MAGIC: Multi-Agent Graph-Attention Communication

Yaru Niu*, Rohan Paleja*, and Matthew Gombolay
Georgia Institute of Technology
Atlanta, GA
yaruniu@gatech.edu, rpaleja3@gatech.edu, matthew.gombolay@cc.gatech.edu

Abstract

High-performing teams learn effective communication strategies to judiciously share information and reduce the cost of communication overhead. Within multi-agent reinforcement learning, synthesizing effective policies requires reasoning about when to communicate, whom to communicate with, and how to process messages. We propose a novel multi-agent reinforcement learning algorithm, Multi-Agent Graph-attentIon Communication (MAGIC), with a graph-attention communication protocol in which we learn 1) a Scheduler to help with the problems of when to communicate and whom to address messages to, and 2) a Message Processor using Graph Attention Networks (GATs) with dynamic graphs to deal with communication signals. The Scheduler consists of a graph attention encoder and a differentiable attention mechanism, which outputs dynamic, differentiable graphs to the Message Processor, which enables the Scheduler and Message Processor to be trained end-to-end. We evaluate our approach on a variety of cooperative tasks, including Google Research Football. Our method outperforms baselines across all domains, achieving $\approx 10.5\%$ *increase in reward in the most challenging domain. We also show MAGIC communicates* 27.4% *more efficiently on average than baselines, is robust to stochasticity, and scales to larger state-action spaces. Finally, we demonstrate MAGIC on a physical, multi-robot testbed. We release our codebase at https://github.com/CORE-Robotics-Lab/MAGIC.*

1. Introduction

Communication is a key component of successful coordination, enabling the agents to convey information and cooperate to collectively achieve shared goals [26, 30]. In high-performing human teams, human experts judiciously choose when to communicate and whom to communicate with, communicating only when beneficial [6, 37, 29]. Each team member exhibits the role of a communicator and message receiver, relaying information to the right teammates and incorporating received information effectively.

There has been recent success in MARL for Multiplayer Online Battle Arena (MOBA) games such as StarCraft II and Dota II [39, 3, 2]. MARL seeks to enable agents to share information to improve team performance [34, 36, 8, 7, 27, 40]. However, most prior work in MARL fails to capture the complex relations among agents, leading to low-performance and inefficient communication. While [34] and [17] are able to efficiently decide when to broadcast messages, agents will broadcast these messages to all other agents without targets, resulting in wasteful communication. Even with targeted communication [7], failure to assess when to communicate results in poor performance, as we display in Section 6. However, determining when to communicate and whom to communicate with is not enough. Selectively utilizing received messages can significantly improve performance. Yet, none of these methods simultaneously address "when" and with "whom", and "how" to communicate while modeling agent interaction topology.

In this paper, we propose Multi-Agent Graph-attentIon Communication (MAGIC), a novel graph communication protocol that determines "when" and "whom" with to communicate via an end-to-end framework. We set a new state-of-the-art in communication-based multi-agent reinforcement learning by modeling the topology of interactions among agents (the local and global characterization of connections between agents [1]) as a dynamic directed graph that accommodates time-varying communication needs and captures the relations between agents. Our proposed framework emulates the features of an effective human-human team through its key components, 1) the Scheduler, which helps each agent to decide when it should communicate and whom it should communicate with, and 2) the Message Processor, which integrates and processes received messages in preparation for decision making. We find MAGIC produces high-performance, cooperative behavior through its efficient communication protocol.

Our Scheduler consists of a graph attention encoder and a differentiable hard attention mechanism to decide when to communicate and whom to communicate with. This infor-

mation is encoded within a directed graph, allowing us to represent the interaction among agents precisely. The Message Processor consisting of a Graph Attention Network (GAT), utilizes received messages and the directed graph to intelligently and efficiently process messages. The encoded messages are then used in each agent's policy, leading to high-performance cooperation and efficient communication, as shown in Section 6. We provide the following contributions:

1. Develop a novel graph-attention communication protocol for MARL that utilizes 1) a Scheduler to solve the problems of when to communicate and whom to address messages to, and 2) a Message Processor using GATs with dynamic directed graphs to integrate and process messages.
2. Enable GATs in the Message Processor to maintain gradients from graph-based operations, which is not supported by standard GATs. In this way, the framework is fully differentiable and can be trained in an end-to-end manner.
3. Outperform prior methods across three domains, including the Google Research Football environment, achieving a 10.5% increase in reward. Further, MAGIC learns to communicate 27.4% more efficiently than the average baseline. These results set a new state-of-the-art in communication-based MARL.
4. Demonstrate our algorithm on physical robots in a 3-vs.-2 soccer scenario on a physical, multi-robot testbed (Section 2 in the supplementary).

2. Related Work

Coordinating multi-agent teams is a challenging computational problem [10, 12, 11, 18, 32, 35]. In multi-agent settings, each agent observes other agents as part of the environment, causing the environment to appear dynamic and non-stationary. Further difficulty arises due to the issue of credit assignment, where it is difficult for each agent to deduce its own contribution to the team's success (especially when there are only global rewards). To solve these multi-agent challenges, many researchers in MARL [5] have pursued centralized training and decentralized execution. Further extensions allow agents to exchange messages during execution, allowing for increased performance. Here, we present recent work in MARL.

MARL with Centralized Critic – Some works extend variants of actor-critic algorithms to multi-agent settings and learn decentralized policy through centralized critics without explicit communication channels [23, 9, 13]. MADDPG [23] is a MARL framework based on Deep Deterministic Policy Gradient, and can be applied in both cooperative and competitive scenarios. COMA [9] extends on-policy actor-critic and proposes a counterfactual baseline to address the credit assignment problem. While these works present critical improvements in the field of MARL, the ability to communicate and process information allows for much-increased performance. As we show in Section 6.4, the ability to communicate results in a 88.9% performance gain for our method.

MARL with Communication – Recent works have enabled agents to communicate and exchange messages during execution. Differentiable Inter-Agent Learning (DIAL) [8] builds up limited-bandwidth differentiable discrete communication channels among agents. CommNet [36] extends to a continuous communication protocol designed for fully cooperative tasks. Agents receive averaged encoded hidden states from other agents and use the messages to make decisions. However, utilizing a sum or average of messages results in some information loss. IC3Net [34] uses a gating mechanism to enable the agents to decide when to communicate, and thus is amenable to competitive scenarios. However, both IC3Net and CommNet process messages with a simple average. The proper integration of these messages is critically important for communication, as displayed by the performance of our results in Section 6. TarMAC [7] achieves targeted communication with a signature-based soft-attention mechanism. The integrated signal for each agent is the weighted mean of values generated by all agents. IMAC [40] uses a scheduler to aggregate compact messages through reweighting all agents' messages, and achieves the state-of-the-art performance on multi-agent communication under limited bandwidth. TarMAC and IMAC do not explicitly consider "when" and "whom" or the topology of agent interactions, which can help save communication resources and process messages efficiently. ATOC [16], employs an attention mechanism to decide if an agent should communicate in its observable field. SchedNet [17] proposes a weight-based scheduler to pick agents who should broadcast their messages. However, both ATOC and SchedNet have to manually configure their communication groups. Our communication protocol intelligently decides when and with whom to communicate through a graph-attention based Scheduler resulting in a large performance gain ($\approx 38\%$ average increase in reward in our most difficult domain) compared to prior work.

MARL using Graph Neural Networks Graph Neural Networks (GNNs) are powerful tools for learning from data with graph structures [31, 41, 42, 43]. To model the interactions between agents, MARL has utilized GNNs to allow for a graph-based representation [33, 15, 24, 20]. DGN [15] represents the multi-agent environment as a graph and employ multi-head dot-production attention as the convolutional kernel to extract relational features between agents. MAGNet [24] learns multi-agent policies in the Pommerman game by utilizing a relevance graph and message passing mechanism. The graphs are static and constructed based on heuristic rules. [33] learns a hierarchical topology of

the communication structure dynamically by electing central agents. HAMA [28] designs a hierarchical graph attention network to model the hierarchical relationships between agents in both cooperative and competitive scenarios. G2ANet [22] combines a hard-attention and a soft attention mechanism to dynamically learns interactions between agents. In this work, we modify standard graph attention networks to be compatible with a differentiable directed graph, allowing us to represent the interactions among agents more accurately during communication.

We improve upon prior frameworks by utilizing a Scheduler to solve the problems of when to communicate and whom to address messages to, and a Message Processor using GATs with dynamic directed graphs to integrate and process messages. In this way, we achieve efficient and targeted message sending and high-performance message comprehension.

3. Preliminaries

3.1. Partially Observable Markov Game

A Markov Game [21] is the multi-agent version of Markov Decision Process (MDP). We are primarily concerned with a partially observable Markov game. A partially observable Markov game (POMG) for N agents can be defined by a set of global states, S, a set of private observations for each agent, $O_1, O_2, \ldots, O_N$, a set of actions for each agent, $A_1, A_2, \ldots, A_N$, and the transition function, $T : S \times A_1 \times \ldots \times A_N \mapsto S$. In each time step, agent i chooses action, $a_i \in A_i$, obtains reward as a function of state, S, and its action $r_i : S \times A_i \mapsto \mathbb{R}$, and receives a local observation $o_i : S \mapsto O_i$. The initial state is defined by a initial state distribution ρ. Agent i aims to maximize its discounted reward $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$, where $\gamma \in [0, 1]$ is a discounted factor. Our work is based on the framework of POMG augmented with communication.

3.2. Reinforcement Learning: Policy Gradients

The Policy Gradient method (Equation 1) is widely used in reinforcement learning (RL) tasks to perform gradient ascent on the agent policy parameters, θ, to optimize the total discounted reward, $J(\theta) = \mathbb{E}_{s\sim p^\pi, a\sim\pi_\theta}[R]$. ρ^π is the state distribution, π_θ is the policy distribution, and $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}, a_{t'})$.

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta}\big[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) R_t\big] \quad (1)$$

In lieu of R_t, we use the advantage function, $A^\pi(s_t, a_t) = R_t - V(s_t)$, to decrease the variance of the estimated policy gradient, where $V(s_t)$ is the value function.

3.3. Graph Neural Networks

In Graph Neural Networks (GNNs), each GNN layer computes the node representation by message passing, where each node aggregates the feature vectors from its neighboring nodes in the graph at the previous layer. The update rule for node representations by a GNN layer is displayed in Equation 2.

$$h_i^{(l)} = \sigma\left(\sum_{j\in N_i} \frac{1}{\sqrt{d_i d_j}} (h_j^{(l-1)} W^{(l)})\right) \quad (2)$$

Here, $h_i^{(l)}$ represents the features of node i, at layer l. N_i is the set of neighboring nodes of node i, $d_i = |N_i|$ is the degree of node i, $W^{(l)}$ is a learnable weighting matrix for layer l, and $\sigma(\cdot)$ is a nonlinear activation function. In this paper, we enable our GATs within the Message Processor to function with differentiable graphs (Section 4.3), allowing for end-to-end training.

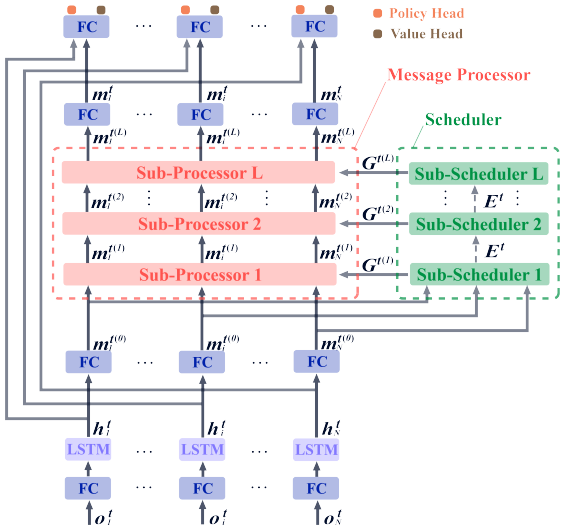


Figure 1. This figure displays the framework of our multi-agent graph-attention communication protocol.

4. Method

In this section, we introduce our proposed Multi-Agent Graph-attentIon Communication protocol, MAGIC. We consider a partially observable setting of N agents, where agent i receives local observation, o_i^t, at time, t, containing local information from the global state, S. The agent, i, learns a communication-based policy, π_i, to output a distribution over actions, $a_i^{(t)} \sim \pi_i$, at each time step, t. Here, we present an overview of our framework, the description of our protocol's key components (i.e., the Scheduler and Message Processor), and our training procedure.

4.1. Overview

Our proposed graph-attention communication protocol is displayed in Figure 1. At time step, t, the observation for each agent, o_i^t, is first encoded using an agent-specific fully-connected layer (FC). The encoded observation is passed

into an agent-specific LSTM cell to generate a hidden state, h_i^t, as shown in Equation 3.

$$h_i^t, c_i^t = LSTM(e(o_i^t), h_i^{t-1}, c_i^{t-1}) \tag{3}$$

In this equation, c_i^t is the cell state for agent, i, at time step, t, and $e(\cdot)$ is a fully-connected layer acting as an encoder for the observation. The hidden state, h_i^t, is then encoded as a message, $m_i^{t(0)} = e_m(h_i^t)$, through the encoder, $e_m(\cdot)$ (a fully-connected layer). Here, the exponent notation for the message, $m_i^{t(0)}$, denotes that message is for agent i, and is prior to any message aggregation or processing. We refer to this stage, where the message has not been processed, as round 0, giving the exponent notation, $t(0)$.

As shown in Figure 1, we define the function module to help agents decide whom to send messages at each time step as the "Scheduler" and define the function module to process messages as "Message Processor." The Scheduler and the Message Processor may include multiple sub-schedulers and sub-processors, respectively. Prior work has termed the procedure of processing messages for multiple iterations as multi-round communication [7]. As multi-round communication has been shown to improve performance, our protocol supports L rounds of communication, where $L \in \mathbb{N}$. A round of communication, l, is defined as a forward pass through a sub-scheduler and sub-processor. As shown in Figure 1, the encoded messages, $m_i^{t(0)}$ are passed into Sub-Scheduler 1 and Sub-Processor 1 (i.e., the sub-scheduler and sub-processor at round 1).

The Sub-Scheduler l (at round, $l \in L$) will output an adjacency matrix, $G^{t(l)}$. $G^{t(l)}$ is a directed graph that indicates the targeted receivers for each agent at time step, t. $G^{t(l)}$ is utilized by the Sub-Processor, l, to produce a set of integrated messages, $\{m_i^{t(l)}\}_1^N$, where $m_i^{t(l)}$ is the integrated message for agent, i, at time step, t. The integrated messages for each agent, i, can be incorporated into agent i's policy (in the case where we are on the last round of communication, $l = L$) or be further processed by more rounds of communication ($l < L$). If the messages are to be further processed, the set of messages outputted from round l, $\{m_i^{t(l)}\}_1^N$, are passed into the Sub-Scheduler $l+1$ and the Sub-Processor $l+1$, producing adjacency matrix $G^{t(l+1)}$ and messages $\{m_i^{t(l+1)}\}_1^N$ respectively.

The message outputted from the Message Processor, $m_i^{t(L)}$, for agent, i, is encoded through a fully-connected layer, $e'_m(\cdot)$, to produce an intelligently integrated message, $m_i^t = e'_m(m_i^{t(L)})$. m_i^t is concatenated with the hidden state, h_i^t, to produce the input feature to the policy head and the value head. The policy head is a fully-connected layer followed by a softmax function. We sample the action for the i-th agent at time step, t, from the policy output distribution: $a_i^t \sim \pi_i(a_i^t|o_i^t)$. The value head is a single fully-connected layer and serves as a baseline function.

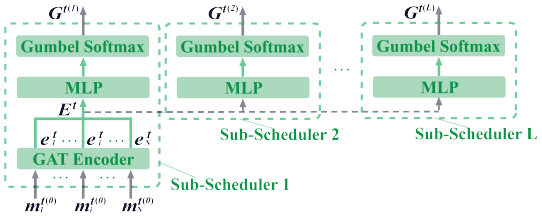


Figure 2. This figure displays the details and components of the Scheduler.

4.2. The Scheduler

The Scheduler decides when each agent should send messages and whom each agent should address messages to, as shown in Figure 2. As a black-box, the Scheduler takes as input the encoded messages, $\{m_i^{t(0)}\}_1^N$, and outputs the directed graphs, $\{G^{t(l)}\}_1^L$, as represented in $f_{Sched}(\cdot)$ shown in Equation 4.

$$\{G^{t(l)}\}_1^L = f_{Sched}\left(m_1^{t(0)}, \cdots, m_N^{t(0)}\right) \tag{4}$$

As noted in Section 4.1, the Scheduler consists of L Sub-Schedulers, each producing an adjacency matrix $G^{t(l)}$. A Sub-Scheduler consists of a GAT encoder and a hard attention mechanism that uses a multi-layer perceptron (MLP) and a Gumbel Softmax function [14]. The GAT encoder helps encode local or global information for an agent efficiently, and it is only used in the first Sub-Scheduler. We adopt the same form of GATs as proposed in [38], where the attention mechanism is expressed in Equation 5.

$$\alpha_{ij}^S = \frac{exp\left(LReL\left(a_S^T[W_S m_i^{t(0)} || W_S m_j^{t(0)}]\right)\right)}{\sum_{k \in N_i^t \cup i} exp\left(LReL\left(a_S^T[W_S m_i^{t(0)} || W_S m_k^{t(0)}]\right)\right)} \tag{5}$$

Here, $LReL(\cdot)$ is the LeakyReLU [25], $a_S \in R^{D'}$ is a weighting vector, $N_i^t \cup i$ is the set of neighboring agents for the i-th agent, including agent i, at time step, t, and $W_S \in R^{D' \times D}$ is a weighting matrix, where D' and D refer to the output feature cardinality and message cardinality, respectively. The node features of each agent are obtained through Equation 6.

$$e_i^t = ELU\left(\sum_{j \in N_i^t \cup i} \alpha_{ij}^S W_S m_j^{t(0)}\right) \tag{6}$$

Here, $ELU(\cdot)$ is the Exponential Linear Unit (ELU) function. After concatenating the node feature vectors pairwise, supposing $E_{i,j}^t = (e_i^t || e_j^t)$, we obtain a matrix $E^t \in \mathbb{R}^{N \times N \times 2D}$, where $E_{i,j}^t$ represents a high-level representation of relational features between the i-th and j-th agent. Setting E^t as the input to an MLP followed by a Gumbel Softmax function, we can get an adjacency matrix $G^{t(l)}$, consisting of binary values, representing a directed graph. If the element $g_{ij}^{t(l)}$ in $G^{t(l)}$ is 1, the j-th agent will send a

message to i-th agent. Otherwise ($g_{ij}^{t(l)} = 0$), the j-th agent will not send any message to i-th agent.

4.3. The Message Processor

The Message Processor helps agents integrate messages for intelligent decision making. As a black-box, it takes in the encoded messages, $\{m_i^{t(0)}\}_1^N$, and the graphs generated by the Scheduler, $G^{t(1)} \cdots, G^{t(L)}$, and outputs the processed messages, $\{m_i^{t(L)}\}_1^N$. We represent the Message Processor in Equation 7.

$$\{m_i^{t(L)}\}_1^N = f_{MP}\left(m_1^{t(0)}, \cdots, m_N^{t(0)}, G^{t(1)}, \cdots, G^{t(L)}\right) \quad (7)$$

As stated in Section 4.1, the Message Processor consists of L Sub-Processors, each producing a set of encoded messages, $\{m_i^{t(l)}\}_1^N$. A Sub-Processor, for a single round of communication, includes a designed GAT layer receiving messages, $\{m_i^{t(l-1)}\}_1^N$, from all agents and the adjacency matrix, $G^{t(l)}$, as input, as shown in Figure 1 in the supplementary. The Sub-Processor helps agents process received messages. In Equation 8, we display the calculation of the attention coefficient in our designed GAT layer, which maintains the gradient from the Scheduler.

$$\alpha_{ij}^P = \frac{g_{ij}^{t(l)} exp\left(LReL\left((a_P^{(l)})^\top [W_P^{(l)} m_i^{t(l-1)} || W_P^{(l)} m_j^{t(l-1)}]\right)\right)}{\sum_{k=1}^N g_{ik}^{t(l)} exp\left(LReL\left((a_P^{(l)})^\top [W_P^{(l)} m_i^{t(l-1)} || W_P^{(l)} m_k^{t(l-1)}]\right)\right)} \quad (8)$$

Here, l is the round of communication, $g_{ij}^{t(l)} \in \{0, 1\}$is a binary value in the adjacency matrix, $G^{t(l)}$, $W_P^{(l)} \in R^{D'' \times D}$is a weighting matrix, and $a_P^{(l)} \in R^{D''}$is a weighting vector. It should be noted that our graphs are capable of a self-loop, where an agent will "send" a message to itself, and use its own message in the integration of received messages. While the calculation of the coefficient for a standard GAT layer is a non-differentiable operation for the graph, using equation 8 allows us to retain the gradient of $g_{ij}^{t(l)}$. Thus, the Scheduler can preserve the gradient flow for end-to-end training, avoiding the need to design an extra loss function to train the Scheduler. In practice, we find using multi-head attention [38] and adding a bias to the output message to be beneficial. The output message of sub-processor l can be obtained via Equation 9.

$$m_i^{t(l)} = ELU\left(\sum_{j=1}^N \alpha_{ij}^P W_P^{(l)} m_j^{t(l-1)}\right) \quad (9)$$

4.4. Training

In our experiments, the parameters of the fully-connected layers and LSTM in the policy network are shared across homogeneous agents to improve training efficiency. We employ a multi-threaded synchronous multi-agent policy gradient [34] and utilize an extra value head in the policy network to estimate the value function, $V_\phi(o_i^t)$, at observation o_i^t, which will serve as a state-independent baseline. In addition to optimizing the discounted total reward with policy gradient, the model also minimizes the squared error between the estimated value and the Monte-Carlo estimate. The two loss functions are balanced by a coefficient, β. We define the overall loss function as $\mathcal{L}(\cdot)$ and the policy function denoted as $\pi_\theta(a_i^t|o_i^t)$. Parameters, θ, of the policy and, ϕ, of the value function, share most of their parameters except the parameters in the policy and value heads. Our model is updated via minimizing the loss function displayed in Equation 10.

$$\nabla_{\theta,\phi}\mathcal{L}(\theta,\phi) = \frac{1}{t_{max}} \sum_{i=1}^N \sum_{t=1}^{t_{max}} [-\nabla_\theta \log \pi_\theta(a_i^t|o_i^t)(R_i^t - V_\phi(o_i^t)) + \beta \nabla_\phi (R_i^t - V_\phi(o_i^t))^2] \quad (10)$$

Here, R_i^t is the discounted total reward for agent i in an episode, and t_{max} is the number of steps taken within a batch. Different threads in training share the parameters, θ, and ϕ, and calculate their own gradients. The threads synchronously accumulate gradients and update θ and ϕ within each batch. A summary of the training procedure of our multi-agent graph-attention communication model and the algorithm is described in Section 1 of the supplementary.

5. Evaluation Environments

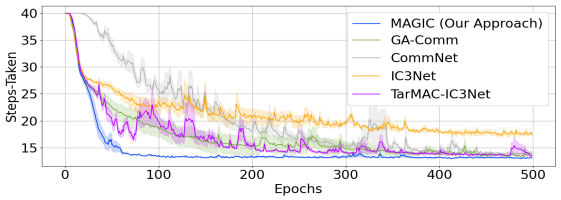
We utilize three domains, including the Predator-Prey [34], Traffic Junction [36] and complex Google Research Football environment [19], to evaluate the utility of proposed communication protocol. Predator-Prey and Traffic Junction are common MARL benchmarks [36, 7]. Google Research Football (GRF) presents a difficult challenge, as it has sparse rewards, stochasticity, and adversarial agents.

5.1. Predator-Prey

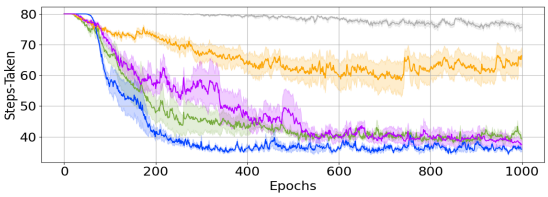
We utilize the predator-prey environment from [34]. Here, there are N predators with limited visions searching for a stationary prey. The predators can take actions up, $down$, $left$, $right$ or $stay$. We utilize the "mixed" mode of Predator-Prey in which the predator incurs a reward -0.05 for each time step until the prey is found. An episode is defined as successful if all the predators find the prey before a predefined maximum time limit. We create two levels of difficulty in this environment. The difficulty varies as the grid size, and the number of predators increase, as more coordination is required to achieve success. The corresponding grid sizes and the number of predators are set to 10×10 with 5 predators and 20×20 with 10 predators. We define a higher-performing algorithm in this domain as one that minimizes the average steps to complete an episode.

5.2. Traffic Junction

The Traffic Junction environment, composed of intersecting routes and cars (agents) with limited vision, requires communication to avoid collisions. Cars enter the traffic junction with a probability p_{arrive}. The maximum number



(a) Low difficulty Predator-Prey Environment with size 10×10 and 5 agents



(b) High difficulty Predator-Prey Environment with size 20×20, 10 agents

Figure 3. This figure displays the average steps taken to finish an episode as training proceeds in each level of the Predator-Prey environment. The shaded regions represent standard error. A lower value for steps taken on the vertical axis is better.

of cars in the environment at a specific time is denoted as N_{max}, which varies across difficulty levels. A car occupies one cell at a time step and can take action "gas" or "brake" on its route.

We validate our algorithm on three difficulty levels. The easy level consists of two, one-way roads on a 7×7 grid with at most five agents ($N_{max} = 5, p_{arrive} = 0.3$). For the medium, the junction consists of two, two-way roads on a 14×14 grid with at most ten agents in the domain ($N_{max} = 10, p_{arrive} = 0.2$). Hard consists of four, two-way roads on a 18×18 grid with at most twenty agents in the domain ($N_{max} = 20$, $p_{arrive} = 0.05$). The success rate (i.e., no collisions within an episode) is used in our evaluation.

5.3. Google Research Football

Our final domain of Google Research Football [19] presents a challenging, mixed cooperative-competitive, multi-agent scenario with high stochasticity and sparse rewards. Google Research Football (GRF) is a physics-based 3D soccer simulator for reinforcement learning. This last domain presents an additional challenge as there are opponent artificial agents (AIs), significantly increasing the complexity of the state-action space. GRF provides 19 actions including moving actions, kicking actions, and other actions such as dribbling, sliding and sprint. GRF provides several pre-defined reward signals, consisting of a scoring and a penalty box proximity reward. The penalty box proximity reward is shaped to push attackers to move forward towards certain locations. Many MARL frameworks have required these highly shaped rewards functions to perform well [19]. However, we choose to use only the scoring reward to verify the ability for our algorithm and baselines to function in a high-complexity stochastic domain with sparse rewards. **Accordingly, the only reward all agents will receive in our evaluation is $+1$ when scoring a goal.** The termination criterion is the team scoring, ball out of bounds, or possession change. We evaluate algorithms in a standard scenario 3 vs. 2 from Football Academy [19], where we have 3 attackers vs. 1 defender, and 1 goalie. The three offending agents are controlled by the MARL algorithm, and the two defending agents are controlled by a built-in AI. We find that utilizing a 3 vs. 2 scenario challenges the robustness of MARL algorithms to stochasticity and sparse rewards. In this domain, we seek to maximize the average success rate (i.e., a goal is scored) and minimize the average steps taken to complete an episode, thereby scoring a goal in the shortest amount of time. We show in section 6.3 that our method outperforms all prior state-of-the-art baselines.

6. Results and Discussion

In this section, we evaluate the performance of our proposed method on three environments, including Predator-Prey [34], Traffic Junction [36], and Google Research Football [19]. We benchmark our approach against a variety of state-of-the-art communication-based MARL baselines, including CommNet [36], IC3Net [34], GA-Comm [22], and TarMAC-IC3Net [7]. We implement our method and baselines on each task, averaging the best performance at convergence over 5 random seeds. Following an analysis of performance, we evaluate MAGIC's communication efficiency, concluding MAGIC presents a new state-of-the-art in both performance and efficiency in communication-based MARL. We provide additional training details within Section 4 of the supplementary.

6.1. Predator-Prey

Figure 3 depicts the average steps taken for the predators to locate the prey. In both the five- and ten-agent cases, our method converges faster and can achieve better performance than the baselines. Our method converges 52% faster than the next-quickest baseline while still achieving the highest performance. Table 1 shows the results of average steps taken to reach the prey at convergence. While approaches

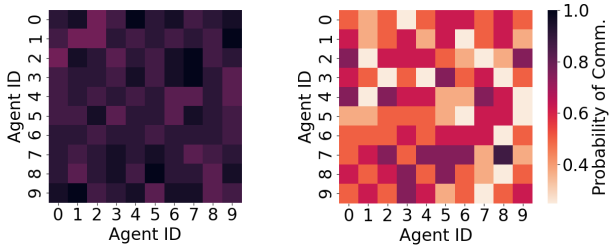
Table 1. The number of steps taken to complete an episode at convergence in Predator-Prey.

Method	10×10, 5 agents	20×20, 10 agents
MAGIC (Our Approach)	**12.72 ± 0.03**	**32.88 ± 0.14**
CommNet [36]	13.16 ± 0.04	73.12 ± 0.68
IC3Net [34]	15.60 ± 0.35	55.13 ± 4.80
TarMAC-IC3Net [7]	13.32 ± 0.11	36.16 ± 0.97
GA-Comm [22]	13.06 ± 0.09	35.78 ± 0.37

Table 2. The success rate at convergence in Traffic Junction.

Method	7×7, $N_{max} = 5$, $p_{arrive} = 0.3$	14×14, $N_{max} = 10$, $p_{arrive} = 0.2$	18×18, $N_{max} = 20$, $p_{arrive} = 0.05$
MAGIC (Our Approach)	**99.9 ± 0.1 %**	**99.9 ± 0.1 %**	**98.0 ± 0.8 %**
CommNet [36]	99.3 ± 0.6%	97.2 ± 0.3%	66.7 ± 1.6%
IC3Net [34]	97.8 ± 1.0%	96.0 ± 0.7%	85.4 ± 2.5%
TarMAC-IC3Net [7]	84.8 ± 4.5%	95.5 ± 1.3%	88.1 ± 1.9%
GA-Comm [22]	95.9 ± 0.1%	97.1 ± 0.7%	95.8 ± 1.1%

such as GA-Comm and TarMAC-IC3Net learn competitive policies for the five agent case, these benchmarks perform much worse than our algorithm in the ten agent case, suggesting a lack of scalability.



(a) Before prey found. (b) After prey found.

Figure 4. Heatmaps of the communication graphs learned by the Scheduler in the Predator-Prey domain.

Predator-Prey Communication Heatmaps - Figure 4 depicts communication heatmaps in Predator-Prey domain with 10 agents in an episode of 31 steps. The color is associated with the probability of communication, with darker colors representing more intensive communication between the two agents. The vertical axis represents message receivers, and the horizontal axis represents message senders. Agent 5 first reaches the prey at step 23, and the other 9 agents quickly reach the prey in the following 7 steps. Figure 4(a) displays the communication before the first agent (agent 5) reaches its prey and 4(b) displays the communication afterwards. We can see that agents communicate with each other intensively before finding the prey. After finding the prey, communication becomes unnecessary, and the learned communication graphs should be sparse, as we see in Figure 4(b). This inspection supports that MAGIC learns to communicate only when beneficial to team performance.

6.2. Traffic Junction

We evaluate our method in the Traffic Junction domain for the cases of a maximum of 5 agents, 10 agents and 20 agents at a junction. Table 2 shows the success rate (i.e., no collision in an episode) for each method at convergence in Traffic Junction. Our algorithm achieves near-perfect performance after convergence, widely outperforming all benchmarks in its success rate. Figure 5 depicts the average number of epochs taken to converges for each method. Our method maintains quick convergence as the number of

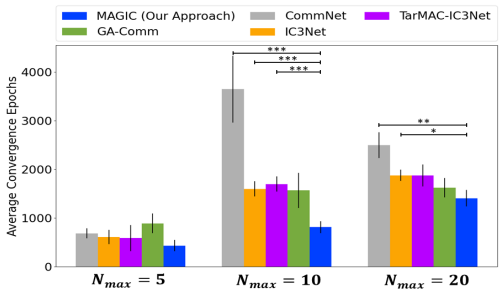


Figure 5. The average number of epochs for convergence in Traffic Junction with standard error bars.

Table 3. The success rate and average steps taken to finish an episode in GRF.

Method	Success Rate	Steps Taken
MAGIC (Our Approach)	**98.2 ± 1.0%**	**34.30 ± 1.34**
Ours (without the Scheduler)	91.0 ± 4.6%	36.31 ± 2.59
CommNet [36]	59.2 ± 13.7%	39.32 ± 2.35
IC3Net [34]	70.0 ± 9.8%	40.37 ± 1.22
TarMAC-IC3Net [7]	73.5 ± 8.3%	41.53 ± 2.80
GA-Comm [22]	88.8 ± 3.9%	39.05 ± 3.05

agents increase. However, several of the benchmarks experience a slowdown in their convergence rate.

Impact of the Message Processor - In Traffic Junction, we allow all agents to communicate to accelerate training for all methods, common in highly vision-limited environments [34]. As we are using complete graphs (i.e., Scheduler is not used), our SOTA performance in the Traffic Junction domain displays that the MP considerably contributes to the success of our algorithm.

6.3. Google Research Football

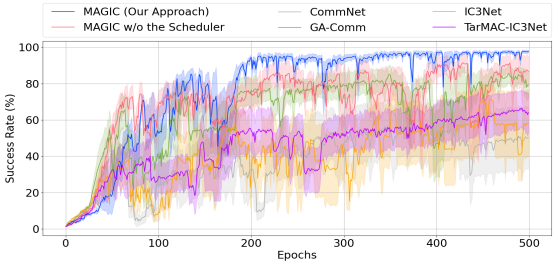


Figure 6. The success rate in GRF as training proceeds.

Lastly, we evaluate our algorithm and several state-of-the-art communication-based MARL baselines in the GRF environment. The results presented provide some insight into each algorithm's ability to handle stochasticity, sparse rewards, and a high-complexity state-action space. We utilize the scoring success rate as our metric of evaluation.

Impact of the Scheduler - We verify the impact of the Scheduler mechanism in MAGIC by testing our method without the use of the Scheduler. As seen, the addition of a Scheduler provides a performance improvement. This

result signifies the importance of determining “when" and “whom" to communicate with.

Figure 6 displays the success rate for offending agents to score as the training proceeds. Our method converges to a higher-performing policy than all the baselines. In our settings, although each agent only has local observations, it is able to completely observe the state space. MAGIC without the ability to utilize a scheduler performs poorly. It is interesting to note that even with unlimited vision, utilizing a complete graph for communication performs much worse than utilizing a scheduler that gives precise and targeted communication. Table 3 displays the success rate and average steps taken to finish an episode at convergence. Our method has a success rate of 98.5%, which is approximately a 10.5% improvement over the next-best baseline of GA-Comm. Our method achieves the lowest number of average steps, scoring 13.8% more quickly than the closest benchmark of GA-Comm. As we have have outperformed all benchmarks within the previous two domains and in the complex GRF environment, we conclude that MAGIC is high-performing, scales well to the number of agents, robust to stochasticity, and performs well with sparse rewards.

6.4. Communication Efficiency

We present an analysis of the communication efficiency of our method in Table 4. Communicating efficiently can save resources and allow for messages to be processed more easily. To gauge the efficiency, we utilize the performance improvement due to communication and divide the communication graph density. The graph densities are determined using the sparsity of the adjacency matrix. A fully-connected graph corresponds to a density of 1, and a graph with no connections corresponds to a density of 0. We perform this analysis within a Predator-Prey domain of grid size $5x5$ with 3 predators, where a performance improvement refers to a reduction in steps. To obtain the performance improvement due to communication, we evaluate a communication-blocked variant of each method and compare the performance to the method itself. All methods use the same message size and one-round of communication to maintain a similar network complexity. Utilizing the performance improvement due to communication divided by the graph density as our metric for communication efficiency, we see that MAGIC is the most efficient. MAGIC communicates 27.4% more efficiently on average than baselines while also achieving the highest performance.

Impact of the combined framework of MAGIC - As we compare each method to its non-communicatory variant, in MAGIC, this results in removing the Scheduler and Message Processor. MAGIC achieves the greatest improvement compared to baselines, displaying the contribution from the combined effects of the Scheduler and Message Processor.

Table 4. Communication efficiency measured as the performance improvement with communication divided by graph density.

Method	Graph Density	Avg. Steps w/ Comms	Performance Improvement	$\frac{\text{Improvement}}{\text{Density}}$
MAGIC (Our Approach)	**0.644**	**8.504**	**7.562**	**11.743**
CommNet [36]	0.667	9.216	6.455	9.681
IC3Net [34]	0.638	9.208	6.421	10.058
TarMAC-IC3Net[7]	0.856	9.376	5.958	6.956
GA-Comm [22]	0.514	9.334	5.868	11.391

6.5. Discussion

Across multiple test domains, we set a new state-of-the-art in communication-based MARL performance, outperforming baselines, including [36, 34, 22, 7]. Across our domains, we achieve an average 5.8% improvement in steps taken (Predator-Prey), average 1.9% improvement in success rate (Traffic Junction), 10.5% improvement in success rate (GRF), and 13.8% improvement in steps taken compared to the closest benchmark across each domain. The strong performance improvement we achieve in GRF suggests our approach is better able to scale to high-dimension state-action spaces while effectively handling stochasticity and sparse rewards. While our architecture shares some attributes as GA-Comm [22], GA-Comm has several drawbacks including large epoch training times due to the complex hard attention structure, a dot-product soft attention mechanism *without scaling*, and the inability to extend to multi-round communication. Our approach takes significantly less compute by avoiding any recurrent structures in the Scheduler. Specifically, GA-Comm requires 2x long to train with 3 agents, 3x long with 5 agents, and 4x as long with 10 agents. Prior work [4] has also shown that additive attention (used in the GAT layers in our Message Processor) outperforms dot-product attention without scaling when the message size is large. MAGIC's Scheduler can explicitly learn and generate different graphs for different rounds of communication, allowing for higher performance. Additionally, MAGIC achieves the highest communication efficiency, outperforming benchmarks by 27.4% on average.

7. Conclusion

In this paper, we propose a novel, end-to-end-trainable, graph-attention communication protocol, MAGIC, that utilizes a Scheduler to solve the problems of when to communicate and whom to address messages to, and a Message Processor to integrate and process messages. We evaluate our method and baselines in several environments, achieving state-of-the-art performance. In GRF, we achieve a 98.5%, near-perfect success rate, while most baselines struggle to reach 70%. Not only does MAGIC produce SOTA results, MAGIC is able to converge 52% faster than the next-quickest baseline, and communicates 27.4% more efficiently than the average baseline.

References

[1] Dhaval Adjodah, Dan Calacci, Abhimanyu Dubey, Anirudh Goyal, Peter Krafft, Esteban Moro, and Alex Pentland. Communication topologies between learning agents in deep reinforcement learning. *arXiv preprint arXiv:1902.06740*, 2019.

[2] In-Chang Baek and Kyung-Joong Kim. Efficient multi-agent reinforcement learning using clustering for many agents. 2019.

[3] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[4] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc Le. Massive exploration of neural machine translation architectures. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1442–1451, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics.

[5] L. Busoniu, R. Babuka, and B. D. Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38:156–172, 2008.

[6] N. Cooke, J. Gorman, Christopher W. Myers, and J. Duran. Interactive team cognition. *Cognitive science*, 37 2:255–85, 2013.

[7] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 1538–1546. PMLR, 2019.

[8] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2137–2145, 2016.

[9] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 2974–2982. AAAI Press, 2018.

[10] M. Gombolay, Anna Bair, Cindy Huang, and J. Shah. Computational design of mixed-initiative human–robot teaming that considers human factors: situational awareness, workload, and workflow preferences. *The International Journal of Robotics Research*, 36:597 – 617, 2017.

[11] M. Gombolay, R. Jensen, Jessica Stigile, T. Golen, N. Shah, Sung-Hyun Son, and J. Shah. Human-machine collaborative optimization via apprenticeship scheduling. *ArXiv*, abs/1805.04220, 2018.

[12] M. Gombolay, R. Wilcox, and J. Shah. Fast scheduling of robot teams performing tasks with temporospatial constraints. *IEEE Transactions on Robotics*, 34:220–239, 2018.

[13] Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2961–2970. PMLR, 2019.

[14] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[15] Jiechuan Jiang, Chen Dun, and Zongqing Lu. Graph convolutional reinforcement learning for multi-agent cooperation. *arXiv preprint arXiv:1810.09202*, 2(3), 2018.

[16] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 7265–7275, 2018.

[17] Daewoo Kim, Sangwoo Moon, David Hostallero, Wan Ju Kang, Taeyoung Lee, Kyunghwan Son, and Yung Yi. Learning to schedule communication in multi-agent reinforcement learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[18] Dong-Ki Kim, Miao Liu, M. Riemer, Chuangchuang Sun, Marwa Abdulhai, G. Habibi, Sebastian Lopez-Cot, G. Tesauro, and Jonathon P. How. A policy gradient algorithm for learning to learn in multiagent reinforcement learning. *ArXiv*, abs/2011.00382, 2020.

[19] Karol Kurach, Anton Raichuk, Piotr Stanczyk, Michal Zajac, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, and Sylvain Gelly. Google research football: A novel reinforcement learning environment. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 4501–4510. AAAI Press, 2020.

[20] Sheng Li, J. Gupta, Peter Morales, R. Allen, and M. Kochenderfer. Deep implicit coordination graphs for multi-agent reinforcement learning. *ArXiv*, abs/2006.11438, 2020.

[21] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994.

[22] Yong Liu, Weixun Wang, Yujing Hu, Jianye Hao, Xingguo Chen, and Yang Gao. Multi-agent game abstraction via graph attention neural network. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7211–7218. AAAI Press, 2020.

[23] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6379–6390, 2017.

[24] Aleksandra Malysheva, Tegg Taekyong Sung, Chae-Bong Sohn, Daniel Kudenko, and Aleksei Shpilman. Deep multi-agent reinforcement learning with relevance graphs. *arXiv preprint arXiv:1811.12557*, 2018.

[25] Chigozie Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *ArXiv*, abs/1811.03378, 2018.

[26] Afshin Oroojlooyjadid and Davood Hajinezhad. A review of cooperative multi-agent deep reinforcement learning. *ArXiv*, abs/1908.03963, 2019.

[27] Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2017.

[28] Heechang Ryu, Hayong Shin, and Jinkyoo Park. Multi-agent actor-critic with hierarchical graph attention network. *arXiv preprint arXiv:1909.12557*, 2019.

[29] E. Salas, N. Cooke, and M. Rosen. On teams, teamwork, and team performance: Discoveries and developments. *Human Factors: The Journal of Human Factors and Ergonomic Society*, 50:540 – 547, 2008.

[30] E. Salas, T. Dickinson, Sharolyn A. Converse, and S. Tannenbaum. Toward an understanding of team performance and training. 1992.

[31] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[32] Esmaeil Seraj and M. Gombolay. Coordinated control of uavs for human-centered active sensing of wildfires. *2020 American Control Conference (ACC)*, pages 1845–1852, 2020.

[33] Junjie Sheng, Xiangfeng Wang, Bo Jin, Junchi Yan, Wenhao Li, Tsung-Hui Chang, Jun Wang, and Hongyuan Zha. Learning structured communication for multi-agent reinforcement learning. *arXiv preprint arXiv:2002.04235*, 2020.

[34] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[35] Laura G. Strickland, Charles E. Pippin, and Matthew Gombolay. *Learning to Steer Swarm-vs.-swarm Engagements*.

[36] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2244–2252, 2016.

[37] Güliz Tokadli and Michael C. Dorneich. Interaction paradigms: from human-human teaming to human-autonomy teaming. *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, pages 1–8, 2019.

[38] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[39] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[40] Rundong Wang, Xu He, Runsheng Yu, Wei Qiu, Bo An, and Zinovi Rabinovich. Learning efficient multi-agent communication: An information bottleneck approach. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 9908–9918. PMLR, 2020.

[41] Zheyuan Wang and M. Gombolay. Heterogeneous graph attention networks for scalable multi-robot scheduling with temporospatial constraints. In *RSS 2020*, 2020.

[42] Zheyuan Wang and M. Gombolay. Learning scheduling policies for multi-robot coordination with graph attention networks. *IEEE Robotics and Automation Letters*, 5:4509–4516, 2020.

[43] Zonghan Wu, Shirui Pan, Fengwen Chen, G. Long, C. Zhang, and P. Yu. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.

Supplementary for MAGIC: Multi-Agent Graph-Attention Communication

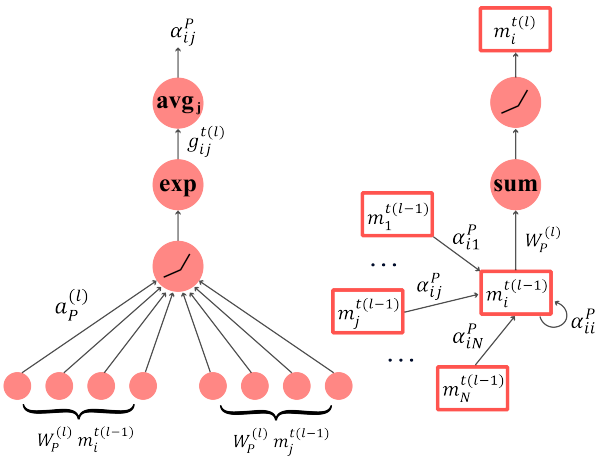


Figure 1. This figure displays the detailed structure of a Sub-Processor.

1. The Algorithm of MAGIC

In Algorithm 1, we start by initializing the number of agents alongside several training parameters, including threads, batch size, maximum steps in an episode, and maximum steps in a batch, shown in Step 1. For each update per thread, we start by initializing a set of thread parameters and a replay buffer, D, as displayed in Step 4. After receiving the initial hidden state and observation for each agent (shown in Steps 7 and 8), we can utilize the Scheduler (containing L sub-schedulers) to output adjacency graphs, determining the communication pattern. The Message Processor (containing L sub-processors) can use these graphs and the encoded messages from round zero to produce messages for each agent, as shown in Steps 13 and 14. Once each agent has its selected message inputs, we can determine an action probability distribution from the policy and perform the action sampled from this distribution, shown in Steps 15 and 16. Storing this information in our replay buffer, we can then complete the episode and proceed to compute gradients with Equation 10, as shown in Step 26. Accumulating gradients across threads, we can update our policy and value function, as shown in Steps 29 and 31.

Algorithm 1 Training Multi-Agent Graph-attentIon Communication (MAGIC)

1: Initialize max updates M, agents N, threads L, max steps in an episode T_e, max steps in a batch T_b
2: **for** update $= 1$ to M **do** $d\theta \leftarrow 0, d\phi \leftarrow 0$
3: **for** thread $k = 1$ to K **do in parallel**
4: Initialize params $\theta_k \leftarrow \theta$, $\phi_k \leftarrow \phi$, buffer D, step-count $t \leftarrow 1$
5: **while** $t < T_b$ **do**
6: Initialize thread step counter $t' \leftarrow 1$
7: Initialize h_i^{t-1}, c_i^{t-1} for each agent i
8: Reset environment and get o_i^t for each agent i
9: **while** $t' < T_e$ and not terminal **do**
10: $h_i^t, c_i^t = LSTM(e(o_i^t), h_i^{t-1}, c_i^{t-1})$ for each agent i
11: $m_i^{t(0)} = e_m(h_i^t)$ for each agent i
12: $\{G^{t(l)}\}_1^L = f_{Sched}\left(m_1^{t(0)}, \cdots, m_N^{t(0)}\right)$
13: $\{m_i^{t(L)}\}_1^N = f_{MP}\left(m_1^{t(0)}, \cdots, m_N^{t(0)}, G^{t(1)}, \cdots, G^{t(L)}\right)$
14: $m_i^t = e'_m(m_i^{t(L)})$ for each agent i
15: Calculate $\pi_{\theta_k}(a_i^t|o_i^t)$ and $V_{\phi_k}(o_i^t)$ for each agent i
16: Perform $a_i^t \sim \pi_{\theta_k}(a_i^t|o_i^t)$ for each agent i
17: Receive r_i^t and o_i^{t+1} for each agent i
18: Store $(o_i^t, a_i^t, \pi_{\theta_k}(a_i^t|o_i^t), V_{\phi_k}(o_i^t), r_i^t, o_i^{t+1})$ in D
19: $t \leftarrow t+1, t' \leftarrow t'+1$
20: **end while**
21: **end while**
22: $t_{max} \leftarrow t$
23: **for** $t = t_{max}, t_{max}-1, \cdots, 1$ **do**
24: $R_i^t = 0$ if o_i^{t+1} is terminal else $R_i^t = r_i^t + \gamma R_i^{t+1}$ using D
25: **end for**
26: Calculate $d\theta_k$ and $d\phi_k$ using D with equation 10
27: **end for**
28: **for** thread $k = 1$ to K **do**
29: Accumulate gradients: $d\theta \leftarrow d\theta + d\theta_k$, $d\phi \leftarrow d\phi + d\phi_k$
30: **end for**
31: Perform update of θ using $d\theta$, and of ϕ using $d\phi$
32: **end for**

2. Physical Robot Demonstration

We present a demonstration of our algorithm in a similar 3-vs.-2 soccer scenario on physical robots in the Robotarium, a remotely accessible swarm robotics research platform [2]. This demonstration displays the feasibility of trajectories produced by our MARL algorithm. We present a depiction of MAGIC's deployed trajectory in Figure 2. A video is attached in this link.

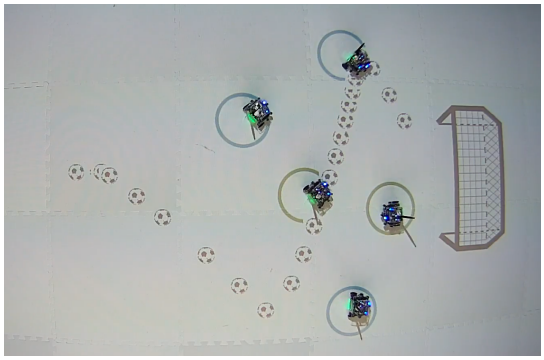


Figure 2. This figure displays a demonstration of our algorithm on physical robots on the Robotarium platform. The display shows a 3 vs. 2 soccer scenario, with blue agents as the attackers, and red agents as defenders.

3. Additional Environment Information

Here, we present additional information about each domain used to benchmark MAGIC against baseline algorithms.

3.1. Predator-Prey

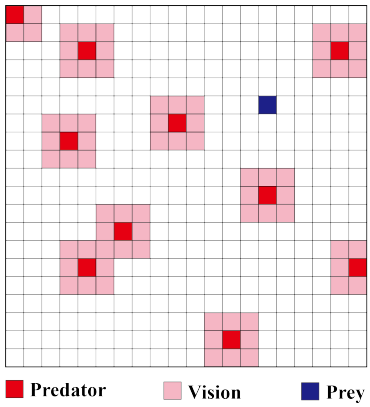


Figure 3. The visualization of the 10-agent Predator-Prey task. The predators (in red) with limited visions (light red region) of size 1 are searching for a randomly initialized fixed prey (in blue).

We utilize the predator-prey environment from [3]. Here, there are N predators with limited visions searching for a stationary prey. A predator or a prey occupies a single cell within the grid world at any time, and its location is initialized randomly at the start of each episode. The state at each point in the grid is the concatenation of a one-hot vector which represents its own location and binary values indicating the presence of predator and prey at this point. The observation of each agent is a concatenated array of the states of all points within the agent's vision. The predators can take actions up, $down$, $left$, $right$ or $stay$. We utilize the 'mixed' mode of Predator-Prey in which the predator incurs a reward -0.05 for each time step until the prey is found. An episode is defined as successful if all the predators find the prey before a predefined maximum time limit. We test two levels of difficulty in this environment. The difficulty varies as the grid size, and the number of predators increases, as more coordination is required to achieve success. The corresponding grid sizes and the number of predators are set to 10×10 with 5 predators and 20×20 with 10 predators. The 10-agent task is shown in Figure 3. We set the maximum steps for an episode (i.e., termination condition) to be 40 and 80, respectively. The vision is set to a unit length. We define a higher-performing algorithm in this domain as one that minimizes the average steps to complete an episode.

3.2. Traffic Junction

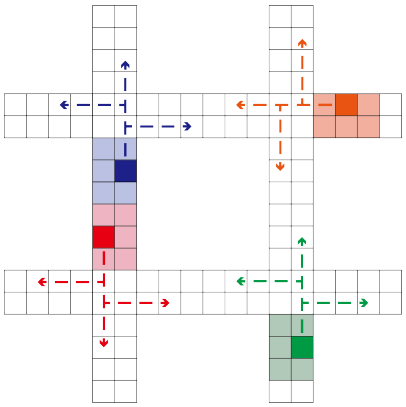


Figure 4. The visualization of the hard level Traffic Junction task. This task consists of four, two-way roads on a 18×18 grid with eight arrival points, each with seven different routes. Each agent is with a limited vision of size 1.

The second domain we utilize is the Traffic Junction environment. This environment, composed of intersecting routes and cars (agents) with limited vision, requires communication to avoid collisions. Cars enter the traffic junction from all entry points at each time step with a probability p_{arrive}, and are randomly assigned a route at the start. The maximum number of cars in the environment at a specific time is denoted by N_{max}, which varies across difficulty levels. A car occupies one cell at a time step and can take action "gas" or "brake" on its route. The state of each cell is the concatenation of a one-hot vector representing its location, and a value indicating the number of cars in this cell. The observation of each car is the concatenation of its previous action, route identifier, and all states of the cells within its vision. Two cars collide if they are in the same location, resulting in a reward of -10 for each car. The simulation terminates once all agents reach the end of its route or if the time surpasses the predefined timeout parameter. Collisions will not incur "death" of agents or terminate the simulation. The agents will only be "dead" when it reaches the end of

Figure 5. The visualization of 3 vs. 2 in Google Research Football. The five people shown in this figure are three offending players, one defending player and the goalie (left to right).

its route. There is a time penalty -0.01τ at each time step, where τ is the number of time steps that have passed since the agent's entry. An episode is considered successful if there are no collisions within the episode.

We validate our algorithm on three difficulty levels. The easy level consists of two, one-way roads on a 7×7 grid. There are two arrival points and two possible routes for each arrival point, and at most five agents ($N_{max} = 5$, $p_{arrive} = 0.3$). For the medium level, the junction consists of two, two-way roads on a 14×14 grid with four arrival points, each with three different routes. Here, there are at most ten agents ($N_{max} = 10$, $p_{arrive} = 0.2$). The hard level, as shown in Figure 4, consists of four, two-way roads on a 18×18 grid with eight arrival points, each with seven different routes, and there are at most twenty agents ($N_{max} = 20$, $p_{arrive} = 0.05$). The average success rate (i.e., no collisions within an episode) is used in our evaluation. We set the limited vision parameter to 1 for both levels. Similar to [3], in Traffic Junction, we fix the gating action to be 1 for IC3Net and TarMAC-IC3Net, set all the hard attention outputs in GA-Comm to be 1, and set all the graphs used by the Message Processor in our method to be complete.

3.3. Google Research Football

Our final domain of Google Research Football [1] presents a challenging, mixed cooperative-competitive, multi-agent scenario with high stochasticity and sparse rewards. Google Research Football (GRF) is a physics-based 3D soccer simulator for reinforcement learning. This domain presents an additional challenge as there are opponent artificial agents (AIs), significantly increasing the complexity of the state-action space. We present a depiction of this environment in Figure 5. To align with the partially observable setting, we extract the local observations from the provided global observations. The local observations include the relative positions of the players on both teams, the relative position of the ball, and one-hot encoding vectors which represent the ball-owned team and the game mode. GRF provides 19 actions including moving actions, kicking actions, and other actions such as dribbling, sliding and sprint. GRF provides several pre-defined reward signals, consisting of a scoring and a penalty box proximity reward. The penalty box proximity reward is shaped to push attackers to move forward towards certain locations. Many MARL frameworks have required these highly shaped rewards functions to perform well [1]. However, we choose to use only the scoring reward to verify the ability of our algorithm and baselines to function in a high-complexity stochastic domain with sparse rewards. Accordingly, the only reward all agents will receive in our evaluation is $+1$ when scoring a goal. The termination criterion is the team scoring, ball out of bounds, or possession change. We evaluate algorithms in a standard scenario 3 vs. 2 from Football Academy [1], as shown in Figure 5, where we have 3 attackers vs. 1 defender, and 1 goalie. The three offending agents are controlled by the MARL algorithm, and the two defending agents are controlled by a built-in AI. We find that utilizing a 3 vs. 2 scenario challenges the robustness of MARL algorithms to stochasticity and sparse rewards. In this domain, we seek to maximize the average success rate (i.e., a goal is scored) and minimize the average steps taken to complete an episode, thereby scoring a goal in the shortest amount of time.

4. Additional Training Details

We distribute the training over 16 threads and each thread runs batch learning with a batch size of 500. The threads share the parameters of the policy network and update synchronously. There are 10 updates in one epoch. We use RMSProp with a learning rate of 0.001 in all the domains except Predator-Prey ten-agent scenario where we use 0.0003. The value coefficient β and discount factor λ are set to 0.01 and 1 respectively. The size of each agent's hidden state for LSTM is 128. The sizes of original encoded messages and the final messages for decision making are 128. 2 or 3 layers of GNNs have been used in practice and shown to work well [4]. Empirically, we find that two rounds of communication achieve the best performance with comparable training speeds to simpler methods such as CommNet and IC3Net. As such, we use two rounds of communication to test the performance of our method in all domains, and the number of heads for the first GAT layer (sub-processor 1) is set to be 4, 4, 1 in Predator-Prey, Traffic Junction and GRF respectively, and the number of heads for the output GAT layer (sub-processor 2) is set to be 1. We use one-round communication for efficiency evaluation for fair comparison, and the number of heads for the GAT layer is 1. The output size of the GAT encoder in the Scheduler is set to 32. We implement our method and baselines on each task over 5 random seeds and average the results.

References

[1] Karol Kurach, Anton Raichuk, Piotr Stanczyk, Michal Zajac, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, and Sylvain Gelly. Google research football: A novel reinforcement learning environment. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 4501–4510. AAAI Press, 2020. 3

[2] Daniel Pickem, E. Squires, and M. Egerstedt. The robotarium : An open , remote-access , multi-robot laboratory. 2016. 1

[3] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 2, 3

[4] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. 3