Cooperative Foraging Behaviour Through Multi-Agent Reinforcement Learning with Graph-Based Communication

Hicham Azmani
Vrije Universiteit Brussel
hicham.azmani@vub.be

Andries Rosseau
Vrije Universiteit Brussel
andries.rosseau@vub.be

Ann Nowé
Vrije Universiteit Brussel
ann.nowe@vub.be

Roxana Rădulescu
Vrije Universiteit Brussel
roxana.radulescu@vub.be

Abstract

According to the social intelligence hypothesis, cooperation is considered a key component of intelligence and is required to solve a wide range of problems, from everyday challenges like scheduling meetings to global challenges like mitigating climate change and providing humanitarian aid. Extending the ability for artificial intelligence (AI) to cooperate well is critical as AI becomes more prevalent in our lives. In recent years, multi-agent reinforcement learning (MARL) has emerged as a powerful approach to model and analyse the problem of cooperation among artificial agents. In this paper, we investigate the impact of communication on cooperation among reinforcement learning agents in social dilemmas. These are settings in which the short-term individual interests are in conflict with the long-term collective ones, thus each individual profits from defecting, but the overall group would benefit if everyone cooperates. We particularly focus on a temporally and spatially extended Stag-Hunt-like social dilemma that models animal foraging behaviour using principles from Optimal Foraging Theory. We propose a method for communication that combines a graph-based attention mechanism with deep reinforcement learning methods. Additionally, we examine several facets of communication, including the effects of the communication topology and the communication range. We find that greater cooperative behaviour can be achieved through graph-based communication using reinforcement learning in social dilemmas. Additionally, we find that during foraging, local communication promotes better cooperation than long-distance communication. Finally, we visualise and investigate the learned attention weights and explain how agents process communications from other agents.

1 Introduction

With artificial intelligence playing an increasingly important role in our lives, it becomes crucial for artificial agents with individual goals to be able to efficiently coordinate and cooperate with each other. This is especially important in social dilemma settings where it is easy for agents to get stuck in detrimental low-reward group dynamics, even though high-reward outcomes are possible through cooperation.

A powerful tool that can enable more cooperative behaviour is communication. In this paper, we look at communication using a Graph-Based Attention mechanism in combination with Deep Reinforcement Learning (RL) agents. We investigate our approach in a temporally and spatially extended Stag-Hunt environment [20, 27, 33], in which communication is a bottleneck for cooperation [9]. Our environment is inspired by Optimal Foraging Theory (OFT) [28], a commonly used framework in behavioural ecology that seeks to mathematically describe animal foraging behaviour. Optimal foraging theory suggests that animals, through natural selection and reproduction, evolved to the point where their foraging behaviour is optimal and fitness is maximised. Since fitness is not easily measured directly, time and energy are often used as substitutes. One model for Optimal Foraging comes from Charnov and Orians [7], who describe a patch exploitation model which examines how long a forager should spend exploiting a food patch before switching to finding a new one. Another model from Sih and Christensen [32] looks at optimal diet theory, and predicts the kinds of food that foragers should choose for optimal foraging, ranging from low-energy foods that agents can forage single-handedly, to high-energy foods which might require coordination with other agents.

These scenarios constitute a Stag-Hunt-like Sequential Social Dilemma (SSD) [20, 11] where individuals have mixed motives, characterised by: (i) when an agent forages individually, it can gather low-energy foods, (ii) when agents cooperate, they can gather more energy-rich foods which are not accessible as an individual, (iii) if an agent attempts to cooperate, but does not find partners, it gets no food at all. These properties give rise to a tension between getting more energy through cooperation, but with the risk of ending up with nothing. While cooperation is the clear best strategy, agents can easily get stuck in a non-cooperative state out of fear [9].

The traditional computational approach for studying animal foraging behaviour is Stochastic Dynamic Programming (SDP) [16]. In order to compute an optimal policy, SDP uses a divide-and-conquer approach dividing the problem into smaller sub-problems, which are solved independently and using backward induction. A major drawback of Stochastic Dynamic Programming is the need of a complete model of the environment with all state transition probabilities and rewards. Moreover, many state space regions are often simply not relevant to find the optimal policy, and SDP is therefore prone to waste a lot of computational resources.

We propose to address this problem by using model-free reinforcement learning [4], which does not require a model of the environment, but rather learns a policy through direct interaction with the environment, balancing exploration and exploitation of the state space. Reinforcement learning can be much more efficient, but once we use non-linear function approximators (like artificial neural networks) to model the policy, or once we enter the multi-agent domain, we loose any guarantees that our agents converge to the optimal policy [34]. However, reinforcement learning with artificial neural networks ('Deep RL') has shown strong performance in practice [25, 12, 36, 31], solving increasingly difficult problems over time. Moreover, the methods of reinforcement learning are originally inspired by the way animals learn [34], so using model-free RL will provide a more biologically accurate model than the model-based planning approach of SDP. Therefore, we model our foraging task as a multi-agent reinforcement learning (MARL) setting. In recent years, MARL has become a staple approach for learning in and analysing temporally complex SSDs [3, 20].

In short, our work presents the following main contributions:

- We create a Sequential Social Dilemma environment inspired by Optimal Foraging Theory, by modifying the Level-Based Foraging environment [8]
- We propose GAPPO (Graph-Attention Proximal Policy Optimisation), a novel MARL method that extends Proximal Policy Optimisation (PPO) [31] to allow agents to communicate with each other based on a Graph-Attention neural network architecture.
- We investigate the capacity of graph-based communication in MARL to promote cooperative behaviour in a temporally and spatially extended Stag-Hunt social dilemma where agents act in a self-interested way. We then compare to approaches without communication.
- We analyse the attention weights of our agents and show that agents can meaningfully filter out the relevant communication messages from other agents for better coordination.

2 Related Work

Due to their capacity to model and process spatial data, there has been an increasing amount of methods using Graph Neural Networks (GNNs) as a communication mechanism in MARL. One of the earliest approaches to use GNNs in MARL was DGN [18], which modelled each agent acting as a node in the graph, and local observations of the agents serving as node features. Through graph convolutions, which serve as a communication mechanism, neighbouring agents are able to exchange their observations which in turn increases their receptive fields and promotes cooperation. Agarwal et al. [1] expanded on this concept by including environment entities as part of the graph.

Instead of limiting communication to only nearby agents, MAGNet [24] and G2ANet [22] employ a learnable method to determine which agents should communicate with one another. The former generates a relevance graph using the self-attention mechanism while the latter uses a hard-attention mechanism to prune edges between agents which do not need to communicate. In a comparable manner, MAGIC [26] employs two separate components to learn a useful communication topology: (1) a scheduler to learn when agents should communicate and with whom; (2) a message processor in the form of a graph attention network, to handle the communication signals.

Out of the discussed methods, the closest to our approach are DGN and G2ANet. We apply DGN's principle of representing the multi-agent environment as a graph and exchanging messages via graph convolutions. However, whereas DGN focuses on fully cooperative (team-based) scenarios, our paper's main goal is to study self-interested agents in a mixed-motive sequential social dilemma setting where cooperation is not directly incentivised. Moreover, we extend the GNN-based PPO algorithm from G2ANet, a policy gradient reinforcement learning algorithm, to promote agent cooperation: unlike G2ANet, in which a central solver learns which agents can communicate, we only allow communication between agents that are in a certain range from each other, which is a restriction frequently found in real-world applications due to the cost of communication. Furthermore, G2ANet, like DGN, again only considers cooperative tasks.

3 Preliminaries

3.1 Partially Observable Stochastic Games

In this work, we model our MARL problem as a Partially Observable Stochastic Game (POSG), a generalisation of Stochastic (Markov) Games [21] to settings where agents are only able to observe parts of the environment's state. A POSG is a tuple $\langle N, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{O}, \mathcal{T}_e \rangle$ [17, 37], where N is the number of agents, $\mathcal{S}$ is the set of all possible global states of the environment, $\mathcal{A} := A_1 \times A_2 \times \ldots \times A_N$ represents the set of actions, where A_i is the action space of agent i, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \triangle(\mathcal{S})$ is the transition probability function between global states, based on the joint action of the agents, $\mathcal{R} := R_1 \times \cdots \times R_N$ is the reward function, where $R_i \colon S \times \mathcal{A} \times S \rightarrow \mathbb{R}$ is the individual reward function of agent i, $\mathcal{O} := O_1 \times O_2 \times \ldots \times O_N$ is the set of observations, with O_i representing the individual observation set for each agent i, and $\mathcal{T}_e : \mathcal{S} \times \mathcal{A} \rightarrow \triangle(\mathcal{O})$ is the observation function, where $\mathcal{T}_e(o|\mathbf{a}, s)$ represents the probability of observing $o \in \mathcal{O}$, given the join action $\mathbf{a} \in \mathcal{A}$ and a new state $s \in \mathcal{S}$ from the environment transition.

The behaviour of an agent is defined by its policy $\pi_i : O_i \times A_i \rightarrow [0, 1]$. An agent's goal is to find a policy π_i which maximises the expected discounted long-term reward:

$$V^{\pi_i} = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_{i,t} \mid \boldsymbol{\pi}\right] \tag{1}$$

where $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$ is the joint policy of the agents acting in the environment, $\gamma \in [0, 1]$ is the discount factor, T is the amount of time steps in an episode, and $r_{i,t} = R_i(s_t, \mathbf{a}_t, s_{t+1})$ is the reward obtained by agent i at time step t, for the joint action $\mathbf{a}_t \in \mathcal{A}$, at state $s_t \in S$ and transitioning to the next state $s_{t+1} \in S$.

3.2 Proximal Policy Optimisation

Proximal Policy Optimisation (PPO) [31] is a policy gradient method for single-agent reinforcement learning. The main benefit of PPO is the introduction of a clipping formula, which prevents unstable learning caused by excessive policy updates. By limiting the change in the probability ratio between the new and old policies using Equation 2, the clipping formula limits the policy update step.

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right], \tag{2}$$

where $L^{\text{CLIP}}(\theta)$ represents the clipped objective function, ϵ a hyperparameter that controls the extent of the clipping, θ the parameters of the policy, $r_t(\theta)$ the probability ratio between the new and old policy at timestep t, and A_t the advantage function which estimates the relative value of a particular action at timestep t.

The PPO algorithm has been used in numerous studies to solve multi-agent reinforcement learning problems, with some demonstrating performance that is comparable with state-of-the-art algorithms in a variety of cooperative challenges [39]. The Independent PPO (IPPO) method is the most straightforward way to use the PPO algorithm in MARL. Using the IPPO algorithm, each agent treats the other agents as being part of the environment and computes its actions based solely on its own local observation [40]. The obtained policy is only conditioned on the agent's own rewards [39].

3.3 Graph Neural Networks

A Graph Neural Network (GNN) is a neural network model suitable to represent and operate on graph structured data. Each node in the graph is characterised by a feature representation that is exchanged with its neighbours. Each node then aggregates and processes the incoming representations, including its own features, through a (potentially shared) neural network, thereby also updating its own representation. This process can undergo several iterations (i.e., network layers), until the nodes use the final features for the designated task (e.g., classification) [6]. Consider a graph $G = \langle \mathcal{V}, \mathcal{E} \rangle$ with a set of nodes $\mathcal{V}$ with d features and a set of edges $\mathcal{E}$. We can update the representation of every node using a GNN layer which expects two inputs: the set of node representations $\{h_i \in \mathbb{R}^d \mid i \in \mathcal{V}\}$ and the set of edges $\mathcal{E}$. A GNN layer then produces a new set of updated node representations $h_i^{'}$ of a potentially different feature dimension $\mathbb{R}^{d'}$: $\{h_i^{'} \in \mathbb{R}^{d'} \mid i \in \mathcal{V}\}$. This set of updated node representations is obtained by applying a parametric function f_θ to every node i and its neighbours $\mathcal{N}_i = \{j \in \mathcal{V} | (j, i) \in \mathcal{E}\}$:

$$h_i^{'} = f_\theta(h_i, AGGREGATE(\{h_j \mid j \in \mathcal{N}_i\})) \tag{3}$$

Most GNN types vary in their definition of the parametric function f_θ and the aggregation function [6].

3.3.1 Graph Attention Networks (GATs)

A popular variant of GNNs are Graph Attention Networks (GATs) [35]. GATs differ from most GNN architectures by computing the updated representation of a node as a weighted average of the representations of its neighbours using the attention mechanism, while most GNN approaches assign equal importance to the representations of its neighbours by using max or mean-pooling as aggregation functions [6].

In order to compute the updated node features h' a GAT layer first computes attention coefficients or scores for every pair of connected nodes i and j using the self-attention mechanism $a : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. These attention coefficients indicate the importance of the features of node j to node i [6]. Veličković et al. [35] use a single-layer feed-forward neural network with a LeakyRelu non-linearity activation function as a scoring function leading to the following equation:

$$e(h_i, h_j) = LeakyReLU(\overrightarrow{a}^T \cdot [Wh_i \mid\mid Wh_j]), \tag{4}$$

where $||$ is concatenation. The weight matrix W represents a linear transformation that is applied to every node i and j of the provided set of node features h to attain sufficient expressive power to compute higher-level features based on the provided input features. Brody et al. [6] notes that the attention computed by Eq. 4 is a form of *static* attention that leads to a global ranking of nodes such that certain influential nodes will always be prioritised over the other nodes regardless of the query. In order to compute a *dynamic* form of attention where each node has a different ranking of nodes it is sufficient to alter the order of the internal operations in GAT by applying the a layer after the LeakyReLU activation function and the W layer after the concatenation which leads to Eq. 5.

$$e(h_i, h_j) = \overrightarrow{a}^T LeakyReLU(W \times [h_i \mid\mid h_j]) \tag{5}$$

Furthermore, softmax is used to normalise the attention scores of node i across all its neighbours $j \in \mathcal{N}_i$ in order to make coefficients easily comparable across different nodes:

$$a_{ij} = softmax_j(e(h_i, h_j)) = \frac{\exp(e(h_i, h_j))}{\sum_{j' \in N_i} \exp(e(h_i, h_{j'}))} \tag{6}$$

Finally, the GAT layer uses the normalised attention coefficients to compute the updated representations of every node i as a linear combination of the neighbouring node representations. The final updated representations produced by the GAT layer are obtained by applying a non-linearity σ [35].

$$\overrightarrow{h'_i} = \sigma\left(\sum_{j \in N_i} a_{ij} W \overrightarrow{h}_j\right) \tag{7}$$

In order to stabilise the learning process we use the multi-headed attention mechanism instead of the single-head attention mechanism. In the case of multi-head attention, we compute Eq. 7 in K independent heads. The results of the K heads are either concatenated in the case of multiple GAT layers or averaged in the case of a single GAT layer. We average the results of the K heads using Eq. 8 as we only consider a single GAT layer.

$$\overrightarrow{h'_i} = \sigma\left(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in N_i} a_{ij}^k W^k \overrightarrow{h_j}\right) \tag{8}$$

4 Methods

A major component of this work is to examine whether graph-based communication can promote cooperation in sequential social dilemmas. Towards this end, we compare both non-communicating agents and communicating agents in a sequential social dilemma inspired by the Optimal Foraging Theory. First, we introduce the multi-agent reinforcement learning social dilemma environment. We then describe GAPPO, our variant of the Proximal Policy Optimisation algorithm extended with a graph-based communication mechanism.

4.1 Model of the Foraging Setting

We use a modified version of the Level-Based Foraging (LBF) environment [2, 8] to create a Sequential Social Dilemma environment that is motivated by Optimal Foraging Theory. Our environment is a mixed-incentive game where self-interested agents (i.e., agents that each have their own individual reward function) are competing for food. Our environment is a rectangular grid world where N randomly placed agents have to collect M randomly placed food items by navigating the world within a certain time. We provide a visualisation of the environment in Figure 3. The action space for the agents is either: do nothing, move to an adjacent grid cell (north, east, south, west) as long as that cell is empty, or capture a food item in an adjacent cell. Each food item is assigned a level of 1 or 2; level 1 food can be captured by one agent, but level 2 food needs the simultaneous presence of (at least) two adjacent agents for the agents to capture that food. Level 1 food gives a reward of 1, and level 2 food gives a reward of 8. One can also provide higher level foods (with an exponentially increasing reward) up until level 4. This represents the natural case where animals can learn to coordinate to get access to improved resources. It is more difficult to obtain these resources, but if successful, the reward is more than what one would get on their own (e.g., learning to hunt larger prey together, or being strong enough together to move an obstacle and reach higher quality foods). The reward is divided evenly between all agents that were simultaneously around the food to capture it. When a food item is captured, it respawns after 5 time steps elsewhere in the environment. Episodes are terminated when the episode has reached $T = 50$ steps. We introduce an energy loss penalty of -0.01 per time step, in accordance with the biological energy loss of agents from their metabolism keeping them alive (and moving them in the world). There is also an energy cost associated with food capture and consumption in Optimal Foraging Theory [23, 13, 30], but one can assume this is already calculated in the food rewards without loss of generalisation. This environment forms a sequential social dilemma, which is a temporally and spatially extended Stag-Hunt game [20, 33, 27] with multiple cooperative levels induced by the multiple levels of food. An in-depth explanation of the observation space, action space, and reward structure of our environment, along with a comparison to the original LBF environment, is given in Appendix B.

4.2 Graph-Attention PPO (GAPPO)

In order to analyse the effects of communication on cooperation we extend the single-agent reinforcement learning algorithm of Proximal Policy Optimisation (PPO) [31] with a GNN-based communication layer.

To allow for inter-agent communication we make use of a Graph Attention Layer. In order to use GATs we first need to model the problem as a graph. We create a node for each agent in the environment, with the features of each node being the local observation of the corresponding agent. The nodes of communicating agents are then connected via an edge to allow them to exchange their local observations. The existence of an edge between two agents will depend on the underlying graph topology. We provide more information on the various topologies we used in Appendix C and provide a comparison between the topologies in Section 5. Once the environment is transformed into a graph, we can allow agents to communicate by updating each node of the graph using the GAT layer. Each node i is updated by a weighted average of the representations of itself and its neighbours $j \in N$ using the attention mechanism of Equation 7. In summary, we first apply a linear transformation, represented by the weight matrix W, to obtain higher-level features. Next, for each node i we compute attention coefficients for all its neighbours j which indicate the importance of the features of node j to node i. When the observations of an agent contain more useful information (e.g., food) we assume that it will receive a larger attention coefficient to indicate its importance. We compute the updated node representation as a weighted sum using the computed attention coefficients.

As described in Section 3.3.1, multi-headed attention is used to stabilise the learning process. Both the IPPO algorithm and the GAT communication mechanism are combined in the architecture presented in Figure 1.

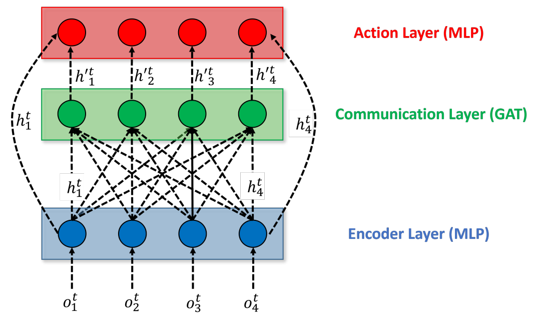


Figure 1: Architecture of GAPPO. Local observations are first passed through an encoder. Encoded observations are then exchanged using the GAT communication mechanism. Finally, probabilities of actions are computed based on the original encoded observations computed by the encoder as well as the updated encoded observations computed by the GAT communication mechanism.

At first, the observations of all agents $o_1^t, o_2^t, ..., o_N^t$ are transformed by an encoding layer into embedded observations $h_1^t, h_2^t, ..., h_N^t$. The encoder in GAPPO is a multi-layer perceptron (MLP) with a single layer that encodes the observations. For more complex observations, such as RGB images, Convolutional Neural Networks (CNN) can be used instead of a simple MLP.

The encoded observations $h_1^t, h_2^t, ..., h_N^t$ are then exchanged with neighbouring agents in the GAT layer. As mentioned earlier the feature of each node is the encoded observation of the corresponding agent. The GAT layer produces updated node representations $h_1^{'t}, h_2^{'t}, ..., h_N^{'t}$ which contain the encoded observation of the agent itself combined with the encoded observations of its neighbours.

Finally, an action layer computes probabilities for each action based on both the encoded observations produced by the encoder $h_1^t, h_2^t, ..., h_N^t$ as well as the updated encoded observations produced by the GAT layer $h_1^{'t}, h_2^{'t}, ..., h_N^{'t}$. The original encoded observations provided by the encoder are provided to the action layer in order to allow the agent to be able to navigate in its neighbourhood easily. Using solely the updated encoded observations leads to situations where agents have difficulty navigating their neighbourhood when surrounded by many other agents as the updated encoded observation contains information from different agents combined.

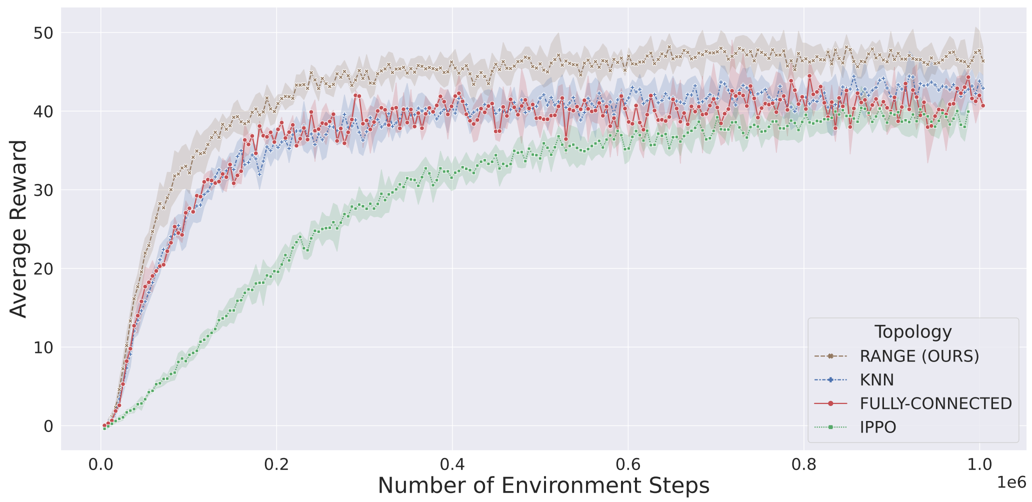


Figure 2: IPPO vs GAPPO with different communication topologies (range, KNN and fully-connected) in a 10x10 grid world with 4 agents. The observation range of the agents is always set to two.

We use parameter-sharing in GAPPO to allow agents to learn a shared encoding of observations which in turn facilitates learning how to combine exchanged encoded observations.

5 Results and Discussion

In this section, we investigate the performance of Graph-Attention Proximal Policy Optimisation (GAPPO) on our sequential social dilemma foraging environment. As a baseline, we use Independent PPO (IPPO), since one of our primary goals is to analyse the **impact of communication** on the agents' learned behaviour. We also perform an in-depth study on how GAPPO is impacted by various **communication topologies** and **communication ranges**, environment sizes and configurations (Appendix - Section E), as well as how meaningful the resulting **attention weights** are for each agent (i.e., can agents' learn to which messages they should pay more attention to in certain contexts?).

We evaluate the *range* communication topology of GAPPO in comparison to a *K-Nearest-Neighbour (KNN)* [14] based communication and a *fully connected* communication topology. Several methods including DIAL [15], TarMAC [10], and SchedNet [19], are built on top of the fully-connected communication topology, while the KNN-based topology was introduced by DGN [18] as a substitute (due to implementation constraints) for the range communication topology we use. We provide more details on the topologies in the Appendix (Section C). We also conduct a scalability analysis in which we evaluate GAPPO's performance in varying environment sizes and food densities in the Appendix (Section E). We repeat each trial five times with a different predetermined random seed to assure fairness. We use $K = 2$ for the KNN-based communication topology which means that each agent always communicates with its two nearest agents.

Impact of Communication. We present the learning curves of IPPO and GAPPO (with various communication topologies) in Figure 2. Firstly, we can observe that GAPPO outperforms IPPO with all communication topologies. GAPPO converges faster than IPPO and achieves a higher average reward, showing that communication enables agents to coordinate and collaborate on gathering higher-level food items and reaching higher average rewards, despite the mixed-inventive scenario.

Communication Topologies. When comparing the performance of the various communication topologies, we find that the *range* communication topology performs better than the others, which suggests that in the LBF environment, local communication fosters better cooperation than long-distance communication. Further analysis of the impact of various communication ranges (see Appendix D) leads us to conclude that the optimal communication range is one that is equal to the agent's observation range which in this case is two. Additionally, we observe that the *KNN*-

based topology, which was used by DGN as a substitute for the range communication topology, performs worse. We propose a dual explanation for why local communication performs better than long-distance communication in the LBF environment. First, we believe local communication is favoured due to the messages' nature. Agents share their local observations which primarily include information relevant to agents in their vicinity. In scenarios involving long-distance communication, agents are occasionally unable to pinpoint where in the environment an observation originates from. Second, the nature of the tasks in the LBF environment, such as picking up food items, strongly favours local cooperation. Agents cooperate by picking up the same food item from adjacent locations. With long-distance communication, agents that are far away from each other can influence one another, but in order to cooperate one of the agents would need to travel a great distance to reach the other agent. With local communication, we limit communication to nearby agents which are easily reachable and thus easy to cooperate with.

5.1 Attention Weight Analysis

In addition to the learning curves, we can also observe how the weights of the GAT layer change over the course of an episode. A GAT layer has the advantage over a straightforward GNN layer in that agents can learn to focus more on particular agents during communication. The attention weights of the GAT layer can therefore provide us with useful insights into how agents communicate.

We describe the general behaviour and communication strategies of trained GAPPO agents with the *range* communication topology across episodes. We provide a recording of the behaviour across a whole episode here. At the start of episodes, agents are mostly scattered around the environment, which makes it difficult for them to communicate with one another and forces them to solely rely on their own observations. When agents encounter food items without being able to communicate with other agents they either pick it up if it is a level 1 food item or wait at the food item until another agent arrives to coordinate in getting the food. Once the agents start exploring the environment and come across other agents, however, we can start analysing the communication behaviour of agents. First, we can see that agents tend to give all of their neighbours' observations equal or nearly equal weights when they are interacting with other agents who are not observing any food items. This can be observed by looking at the weights of $agent_3$ in Figure 3. Neither $agent_3$ nor its neighbour, $agent_1$, can see a food item which leads $agent_3$ to assign almost equal weights to both observations. This leads to agents mostly exploring the environment by moving in random directions looking for food items, but often staying close enough to each other in order to be able to communicate once they locate a food item.

When agents do encounter food items we can see agents start attributing different weights to different agents. If an agent observes a food item itself it starts assigning more importance to its own local observation while reducing the attention it gives to the obtained observations of its neighbours, as seen in Figure 3. $Agent_0$ in this case encounters a food item of level two which it tries to pick up. Even though $agent_0$ can communicate with other agents such as $agent_1$, we can see that it assigns 89% of its attention to its own observation. This can lead to groups of agents splitting up once multiple agents of the group encounter food items. When agents do not encounter food items themselves but instead one of their neighbours encounters a food item we can see that the observation of that agent gets significantly more attention and the agent moves towards that agent. In Figure 3, $agent_1$ cannot directly observe any food items due to its observation range of two. However, because it can communicate with $agent_0$, which is trying to pick up a food item of level two, it assigns 75% of its attention to the observation of $agent_0$ and starts moving towards $agent_0$ and the food item. Even when using other communication topologies like the KNN-based and fully-connected topologies, we observe that agents still learn to prioritise observations which include items of interest like food. However, they now struggle to consistently move in the right direction as they cannot always pinpoint the location of the food in the environment, which further reinforces our intuitions.

Based on both the learning curves and attention weights analysis, we can observe that local communication promotes cooperation in our foraging environment. Firstly, we can observe that agents learn to stay within each other's communication range to be able to discover food items more efficiently which in turn leads to a higher average reward. This is in line with the emergence of group or swarm foraging in Foraging Theory and has been observed among many animal species, including lions [29] and ants [5]. Second, we observe that agents learn an explainable communication strategy in which they prioritise observations of agents that are registering food over others.

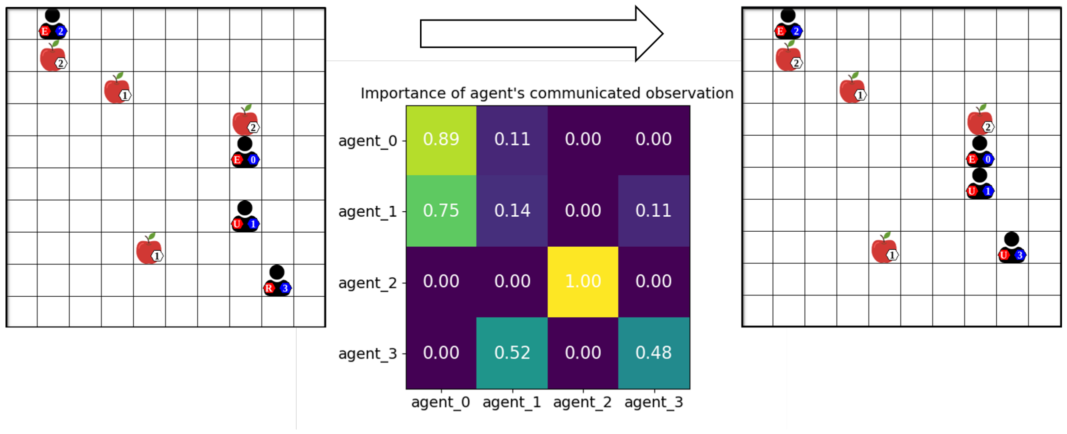


Figure 3: Example of 10x10 environment with 4 agents. $Agent_0$ first observes a high-level food item which leads it to ignore other agents and focus on its own observation. When $agent_0$ communicates this to $agent_1$, $agent_1$ prioritises the observation of $agent_0$ as it does not see any food item itself and starts moving towards $agent_0$. On the other hand, $agent_3$ assigns around equal weights to its own observation and the observation of $agent_1$ as neither of them can see food.

6 Conclusion

In this paper, we analyse the effect of graph-based communication in multi-agent reinforcement learning social dilemmas. We chose to model animal foraging behaviour as a sequential social dilemma by extending and modifying the level-based foraging environment, an existing multi-agent reinforcement learning environment that simulates animal foraging behaviour. We propose a graph-based communication method, GAPPO (Graph-Attention Proximal Policy Optimisation), as well as a visualisation method which allows us to gain a deeper knowledge of the learned communication strategies. Finally, we analysed the effects of various aspects of communication such as the communication topology and communication range on the performance of the agents. We conclude that graph-based communication with attention can be a strong cooperative enabler in temporally and spatially extended social dilemmas. In the case of a foraging environment, local communication leads to better cooperation than long-distance communication. Finally, we observed that the learned communication strategy can be explained through a visualisation of the attention weights of the graph-based communication mechanism.

Future Work

We consider two directions for future work on this topic. First, future research may try to improve the graph-based communication system. In this work, we only consider a basic permanent communication policy. Future work may include more intricate communication policies, such as individual and global communication policies that let agents select which other agents they want to communicate with. One can also allow agents to share other information in addition to their local observations. While there exist methods that implement individual and global communication policies, they have only been analysed in the context of teams of cooperative agents. In social dilemmas, arguably the hardest class of mixed-motives problems, agents can show much more complex or even manipulative communication strategies and can be incentivised to withhold information.

Second, future research may aim to increase the environment's realism. Currently, the environment incorporates only a few of the fundamental ideas underlying animal foraging theory. Some factors are not taken into consideration, such as the decreasing marginal benefits of foraging, or the possibility of an animal dying from starvation or becoming prey itself. We describe reinforcement learning as a promising model for animal foraging behaviours, but for reinforcement learning to be a viable method for behavioural ecologists, models will need to be extended to include more ecological factors as well.

Acknowledgments Roxana Rădulescu is supported by the Research Foundation – Flanders (FWO), grant number 1286223N. Andries Rosseau is supported by funding from the Optimesh ICON project (grant code VLAFLX7). The resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the FWO and the Flemish Government.

References

[1] Agarwal, A., Kumar, S., and Sycara, K. (2019). Learning transferable cooperative behavior in multi-agent teams. *arXiv preprint arXiv:1906.01202*.

[2] Albrecht, S. V. and Ramamoorthy, S. (2015). A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems. *arXiv preprint arXiv:1506.01170*.

[3] Atrazhev, P. and Musilek, P. (2022). Investigating effects of centralized learning decentralized execution on team coordination in the level based foraging environment as a sequential social dilemma. In *Advances in Practical Applications of Agents, Multi-Agent Systems, and Complex Systems Simulation. The PAAMS Collection: 20th International Conference, PAAMS 2022, L'Aquila, Italy, July 13–15, 2022, Proceedings*, pages 15–23. Springer.

[4] Barto, A. G., Sutton, R. S., and Watkins, C. (1989). *Learning and sequential decision making*. University of Massachusetts Amherst, MA.

[5] Beckers, R., Goss, S., Deneubourg, J.-L., and Pasteels, J.-M. (1989). Colony size, communication and ant foraging strategy. *Psyche*, 96(3-4):239–256.

[6] Brody, S., Alon, U., and Yahav, E. (2021). How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*.

[7] Charnov, E. and Orians, G. H. (2006). Optimal foraging: some theoretical explorations.

[8] Christianos, F., Schäfer, L., and Albrecht, S. (2020). Shared experience actor-critic for multi-agent reinforcement learning. *Advances in neural information processing systems*, 33:10707–10717.

[9] Dafoe, A., Hughes, E., Bachrach, Y., Collins, T., McKee, K. R., Leibo, J. Z., Larson, K., and Graepel, T. (2020). Open problems in cooperative ai. *arXiv preprint arXiv:2012.08630*.

[10] Das, A., Gervet, T., Romoff, J., Batra, D., Parikh, D., Rabbat, M., and Pineau, J. (2019). Tarmac: Targeted multi-agent communication. In *International Conference on Machine Learning*, pages 1538–1546. PMLR.

[11] Dawes, R. M. (1980). Social dilemmas. *Annual review of psychology*, 31(1):169–193.

[12] Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., de Las Casas, D., et al. (2022). Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419.

[13] Emlen, J. M. (1966). The role of time and energy in food preference. *The American Naturalist*, 100(916):611–617.

[14] Fix, E. (1985). *Discriminatory analysis: nonparametric discrimination, consistency properties*, volume 1. USAF school of Aviation Medicine.

[15] Foerster, J., Assael, I. A., De Freitas, N., and Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 29.

[16] Frankenhuis, W. E., Panchanathan, K., and Barto, A. G. (2019). Enriching behavioral ecology with reinforcement learning methods. *Behavioural Processes*, 161:94–100.

[17] Hansen, E. A., Bernstein, D. S., and Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games. In *Proceedings of the 19th National Conference on Artifical Intelligence*, AAAI'04, pages 709–715. AAAI Press.

[18] Jiang, J., Dun, C., Huang, T., and Lu, Z. (2018). Graph convolutional reinforcement learning. *arXiv preprint arXiv:1810.09202*.

[19] Kim, D., Moon, S., Hostallero, D., Kang, W. J., Lee, T., Son, K., and Yi, Y. (2019). Learning to schedule communication in multi-agent reinforcement learning. *arXiv preprint arXiv:1902.01554*.

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

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

[22] Liu, Y., Wang, W., Hu, Y., Hao, J., Chen, X., and Gao, Y. (2020). Multi-agent game abstraction via graph attention neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7211–7218.

[23] MacArthur, R. H. and Pianka, E. R. (1966). On optimal use of a patchy environment. *The American Naturalist*, 100(916):603–609.

[24] Malysheva, A., Kudenko, D., and Shpilman, A. (2019). Magnet: Multi-agent graph network for deep multi-agent reinforcement learning. In *2019 XVI International Symposium" Problems of Redundancy in Information and Control Systems"(REDUNDANCY)*, pages 171–176. IEEE.

[25] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

[26] Niu, Y., Paleja, R. R., and Gombolay, M. C. (2021). Multi-agent graph-attention communication and teaming. In *AAMAS*, pages 964–973.

[27] Owen, G. (2013). *Game theory*. Emerald Group Publishing.

[28] Perry, G. and Pianka, E. R. (1997). Animal foraging: past, present and future. *Trends in Ecology & Evolution*, 12(9):360–364.

[29] Scheel, D. and Packer, C. (1991). Group hunting behaviour of lions: a search for cooperation. *Animal behaviour*, 41(4):697–709.

[30] Schoener, T. W. (1971). Theory of feeding strategies. *Annual review of ecology and systematics*, 2(1):369–404.

[31] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[32] Sih, A. and Christensen, B. (2001). Optimal diet theory: when does it work, and when and why does it fail? *Animal behaviour*, 61(2):379–390.

[33] Skyrms, B. (2001). The stag hunt. In *Proceedings and Addresses of the American Philosophical Association*, volume 75, pages 31–41. JSTOR.

[34] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[35] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903*.

[36] Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354.

[37] Wiggers, A. J., Oliehoek, F. A., and Roijers, D. M. (2016). Structure in the value function of two-player zero-sum games of incomplete information. In *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, pages 1628–1629. IOS Press.

[38] Wong, A., Bäck, T., Kononova, A. V., and Plaat, A. (2021). Multiagent deep reinforcement learning: Challenges and directions towards human-like approaches. *CoRR*, abs/2106.15691.

[39] Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., and Wu, Y. (2022). The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624.

[40] Zhang, K., Yang, Z., and Başar, T. (2021). Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control*, pages 321–384.

A Hyperparameters

We used the RLlib Reinforcement Learning library, which allows us to create a flexible and extendable experimental framework. We include the hyperparameters for both IPPO and GAPPO in Table 1 for reproducibility. Each experiment was repeated five times using the same hyperparameters with a different predetermined random seed to assure fairness. Additionally, we provide the codebase with the configuration files used for each experiment on Github.

Hyperparameter	**IPPO**	**GAPPO**
discount	0.99	0.99
batch size	4000	4000
minibatch size	128	128
num_sgd_iter	30	30
kl_coeff	0.2	0.2
kl_target	0.1	0.1
λ (GAE)	1.0	1.0
optimiser	Adam	Adam
learning rate	$5e-05$	$5e-05$
Number of layers in encoder	1	1
Hidden units of encoder layers	128	128
Encoder activation function	ReLU	ReLU
Number of GAT layers	/	1
Hidden units of GAT layer	/	128
GAT activation function	/	ReLU
Number of heads in GAT layer	/	2

Table 1: Hyperparameters during training.

B Foraging Environment

In this section, we provide a thorough overview of the observation space, action space, and reward structure of the foraging environment and compare it to the original LBF environment.

B.1 Observation Space

The LBF environment provides both full and partial observability. Original studies on the LBF environment were primarily conducted under the assumption that the environment was completely observable [8]. Instead, we focus on the partially observable variant. We use our Graph-Attention Proximal Policy Optimisation (GAPPO) method to allow the agents to exchange their partial observations in order to obtain a better global view of the environment, which is one of the ways to address the challenge of partial observability [38].

The LBF environment provides two types of observations: triplet observations and grid observations. Consider the environment in Figure 4 which we will use to describe the two types of observations.

When using triplet observations, the observation of each agent consists of a fixed-size array of size $3 \times (N + M)$ with N being the number of agents and M the amount of food. The first $3 \times M$ cells are used to store x-, y-, and levels of food items. The remaining cells are used to store x-, y-, and levels of agents, with the first three cells always being reserved for the agent of whom we are analysing the observation. Figure 5 provides the triplet observation of $agent_0$ in the example environment of Figure 4.

The second type of observation is grid observation. When using grid observations each agent receives a layered $(2 * observation\_range + 1) \times (2 * observation\_range + 1) \times 3$ grid observation. This is because we allow the agent to see $observation\_range$ grid cells in each direction and provide 3 layers. The three layers are a food layer, where only food items are shown, an agents layer, where only agent IDs are shown, and finally, a wall layer where accessible cells are indicated. Figure 6 provides the grid observation of $agent_0$ in the example environment of Figure 4.

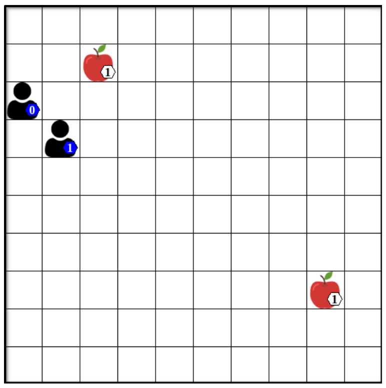
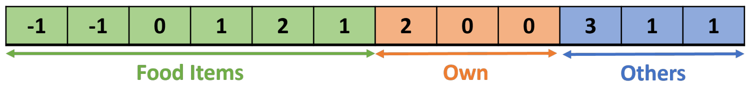


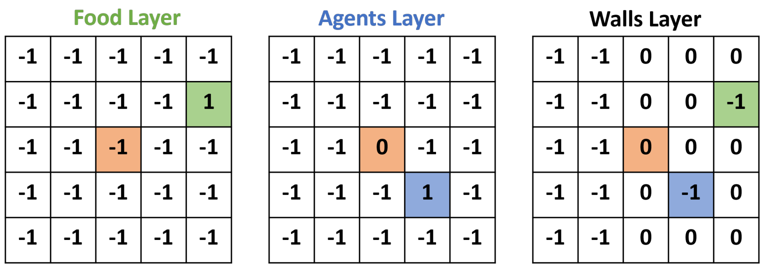
Figure 4: Example 10x10 LBF environment with 2 agents and 2 low-level food items.



-1	-1	0	1	2	1	2	0	0	3	1	1

Food Items
Own
Others

Figure 5: Example of the triplet observation of $agent_0$. The first six cells are used to store the x- and y-coordinates of the two food items as well as their level. When a food item is outside of the agent's observation range it is indicated by -1 x- and y-coordinates. The next 3 cells are used for $agent_0$'s own x-coordinate, y-coordinate, and ID. The remaining cells are used to indicate the positions and IDs of the other agents in the environment. Agents that are outside of the agent's observation range are also indicated by -1 x- and y-coordinates.



Food Layer

-1	-1	-1	-1	-1
-1	-1	-1	-1	1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1

Agents Layer

-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	0	-1	-1
-1	-1	-1	1	-1
-1	-1	-1	-1	-1

Walls Layer

-1	-1	0	0	0
-1	-1	0	0	-1
-1	-1	0	0	0
-1	-1	0	-1	0
-1	-1	0	0	0

Figure 6: Example of the grid observation of $agent_0$. The observation consists of 3 layers stacked on top of each other. The agent is always in the centre of the $(2 * observation_range + 1) \times (2 * observation_range + 1)$ grid. The first layer only contains food item levels if a food item is present in that cell. The second layer only contains agent IDs at the agents' positions. The third layer indicates cells that agents can move to. Cells that are outside the grid are not accessible. Similarly, cells which contain a food item or another agent are not accessible.

In this work, we focus exclusively on grid observations.

B.2 Action Space

Each agent has a choice of six different actions during each time step, including moving in any of the four directions (North, East, South, or West), doing nothing, or attempting to load/capture a nearby food item (there is one action for capturing, no matter which direction you come from).

We start with the 'load' action, which allows agents to pick up food items. Any of the four adjacent locations is available for agents to pick up food items. Agents are only rewarded when the food item is successfully loaded while they are picking it up. When multiple agents are picking up the same food item at the same time, the total reward of the food item is split equally among the agents, whereas in the original LBF environment, all agents obtain the full reward associated with a food item.

In addition to the 'load' action, agents can also use one of the 4 moving actions to move to an adjacent grid cell, or remain still. As long as the cell is still in the grid and free, agents are free to move to any of the four cells that are close by. A location is considered free if it is within the grid and is free of another agent or any food. The free cells are provided to the agent in the grid observation, as can be seen in Figure 6.

As all agents move simultaneously and treat one another as being part of the environment, one issue with the moving actions is that agents may attempt to move into the same free cell. We prevent these collisions by cancelling both agents' actions, causing the two agents to remain in the same position for the following time step. Consider the example environment in Figure 4. If $agent_0$ decides to move right and $agent_1$ decides to move up they are both moving to the same cell, which leads to a collision. In order to avoid this collision we cancel the actions of both agents. Both $agent_0$ and $agent_1$, therefore, keep their current positions. Due to the cancellation of movements in the case of collisions, agents can sometimes find themselves in a deadlock situation where they are blocking each other in successive time steps because they have no way of knowing what actions other agents chose in the previous time step or what they intend to carry out in this time step. However, this phenomenon usually becomes much less frequent as training progresses.

B.3 Reward Structure

Instead of a reward being shared globally among all agents or a team of agents, each agent receives their own reward. Agents only receive positive rewards when they pick up food; all other rewards are negative. When a food item is picked up, the agents that took part in collecting it are rewarded. Every food level has a reward attached to it that corresponds to the amount of energy it produces when consumed. Higher-level food types that require the coordination of multiple agents produce a higher reward than lower-level food types that can be picked up by a single agent. When a group of agents picks up a food item, the reward associated with that food item is divided equally among the number of agents that participated in the pickup. In the case of level one food that can be picked up by a single agent, an agent is rewarded more if it consumes the food item on its own than if it shares it with other agents. Therefore, the agents must cooperate in order to pick up higher-level food, which can only be picked up by groups of agents, while they compete for lower-level food.

Agents in our environment consume energy while looking for food, in a manner similar to the principle of Optimal Foraging Theory. We model this by giving each agent a penalty of -0.01 in each time step. By incurring a -0.01 penalty for each time step spent waiting for another agent to assist in picking up the food, as opposed to continuously searching for level one food that can be picked up by a single agent, this penalty makes opting for a higher level food item a more costly option. Another cost comes simply from waiting, and therefore not being able to eat lower level foods in the meantime, thereby missing out on rewards as well. The amount of food is constrained in the original LBF environment because food does not re-spawn, which means that a waiting agent will eventually receive assistance from the other agents when low-level food has been depleted. However, in our variation, the other agents can decide to pursue other food items instead of helping the waiting agent, which makes the environment more like a stag-hunt setting.

C Analysing the Communication Topology

In Section 5 we present the learning curves of IPPO and GAPPO with various communication topologies. In Figure 7 we present the used topologies through an example.

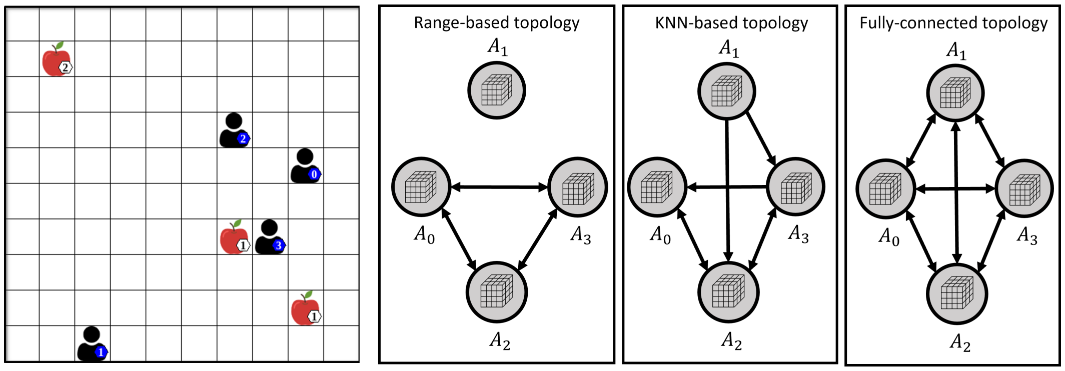


Figure 7: Example of different communication topologies in GAPPO.

We consider three types of communication topologies. The range-based topology connects all agents within a certain observation range of each other, in this case, two (meaning 5 x 5 grid cells). It is possible for agents to not communicate with any other agent when they are isolated, such as $Agent_1$ in Figure 7. In the KNN-based topology, each agent is allowed to send its observations to the closest K agents. While the range-based communication topology is symmetrical, agents can always communicate in both directions; this is not the case for the KNN-based communication topology. $Agent_1$ for example sends its observations to $Agent_3$ as it is one of its closest agents, but $Agent_3$ sends its observations to $Agent_1$ and $Agent_2$ instead. The final communication topology is the fully-connected communication topology. Under this communication topology, all agents communicate with one another.

D Analysing the Communication Range

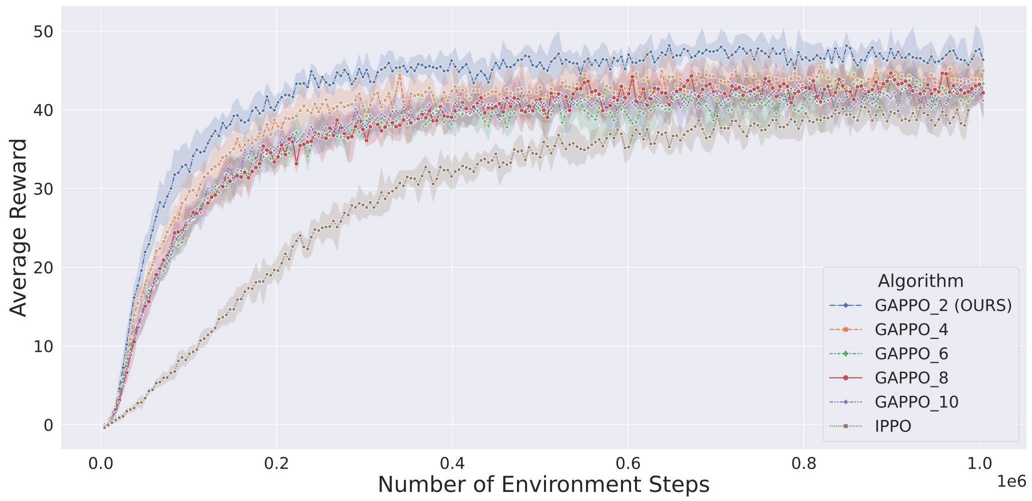


Figure 8: IPPO vs GAPPO with different communication ranges in a 10x10 LBF grid world with 4 agents. All agents have a observation range of two.

We present the learning curves for IPPO and GAPPO with different communication topologies in Section 5. We mention that the best-performing GAPPO communication topology is the range-based topology with a communication range of two, which is equal to the observation range of the agents.

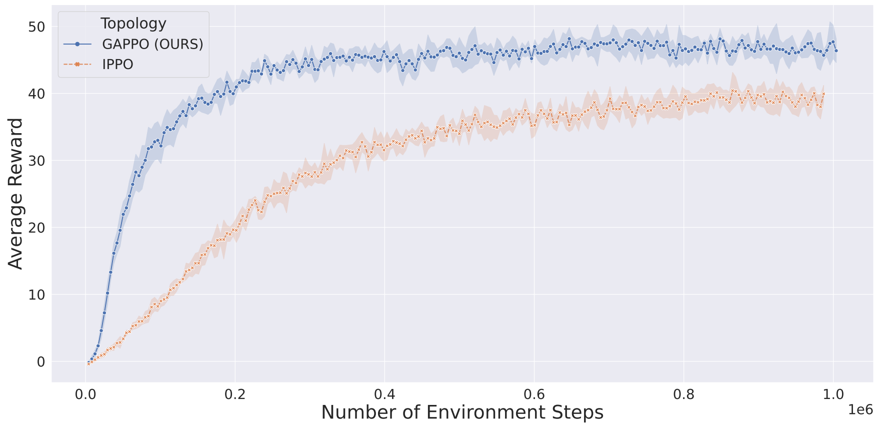
This was concluded from an experiment where we investigated the effect of different communication ranges. We present the learning curves of this analysis in Figure 8.

We can observe in Figure 8 that GAPPO with a range of two outperforms other larger communication ranges. We believe that the reasons for this phenomenon are the same as the ones explained in Section 5 for the poor performance of the KNN-based and fully-connected topologies.

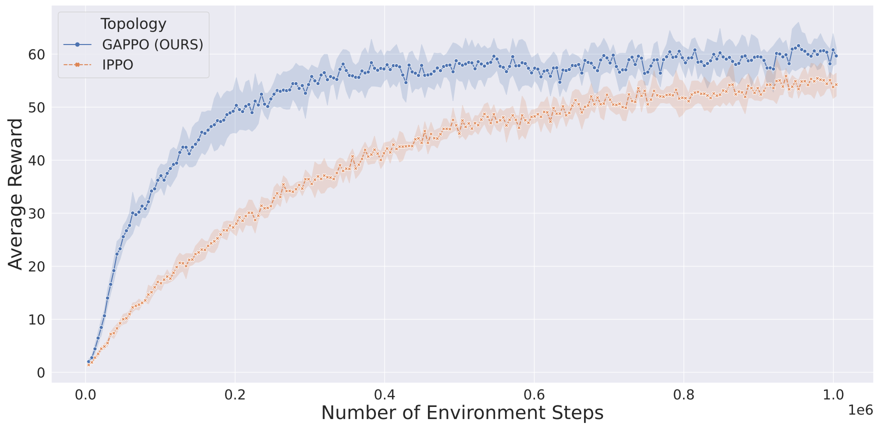
E Scalability Analysis

We examine how GAPPO performs in various environmental conditions. We compare GAPPO and IPPO in a selection of food densities and environment sizes.

We consider two environments, a small environment of size 10×10 and a large environment of size 20×20. In both environments, we consider low and high-food density variants. For the small environment, we introduce 8 food items (four of level 1 and four of level 2) in the high-food density setting and 4 food items (two of level 1 and two of level 2) in the low-density food setting. In the large environment, we define 16 food items (eight of level 1 and eight of level 2) in the high-food density setting and 8 food items (four of level 1 and four of level 2) in the low-food density setting. In all environmental conditions, we maintain a fixed number of four agents. We present the learning curves of GAPPO and IPPO in the different environmental conditions in Figures 9 and 10.

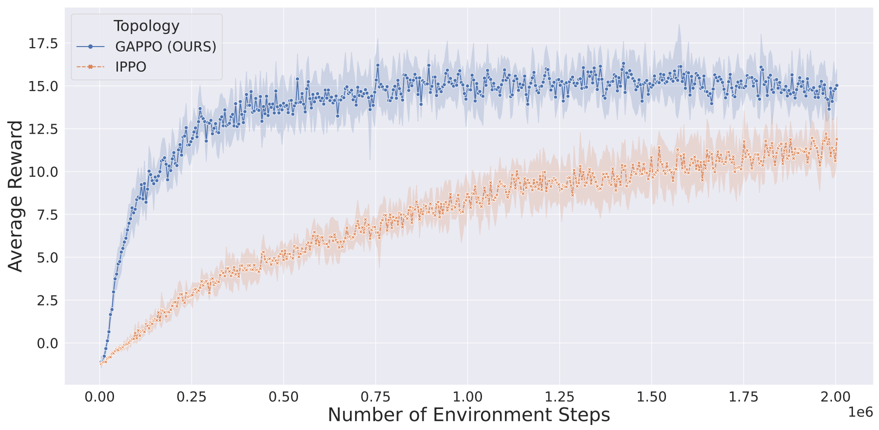


(a) Small environment with low food density

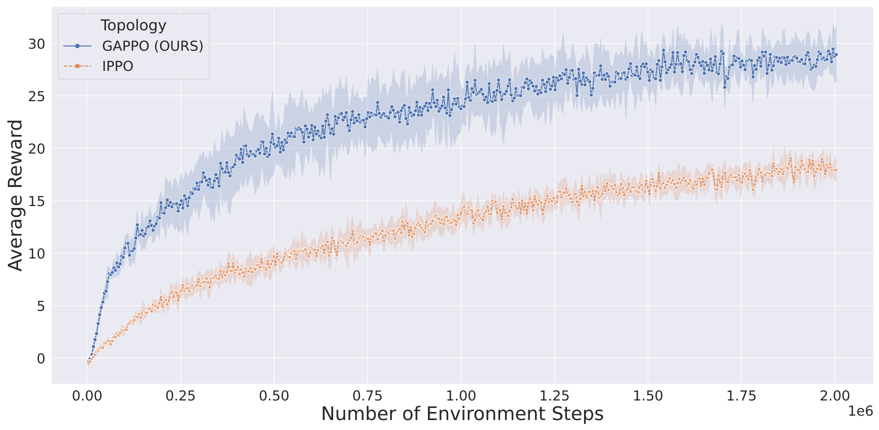


(b) Small environment with high food density

Figure 9: Learning curves of IPPO and GAPPO under varying environmental conditions (small environment 10x10).



(a) Large environment with low food density



(b) Large environment with high food density

Figure 10: Learning curves of IPPO and GAPPO under varying environmental conditions (large environment 20x20).

We can see that the results are scale-invariant across the different environment sizes and food densities. We observe that GAPPO outperforms IPPO in all settings by achieving a higher average reward and converging faster. Additionally, we observe that in the smaller environment, GAPPO performs better in an environment with low food density, requiring more cooperation between the agents to reliably obtain rewards. However, in a larger environment, IPPO and GAPPO appear to receive significantly fewer rewards than their counterpart with a high food density, which is not the case in a smaller environment where both settings lead to similar average rewards. We conclude that the sparse distribution of food and agents in the big environment makes it harder for agents to learn to cooperate together.