent coordination. These works use models of *cooperative* multi-agent systems whereby agents strive to achieve a shared global goal. When considering agents with self-interested local objectives, the standard design choice is to model these as separate learning systems (albeit sharing the same environment). Such a design choice, however, precludes the existence of a single, differentiable communication channel, and consequently prohibits the learning of inter-agent communication strategies. In this work, we address this gap by presenting a learning model that accommodates individual non-shared rewards and a differentiable communication channel that is common among all agents. We focus on the case where agents have self-interested objectives, and develop a learning algorithm that elicits the emergence of adversarial communications. We perform experiments on multi-agent coverage and path planning problems, and employ a post-hoc interpretability technique to visualize the messages that agents communicate to each other. We show how a single self-interested agent is capable of learning highly manipulative communication strategies that allows it to significantly outperform a cooperative team of agents.

Keywords: Graph Neural Networks, Multi-Agent Reinforcement Learning, Adversarial Communication, Interpretability

1 Introduction

Multi-agent reinforcement learning models arise as a natural solution to problems where a common environment is influenced by the joint actions of multiple decision-making agents. Such solutions have been applied to a number of domains including traffic systems [1, 2], dynamic supply-demand matching [3], and multi-robot control [4, 5]. In fully decentralized systems, agents not only need to learn how to behave cooperatively, but also, how to *communicate* to most effectively coordinate their actions in pursuit of a common goal. Even though effective communication is key to successful decentralized coordination, the way a problem benefits from communication is not necessarily predetermined, especially in complex multi-agent settings where the optimal strategy is unknown. Reinforcement learning has become one of the most promising avenues to solve such problems.

Capturing information that enables complex inter-agent coordination requires new kinds of Neural Network (NN) architectures. Graph Neural Networks (GNNs) exploit the fact that inter-agent relationships can be represented as graphs, which provide a mathematical description of the network topology. In multi-agent systems, an agent is modeled as a node in the graph, the connectivity of agents as edges, and the internal state of an agent as a graph signal. In recent years, a range of approaches towards learning explicit communication were made [6, 7, 8, 9]. The key attribute of GNNs is that they operate in a localized manner, whereby information is shared over a multi-hop communi-

4th Conference on Robot Learning (CoRL 2020), Cambridge MA, USA.

cation network through explicit communication with nearby neighbors only, hence resulting in fully decentralizable policies.

One particularly promising approach leverages Graph Convolutional Neural Networks (GCNNs), which utilize *graph convolutions* to incorporate a graph structure into the learning process by concatenating layers of graph convolutions and nonlinearities [10, 4]. Recent work leverages GCNNs to automatically synthesize local communication and decision-making policies for solving complex multi-agent coordination problems [7, 8]. These learning approaches assume full cooperation, whereby all agents share the same goal of maximising a global reward. Yet there is a dearth of work that explores whether agents can utilize machine learning to synthesize communication policies that are not only cooperative, but instead, are *non-cooperative* or even *adversarial*. The goal of this paper is to demonstrate how self-interested agents can learn such adversarial communication, without explicitly optimizing for it. Crucially, we posit that understanding how adversarial communication emerges is the first step towards developing methods that can deal with it in real-world situations.

Contributions. The main contribution of this paper is an evaluation of the hypothesis that non-cooperative agents can learn manipulative communication policies. To achieve this goal, we developed a new multi-agent learning model that integrates heterogeneous, potentially self-interested policies that share a differentiable communication channel. This model consists of three key components, *(i)* a monolithic, decentralizable neural architecture that accommodates multiple distinct reward functions and a common differentiable communication channel, *(ii)* a reinforcement learning algorithm that elicits the emergence of adversarial communications, and *(iii)*, a post-hoc interpretability technique that enables the visualization of communicated messages. Our code is publicly available [1].

The experimental evaluation is based on a multi-agent system with a mix of cooperative and self-interested agents, and demonstrates the effectiveness of the learning scheme in multi-agent coverage and path planning problems. Results show that it is possible to learn highly effective communication strategies capable of manipulating other agents to behave in such a way that it benefits the self-interested agents. Overall, we demonstrate that adversarial communication emerges when local rewards are drawn from a finite pool, or when resources are in contention. We also show that self-interested agents that communicate manipulatively, however, need not be adversarial by design; they are simply programmed to disregard other agents' rewards.

2 Related Work

We briefly review work in cooperative and non-cooperative multi-agent reinforcement learning, with a focus on approaches that model communication between agents.

Cooperative **multi-agent reinforcement learning**. Cooperation enables agents to achieve feats together that no individual agent can achieve on its own. Yet independently learning agents perform poorly in practice [11], since agents' policies change during training, resulting in a non-stationary environment. Hence, the majority of recent work leans on joint learning paradigms [12, 13, 14, 15]. These approaches avoid the need for explicit communication by making strong assumptions about the visibility of other agents and the environment. Some other approaches use communication, but with a predetermined protocol [16, 17].

Early work on learning communication considers *discrete* communication, through signal binarization [6], or categorical communication emissions [18]. The former approaches demonstrate emergent communication among few agents; scaling the learning process to larger agent teams requires innovations in the structure of the learnt communication models. The approach in [19] presents a more scalable approach by instantiating a GNN-inspired construction for learning continuous communication. Other, more recent work demonstrates the use of GNNs for learning communication policies that lead to successful multi-agent coordination in partially observed environments [8, 7, 4].

Non-cooperative **multi-agent reinforcement learning**. Most work on non-cooperative multi-agent systems does not model *learnable* communication policies, since the assumption is made that agent behaviors evolve as a function of consequences observed in the environment. *Social dilemma* problems represent one type of non-cooperative system, where the collectively best outcomes are not aligned with individualistic decisions. Descriptive results were obtained for sequential social dilem-

[1] https://github.com/proroklab/adversarial_comms

mas [20] and common-pool resource problems [21]. Other work focuses on the development of learning algorithms for non-cooperative multi-player games [22, 23]. Yet none of these approaches include dedicated communication channels between agents.

More closely related to our work, the work in [24] presents a learning scheme for mixed cooperative-competitive settings. The approach enables speaker agents to output semantic information, which is, in turn, observed by listener agents. In contrast to our approach, this type of communication is not differentiable, and assumes time-invariant fully connected agent topologies. To date, there is a lack of work in non-cooperative multi-agent reinforcement learning with continuous differentiable communication.

3 Preliminaries

This work considers the presence of an explicit communication channel between agents. As such, we first introduce Aggregation Graph Neural Networks (AGNNs) [10], which provide a decentralizable architecture to learn communication policies which are fully differentiable. [2] We then proceed with the introduction of Vanilla Policy Gradient (VPG) [27] for groups of independent *self-interested* agents.

3.1 Aggregation Graph Neural Networks for Multi-Agent Communication

We model inter-agent communication through a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$. The node set $\mathcal{V} = \{1, \dots, N\}$ represents individual agents, and the edge set $\mathcal{E} = \mathcal{V} \times \mathcal{V}$ represents inter-agent communication links. The set of neighboring agents $\mathcal{N}_i$ that can communicate with agent $i \in \mathcal{V}$ is defined as $\mathcal{N}_i = \{j \in \mathcal{V} : (j, i) \in \mathcal{E}\}$. The adjacency matrix $\mathbf{S} \in \mathbb{R}^{N \times N}$ indicates the connectivity of the graph with the entry $[\mathbf{S}]_{ij}$ equal to one if node $j \in \mathcal{N}_i$ (and zero otherwise). We can make the dependency on the edge set $\mathcal{E}$ explicit with $\mathbf{S}_{\mathcal{E}}$.

The set of messages transmitted by all robots is denoted $\mathbf{X} \in \mathbb{R}^{N \times F}$. Hence, the message (or *datum*) sent by agent i is $\mathbf{x}_i = [\mathbf{X}]_i$ (the ith row of matrix $\mathbf{X}$). AGNNs operate over multiple communication hops $k \in 0, \dots, K$ and the connectivity at each hop k can be computed by elevating the adjacency matrix to the power k (i.e., $\mathbf{S}^k$). For each hop k, messages are aggregated using a permutation-invariant operation (e.g., sum or average) and the next message is computed from this aggregated data. More formally, the *graph shift operator* $\mathbf{S}$ shifts the signal $\mathbf{X}$ over the nodes and a series of learnable *filter taps* $\mathbf{H}_k \in \mathbb{R}^{F \times F'}$ aggregates data from multiple hops, such that

$$[\mathbf{S}\mathbf{X}]_i = \sum_{j \in \mathcal{N}_i} [\mathbf{S}]_{ij} \mathbf{x}_j \quad \text{and} \quad \mathbf{X}' = g_\eta(\mathbf{X}; \mathbf{S}) = \sum_{k=0}^{K} \mathbf{S}^k \mathbf{X} \mathbf{H}_k \tag{1}$$

where g is a function parameterized by $\eta = \{\mathbf{H}_k\}_{k=1}^{K}$ and where $\mathbf{X}'$ summarizes the data received by all agents. A non-linearity σ is applied and the process is cascaded L times:

$$\mathbf{X}_l = \sigma\left(g_{\eta_l}(\mathbf{X}_{l-1}; \mathbf{S})\right) \text{ with } \mathbf{X}_0 = \mathbf{X}. \tag{2}$$

As indicated by the first equality in Eq. 1, this sequence of operations is decentralizable, and can be executed locally at each agent [10]. We also note that, since η_l for $l \in \{1, \dots, L\}$ is shared among agents, this formulation can only accommodate *homogeneous* teams of agents— a limitation that we lift in Sec. 4.1 .

3.2 Independent Vanilla Policy Gradient

We consider multi-agent systems that are partially observable and operate in a decentralized manner. Each agent aims to maximise a local reward with no access to the true global state. In Sec. 4.2, we allow agents to communicate through the exchange of explicit messages.

Markov Decision Process (MDP). We formulate a stochastic game defined by the tuple $\langle \mathcal{V}, \mathcal{S}, \mathcal{A}, P, \{R^i\}_{i \in \mathcal{V}}, \{\mathcal{Z}^i\}_{i \in \mathcal{V}}, Z, \mathcal{T}, T, \gamma \rangle$, in which N agents identified by $i \in \mathcal{V} \equiv \{1, \dots, N\}$

[2] An AGNN is a particular instantiation of GNNs [25, 26]. Our method can be used with any GNN variant.

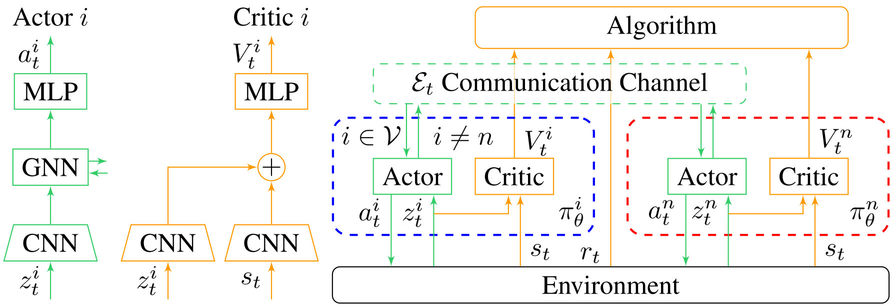


Figure 1: Architectural overview: The actor or the policy π_θ^i determines the action a_t^i from the local observation z_t^i and the multi-hop communication messages from other agents, which in turn depend on all agents' observations z_t and the communication topology $\mathcal{E}_t$. Green denotes components that can be executed locally, and orange denotes components that are required for centralized training only. In this figure and the rest of this paper we highlight components and results for the cooperative team in blue and the self-interested agent in red.

choose sequential actions. Similarly to the formulation in [14], the environment has a true state $s_t \in \mathcal{S}$. At each time step t, each agent executes action $a_t^i \in \mathcal{A}$. Together, the agents form a joint action $\mathbf{a}_t \in \mathcal{A}^N$. The state transition probability function is $P(s_{t+1}|s_t, \mathbf{a}_t) : \mathcal{S} \times \mathcal{A}^N \times \mathcal{S} \to [0,1]$. We consider a partially observable setting, in which each agent i draws observations $z_t^i \in \mathcal{Z}^i$ according to the observation function $Z^i(s_t) : \mathcal{S} \to \mathcal{Z}$, forming a joint observation $\mathbf{z}_t \in \mathcal{Z}^N$. The discount factor is denoted by $\gamma \in [0,1)$. To the contrary of [14], agents in our system observe a local reward r_t^i drawn from $R^i(s_t, a_t^i) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ (i.e., it is not a global reward given to the whole team). The agents' communication topology at time step t, denoted by $\mathcal{E}_t \in \mathcal{T}$, is drawn according to $T(s_t) : \mathcal{S} \to \mathcal{T}$.

Multi-Agent Vanilla Policy Gradient (VPG). Next, we detail VPG applied to independent self-interested agents with a centralized critic—a setting similar to [14], but with individual agent-specific rewards. Without explicit communication, each agent learns a local policy $\pi_{\theta^i}^i(a_t^i|z_t^i) : \mathcal{Z} \times \mathcal{A} \to [0,1]$ parameterized by a local set of parameters θ^i. The induced joint policy is $\boldsymbol{\pi}_\theta(\mathbf{a}_t|\mathbf{z}_t) = \prod_{i\in\mathcal{V}} \pi_{\theta^i}^i(a_t^i|z_t^i)$ with $\theta = \{\theta^i\}_{i\in\mathcal{V}}$. The discounted return for each agent i is $G_t^i = \sum_{l=0}^{\infty} \gamma^l r_{t+l}^i$. The centralized value functions that estimate the value of each agent i in state s_t are $V^{\boldsymbol{\pi},i}(s_t) = \mathbb{E}_t[G_t^i|s_t]$, and the corresponding action-value functions are $Q^{\boldsymbol{\pi},i}(s_t, \mathbf{a}_t) = \mathbb{E}_t[G_t^i|s_t, \mathbf{a}_t]$ (we now omit $\boldsymbol{\pi}$ and simply write V^i and Q^i). The advantage functions are then given by $A^i(s_t, \mathbf{a}_t) = Q^i(s_t, \mathbf{a}_t) - V^i(s_t)$. There exist many techniques to estimate the advantage. For example, Schulman et al. [27] define the *Generalized Advantage Estimate* as $\hat{A}^i(s_t) = \sum_{l=0}^{\infty}(\gamma\lambda)^l \delta_{t+l}^i$ where $\lambda \in [0,1]$ and where the TD-residual is $\delta_t^i = r_t^i + \gamma V^i(s_{t+1}) - V^i(s_t)$. The value functions themselves can be estimated by parameterized functions V_ϕ^i (with parameters ϕ) by minimizing $\mathbb{E}_t[\|V_\phi^i(s_t) - G_t^i\|^2]$ (again, there are many approaches to estimate value functions [28]). However, since this is not the focus of this work, we will now assume that we have access to a centralized advantage estimate $\hat{A}^i(s_t)$ for each agent i. Finally, the policy of agent i can be improved through gradient ascent using the policy gradient

$$g^i = \mathbb{E}_t\left[\nabla_\theta \log \pi_{\theta^i}^i(a_t^i|z_t^i)\hat{A}^i(s_t)\right]. \tag{3}$$

4 Methodology

The objective of this work is to demonstrate the emergence of adversarial communication. Towards this end, we consider a mixed setup with a team of cooperative agents and one self-interested agent. We first propose a modification to the AGNN elaborated in Sec. 3.1, to allow for multi-agent systems with heterogeneous policies. We then modify VPG to train the team of cooperative agents to collaborate using explicit communication. Finally, we introduce the self-interested agent and train it to communicate with the cooperative team. An overview of the architecture is illustrated in Fig. 1; details of the algorithms are given in B.

4.1 Heterogeneous AGNN

In order to use the AGNN architecture in a heterogeneous setting, we need to generalize the homogeneous formalization defined in Eq. 1. In the homogeneous setting, a single set of parameters is given as $\eta = \{\mathbf{H}_k\}_{k=1}^{K}$ where $\mathbf{H}_k$ describes a trainable *filter tap* for hop k. To allow for locally unique communication policies, we introduce different *filter taps* for individual agents. Eq. 1 becomes

$$\mathbf{X}' = g_\eta(\mathbf{X}; \mathbf{S}) = \sum_{k=0}^{K} \begin{bmatrix} [\mathbf{S}^k]_1^\mathsf{T} \mathbf{X} \mathbf{H}_k^1 \\ \vdots \\ [\mathbf{S}^k]_N^\mathsf{T} \mathbf{X} \mathbf{H}_k^N \end{bmatrix} \tag{4}$$

where $\eta = \{\eta^i\}_{i \in \mathcal{V}}$ and $\eta^i = \{\mathbf{H}_k^i\}_{k=1}^{K}$. It is important to observe that each aggregation $[\mathbf{S}^k]_i^\mathsf{T} \mathbf{X}$ only considers messages sent by agents k hops away from i and that this formulation remains decentralizable. Note that not all η^i need to be distinct, i.e., a sub-group of agents can share the same communication policy.

4.2 Cooperative Learning

We first consider a team of cooperative homogeneous agents with individual rewards and explicit communication, and formulate a local decentralized policy.

Local Policy. Each agent encodes its local observation z_t^i using an encoder $f_\nu(z_t^i) : \mathcal{Z} \to \mathbb{R}^F$ (e.g., using a Convolutional Neural Network (CNN)). The encoded observations are grouped as $\mathbf{X} = [f_\nu(z_t^1), \dots, f_\nu(z_t^N)]^\mathsf{T}$, and each local encoding is then shared with neighboring agents by applying $\mathbf{X}' = \sigma(g_\eta(\mathbf{X}, \mathbf{S}_{\mathcal{E}_t})) : \mathbb{R}^{N \times F} \times \mathbb{R}^{N \times N} \to \mathbb{R}^{N \times F'}$ (i.e., using an AGNN). The aggregated data is then used to output a distribution over actions using $h_\mu([\mathbf{X}]_i) : \mathbb{R}^{F'} \to \Delta^{|\mathcal{A}|}$ (where Δ represents the simplex), e.g., using a Multi Layer Perceptron (MLP) with a softmax output. Overall, the local policy is defined by $\pi_\theta^i(a_t^i | z_t, \mathcal{E}_t) = h_\mu([\sigma(g_\eta([f_\nu(z_t^1), \dots, f_\nu(z_t^N)])^\mathsf{T}, \mathbf{S}_{\mathcal{E}_t})]_i)$ with $\theta = \{\nu, \eta, \mu\}$ (we omit the explicit dependence on $\mathcal{E}_t$ and write $\pi_\theta^i(a_t^i | z_t)$). Despite being decentralized and locally executable, each agent's policy explicitly depends on the observations made by neighboring agents (due to the explicit communication).

Cooperative Policy Gradient. To train the cooperative group of agents, we modify VPG (Sec. 3.2) to reinforce local, individual actions that lead to increased rewards for other agents.

Lemma 1 *Given an actor-critic algorithm with a compatible TD(1) critic that follows the cooperative policy gradient*

$$g_k^i = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{j \in \mathcal{V}} \nabla_\theta \log \pi_\theta^j (a^j | z) A^i(s) \right] \tag{5}$$

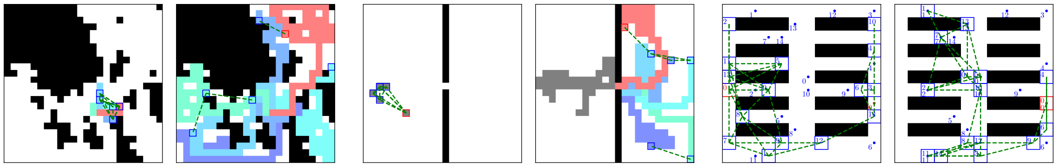
for each agent $i \in \mathcal{V}$ at each iteration k, this gradient converges to a local maximum of the expected sums of returns of all agents with probability one.

The proof is provided in A. This lemma highlights that not only does the gradient converge, but also that the joint policy maximizes the sum of cumulative rewards.

4.3 Self-Interested Learning

After training the cooperative policy, we replace one of the agents with a self-interested agent. This agent's goal is simply to maximize its own reward (disregarding the rewards of others). If rewards are drawn from a finite pool (e.g., agents compete for resources), we expect the self-interested agent to learn to communicate erroneous information. In other words, it will start lying about its state and observations to mislead other agents, thereby increasing its own access to the limited rewards.

Local Policy. For clarity, we denote the parameters of all cooperative agents by θ^c and the parameters of the self-interested agent by θ^n. The index of the self-interested agent is denoted by n. The policy of each agent i is denoted by $\pi_{\theta^c \theta^n}^i (a_t^i | z_t)$. It depends on both θ^c and θ^n for all agents, as messages exchanged between agents are inter-dependent.



(a) Standard coverage.

(b) Coverage in split environment.

(c) Path planning.

Figure 2: Overview of grid-world environments used in our experiments. Cooperative and self-interested agents are visualized as blue and red squares, respectively. Black cells correspond to obstacles. In the coverage environments, different colors indicate the coverage achieved by individual agents. In the path planning environment, labeled goal locations are indicated by circles.

Self-Interested Policy Gradient. The goal of this learning procedure is to learn θ^n to maximize the expected return of the self-interested agent under policy $\pi^n_{\theta^c\theta^n}(a^n_t|z_t)$ where θ^c is fixed. Similarly to Sec. 4.2, we modify the policy gradient of the self-interested agent to account for its advantage across actions performed by other agents.

Lemma 2 *Given an actor-critic algorithm with a compatible TD(1) critic that follows the self-interested policy gradient*

$$g^n_k = \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{j\in\mathcal{V}} \nabla_{\theta^n} \log \pi^j_{\theta^c\theta^n}(a^j|z)A^n(s)\right] \tag{6}$$

for a self-interested agent $n \in \mathcal{V}$ at each iteration k, this gradient converges to a local maximum of the expected returns of the self-interested agent with probability one.

The proof is provided in A. Note how this gradient only affects the parameters θ^n of the self-interested policy.

4.4 White-Box Analysis

The encoder f_{ν^c} of the cooperative policy $\pi^i_{\theta^c}$, where ν^c refers to the parameters of the cooperative encoder, transforms the local observation z^i_t into an encoding $[\mathbf{X}_t]_i$ which is used as input to the AGNN and constitutes the first message being sent by agent i at time step t. We hypothesize that the self-interested agent creates an alternative encoding that not only helps it gather more rewards individually, but also, influences cooperative agents' behavior towards its self-interested goal. To verify this hypothesis, we perform a white-box analysis by sampling observations z^i_t and feature vectors $[\mathbf{X}_t]_i$ for all cooperative agents, and by training an interpreter f^{-1}_{ψ} that minimizes the reconstruction error such that $f^{-1}_{\psi} \circ f_{\nu^c}(z^i_t) \approx z^i_t$ for all z^i_t. We provide more details in D.

5 Experiments

We validate our proposed learning scheme and architecture on three case studies requiring communication between agents. For all our experiments, we use a PPO variation of our algorithms [29] and employ distributed training based on Ray [30] and RLlib [31]. A movie of our experiments is available at `https://youtu.be/o1Nq9XoSU6U`.

Setup. We evaluate our learning scheme in a custom grid-world (see Fig. 2). Agents can communicate if they are closer than some predefined distance, thus defining the communication topology $\mathcal{E}_t$. Each agent has a local field of view. We perform three experiments, **(1)** with a purely cooperative team, **(2)** with the introduction of one self-interested agent holding the cooperative team's policy fixed, and **(3)**, with the cooperative team allowed to re-adapt to the self-interested agent. In experiment (2), we perform two variants, one where the self-interested agent can communicate to the other agents, and one where it cannot. We evaluate average agent performance, and use our white-box interpreter to understand the nature of messages sent.

		Cooperative		Introduction of SI agent		Re-adaptation
Task		w/ comms	w/o comms	w/ adv comms	w/o adv comms	w/ adv comms
Coverage	C	67.1 ± 2.8	63.2 ± 5.9	45.6 ± 5.2	58.6 ± 2.3	60.8 ± 2.7
	SI	N/A	N/A	103.7 ± 21.1	45.5 ± 10.0	32.9 ± 13.3
Split Coverage	C	52.9 ± 0.5	38.9 ± 11.0	34.8 ± 3.0	47.5 ± 1.9	50.8 ± 1.4
	SI	N/A	N/A	90.9 ± 15.0	27.6 ± 9.4	10.4 ± 6.8
Path planning	C	29.1 ± 6.0	22.4 ± 6.6	8.9 ± 4.3	28.0 ± 6.6	27.1 ± 5.0
	SI	N/A	N/A	37.7 ± 10.1	17.8 ± 19.8	5.6 ± 13.8

Table 1: Average return for all agents of each group (cooperative and self-interested (SI)) over 100 episodes at the end of training for all experiments given with a 1σ standard deviation. Rows show the tasks and agent groups, cooperative (C) and self-interested (SI); columns show the three experiment types, (1)-(3), and communication variants, as described in Sec. 5.

Tasks. We consider the following three tasks, as depicted in Fig. 2. More details on each task are given in D.

Coverage in non-convex environments: An agent is required to visit all free cells in the environment as quickly as possible; it is rewarded for moving into a cell that has not yet been covered by any other agent (including itself). The observation is described by a three-channel tensor consisting of the local obstacle map, the agent's own coverage, and the agent's position. We use $N = 6$ agents.

Coverage in split environments: An agent is required to visit all free cells in the *right-hand* side of the environment as quickly as possible; it is rewarded for covering new cells in that sub-area. The observation is the same as for the prior task. We use $N = 6$ agents.

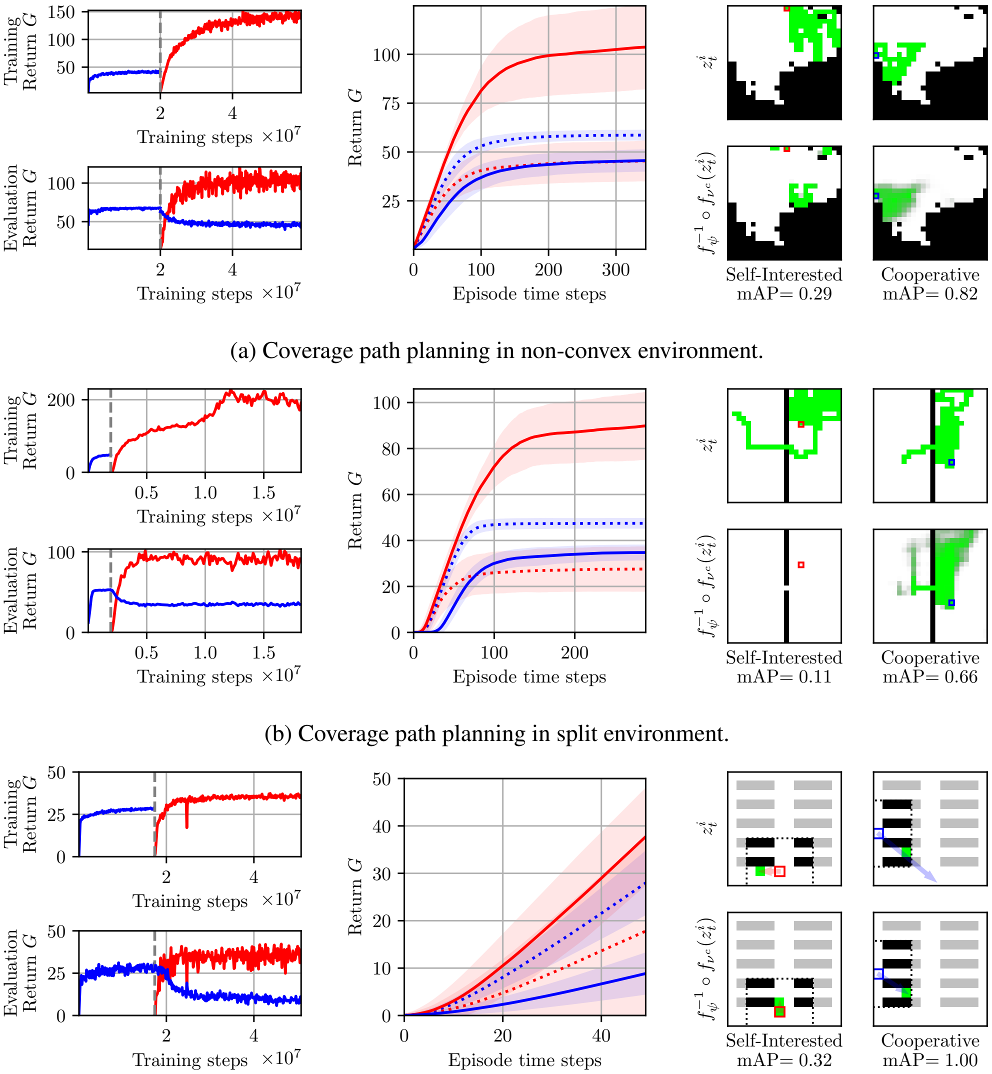
Path planning: An agent is required to navigate to its assigned goal. Only one agent can occupy any given cell at the same time, hence, agents must learn to avoid each other. An agent is rewarded for each time-step that it is located at its designated goal (episodes have fixed horizons). The observation is similar to the coverage tasks, except the channel containing the agent's coverage, which is replaced with a map containing its goal. We use $N = 16$ agents.

Results. The results are summarized in Tab. 1 and visualized in Fig. 3. In a purely cooperative team (first two columns), agents are able to improve their average performance by utilizing explicit communication to coordinate. After the introduction of a self-interested agent, average performance for the cooperative team decreases; this loss is significant in the case where adversarial communication is enabled. The self-interested agent is able to significantly outperform the cooperative team. When communicating, its performance improves by 128% for non-convex coverage, by 229% for split coverage, and by 112% for path planning. After re-adaptation, the cooperative team is able to recoup its performance loss, and reach a level that is on par with the purely cooperative case (with communication). Fig. 3 shows performance during training (first column) and final testing performance over an episode (second column). For all tasks, the mean Average Precision (mAP) of the white-box interpreter on the test set is significantly higher for the cooperative agents than for the self-interested agent, indicating that the self-interested agent learns an encoding that differs from the cooperative policy's encoding of local observations. We include additional experiments in C.

Discussion. Allowing the self-interested agent to learn its policy while holding all other agents' policies fixed leads to manipulative behavior that is made possible through adversarial communication. We showed that this observation is valid across different task settings, with performance improvements in the range of 112%–229% for the self-interested agent when communication is enabled. Conversely, adversarial communication is neutralized when other agents are able to adapt to the self-interested policy. Overall, we demonstrate that adversarial communication emerges when local rewards are drawn from a finite pool, or when resources are in contention; self-interested agents that communicate manipulatively, however, are not necessarily adversarial by design, they are simply programmed to disregard other agents' rewards.

6 Conclusion

We proposed a novel model for learning to communicate in multi-agent systems with multiple, potentially conflicting incentives that are driven by local agent-specific rewards. The main attribute of our model is that it is capable of accommodating multiple agent objectives while maintaining a common differentiable communication channel. We demonstrated the emergence of adversarial communication and listed conditions under which this was observed. Post-hoc interpretations indicated devious encodings in messages sent through self-interested agents. Future work will address co-optimization schemes, the study of equilibria, and a generalization of the proposed methods to arbitrary proportions of cooperative vs. self-interested agents.



(a) Coverage path planning in non-convex environment.

(b) Coverage path planning in split environment.

(c) Path planning.

Figure 3: The sub-panels show results for the three considered tasks. *1st column:* Mean episodic reward normalized per agent during training; blue for the cooperative agents and red for the self-interested agent. The top plot shows the sequential training of the cooperative team followed by the training of the self-interested agent while holding the cooperative policies fixed. The bottom plot shows the reward during training evaluation for a fixed episode length. *2nd column*: Mean test reward per agent group (self-interested or cooperative) over 100 episodes, throughout an episode. The solid curves show the return with adversarial communications and the dashed curves without adversarial communications. All curves are given with a 1σ standard deviation. *3rd column*: Visualization of the white-box analysis on an example. The top row shows the true local observation for the self-interested agent and for a single cooperative agent. The bottom row shows the reconstruction of the message according to the trained interpreter.

Acknowledgments

We gratefully acknowledge the support of ARL grant DCIST CRA W911NF-17-2-0181, and the Engineering and Physical Sciences Research Council (grant EP/S015493/1). The research is funded in part by Arm, and in part by Amazon.com Inc. Their support is gratefully acknowledged.

References

[1] C. Wu, A. Kreidieh, E. Vinitsky, and A. M. Bayen. Emergent behaviors in mixed-autonomy traffic. In S. Levine, V. Vanhoucke, and K. Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 398–407. PMLR, 13–15 Nov 2017. URL http://proceedings.mlr.press/v78/wu17a.html.

[2] N. Hyldmar, Y. He, and A. Prorok. A fleet of miniature cars for experiments in cooperative driving. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3238–3244. IEEE, 2019.

[3] D. T. Nguyen, A. Kumar, and H. C. Lau. Credit assignment for collective multiagent RL with global rewards. In *Advances in Neural Information Processing Systems*, pages 8102–8113, 2018.

[4] A. Khan, E. Tolstaya, A. Ribeiro, and V. Kumar. Graph policy gradients for large scale robot control. In L. P. Kaelbling, D. Kragic, and K. Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 823–834. PMLR, 30 Oct–01 Nov 2020. URL http://proceedings.mlr.press/v100/khan20a.html.

[5] J. Paulos, S. W. Chen, D. Shishika, and V. Kumar. Decentralization of multiagent policies by learning what to communicate. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7990–7996. IEEE, 2019.

[6] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2137–2145. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6042-learning-to-communicate-with-deep-multi-agent-reinforcement-learning.pdf.

[7] Q. Li, F. Gama, A. Ribeiro, and A. Prorok. Graph neural networks for decentralized multi-robot path planning. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[8] E. Tolstaya, F. Gama, J. Paulos, G. Pappas, V. Kumar, and A. Ribeiro. Learning decentralized controllers for robot swarms with graph neural networks. In *Conference on Robot Learning*, pages 671–682, 2020.

[9] A. Prorok. Graph neural networks for learning robot team coordination. Federated AI for robotics workshop. Technical report, IJCAI-ECAI/ICML/AAMAS 2018, 2018. URL https://arxiv.org/abs/1805.03737.

[10] F. Gama, E. Isufi, G. Leus, and A. Ribeiro. From graph filters to graph neural networks. *arXiv preprint arXiv:2003.03777*, 2020. URL https://arxiv.org/pdf/2003.03777.

[11] L. Matignon, G. J. Laurent, and N. Le Fort-Piat. Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. *The Knowledge Engineering Review*, 27(1): 1–31, 2012. doi:10.1017/S0269888912000057.

[12] J. K. Gupta, M. Egorov, and M. Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In G. Sukthankar and J. A. Rodriguez-Aguilar, editors, *Autonomous Agents and Multiagent Systems*, pages 66–83, Cham, 2017. Springer International Publishing. ISBN 978-3-319-71682-4.

[13] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. N. Foerster, and S. Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *ICML*, pages 4292–4301, 2018. URL http://proceedings.mlr.press/v80/rashid18a.html.

[14] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual multi-agent policy gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018. URL https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17193.

[15] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 2681–2690. JMLR.org, 2017.

[16] D. Maravall, J. de Lope, and R. Domínguez. Coordination of communication in robot teams by reinforcement learning. In J. M. Ferrández, J. R. Álvarez Sánchez, F. de la Paz, and F. J. Toledo, editors, *Foundations on Natural and Artificial Computation*, pages 156–164, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-21344-1.

[17] C. Zhang and V. Lesser. Coordinating multi-agent reinforcement learning with limited communication. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS '13, pages 1101–1108, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450319935.

[18] I. Mordatch and P. Abbeel. Emergence of grounded compositional language in multi-agent populations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[19] S. Sukhbaatar, A. Szlam, and R. Fergus. Learning multiagent communication with backpropagation. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 2252–2260, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

[20] J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel. Multi-agent reinforcement learning in sequential social dilemmas. *arXiv preprint arXiv:1702.03037*, 2017.

[21] J. Perolat, J. Z. Leibo, V. Zambaldi, C. Beattie, K. Tuyls, and T. Graepel. A multi-agent reinforcement learning model of common-pool resource appropriation. In *Advances in Neural Information Processing Systems*, pages 3643–3652, 2017.

[22] J. Serrino, M. Kleiman-Weiner, D. C. Parkes, and J. Tenenbaum. Finding friend and foe in multi-agent games. In *Advances in Neural Information Processing Systems*, pages 1251–1261, 2019.

[23] P. Paquette, Y. Lu, S. S. Bocco, M. Smith, O.-G. Satya, J. K. Kummerfeld, J. Pineau, S. Singh, and A. C. Courville. No-press diplomacy: Modeling multi-agent gameplay. In *Advances in Neural Information Processing Systems*, pages 4474–4485, 2019.

[24] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*, 2017.

[25] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and K. kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 4509–4517, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

[26] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.

[27] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[28] D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995.

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL `http://arxiv.org/abs/1707.06347`.

[30] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan, and I. Stoica. Ray: A distributed framework for emerging ai applications. In *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation*, OSDI'18, pages 561–577, USA, 2018. USENIX Association. ISBN 9781931971478.

[31] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. E. Gonzalez, M. I. Jordan, and I. Stoica. RLlib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.

[32] V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. In S. A. Solla, T. K. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 1008–1014. MIT Press, 2000. URL `http://papers.nips.cc/paper/1786-actor-critic-algorithms.pdf`.

[33] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[34] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro. Convolutional neural network architectures for signals supported on graphs. *IEEE Transactions on Signal Processing*, 67(4):1034–1049, 02 2019. ISSN 1941-0476. doi:10.1109/tsp.2018.2887403.

[35] E. Galceran and M. Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258 – 1276, 2013. ISSN 0921-8890. doi:https://doi.org/10.1016/j.robot.2013.09.004. URL `http://www.sciencedirect.com/science/article/pii/S092188901300167X`.

Appendices

A Proofs

Lemma 3 *Given an actor-critic algorithm with a compatible TD(1) critic that follows the cooperative policy gradient*

$$g_k^i = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{j \in \mathcal{V}} \nabla_\theta \log \pi_\theta^j (a^j | z) A^i(s) \right] \tag{7}$$

for each agent $i \in \mathcal{V}$ at each iteration k, this gradient converges to a local maximum of the expected sums of returns of all agents with probability one.

Proof: The total gradient applied to θ is given by

$$g = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} \nabla_\theta \log \pi_\theta^j (a^j | z) A^i(s) \right] = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{i \in \mathcal{V}} A^i(s) \sum_{j \in \mathcal{V}} \nabla_\theta \log \pi_\theta^j (a^j | z) \right] \tag{8}$$

$$= \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{i \in \mathcal{V}} A^i(s) \nabla_\theta \log \prod_{j \in \mathcal{V}} \pi_\theta^j (a^j | z) \right] = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{i \in \mathcal{V}} A^i(s) \nabla_\theta \log \boldsymbol{\pi}_\theta (a | z) \right]. \tag{9}$$

If we consider the sum of rewards $r_t = \sum_{i \in \mathcal{V}} r_t^i$ as the joint reward obtained by the joint policy $\boldsymbol{\pi}$, the joint advantage estimate is $A(s) = \sum_{i \in \mathcal{V}} A^i(s)$. Hence the individual policy gradients lead to a joint policy gradient, which is known to converge to a local maximum of the expected return $G_t = \sum_{i \in \mathcal{V}} G_t^i$ if *(i)* $\boldsymbol{\pi}$ is differentiable, *(ii)* the update timescales are sufficiently slow, and *(iii)* the advantage estimate uses a representation compatible with $\boldsymbol{\pi}$ [32]. □

Lemma 4 *Given an actor-critic algorithm with a compatible TD(1) critic that follows the self-interested policy gradient*

$$g_k^n = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{j \in \mathcal{V}} \nabla_{\theta^n} \log \pi_{\theta^c \theta^n}^j (a^j | z) A^n(s) \right] \tag{10}$$

for a self-interested agent $n \in \mathcal{V}$ at each iteration k, this gradient converges to a local maximum of the expected returns of the self-interested agent with probability one.

Proof: The gradient applied to θ is given by

$$g^n = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{j \in \mathcal{V}} \nabla_{\theta^n} \log \pi_{\theta^c \theta^n}^j (a^j | z) A^n(s) \right] = \mathbb{E}_{\boldsymbol{\pi}} \left[A^n(s) \sum_{j \in \mathcal{V}} \nabla_{\theta^n} \log \pi_{\theta^c \theta^n}^j (a^j | z) \right] \tag{11}$$

$$= \mathbb{E}_{\boldsymbol{\pi}} \left[A^n(s) \nabla_{\theta^n} \log \prod_{j \in \mathcal{V}} \pi_{\theta^c \theta^n}^j (a^j | z) \right] = \mathbb{E}_{\boldsymbol{\pi}} \left[A^n(s) \nabla_{\theta^n} \log \boldsymbol{\pi}_{\theta^c \theta^n} (a | z) \right]. \tag{12}$$

If we consider the self-interested agent's reward $r_t = r_t^n$ as the joint reward obtained by the joint policy $\boldsymbol{\pi}$, the joint advantage estimate is $A(s) = A^n(s)$. Hence, the self-interested policy gradient leads to a joint policy gradient which is known to converge to a local maximum of the expected return $G_t = G_t^n$ if *(i)* $\boldsymbol{\pi}$ is differentiable, *(ii)* the update timescales are sufficiently slow, and *(iii)* the advantage estimate uses a representation compatible with $\boldsymbol{\pi}$ [32]. □

```
Algorithm 1: Cooperative Policy Gradient
Input: Initial policy parameters $\theta$ and value parameters used to estimate the advantage
for $k \leftarrow 1, 2, \ldots$ do
    Collect set of trajectories $\mathcal{D}_t$ by running policies $\pi^i_\theta$
    Compute advantage estimates $\hat{A}^i(s_t)$ for all time steps
    Estimate the policy gradient as $g_k = \sum_{i \in \mathcal{V}} g^i_k$ with
      $g^i_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t \in \tau} \sum_{j \in \mathcal{V}} \nabla_\theta \log \pi^j_\theta(a^j_t | z_t) A^i(s_t)$
    Compute policy update $\theta \leftarrow \theta + \alpha_k g_k$
    Fit value functions
end
```

```
Algorithm 2: Self-Interested Policy Gradient
Input: Initial policy parameters $\theta^n$ and value parameters used to estimate the advantage
for $k \leftarrow 1, 2, \ldots$ do
    Collect set of trajectories $\mathcal{D}_t$ by running policies $\pi^i_{\theta^c \theta^n}$
    Compute advantage estimate $\hat{A}^n(s_t)$ for all time steps
    Estimate the policy gradient as
      $g^n_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t \in \tau} \sum_{j \in \mathcal{V}} \nabla_{\theta^n} \log \pi^j_{\theta^c \theta^n}(a^j_t | z_t) A^n(s_t)$
    Compute policy update $\theta^n \leftarrow \theta^n + \alpha_k g^n_k$
    Fit value function
end
```

B Learning Algorithms

Alg. 1 and Alg. 2 detail the algorithms for the Cooperative Policy Gradient and Self-Interested Policy Gradient, respectively.

C Additional Experiments

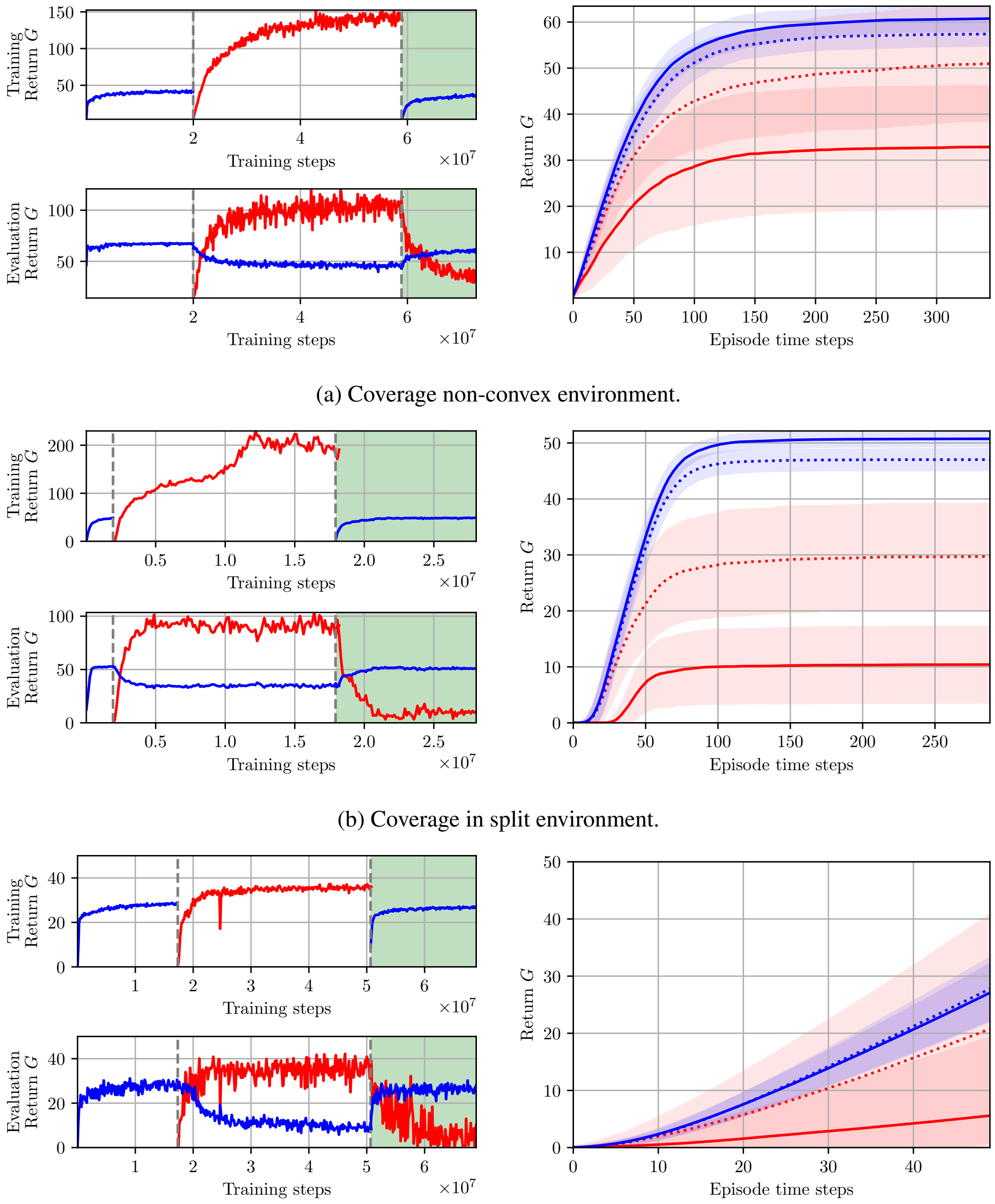
In this section, we provide additional results. In particular, we show *(i)* how cooperative agents can re-adapt to counteract adversarial communication (i.e., *"fool me once, shame on you; fool me twice, shame on me"*), *(ii)* the nature of communicated messages and their effect on agents' internal representations, and *(iii)* that adversarial communication only arises when there is resource contention.

Fig. 4 demonstrates how cooperative agents learn to counteract adversarial communications, when allowed to continue their training in the presence of the self-interested agent. The test results show no performance loss for the cooperative agents when adversarial communication is enabled (solid blue curve), whereas the self-interested agent now performs significantly worse.

Fig. 5, Fig. 6 and Fig. 7 show the interpretation of messages sent, for the three considered tasks, respectively. Overall, the panels confirm that messages sent by the cooperative agents are truthful, yet that messages sent by the self-interested agent are false.

Fig. 8 and Fig. 9 show how the messages sent by the self-interested agent impact the local estimates of global coverage. They demonstrate that when adversarial communication is enabled, agents' local estimates are manipulated to represent faulty information.

Fig. 10 shows an example of a physically split, non-shared environment. The results demonstrate that in environments without resource contention, there is no performance difference *with* and *without* adversarial communication. For this figure, we trained an interpreter on the final output of the AGNN.



(a) Coverage non-convex environment.

(b) Coverage in split environment.

(c) Path planning.

Figure 4: Cooperative agents continue their training, and learn to counteract adversarial communications, for all tasks in (a)-(c). *Left:* Performance during training and evaluation. Re-adaptation occurs during the *green* phase. *Right:* Performance during testing with the final model throughout an episode. We perform 100 episode runs; the solid curves show the return with adversarial communications and the dashed curves without adversarial communications. All curves are given with a 1σ standard deviation.

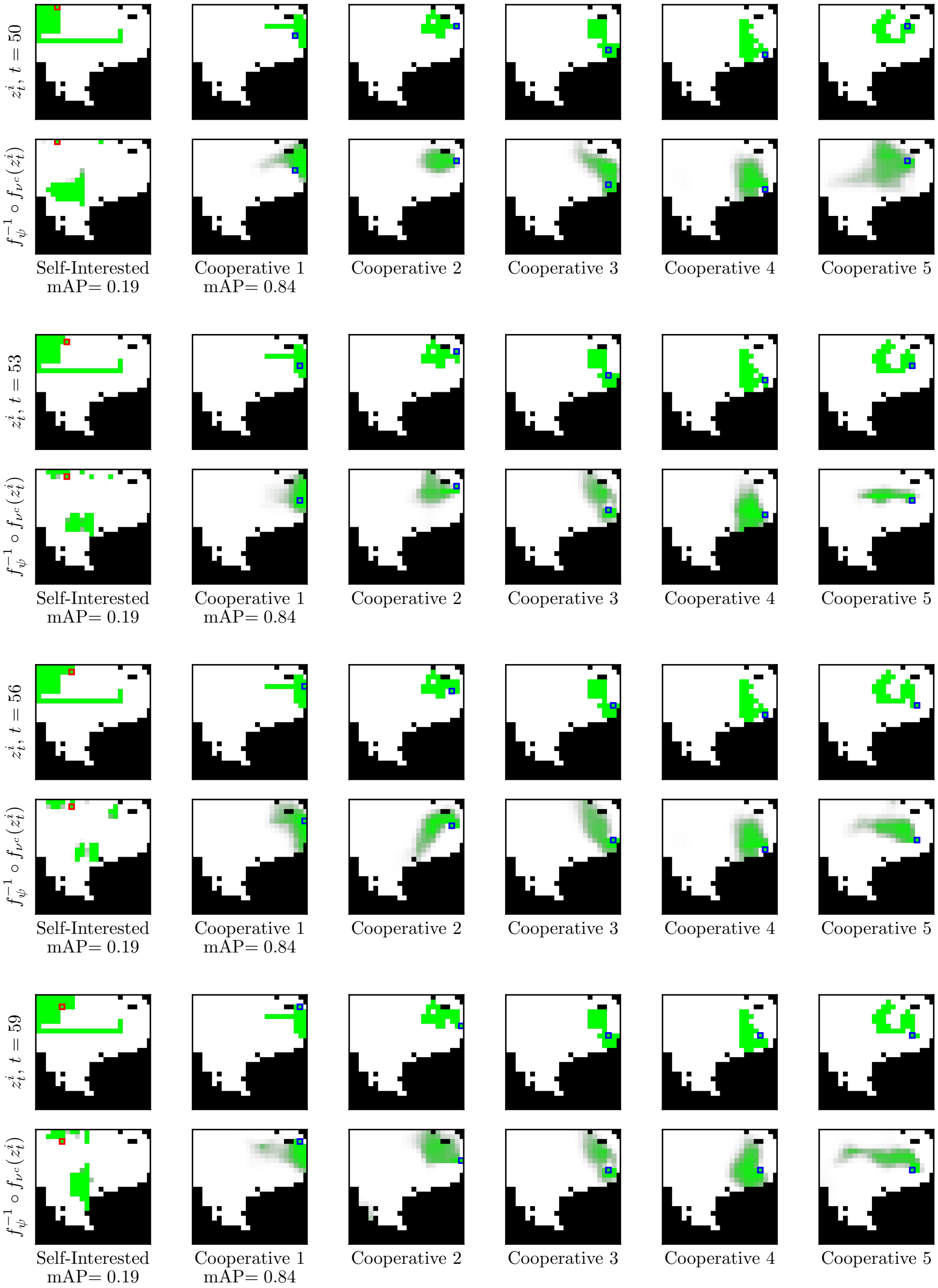


Figure 5: Sequence of interpreted messages for non-convex coverage (with adversarial communication). Column 1 shows the self-interested agent (in red) and columns 2-6 show five cooperative agents (in blue). We consider 4 successive instances in time (top to bottom). For each time instance, we show two rows: the top row shows the true local coverage, and the bottom row shows the communicated local coverage. The adversarial agent clearly sends false information, whereas cooperative agents are truthful.

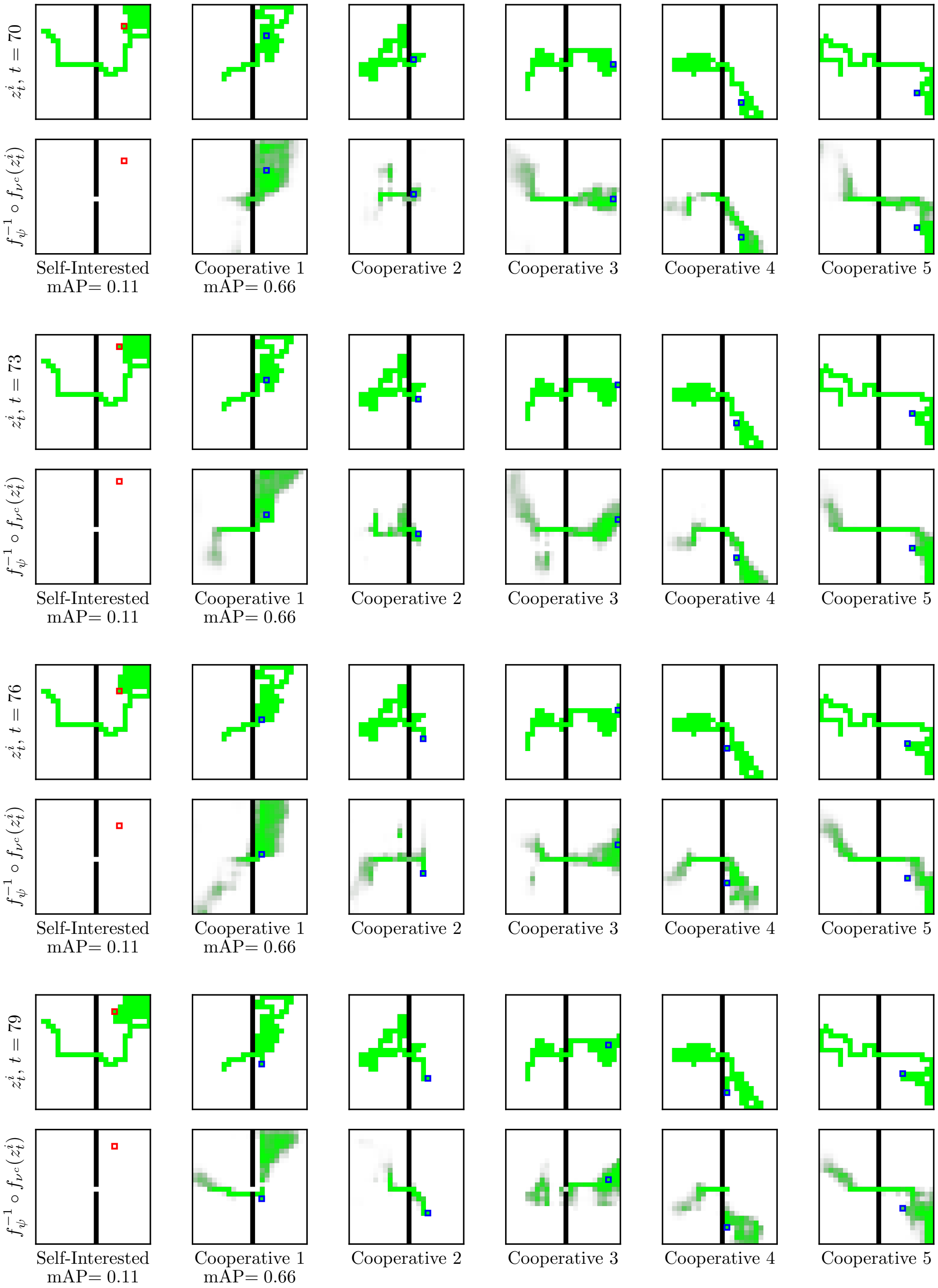


Figure 6: Sequence of interpreted messages for split coverage (with adversarial communication). Column 1 shows the self-interested agent (in red) and columns 2-6 show five cooperative agents (in blue). We consider 4 successive instances in time (top to bottom). For each time instance, we show two rows: the top row shows the true local coverage, and the bottom row shows the communicated local coverage. The adversarial agent clearly sends false information, whereas cooperative agents are truthful.

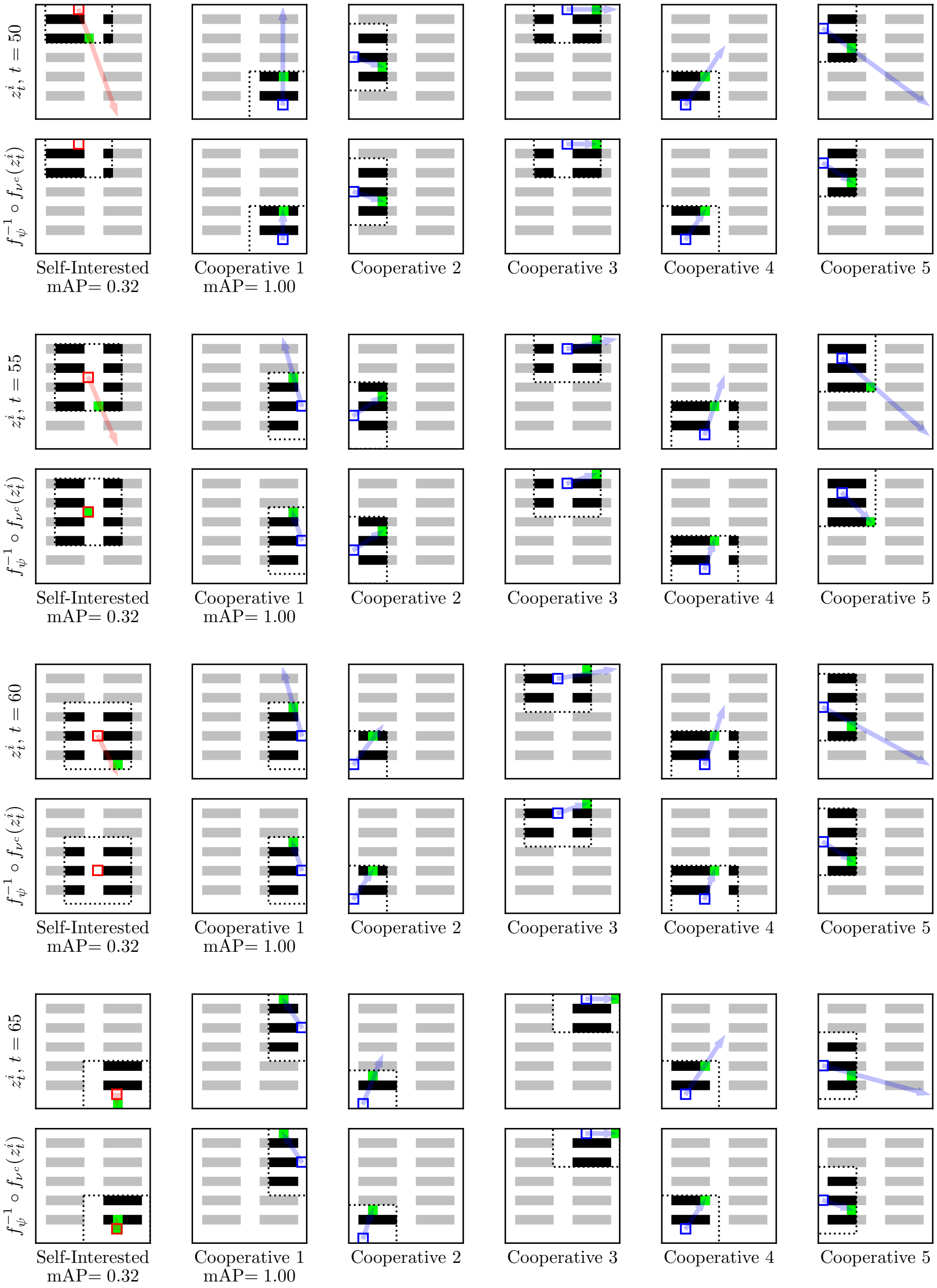


Figure 7: Sequence of interpreted messages for the path planning task (with adversarial communication). Column 1 shows the self-interested agent (in red) and columns 2-6 show five cooperative agents (in blue). We consider 4 successive instances in time (top to bottom). For each time instance, we show two rows: the top row shows the agent's true goal, and the bottom row shows the communicated information. When the goal lies outside the field-of-view, it is projected to its perimeter. The adversarial agent clearly sends false information about its goal, whereas cooperative agents are truthful.

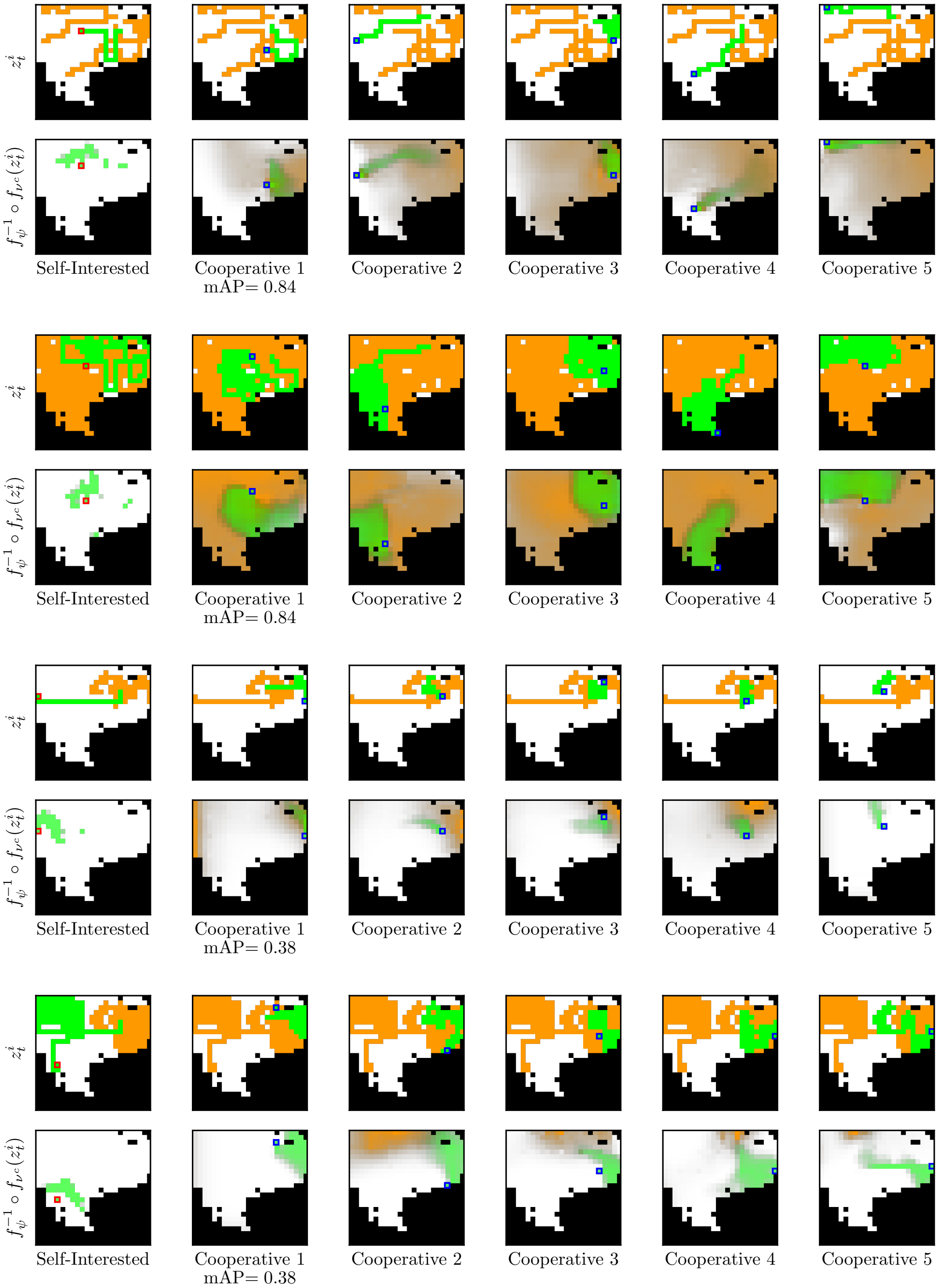


Figure 8: We train the interpreter on the output of the AGNN to reconstruct the local representation of the global coverage (in orange). Each agent's local coverage is shown in green. The first two result sets are obtained without adversarial communication, and the bottom two with. Column 1 shows the self-interested agent (in red) and columns 2-6 show five cooperative agents (in blue). The panels show how the self-interested agent is able to manipulate the local estimate of global coverage.

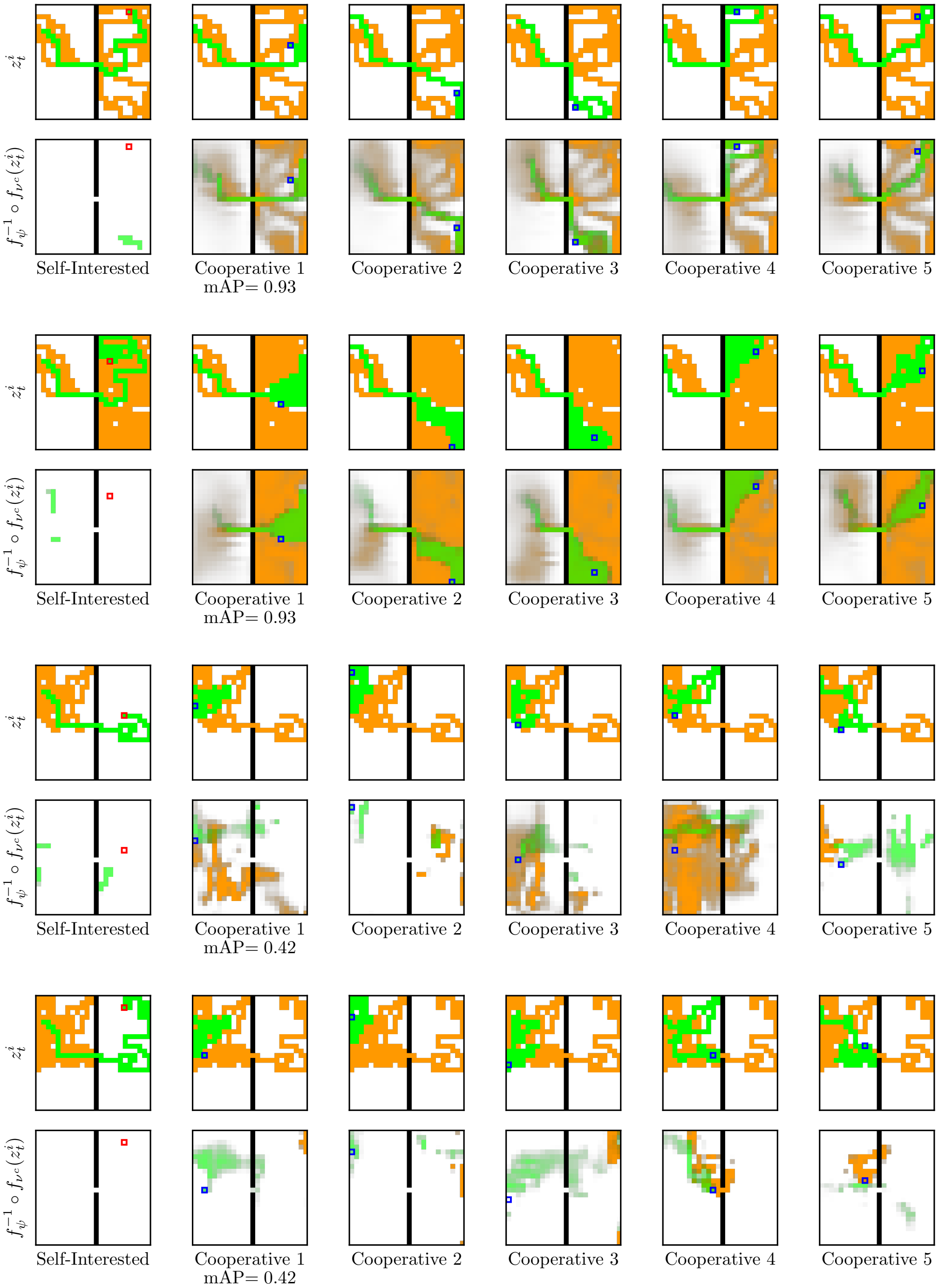


Figure 9: We train the interpreter on the output of the AGNN to reconstruct the local representation of the global coverage (in orange). Each agent's local coverage is shown in green. The first two result sets are obtained without adversarial communication, and the bottom two with. Column 1 shows the self-interested agent (in red) and columns 2-6 show five cooperative agents (in blue). The panels show how the self-interested agent is able to manipulate the local estimate of global coverage.

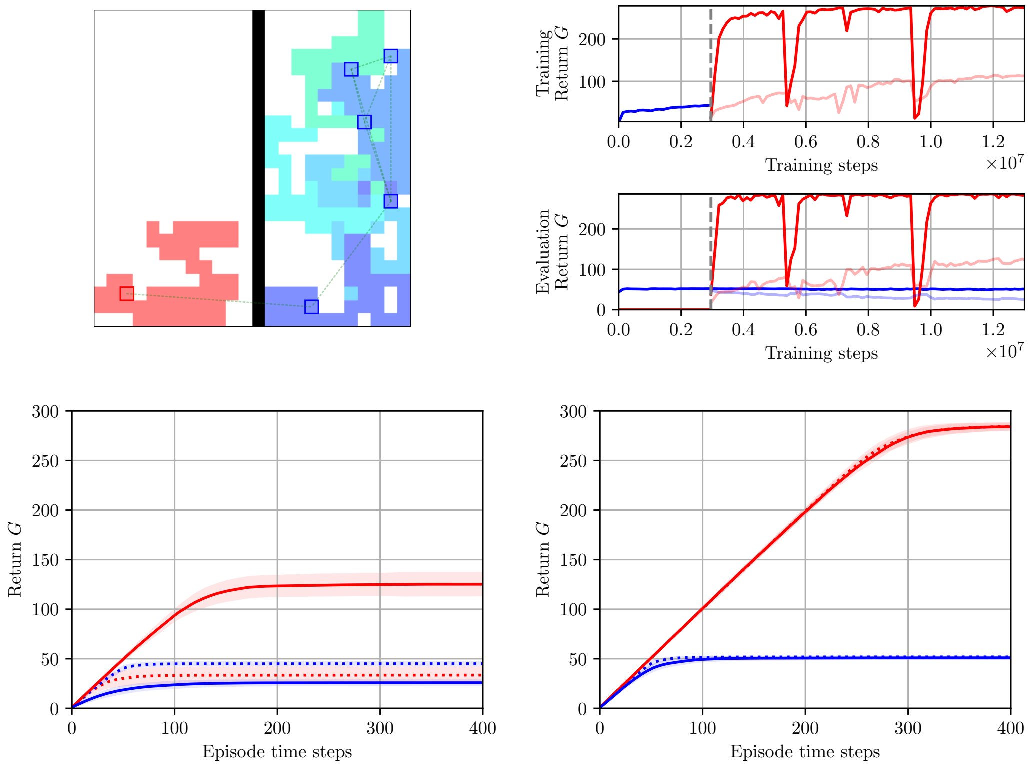


Figure 10: Adversarial training in a non-competitive environment where resources are non-shared. Red curves show the return for the self-interested agent and blue curves for the cooperative agent group. *Top left:* An example of the environment, which is physically split (the communication graph remains connected). *Top right:* The solid curves show the training and evaluation return of the non-competitive environment (self-interested agent placed in left half) and the light curves show the return for the competitive environment (self-interested agent placed together with cooperative agents in the right half). The plot is separated in two areas, left shows cooperative training and right self-interested training. The blue solid curve in the evaluation plot continues at the same level over the course of the training while the red curve quickly rises, indicating that no adversarial communication is learned as the cooperative agents consistently cover the same area on the right side while the self-interested agent learns to cover the left side on its own. In contrast, the blue light curve decreases while the red light curve increases, indicating that the self-interested agent learns adversarial communications as it gradually covers more area that, in turn, cannot be covered by the cooperative agents. *Bottom:* Both plots in the bottom show the testing performance throughout 100 episode runs on the final model. We provide 1σ error bars; the solid curves show the return with adversarial communications and the dashed curves without adversarial communications. *Bottom left:* Testing performance for baseline in competitive agent placement (self-interested agent shares right half with cooperative agents). There is a significant difference in performance with and without adversarial communications. *Bottom right:* When placing the self-interested agent in the separated half as depicted in top left, the difference in performance with and without adversarial communications disappears. Note that the evaluation magnitude of the return of the cooperative agents and self-interested agent differs because the area is split in half while the team sizes are unbalanced, resulting in a differing per-agent average return.

	Learning Rate	γ	ϵ	λ	Training Batch Size	Minibatch Size	SGD Iterations
Coverage (both)	$5 \cdot 10^{-4}$	0.9	0.2	0.95	5000	1000	5
Path planning	$4 \cdot 10^{-4}$	0.99	0.2	0.95	5000	1000	5

Table 2: Overview of training and PPO hyperparameters. The discount factor is denoted as γ, the PPO clipping parameter is denoted as ϵ and the GAE bias-variance parameter is denoted as λ. SGD iterations refer to the number of consecutive stochastic gradient descent optimization iterations for each training batch.

D Implementation Details

Hyperparameters. We adapt Proximal Policy Optimization (PPO) to integrate our policy gradients and optimize using minibatch stochastic gradient descent as implemented in RLlib [31]. The chosen hyperparameters for both experiments are shown in Tab. 2.

Environment setup. The environment of size $W \in \mathbb{N}$, $H \in \mathbb{N}$ is populated with N agents $\mathcal{V} = \{1, \ldots, N\}$. Each agent i is described at discrete time t by its position $\mathbf{p}_t^i \in \mathbb{N}^2$ and a map of the environment $\mathbf{M}_t^i \in \mathbb{B}^{W \times H}$. Agents i and j can communicate to each other only if $\|\mathbf{p}_t^i - \mathbf{p}_t^i\| < d$, resulting in the communication graph $\mathcal{G}_t$. The graph $\mathcal{G}_t$ is represented as adjacency matrix $\mathbf{S}_t$. The communication range d is a hyperparameter of the environment. The adjacency matrix $\mathbf{S}_t$ is constructed for the adapted communication range and normalized to avoid exploding or vanishing gradients and therefore, numerical instability [33, 34].

At any time step, an agent can either move in one of four directions or wait. Agents' actions are constrained by the environment to prohibit collisions with obstacles or the world's margin. Each agent i has a field of view (FOV) of width $W_{\text{FOV}} \in \mathbb{N}$ and height $H_{\text{FOV}} \in \mathbb{N}$. Each agent's partial observation can be described as tensor $z_t^i \in \mathbb{R}^{2 \times W_{FOV} \times H_{FOV}}$ and consists of channels for a local world map $\mathbf{M}_t^i$ (i.e., obstacle positions) and, depending on the experiment, an optional map $\mathbf{C}_t^i$ containing the agent's local information such as coverage or goal. We integrate the agent's position $\mathbf{p}_t^i$ implicitly into the observation by shifting and cropping $\mathbf{M}_t^i$ and $\mathbf{C}_t^i$ to the agent-relative field of view so that the agent is centered. We pad the observation so that the map outside the dimensions of $\mathbf{M}_t^i$ is perceived as occupied with obstacles and the optional map $\mathbf{C}_t^i$ with zeros (i.e., no coverage and no goals).

Non-convex coverage. The coverage path planning problem is related to the *covering salesman problem* in which an agent is required to visit all points in the target environment (an area or volume) as quickly as possible while avoiding obstacles [35]. Each agent keeps track of its local coverage $\mathbf{C}_t^i \in \mathbb{B}^{W \times H}$. The global coverage $\mathbf{C}_t \in \mathbb{B}^{W \times H}$ is the logical disjunction $\mathbf{C}_t = \bigvee_{i=1}^{N} \mathbf{C}_t^i$ of all agents' coverages.

The grid-world has a size of $W = H = 24$ and is occupied with 40% obstacles that are randomly generated while assuring connectivity of the environment. Each agent receives a reward of 1 if it moves towards a cell that has not yet been covered by any agent. To incentivize fast coverage, in which all agents continuously cover new cells, and to increase sample efficiency, we use a dynamic episode length in which we terminate an episode if any agent has not covered a new cell for 10 time steps.

We train the cooperative policy for 20M training environment time steps and the adversarial policy for 40M time steps. After completing the training, we evaluate with a fixed episode length $T = \lceil W \cdot H \cdot 0.6 \rceil = 346$, which is the minimum time required to visit every free cell for a single agent.

Split coverage. The problem is similar to the non-convex coverage, but in this experiment, the agents only receive a reward for covering the *right-hand side* of the environment. Hence, the agents have to learn to first move to that area and then coordinate to cover it.

The grid-world has a size of $W = H = 24$. The environment has a fixed layout split vertically in two halves by a line of obstacles. The two halves are connected through a single cell. We initialize the agent's positions similarly to the previous experiment but constrain them to be placed in the left half. The reward is similar to the non-convex coverage, but the agents are only rewarded for covering the right side of the environment. The early termination is similar, but is only triggered if at least one agent has reached the right side. Otherwise, the episode ends after a fixed time.

We train the cooperative policy for 2M training environment time steps and the adversarial policy for 15M time steps. We perform an evaluation with a fixed episode length $T = \lceil (W \cdot H)/2 \rceil = 288$ which is the time required to cover every cell in one half for a single agent.

Path planning. In the path planning scenario, each agent is required to navigate to a labelled goal $\mathbf{g}_t^i \in \mathbb{N}^2$. In contrast to the coverage scenarios, only one agent can occupy a single cell at the same time. Since each agent is only aware of the relative environment layout and its own goal position, but not of other agent's positions, the agents have to communicate to coordinate and determine the most efficient set of paths.

The grid-world has a size of $W = H = 12$. The environment resembles a warehouse layout with obstacles aligned in a regular grid. The agents are placed uniformly at random locations in the environment. Each agent receives a local reward of 1 at each time step t if $\mathbf{p}_t^i = \mathbf{g}_t^i$. Each episode has a horizon of $T = 50$ time steps. Therefore the maximal possible mean reward per agent is T if each agent immediately reaches its goal position.

We train the cooperative policy for 16M training environment time steps and the adversarial policy for 35M time steps. The episode length of T during evaluation is similar to during training.

White-Box Analysis. We collect 50K training, 10K validation and 5K testing samples. To reduce correlation between samples, we sample at every time step t with a probability of 10% and discard all other samples.

The neural network architecture f_ψ^{-1} consists of multiple combinations of 2D convolutions, LeakyReLU activation, 2-upsampling and zero padding where the specific hyperparameters depend on the input feature size and the desired output size. The final convolution has one output channel and a sigmoid activation. We use a binary cross-entropy loss. We are only interested in the features that are essential for an efficient collaboration between agents and therefore mask the loss and evaluation metrics correspondingly.

We train with a batch size of 64 for 200 training epochs. Every 10 training epochs, we evaluate the classification performance as mAP on the validation set. We avoid overfitting by check-pointing the model with the best mAP computed on the validation set. All interpreter visualizations in this paper are created from a subset of the testing samples.