Multi-Agent Reinforcement Learning in Sparsely Connected Cooperative Environments

- Final Report -

Pankayaraj Pathmanathan
Yuvini Sumanasekera
Chandima Samarasinghe

Department of Computer Engineering
University of Peradeniya

Final Year Project (courses CO421 & CO425) report submitted as a requirement of the degree of
B.Sc.Eng. in Computer Engineering

June 2020

Supervisors: Dr. Dhammika Elkaduwe (University of Peradeniya), Dr. Upul Jayasinghe (University of Peradeniya) and Dr. D. H. S. Maithripala (University of Peradeniya)

We would like to dedicate this thesis to our project supervisors, academia and colleagues who guided us in this process. We are grateful for their support, advise and endless encouragement.

Declaration

We hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is our own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgments.

Pankayaraj Pathmanathan
Yuvini Sumanasekera
Chandima Samarasinghe
June 2020

Acknowledgements

First and foremost we would like to thank our supervisors, Dr. Dhammika Elkaduwe, Dr. Upul Jayasinghe and Dr D.H.S Maithripala for their continuous support and guidance in the successful delivery of the project. Their invaluable feedback and constructive criticism ensured that the project progressed in the right direction and its successful completion.

At the same time, we wish to express our appreciation to Prof. Roshan Ragel, Dr. Sampath Deegalla, Dr. Damayanthi Herath, and Dr. Pradeepa Bandaranayake for their guidance and helpful discussions regarding the final year project.

We are also grateful to all members of the academic staff and non-academic staff at the Department of Computer Engineering, University of Peradeniya, for their support extended in numerous ways.

Finally, our heartfelt gratitude goes out to our families who have always been encouraging and supporting us throughout our academic career.

Abstract

In recent years, the consensus among adaptive agents within multi-agent systems (MAS) has been an emerging area of research in the field of autonomous control. Reinforcement Learning (RL) has gained immense interest in this line of work as it aims to learn optimal cooperative policies through trial and error by dynamically interacting with the environment. However, in practice, connectivity within the multi-agent network may be sparse and the agents are often subjected to partial observability. This can result in the learning of sub-optimal policies. In this work, we consider the problem of learning optimal policies in cooperative multi-agent environments in the face of partial observability and sparse connectivity. The proposed model exploits the inherent graph-like structure of multi-agent systems. Graph Neural Networks (GNNs) are utilized to extract spatial dependencies and temporal dynamics of the underlying graph. Such spatio-temporal information is exploited to generate better state representations so as to facilitate the learning of more robust policies. Empirically, we demonstrate the effectiveness of the proposed model on a variety of well-known cooperative control tasks. Finally, we conclude the paper with proposals on possible research directions.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

CTDE	Centralised Training with Decentralised Execution
DL	Deep Learning
GNN	Graph Neural Network
GRU	Gated Recurrent Unit
LSTM	Long Short Term Memory
MARL	Multi-Agent Reinforcement Learning
MAS	Multi-Agent System
MDP	Markov Decision Process
ML	Machine Learning
NN	Neural Network
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
RNN	Recurrent Neural Network

Chapter 1

Introduction

Multi-agent systems (MAS) consist of a network of loosely connected autonomous agents interacting with one another to achieve some objective(s). These systems are widely used for solving problems that are beyond the capabilities of single-agent systems. The efficiency, robustness, reliability, and scalability offered by such systems, make them appealing to a variety of application domains [2]. In recent years, the consensus among the adaptive agents within MAS has been an emerging area of research in the field of autonomous control. In practice, such agents may operate in environments with complex dynamics and high degrees of uncertainty. Thus, defining which agent behaviours are optimal for a given situation, ahead of time, is impractical or often impossible [3]. As opposed to relying on a predefined behaviour strategy, Reinforcement Learning (RL) enforces the learning of optimal behaviour through trial and error, by allowing the agent (i.e. the learner) to dynamically interact with the environment [4]. With the success of RL in single-agent domains, the focus of research has shifted towards developing RL based techniques that take into account the dynamicity and uncertainty of the environment, as well as complexities arising from agent-to-agent interactions in multi-agent domains. Accompanied by the recent popularity of deep learning (DL) techniques, multi-agent reinforcement learning (MARL) has diversified into numerous real-world applications such as robotics [5], network management and packet routing [6, 7], resource management [8, 9], power grid control [10, 11], etc.

Many multi-agent systems—from social networks to molecular structures—inherently exhibit the structure of graphs. Moreover, graphs provide rich and expressive means of modelling and reasoning about multi-agent systems. Prior to end-to-end learning, traditional machine learning methodologies typically relied on handcrafted feature engineering to extract meaningful latent representations from data [12], [13]. Graph neural networks (GNN) [14] leverage the capabilities of existing DL techniques to learn from non-Euclidean graph-structured data. Spatio-temporal dependencies reflect the dynamics and correlations of the underlying multi-agent system across both space and time. Thus, capturing such dependencies can provide crucial insight into the mutual interplay among the agents, the evolution of agent behaviour overtime, etc. This facilitates agents to learn more robust coordination strategies coherently. In this work, we propose a reinforcement model, where spatio-temporal dependencies of the underlying graph structure of the multi-agent system are exploited to leverage the learning of optimal policies in cooperative environments.

1.1 Background

1.1.1 Reinforcement Learning

In contrast to classical machine learning techniques, reinforcement learning approaches attempt to learn useful behaviour through dynamic interactions between a "goal-directed" agent (i.e. the learner and decision-maker) and the environment [4]. RL problems are formally modelled as Markov decision processes (MDPs) which can be defined by a tuple $M = \langle S, A, T, R, \gamma \rangle$. Thus, as the agent explores potential behaviour strategies, based on the observed state and the agent's selected action at each time-step, it receives a feedback signal from the environment in the form of a numerical reward (positive or negative). Figure 1.1 illustrates this agent-environment interaction.

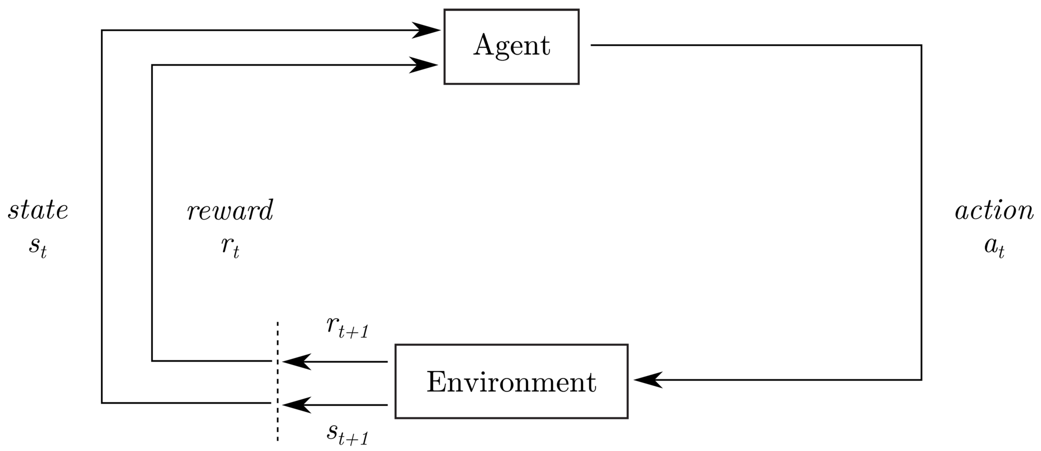

Fig. 1.1 Agent-environment interaction in Reinforcement Learning

Assume that the agent's state and action space is denoted by S and A respectively. The probability of reaching the successor state $s' \in S$ by taking action $a \in A$ from a given state $s \in S$, is denoted by $T : S \times A \times S \to [0, 1]$. The agent receives an immediate scalar reward r according to the reward function $R : S \times A \rightarrow \mathbb{R}$. In general, the agent seeks to maximize its sum of discounted future rewards (i.e. the return G_t). A discount rate $\gamma \in [0, 1]$ is introduced, which determines the short/long sightedness of the agent in terms of reward accumulation.

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots \tag{1.1}$$

A policy π fully defines the agent's behaviour function. Specifically, a stochastic policy; $\pi(a|s)$, maps a given state to a probability distribution over the actions. Every state's or state-action pair's quality is valued as an expectation of the return (G_t) starting from that particular state or state-action pair and following a certain policy $\pi(a|s)$ thereafter. These expectations are known as the state-value function $V^\pi(s)$ and action-value function $Q^\pi(s, a)$ respectively.

$$V^\pi(s) = \mathbb{E}_\pi \left[G_t | s_t = s\right] \tag{1.2}$$

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[G_t | s_t = s, a_t = a\right] \tag{1.3}$$

The expected rewards generated by following a parameterized policy $\pi(a|s; \theta)$ over time, can be considered as the objective function (1.4).

$$J(\theta) = \mathbb{E}_{s_t \sim p^\pi, a_t \sim \pi} \left[\sum_t R\left(s_t, a_t\right)\right] \tag{1.4}$$

where p^π denotes the state distribution for policy $\pi(a|s; \theta)$. Thus, the ultimate goal of RL boils down to learning a policy $\pi(a|s; \theta)$ that maximizes this objective function (1.4). RL algorithms can be classified into three categories; value-based, policy-based, and actor-critic methods. The following three subsections briefly discusses each of these categories.

Value Based Methods

In value-based approaches, the value function is explicitly modelled and improved, which is subsequently used to extract the agent's optimal policy. One popular value-based method is Deep Q-Network (DQN) [15], where a parameterized function approximator; $Q(s, a; w)$ is used to represent $Q^\pi(s, a)$.

$$Q(s, a; w) = \mathbb{E}_{s'} \left[R(s, a) + \gamma \mathbb{E}_{a' \sim \pi} \left[Q\left(s', a'\right)\right]\right] \tag{1.5}$$

The optimal policy is obtained by minimizing the loss function $\mathcal{L}(w)$ (1.6).

$$\mathcal{L}\left(w\right) = \mathbb{E}_{s,a,r,s'\sim\mathcal{D}}\left[\left(Q\left(s,a;w\right) - y\right)^2\right] \tag{1.6}$$

where $y = r + \gamma \max_{a'} Q\left(s', a'; w^-\right)$. Here, $\mathcal{D}$ is a replay buffer that stores recent experience tuples; (s, a, r, s'), and $Q\left(s, a; w^-\right)$ is a parameterized target network. Maintaining an experience replay buffer and a separate target network improves the stability and efficiency of learning [15].

Policy Based Methods

In policy-based methods, the policy $\pi(a|s;\theta)$ is directly modelled and improved over time to obtain the optimal policy. As opposed to value-based techniques, the value function is not explicitly modelled. The objective function (1.4) is maximized by adjusting θ in the direction of the policy gradient (1.7).

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim p^\pi, a\sim\pi}\left[\nabla_\theta \log \pi(a|s;\theta) R\right] \tag{1.7}$$

Based on the policy gradient theorem [4], for any differentiable policy $\pi(a|s;\theta)$, R can be replaced with $Q\left(s, a; w\right)$. An example of this approach is the REINFORCE [16] algorithm where multiple trajectories are sampled while updating the policy using the estimated gradient.

Actor-Critic Methods

Actor-critic methods can be considered as a hybrid approach where both the policy and the value function are explicitly modelled. The critic $Q\left(s, a; w\right)$ estimates the value function (value-based), which is then used to update the actor $\pi(a|s;\theta)$ in the direction of policy improvement (policy-based) [4]. DDPG [17] and PPO [18] are well-known examples of actor-critic algorithms.

1.1.2 Multi-Agent Reinforcement Learning (MARL)

In a multi-agent learning setup, several adaptive agents co-exist in a common environment. Thus, the behaviour strategy of one agent is influenced not only by the environment but also by other agents' behaviours. RL can be applied in such settings to learn complex physical and/or communicative strategies through dynamic interactions with the environment and with other agents. One of the most commonly used frameworks; Markov games [19] is a generalisation of MDPs to a multi-agent learning setup. Often times, as the agents are distributed across the environment and due to their limited sensing capabilities, partial observability can occur. Under such conditions, a partially observable Markov game is defined by a tuple $G = \langle S, A, T, R, O, Z, n, \gamma \rangle$, where S denotes the finite set of states of the environment. Each of the n agents chooses actions sequentially. A_i and O_i denote the finite sets of actions and observations available to agent i respectively, where $i = 1, 2, \ldots, n$. The local observation perceived by agent i; $o \in O_i$ is generated from the underlying state according to the observation function $Z : S \times A \rightarrow O$. Each agent i chooses an action $a \in A_i$ which contributes to the joint action $A = A_1 \times A_2 \times \ldots \times A_n$. This in turn induces a state transition in the environment according to the state transition probability function $T : S \times A \times S \rightarrow [0, 1]$. Each agent i receives an immediate reward as a function of the global state and the joint action, $R_i : S \times A \rightarrow \mathbb{R}$.

RL problems formulated using MDPs are under the assumption that the environment is stationary from the point of view of any given agent. In most single-agent settings, given sufficient exploration, the convergence of the underlying RL algorithm is dependant on this assumption [20]. In a multi-agent setting, if each agent were to learn its optimal behaviour independently, the remaining agents would be treated as part of the environment. However, as learning occurs concurrently, due to the actions of the other agents, now the environment is no longer stationary from the perspective of any given agent, thus violating the stated assumption. Therefore, the presence of any adaptive agents other than the agent of interest leads to this non-stationarity problem [3]. The notion of centralised training with decentralised execution (CTDE) has been extensively utilised in solving MARL problems, as it addresses the issue of non-stationarity [21], [22]. Here, a centralized critic that has access to the observations and actions of all the agents, is exploited during training. Thus, as learning progresses, the environment becomes stationary for any given agent. However, during execution, the actors solely rely on local action-observation histories. Thus, while additional information is available for an agent during training, execution is conditioned only on local information.

As actor-critic methods utilise two separate models, such algorithms naturally fit into the CTDE framework. Particularly, the actor model is used to make decisions (i.e. action selection) and the critic model is used to evaluate state-action pairs to guide the policy in the direction of optimality during the training phase. At execution, the only model of interest is that of the actor (policy), therefore, the critic model can be discarded at this stage. Thus, CTDE not only facilitates the learning of complex behaviours in the presence of adaptive agents but also enables any given agent to act independently of the other agents' actions and/or observations while delivering the expected results through coordinated behaviour.

1.2 Problem Statement

The goal of the research is to develop a MARL model to leverage the learning of optimal agent policies so as to establish consensus among the agents to carry out some control task cooperatively. A comprehensive analysis of literature related to previous studies in the domain deduced that modelling multi-agent systems so as to encapsulate the dynamicity of the learning environment is often a challenge. Motivated by the recent advancements in deep learning, graph-based approaches show great promise in this line of work. Thus, one of the objectives of this research was to incorporate GNNs to leverage the capabilities of MARL to generate more robust control policies. In practice, agents are often limited by their sensing capabilities which result in partial observability. As an observation is not a complete representation of the underlying state of the environment, inferring the actual state using the joint observations of all agents, is often inadequate. Moreover, as $Q(o, a|\theta) \neq Q(s, a|\theta)$, decisions based on such partial observations can lead to sub-optimal policies [23]. Sub-optimal policies may also stem from sparse connectivity within multi-agent networks that result from poor network connectivity or limited bandwidth of the communication channels or sparse interactions among agents. Further, due to computational intractability and cost of communication, it is often infeasible to consider information from all agents. Thus, a primary aim of the research was to ensure that the proposed model was reflective of such practical constraints prevalent in real-world applications. Inspired by some of the recent work in this line of work, the design of the spatial dependency module of the proposed model closely followed the work of Agarwal et al. [24].

In this work, we propose a reinforcement learning model, in which the multi-agent system is modelled as a graph. The nodes may represent the agents or environment entities, while the edges represent the communicative dependencies between nodes. The proposed model utilises spatio-temporal dependency modelling to generate better state representations for the learning agents within the system. Spatial dependency modelling includes an iterative message-propagation mechanism to ensure information propagation through the network via intermediary nodes. Temporal dependency modelling utilises a recurrent structure which retains information and propagates such information through time. Thus, as opposed to raw local observations, spatio-temporal dependency modelling generates state representations which are more reflective of the spatial influences of the agent-agent interactions as well as the underlying temporal dynamics of the system. Such state representations are thereby utilised by the agents in learning optimal policies to perform some cooperative control task. Notably, this work intends to solely conduct the training and evaluation of the proposed model in a simulated particle environment and thus, considers the transfer of the simulated model to the real-world applications as a potential future work.

1.3 Outline of the Report

The concluded introductory chapter is followed by the second chapter; the literature review. The chapter is composed by conducting a broad review of previous studies carried out in the problem domain, which are relevant for the research. Existing work on MARL is identified to be two-fold depending on how consensus is acquired among the learning agents and is thus organized under separate subsections; learning-for-consensus and learning-for-communication. The review will further focus on past work that has integrated GNNs into the MARL architecture. The contributions and limitations of the studies will be evaluated to identify the gaps in the domain that needs to be addressed.

The third chapter describes the design methodology of the research. A high-level overview of the proposed model will be introduced, followed by a detailed description of its constituent modules. Further, the design considerations and the justifications behind selecting certain components/techniques will be discussed as well.

The fourth chapter describes the experimental setup and implementation details of the study. This will outline the environment and the control tasks utilised in the study.

The final two chapters cover the discussion and the conclusion of the study. The fifth chapter will provide the experimental results and a subsequent analysis of the results. The benefits, limitations and feasibility of the implemented solution will be discussed in this chapter. To conclude the report, the sixth chapter will provide the conclusions of the research and possible avenues for future research.

Chapter 2

Related work

2.1 MARL

2.1.1 Learning-for-consensus

In learning-for-consensus approaches, during the execution of the learned policies, agents' actions are solely based on local observations. Coordination is facilitated by allowing each agent to exploit information of other agents during the training phase [25], [26]. Much of the recent work in this line of research incorporates CTDE as it provides a natural paradigm for such learning settings. MADDPG [27] was devised as an extension of DDPG [17] to multi-agent cooperative, competitive and mixed environments with continuous action spaces. Each agent utilises a separate centralised critic augmented with additional information regarding other agents' policies, thus removing the effect of non-stationarity. The actors take actions based on their local observations. Foerster et al. proposed COMA [22], which relies on a fully centralized critic. COMA specifically addresses the credit assignment problem commonly prevalent in cooperative multi-agent learning settings, where agents must coordinate their behaviours to achieve some common goal. The model utilises a counterfactual baseline which marginalizes out the action of a given agent such that its effect on the global reward can be evaluated. CM3 [28] follows a similar approach within a multi-goal environment, where the impact of each agent's behaviour on the remaining agents' goal attainment is determined. Yang et al. [29] argued that the input space of a centralized critic grows exponentially with the increasing number of agents. Thus, the authors proposed a mean-field MARL approach which approximates the centralised critic by one which considers only the pairwise local interactions between a given agent and its neighbours. However, this approximation can result in the loss of information beneficial for learning effective cooperation strategies. Given that any particular agent is not affected by all agents at all times, an alternative approach was proposed in MAAC [30]. The centralized critic employs an attention mechanism that allows each agent to dynamically select agents to attend to during training. As the centralized critic does not consider aggregated information from all the other agents at any given time as in [27] and [22], the input space is shown to scale linearly with the number of agents.

2.1.2 Learning-for-communication

Learning-for-communication approaches achieve coordination by sharing information among agents via explicit communication protocols [25], [26]. With the popularity in DRL techniques, a variety of end-to-end learning architectures have been proposed to facilitate communication among agents. BiCNet [31] introduces a vectorised version of the multi-agent actor-critic formulation, based on bi-directional RNNs. The bi-directional recurrent structure acts as a communication channel which facilitates information sharing among heterogeneous agents. CommNet [32] considers the notion of continuous communication to learn fully cooperative tasks. Each agent learns a communication protocol in conjunction with its policy. Through a multi-cycle communication mechanism, local state observations and the average of incoming messages determine agents' actions. Both [31] and [32] are under the assumption that all the agents communicate with one another at any given time. A few notable work involves the notion of targeted communication whereby agents learn when, what and whom to communicate with, using attention mechanisms. Enabling agents to actively differentiate and select useful information from globally shared information helps in developing more efficient and effective communication strategies which improves consensus among agents. IC3Net [33] is an extension of [32], where the continuous communication model is controlled using a gating mechanism. This enables agents to learn when and what to communicate with other agents. Moreover, it makes amendments to some of the shortcomings of [32]. Notably, the credit assignment problem and the model's lack of applicability in competitive and mixed multi-agent settings. Jiang et al. proposed ATOC [34], a scalable communication model for partially observable distributed learning settings. An attention layer fully determines whether communication is required for a given agent at each time-step. A bi-directional Long Short Term Memory (LSTM) structure facilitates communication by connecting each agent to the other agents within its receptive field. TarMAC [35] utilises a soft attention mechanism to determine which agents to communicate with and what messages to transmit, in partially-observable cooperative learning settings. A signature encoded with agent-specific information is sent alongside the message which ensures that the broadcasted message reaches the intended agents.

2.2 MARL with GNNs

Generalizing neural networks to graph-structured data is currently an active area of deep learning research. However, the research efforts to incorporate GNNs to MARL problems have been limited thus far. Wang et al. proposed NerveNet [36], that utilises GNNs to learn policies by exploiting the graph-like body structure of human/robot agents. Actions for different body parts of the agent are predicted via information propagation over the underlying structure. The ability of such learned policies to be generalised to other learning tasks is also highlighted. DGN [37] introduces a graph convolution network (GCN) based approach to facilitate the learning of cooperative policies. Convolution with relation kernels is utilised to capture relation representations of the underlying graph that are exploited to better understand the mutual interplay among the learning agents. HAMA [25] considers a mixed learning setting with both cooperative and competitive agents that are clustered based on some pre-defined rules. The authors introduced a hierarchical graph attention network-based method to model both the intra-group relationships as well as the inter-group relationships among the learning agents. Considering effective communication as an integral part of multi-agent cooperation, Sheng et al. proposed LSC [38]. LSC utilises hierarchical graph neural networks to enable both intra-group and inter-group message generation and message propagation among dynamically clustered agents. A notable observation of all the aforementioned studies is that, in order to facilitate relation reasoning within the multi-agent system, the extraction of latent features is limited to the spatial domain, with no consideration to that of the temporal domain. In contrast, Wang et al. introduced STMARL [39] as a framework for multi-intersection traffic light control that utlizes spatio-temporal dependency modelling. A recurrent neural network structure is utilised to exploit historical traffic information while a GNN-based structure is utilised to facilitate relation reasoning among the traffic lights. The authors argue that the influence mechanism among multi-intersection traffic lights can be better understood via spatio-temporal dependency modelling. Hu et al. [40] proposed a progressive relation learning framework for group activity recognition and analysis. In order to explicitly model the relations among individuals within the group, a semantic relation graph is constructed. Two agents utilize RL to progressively refine the low-level spatio-temporal features and high-level semantic relations of group activities. The work demonstrates the implicit coordination between the two learning agents which affects the performance of the underlying group activity recognition task.

Chapter 3

Methodology

Figure 3.1 illustrates the pipeline of our proposed model. In the following subsections, we introduce each of the constituent modules in detail.

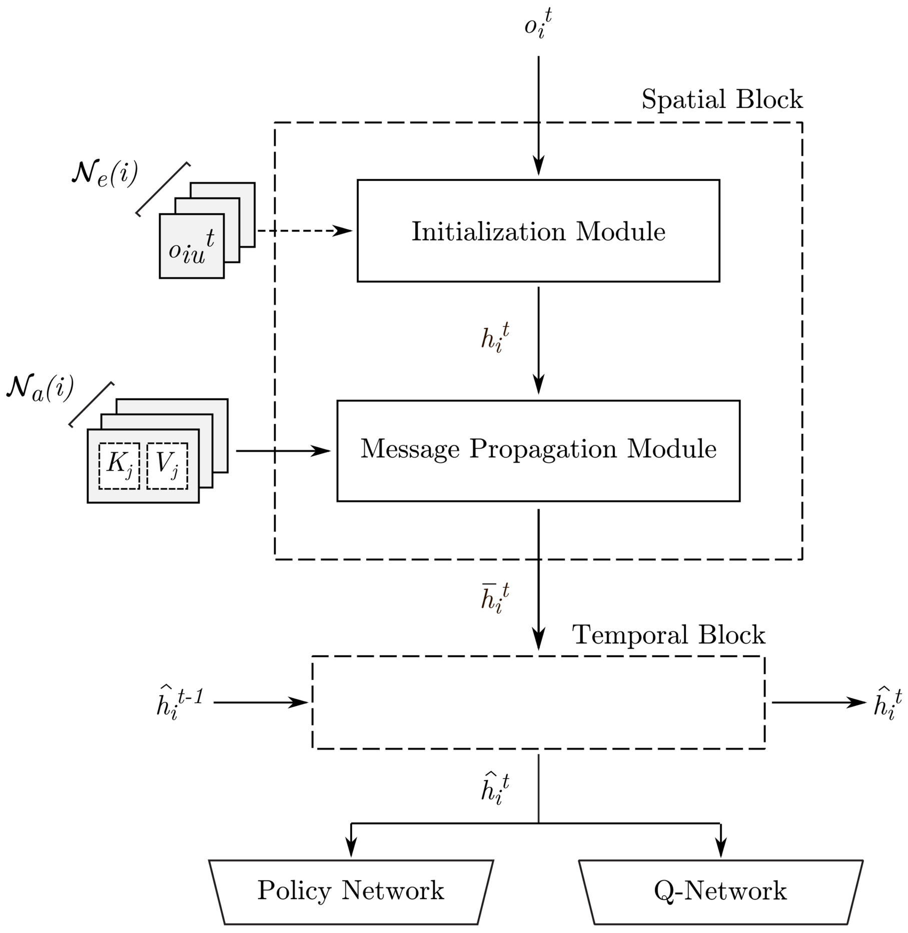

Fig. 3.1 Overall architecture of the model for a given agent i

3.1 Graph Construction

We organize the multi-agent system as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Each node $v \in \mathcal{V}$ represents either an agent or an entity. Here, entities represent certain objects in the environment such as landmarks, obstacles, etc. In practice, an agent is more likely to interact and be influenced by agents/entities in its immediate vicinity. Therefore, the communication range of an agent is specified as a function of proximity, which consequently determines its neighbourhood. We denote the neighbourhood of agent i as $\mathcal{N}(i)$, determined by the communication range of agent i. Thus, in $\mathcal{G}$, there exists an edge $(i, v) \in \mathcal{E}$ between agent i and node v (agent or entity), if $v \in \mathcal{N}(i)$. Further, for a given agent i, its neighbourhood of agents and neighbourhood of entities are denoted by $\mathcal{N}_a(i)$ and $\mathcal{N}_e(i)$ respectively. Note that, $\mathcal{N}(i) = \mathcal{N}_a(i) \cup \mathcal{N}_e(i)$

3.2 Spatial Dependency Modelling

3.2.1 Initialization Module

Figure 3.2 illustrates the pipeline of the initialization module for a given agent i. At any given time step t, the local observation o_i^t perceived by agent i, constitutes of its position and velocity. The raw input o_i^t is encoded into a fixed-length state vector $x_i^t = \phi_S(o_i^t)$, where $\phi_S(.)$ denotes the state encoding function.

For each entity $u \in \mathcal{N}_e(i)$, agent i computes the encoding $e_{iu}^t = \phi_E(o_{iu}^t)$, where $\phi_E(.)$ denotes the entity encoding function. Here, o_{iu}^t represents i's observation of u (i.e. the relative position of entity u with respect to agent i). As o_{iu}^t is determined by considering a given agent's own perception of its neighbouring entities, explicit communication between the agent and entities is not required.

Each agent i, linearly transforms the encoding x_i^t into $\tilde{Q}_i^t$ and every encoding e_{iu}^t into K_{iu}^t and V_{iu}^t. The graph attention mechanism [41] is utilised to compute attention weights between $\tilde{Q}_i^t$ and all K_{iu}^t values. A fixed-length output vector $\bar{e}_i^t$ is computed as a weighted sum of the V_{iu}^t values with the attention weights. The node feature vector h_i^t associated with each agent i at time step t, is obtained by concatenating x_i^t with $\bar{e}_i^t$.

$$h_i^t = [\ x_i^t \ || \ \bar{e}_i^t\ ] \tag{3.1}$$

where $||$ denotes the concatenating operation. Intuitively, h_i^t represents agent i's perception of its own local state as well as the entities within its neighbourhood.

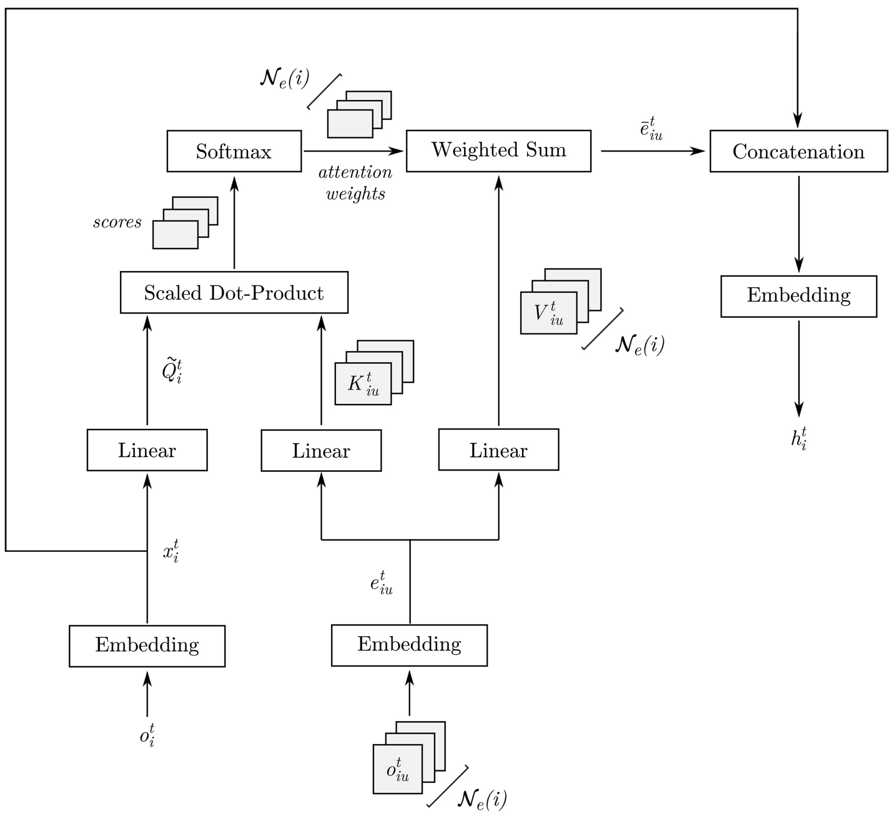

Fig. 3.2 Architecture of the initialization module for a given agent i

3.2.2 Message Propagation Module

At each time step t, an internal propagation process is repeated several times to capture the spatial correlations among agents coherently. Thus, the node feature vector h^t_i computed above is considered as the node feature vector of agent i at propagation step 0 (in time step t) and is re-written as $h^{t,0}_i$. Note that, in deriving the propagation module, we omit time t for notational simplicity. Thus, $h^{t,0}_i \to h^0_i$.

At every propagation step d, each agent $i \in \mathcal{V}$ linearly projects its current node feature vector to key, value and query representations denoted by K_i, V_i and Q_i respectively. Thereafter, each agent i receives a message containing the query-value pair (K_j, V_j) from every agent $j \in \mathcal{N}_a(i)$. For a given agent i, let $\mathcal{N}_a(i)_+$ denote $\mathcal{N}_a(i)$ including i. Upon receiving all the messages from its corresponding neighbours, agent i computes a score using the scaled dot-product attention mechanism [42] for each agent $j \in \mathcal{N}_a(i)_+$. These scores indicate the importance of a given neighbour's information to the agent of interest. These scores are subsequently used to obtain the respective attention weights, $\alpha_{ij} = \text{softmax}\left(\frac{Q_i K_j^\top}{\sqrt{d_K}}\right)$, where d_K is the dimensionality of K. The aggregated message vector m_i^d is computed by taking the weighted sum of the neighbours' values V_j and passing the output through a dense layer.

$$m_i^d = W_o \sum_{j \in \mathcal{N}(i)_+} \alpha_{ij} V_j \tag{3.2}$$

where W_o denotes the trainable parameter weights of the dense layer. The node feature vector of every agent is updated based on the current node feature and the aggregated message vector.

$$h_i^{d+1} = \phi_U([\ h_i^d \ || \ m_i^d\ ]) \tag{3.3}$$

where $\phi_U(.)$ denotes the update function.

The above process represents a single propagation step (one-hop) in which a given agent i acquires information from its neighbouring agents (Figure 3.3). Thus, repeating the process a fixed number of steps (a multi-hop propagation mechanism) enables a given agent to gradually increase its receptive field. Specifically, for a sparsely connected multi-agent network, this enables an agent to propagate information to agents outside its communication range.

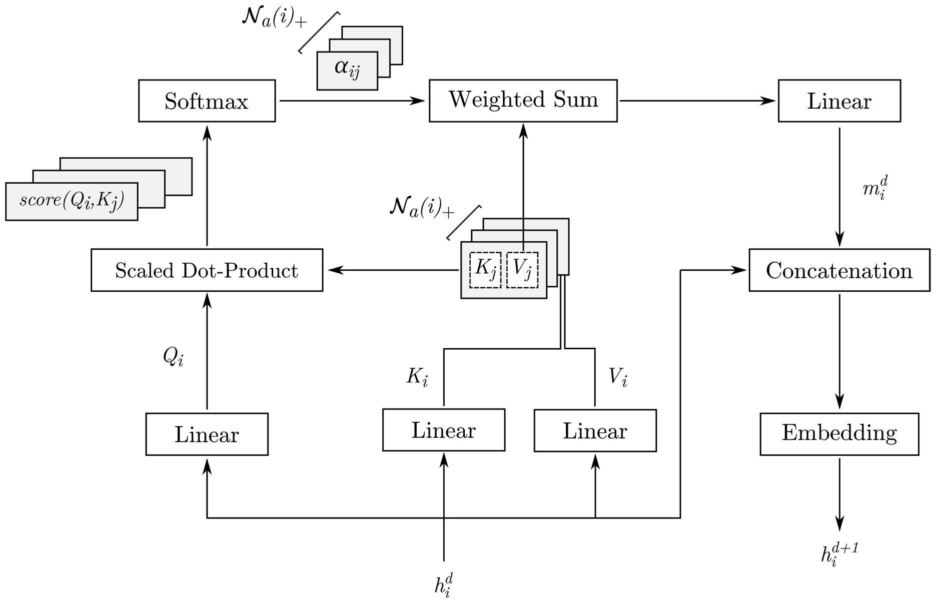

Fig. 3.3 The message propagation module illustrating a single propagation step for a given agent i.

3.3 Temporal Dynamics Modelling

Let $\bar{h}_i^t$ denote the final node feature vector of agent i obtained after a fixed number of propagation steps (at time step t). We incorporate historical information in an attempt to capture temporal dynamic features.

$$\hat{h}_i^t = \phi_T(\bar{h}_i^t,\ \hat{h}_i^{t-1}) \tag{3.4}$$

Two possible alternatives; GRUs [43] and LSTMs [1] were considered when selecting a suitable recurrent structure for $\phi_T(.)$, where the selection was carried out on the basis of accuracy and memory efficiency of the underlying architecture. The corresponding system of equations which composes $\phi_T(.)$ upon adopting a GRU or LSTM architecture are shown by equations (3.5) and (3.6) respectively. Note that, we omit i for notational simplicity.

$$\begin{aligned}
z^t &= \sigma\left(W_z\bar{h}^t + U_z\hat{h}^{t-1} + b_z\right)\\
r^t &= \sigma\left(W_r\bar{h}^t + U_r\hat{h}^{t-1} + b_r\right)\\
\ddot{h}^t &= \tanh\left(W_h\bar{h}^t + U_h\left(r_t \odot \hat{h}^{t-1}\right) + b_h\right)\\
\hat{h}^t &= \left(1 - z^t\right) \odot \hat{h}^{t-1} + z_t \odot \ddot{h}_t
\end{aligned} \tag{3.5}$$

$$\begin{aligned}
f^t &= \sigma\left(W_f\bar{h}^t + U_f\hat{h}^{t-1} + b_f\right)\\
u^t &= \sigma\left(W_u\bar{h}^t + U_u\hat{h}^{t-1} + b_u\right)\\
o^t &= \sigma\left(W_o\bar{h}^t + U_o\hat{h}^{t-1} + b_o\right)\\
\tilde{c}^t &= \tanh\left(W_c\bar{h}^t + U_c\hat{h}^{t-1} + b_c\right)\\
c^t &= f^t \odot c^{t-1} + u^t \odot \tilde{c}^t\\
\hat{h}^t &= o^t \odot \tanh\left(c^t\right)
\end{aligned} \tag{3.6}$$

where W_z, W_r, W_f, W_u, W_o, W_c, U_z, U_r, U_f, U_u, U_o, U_c denotes the corresponding weights and b_z, b_r, b_f, b_u, b_o, b_c denotes the corresponding bias terms. σ denotes the non-linearity function and $\odot$ denotes the Hadamard product.

As given in (3.5), the GRU is composed of a simpler structure with only update gates z^t and reset gates r^t. The number of parameters it takes is smaller compared to LSTM (3.6), which consists of three gates; the forget gate f^t, input gate u^t, and output gate o^t. Therefore, in terms of memory efficiency, GRU may be considered as a better option. However, the presence of a forget gate in LSTM makes it more suitable for long sequences. Particularly, within our model, sequences refer to entire episodes. Thus, being able to capture long sequences more accurately is of paramount importance. Therefore, we incorporate the LSTM architecture (Figure 3.4) as the temporal unit of the underlying model despite the potential burden on memory.

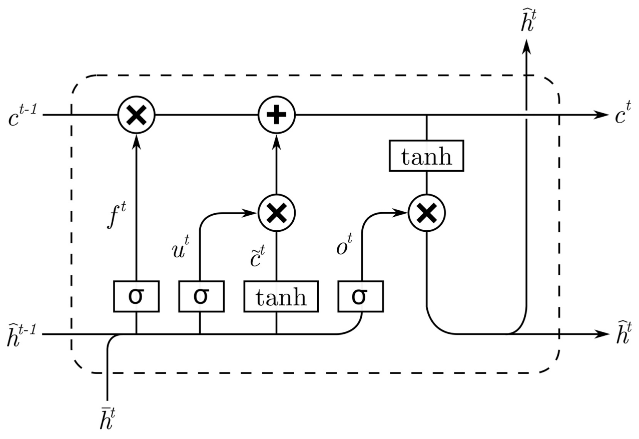

Fig. 3.4 Architecture of the basic LSTM [1] memory cell utilised in temporal dependency modelling.

Intuitively, on the algorithm's perspective the output of the temporal unit $\hat{h}_i^t$ represents an enriched representation of the raw observation o_i^t perceived by a given agent i. That is, $\hat{h}_i^t$ is more reflective of the spatial influences of the agent-agent interactions as well as the underlying temporal dynamics of the system. During training, $\hat{h}_i^t$ is fed into the corresponding critic network and the actor network for a given agent i to learn optimal policies that facilitates in carrying out the underlying control tasks cooperatively.

3.4 Policy Learning

At this stage, the comprehensive features generated via spatio-temporal dependency modelling are exploited to train decentralised agent policies. The encoding $\hat{h}_i^t$ is fed into the critic network and the actor network of the corresponding agent i, which estimate the state value and a probability distribution over all possible actions, respectively. Upon sampling an action from the distribution, each agent will act accordingly and subsequently receive a joint reward from the environment. In this work, we consider the agents to be homogeneous and thus, share all learnable parameters including those of the encoding functions, update functions, actor and critic networks. As each agent perceives its local environment differently and attends to incoming messages from its neighbourhood differently, sharing parameters does not hinder an agent's ability to act autonomously.

3.4.1 Policy Optimization

Generally, in the process of learning, vanilla policy gradient optimization techniques may lead to less optimal policies. Moreover, considering the degree of non-stationarity that is prevalent within the environment due to the sparse connectivity, it is preferable if there is a guarantee of policy improvement following every update.

$$\eta(\tilde{\pi}) = \eta(\pi) + \mathbb{E}_{s_0, a_0, \ldots \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t A_\pi\left(s_t, a_t\right)\right] \tag{3.7}$$

where A_π denotes the advantage function corresponding to policy π, and $\eta(\pi)$ denotes the expected reward return following the policy π [44].

TRPO [44] is one such algorithm that builds upon the identity (3.7) for policy improvement derived in [45] to guarantee a policy improvement at every step. However, this algorithm relies on the roll-out of multiple trajectories from the current policy, prior to making a single update. Moreover, it also involves an expensive calculation of a Fisher discriminant matrix. Given the expensive pre-possessing carried out within our proposed scheme and the memory burden of the LSTM resulting from the underlying dynamicity, the addition of yet another heavy computation was considered to be unsuitable. Thus, we opted for PPO [18] as the policy optimization technique, which does the same monotonic improvement but with simple first-order terms by the introduction of a clipping mechanism.

Chapter 4

Experimental Setup and Implementation

4.1 Simulation Environment

For the experiments, we consider the two-dimensional multi-agent particle environment proposed in [27] with a discretized action space. The environments constitute of A agents and E entities, where the cooperating agents attempt to carryout out some pre-defined control task. Particularly, we consider three well-known multi-agent cooperative control tasks [24]; target coverage, formation control in terms of polygonal and linear pattern generation (Figure 4.1). We briefly describe each of the three aforementioned environments below.

1. **Coverage Control.** In an environment with N entities, each of the N agents attempts to navigate to a distinct entity, without colliding with one another. As a particular agent is not pre-assigned some entity beforehand, each agent must infer a suitable target entity via consensus.

2. **Polygonal Formation Control.** In this environment, N cooperating agents attempt to position themselves evenly around one entity, such that they form an N-sided regular polygon with the entity at its centre.

3. **Linear Formation Control.** In this environment, N cooperating agents attempt to evenly position themselves along a straight-line between the two entities.

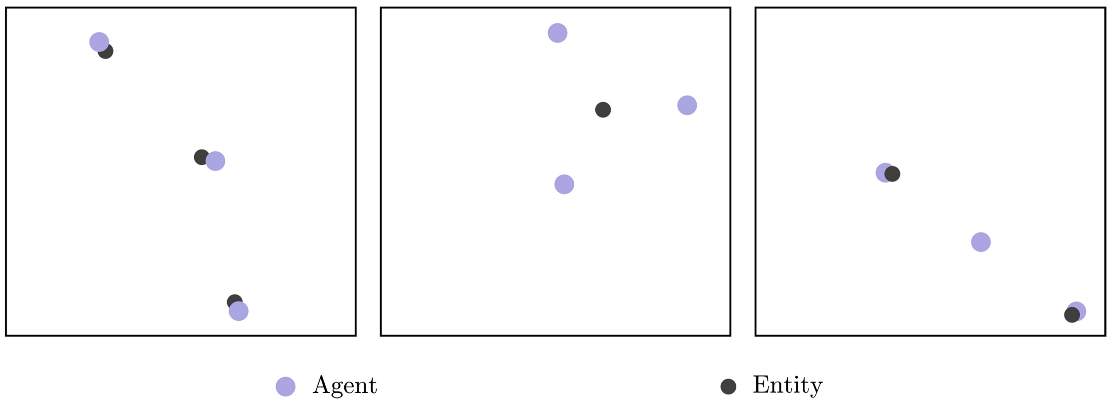

Fig. 4.1 Experimental environments and the associated control tasks. (a) Coverage control, (b) Polygonal formation control, and (c) Linear formation control.

4.2 Implementation Details

4.2.1 Node Features

In this work, the local observation of a particular agent constitutes of its position ((x, y) location) and its velocity. In order to incorporate the notion of a sparsely connected network, we limit the communication range of each agent to a radius of 1 unit. As entities are considered to be stationary, the corresponding node feature of an entity constitutes of its position ((x, y) location) only. Thus, o^t_{iu} denotes the relative position of entity u with respect to agent i an time t. Therefore, the corresponding dimensions of the inputs fed into $\phi_S(.)$ and $\phi_E(.)$ are 4 and 2 units, respectively.

4.2.2 Architecture and Training

Both encoding functions; $\phi_S(.)$ and $\phi_E(.)$ embed their respective inputs into a 128-dimensional vector with a fully connected MLP layer using ReLU non-linearity activation function. The attention layer in the message propagation module uses 128-dimensional keys, values and queries. The update function $\phi_U(.)$ is yet another MLP layer with 128 neurons and ReLU activation. The temporal data is enriched using an LSTM module with 128 neurons (denoted by the function $\phi_T(.)$). This pre-processed data is then used as input for both the critic (value) and actor (policy) networks. Both the inputs and outputs of the temporal unit are concatenated and fed into the critic network and passed through a 128-dimensional MLP layer from which a single value function is obtained. Similarly, the actor network is also fed with the same input as the critic network and subsequently passed through a 128-dimensional MLP layer. The output is fed through a categorical distribution to obtain the probabilities over the possible actions. In all of the aforementioned layers, the ReLU non-linearity activation function is used. As mentioned earlier, we consider the agents to be homogeneous and thus, share all learnable parameters including those of the encoding functions, update functions, actor and critic networks. As each perceives its local environment differently and attends to incoming messages from its neighbourhood differently, sharing parameters does not hinder an agent's ability to act autonomously.

Each episode is composed of 50 time-steps and the algorithm is evaluated after every 30 episodes during training. The corresponding success rate is employed as the final performance metric of the algorithm. The algorithms are trained until convergence. The Adam optimizer with a learning rate of 0.001 is utilised to optimize the neural networks. Each PPO update is done after 128 time-steps.

Chapter 5

Results and Analysis

We employ two other models; [27] and [24], as baselines to validate the performance of the proposed model. In [27], the model learns a centralised critic for each agent with access to joint observations and actions from all agents during learning and thus, does not rely on some explicit communication mechanism. In, [24], the model employs a message propagation mechanism (complementary to that used in our proposed model for spatial dependency modelling) to learn transferable policies, however, does not consider the temporal dynamics and dependencies of the underlying system during policy learning. We analyze and evaluate the models on the aforementioned control tasks; target coverage, formation control in terms of polygonal and linear pattern generation.

We employ two performance metrics; success rate ($S\%$) and time (T) to compare the performance of the proposed model against the aforementioned baselines. The success rate ($S\%$) denotes the percentage of the number of episodes in which the underlying objective was achieved over the total number of episodes. The metric, time (T) denotes the number of time-steps that were required to achieve the underlying objective. Here, an objective refers to the cooperative control task the adaptive agents are expected to perform in the respective environments.

Further, in terms of communication, we consider two cases; restricted communication (RC) and unrestricted communication (UC). In RC settings, the communication constraints for a given agent highlighted throughout this paper hold. That is, each agent is only allowed to directly communicate with its neighbours. In UC settings, such constraints do not hold, thus, each agent can communicate with all the other agents and entities within the population (in which case the MAS can be represented by a fully connected graph $\mathcal{G}$).

Table 5.1 and Table 5.2 summarize the performance of the different models under the aforementioned learning settings with $A = 3$ and $A = 6$ agents, respectively.

Table 5.1 Performance comparisons in learning environments with A= 3 agents

Control Task	Algorithm	Observability	Communication	T	$S\%$
Coverage	Lowe et al. [27]	Full	N/A	17.89	95
	Yang et al. [24]	Partial	UC	14.12	100
		Partial	RC	14.22	98
	Ours	Partial	UC	**9.27**	**100**
		Partial	RC	**11.36**	**100**
Polygonal Formation	Lowe et al. [27]	Full	N/A	15.66	95
	Yang et al. [24]	Partial	UC	13.56	100
		Partial	RC	12.97	98
	Ours	Partial	UC	**8.10**	**100**
		Partial	RC	**11.00**	**100**
Linear Formation	Lowe et al. [27]	Full	N/A	35.84	58
	Yang et al. [24]	Partial	UC	15.14	98
		Partial	RC	15.24	97
	Ours	Partial	UC	**9.77**	**100**
		Partial	RC	**12.03**	**100**

Higher $S\%$ values indicate the ability of the agents to learn effective policies autonomously to execute the underlying control task cooperatively. Lower T values indicate the ability of the agents to learn such policies quickly (i.e. faster converging policies). The results indicate that our proposed model outperforms the others baselines against the two performance metrics, under both restrictive and non-restrictive communication conditions whilst being subjected to partial observability.

MADDPG [27] requires a considerably longer time to learn suitable policies, yet results in lower success rates compared to the other two models, which is particularly evident in the line formation control task. Moreover, as the agent population increases, MADDPG fails to converge sufficiently quickly compared to the other two models, despite the agents having full observability (i.e. access to other agents' information) during training. The baseline model [24] learns reasonable policies in all control tasks, however, it requires relatively more time as reflected by the performance metric T. The proposed model strictly dominates the baseline models, both in terms of training speed and consistently achieving the underlying control objective across all learning settings. This is attributed to effectively extracting latent features across the temporal domain in addition to the spatial domain which evidently enables an agent to implicitly infer the behaviour strategies of other agents, thereby providing a faster convergence to the underlying objective.

Thus, the experimental results indicate the significance of exploiting spatio-temporal dependency modelling to compensate for the loss of information due to sparse connectivity and partial observability, to facilitate adaptive agents to quickly learn more meaningful and robust policies.

Table 5.2 Performance comparisons in learning environments with A= 6 agents

Control Task	Algorithm	Observability	Communication	T	$S\%$
Coverage	Lowe et al. [27]	Full	N/A	50	0
	Yang et al. [24]	Partial	UC	20.47	93
		Partial	RC	48.32	5
	Ours	Partial	UC	**16.32**	**96**
		Partial	RC	**17.53**	**93**
Polygonal Formation	Lowe et al. [27]	Full	N/A	50	0
	Yang et al. [24]	Partial	UC	14.22	100
		Partial	RC	14.26	100
	Ours	Partial	UC	**11.17**	**100**
		Partial	RC	**10.07**	**100**
Linear Formation	Lowe et al. [27]	Full	N/A	50	0
	Yang et al. [24]	Partial	UC	16.31	100
		Partial	RC	17.07	100
	Ours	Partial	UC	**9.77**	**100**
		Partial	RC	**12.33**	**100**

5.1 Preliminary Directions

5.1.1 Pairwise Approximation of the Q-Function in MARL

We first approached the notion of sparse connectivity within a MAS by considering the smallest unit of interacting agents within any learning environment; a pair of agents. Following the CTDE framework, in a MAS with N agents, each agent $j \in \{1, 2 \ldots, N\}$ utilises a centralised critic in the form of $Q_j(s, \mathbf{a})$, where $\mathbf{a} = \{a_1, \ldots, a_N\}$ denotes the joint action. In a sparsely connected learning environment, the accessibility to the joint action $\mathbf{a}$ may be limited for certain agents. Moreover, as the dimensionality of the joint action **a** grows proportionally with the number of agents, considering the associated computational cost and complexity, learning the standard Q-function may be infeasible. Following the work of Yang et al. [29], the standard Q-function can be approximated by the mean effect of the local interactions within a given agent's neighbourhood. However, as the size of the neighbourhood can vary dynamically, we assume that at a given time, any given agent j is guaranteed to interact at least with one of its neighbours (say with the neighbouring agent k). Thus, the N-agent stochastic game can be reduced to multiple 2-agent stochastic games. For any given pair of agents $j, k \in N$, we define the "pairwise Q-function" as follows.

$$Q_j\left(s, a_j, a_k\right) = \mathbb{E}_{\pi_{jk}}\left[R_j^t \mid s^t = s, a_j^t = a_j, a_k^t = a_k\right] \tag{5.1}$$

where $\pi_{jk} = [\pi_j \; \pi_k]$.

As we focus on a problem setting with CTDE, the policies are independent. Thus,

$$\pi\left(a_j, a_k|s\right) = \pi_j\left(a|s\right)\pi_k\left(a|s\right) \tag{5.2}$$

A potential issue that arises from this setting is that from the perspective of the pair of agents, it is assumed that the actions of the agent-pair are solely responsible for the global reward, whereas the true pairwise Q-function of a pair would be given as,

$$Q_j^{true}\left(s, a_j, a_k\right) = \sum_{i \in X(i)} \pi\left(\{a_i\}_i \mid s\right) Q\left(s, a_1, \ldots, a_i, \ldots, a_n\right)$$

where $X(i) = j^{-1} \cap k^{-1}$ represents the index set of all the agents other than agent j and k. As the policies are considered to be independent under the CTDE paradigm,

$$\pi\left(\{a_i\}_i \mid s\right) = \pi_1\left(a \mid s\right) \cdot \pi_2\left(a \mid s\right) \cdot \ldots \cdot \pi_i\left(a \mid s\right) \cdot \ldots \cdot \pi_n\left(a \mid s\right)$$

This is obtained by keeping the action-pair of interest constant and marginalising out the actions of all the other agents. One particular issue with this approach is that, given that we have the estimation for all pairs, minimizing the difference between the true pairwise Q-function and our estimation. A potential solution for this would be to formulate a credit function.

We formulate an action-independent baseline function from the estimated pairwise Q-function (we provide the complete proof in the Appendix).

$$b(s) = \sum_j \pi_j(a|s) \cdot \sum_k \pi_k(a|s) \cdot Q_j(s, a_j, a_k) \tag{5.3}$$

With the above baseline (5.3), we can now define an advantage function as follows.

$$A_j(s, \mathbf{a}) = Q_j(s, \mathbf{a}) - b(s) \tag{5.4}$$

Intuitively, the idea here is that if a particular action-pair of interest is responsible for the change in the Q-function, then the baseline would not be able to reduce the advantage function. However, when the particular action-pair is not responsible for the change in the Q-function and it was due to the actions of other agent pairs, the baseline would reduce the advantage function value. However, with this setup advantage function is dependant on the global Q-function, $Q(s, \mathbf{a})$. Thus, in order to utilise the advantage function to proceed in meaningful directions, we still require a valid estimate to the global Q-function in sparsely connected learning environments. Consequently, we shifted the focus of the research to GNN-based methods to first obtain an estimate to the global Q-function and this was the direction that was subsequently covered in Chapter 3.

Chapter 6

Conclusions and Future Works

In many multi-agent systems, establishing consensus among adaptive agents to solve some complex control task is a challenging problem. In this work, we introduce a MARL model to leverage the learning of optimal cooperative agent policies by exploiting the inherent graph-like structure of multi-agent systems. The proposed model utilises spatio-temporal dependency modelling to compensate for the loss of information due to sparse connectivity and partial observability prevalent within real-world multi-agent learning environments. Spatial dependency modelling includes an iterative message-propagation mechanism to ensure information propagation through the network via intermediary nodes. Temporal dependency modelling utilises a recurrent structure which retains information and propagates such information through time. Thus, as opposed to raw local observations, spatio-temporal dependency modelling generates state representations which are more reflective of the spatial influences of the agent-agent interactions as well as the underlying temporal dynamics of the system. Such state representations are thereby utilised by the agents in learning meaningful policies to perform some cooperative control task. Empirically, we demonstrate the effectiveness of the proposed model on a variety of well-known cooperative control tasks. Further, we show that our model outperforms other MARL models given its ability to generate faster converging policies and learn more robust coordination strategies coherently.

In terms of future work, one potential avenue would be to incorporate the pairwise Q-function and the baseline function which were formulated as part of the preliminary work of this project. As we now have access to an estimation of the centralised Q-function, we can verify and validate the viability of this approach and the potential advantages it may present. Moreover, we can extend the model to incorporate edge features in addition to node features during spatial dependency modelling, to further enrich the state representations. For instance, in a learning environment with heterogeneous agents, edge features can be exploited to be reflective of the nature of the relationships among agents. As of now, the algorithm was validated against simple and primitive environment settings. Thus, we can further optimize the algorithm in large, complex and diversified environments such that it can be utilised in real-world applications.

References

[1] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[2] P. Balaji and D. Srinivasan, “An introduction to multi-agent systems,” in *Innovations in multi-agent systems and applications-1*, pp. 1–27, Springer, 2010.

[3] L. Bu, R. Babu, B. De Schutter, *et al.*, “A comprehensive survey of multiagent reinforcement learning,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.

[4] R. S. Sutton, A. G. Barto, *et al.*, *Introduction to reinforcement learning*, vol. 2. MIT press Cambridge, 1998.

[5] J. Kober, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[6] J. A. Boyan and M. L. Littman, “Packet routing in dynamically changing networks: A reinforcement learning approach,” in *Advances in neural information processing systems*, pp. 671–678, 1994.

[7] D. Ye, M. Zhang, and Y. Yang, “A multi-agent framework for packet routing in wireless sensor networks,” *sensors*, vol. 15, no. 5, pp. 10026–10047, 2015.

[8] Y. Wei and M. Zhao, “A reinforcement learning-based approach to dynamic job-shop scheduling,” *Acta Automatica Sinica*, vol. 31, no. 5, p. 765, 2005.

[9] D. B. Noureddine, A. Gharbi, and S. B. Ahmed, “Multi-agent deep reinforcement learning for task allocation in dynamic environment.,” in *ICSOFT*, pp. 17–26, 2017.

[10] J. G. Schneider, W.-K. Wong, A. W. Moore, and M. A. Riedmiller, “Distributed value functions,” in *Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 371–378, 1999.

[11] M. Riedmiller, A. Moore, and J. Schneider, “Reinforcement learning for cooperating and communicating reactive agents in electrical power grids,” in *Workshop on Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, pp. 137–149, Springer, 2000.

[12] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, “Graph neural networks: A review of methods and applications,” *arXiv preprint arXiv:1812.08434*, 2018.

[13] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, “A comprehensive survey on graph neural networks,” *arXiv preprint arXiv:1901.00596*, 2019.

[14] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

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

[16] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[17] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. M. O. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *CoRR*, vol. abs/1509.02971, 2015.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *ArXiv*, vol. abs/1707.06347, 2017.

[19] J. van der Wal, *Stochastic Dynamic Programming: Successive Approximations and Nearly Optimal Strategies for Markov Decision Processes and Markov Games*. Mathematical Centre tracts, Mathematisch Centrum, 1981.

[20] K. Tuyls and G. Weiss, “Multiagent learning: Basics, challenges, and prospects,” *Ai Magazine*, vol. 33, no. 3, pp. 41–41, 2012.

[21] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, “Deep reinforcement learning for multi-agent systems: A review of challenges, solutions and applications,” *arXiv preprint arXiv:1812.11794*, 2018.

[22] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, “Counterfactual multi-agent policy gradients,” in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[23] F. S. Melo and M. Veloso, “Learning of coordination: Exploiting sparse interactions in multiagent systems,” in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 773–780, International Foundation for Autonomous Agents and Multiagent Systems, 2009.

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

[25] H. Ryu, H. Shin, and J. Park, “Multi-agent actor-critic with hierarchical graph attention network,” *arXiv preprint arXiv:1909.12557*, 2019.

[26] J. Sheng, X. Wang, B. Jin, J. Yan, W. Li, T.-H. Chang, J. Wang, and H. Zha, “Learning structured communication for multi-agent reinforcement learning,” *arXiv preprint arXiv:2002.04235*, 2020.

[27] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” in *Advances in neural information processing systems*, pp. 6379–6390, 2017.

[28] J. Yang, A. Nakhaei, D. Isele, K. Fujimura, and H. Zha, “Cm3: Cooperative multi-goal multi-stage multi-agent reinforcement learning,” *arXiv preprint arXiv:1809.05188*, 2018.

[29] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, “Mean field multi-agent reinforcement learning,” *arXiv preprint arXiv:1802.05438*, 2018.

[30] S. Iqbal and F. Sha, “Actor-attention-critic for multi-agent reinforcement learning,” *arXiv preprint arXiv:1810.02912*, 2018.

[31] P. Peng, Q. Yuan, Y. Wen, Y. Yang, Z. Tang, H. Long, and J. Wang, “Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games,” *arXiv preprint arXiv:1703.10069*, vol. 2, 2017.

[32] S. Sukhbaatar, R. Fergus, *et al.*, “Learning multiagent communication with backpropagation,” in *Advances in neural information processing systems*, pp. 2244–2252, 2016.

[33] A. Singh, T. Jain, and S. Sukhbaatar, “Learning when to communicate at scale in multiagent cooperative and competitive tasks,” *arXiv preprint arXiv:1812.09755*, 2018.

[34] J. Jiang and Z. Lu, “Learning attentional communication for multi-agent cooperation,” in *Advances in neural information processing systems*, pp. 7254–7264, 2018.

[35] A. Das, T. Gervet, J. Romoff, D. Batra, D. Parikh, M. Rabbat, and J. Pineau, “Tarmac: Targeted multi-agent communication,” *arXiv preprint arXiv:1810.11187*, 2018.

[36] T. Wang, R. Liao, J. Ba, and S. Fidler, “Nervenet: Learning structured policy with graph neural networks,” in *International Conference on Learning Representations*, 2018.

[37] J. Jiang, C. Dun, and Z. Lu, “Graph convolutional reinforcement learning for multi-agent cooperation,” *arXiv preprint arXiv:1810.09202*, vol. 2, no. 3, 2018.

[38] J. Sheng, X. Wang, B. Jin, J. Yan, W. Li, T.-H. Chang, J. Wang, and H. Zha, “Learning structured communication for multi-agent reinforcement learning,” *arXiv preprint arXiv:2002.04235*, 2020.

[39] Y. Wang, T. Xu, X. Niu, C. Tan, E. Chen, and H. Xiong, “Stmarl: A spatio-temporal multi-agent reinforcement learning approach for traffic light control,” *arXiv preprint arXiv:1908.10577*, 2019.

[40] G. Hu, B. Cui, Y. He, and S. Yu, “Progressive relation learning for group activity recognition,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 980–989, 2020.

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

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[43] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, “On the properties of neural machine translation: Encoder-decoder approaches,” *arXiv preprint arXiv:1409.1259*, 2014.

[44] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in *International conference on machine learning*, pp. 1889–1897, 2015.

[45] S. Kakade and J. Langford, “Approximately optimal approximate reinforcement learning,” in *ICML*, vol. 2, pp. 267–274, 2002.

Appendix A

Pairwise Approximation of the Q-Function in MARL

In a sparsely connected MAS, the smallest unit of interacting agents within the learning environment is a pair of agents. Following this line of thought, we reduce an N-agent stochastic game to multiple 2-agent stochastic games. Similar to the approach used in independent actor-critic (IAC) [22] which assumes a 1-agent stochastic game, we formulate the 2-agent stochastic game as follows. For any given pair of agents $j, k \in N$, we define the "pairwise Q-function" as,

$$Q_j\left(s, a_j, a_k\right) = \mathbb{E}_{\pi_{jk}}\left[R_j^t \mid s^t = s, a_j^t = a_j, a_k^t = a_k\right] \tag{A.1}$$

where $\pi_{jk} = [\pi_j \ \pi_k]$

As we focus on decentralized execution, the policies are considered to be independent. Thus,

$$\pi\left(a_j, a_k | s\right) = \pi_j\left(a|s\right)\pi_k\left(a|s\right) \tag{A.2}$$

From the Bellman expectation equation [4],

$$Q_j\left(s, a_j, a_k\right) = E_{\pi_{jk}}\left[R_j^t + \gamma Q_j\left(s^{t+1}, a_j^{t+1}, a_k^{t+1}\right) \mid s^t = s, a_j^t = a_j, a_k^t = a_k\right] \tag{A.3}$$

where $a_j^{t+1} \sim \pi_j\left(a|s^{t+1}\right)$ and $a_k^{t+1} \sim \pi_k\left(a|s^{t+1}\right)$

Thus, this pairwise Q-function parameterized by θ_{Q_c}, can be learnt by minimizing the following loss function,

$$\mathcal{L}\left(\theta_{Q_c}\right) = \mathbb{E}\left[y - Q_j\left(s^t, a_j^t, a_k^t \ ; \ \theta_{Q_c}\right)\right]. \tag{A.4}$$

where

$$y = R_j^t + \gamma Q_j\left(s^{t+1}, \pi\left(a_j|s^{t+1}\right), \pi\left(a_k|s^{t+1}\right)\ ;\ \theta_{Q_c}\right). \tag{A.5}$$

Furthermore, if we marginalize a_j, a_k out, we obtain,

$$\begin{aligned}
\sum_{jk} \pi(a_j, a_k|s) \cdot Q_j(s, a_j, a_k) &= \sum_{jk} \pi_j(a|s) \cdot \pi_k(a|s) \cdot Q_j(s, a_j, a_k) \\
&= \sum_{j} \pi_j(a|s) \cdot \sum_{k} \pi_k(a|s) \cdot Q_j(s, a_j, a_k) \\
&= \sum_{j} \pi_j(a|s) \cdot Q_j(s, a_j) \\
&= V_j(s)
\end{aligned}$$

An action-independent baseline function can be constructed from the pairwise Q-function as follows.

$$b(s) = \sum_{j} \pi_j(a|s) \cdot \sum_{k} \pi_k(a|s) \cdot Q_j(s, a_j, a_k) \tag{A.6}$$

A.1 Proposition

The baseline function $b(s)$ (A.6) does not introduce any bias into the gradient calculation. $\mathbb{E}_{\pi_{jk}}\left[b(s)\nabla_\theta \log \pi(a|s)\right]$ evaluates to zero, and hence has no effect on the gradient update.

We follow the proof of the policy gradient theorem [4],

$$\nabla E_{\pi_{jk}}[b(s)] = \sum_{s\in S} d^{\pi_{jk}}(s) \cdot \sum \nabla_\theta \pi_{jk}(a|s) \cdot b(s)$$

where $d^{\pi_{jk}}(s)$ denotes the state distribution following π_{jk}.
Since $b(s)$ is independent of actions,

$$\begin{aligned}
\nabla E_{\pi_{jk}}[b(s)] &= \sum_{s\in S} d^{\pi_{jk}}(s) \cdot \left(\sum \nabla_\theta \pi_{jk}(a|s)\right) \cdot b(s) \\
&= \sum_{s\in S} d^{\pi_{jk}}(s) \cdot \left(\sum \nabla_\theta 1\right) \cdot b(s) \\
&= \sum_{s\in S} d^{\pi_{jk}}(s) \cdot 0 \cdot b(s) \\
&= 0
\end{aligned}$$