Towards Cooperative MARL in Industrial Domains

By

JONATHAN DAVID THOMAS

Department of Electrical and Electronic Engineering
UNIVERSITY OF BRISTOL

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of DOCTOR OF PHILOSOPHY in the Faculty of Engineering.

MARCH 2023

Word count: Forty one thousand seven hundred and twelve

Abstract

This thesis investigates the application of Deep Multi-Agent Reinforcement Learning (DMARL) to problems within telecommunications and logistics. These sectors are exemplary of a common class of industrial systems that are comprised of large numbers of interconnected and interdependent assets. Traditionally, optimisation of these systems is achieved through the utilisation of heuristic and/or human expertise. However, due to their inherent complexity and dimensionality, these approaches are often significantly sub-optimal. Deep Reinforcement Learning (DRL) has proved successful in several real-world problems, but the aforementioned characteristic of these domains precludes its direct application. Alternatively, we can instantiate each asset as an agent and apply DMARL methodologies. This addresses the dimensionality but requires cooperative behaviours to be induced. Herein, we detail our efforts deriving novel cooperative DMARL solutions to representative problems in our target industrial domains. Our telecommunications work considers network maintenance planning, which requires a finite amount of maintenance resources to be assigned among network equipment. The logistics work explores the order-picking problem, in which human and robot workers must collaborate to collect and deliver items distributed around a commercial warehouse. In both cases, we develop and empirically validate novel DMARL algorithms within simulated environments and demonstrate improvements over relevant industrial heuristics. Furthermore, inspired by these domains, we anticipate language playing a key part in future industrial systems as a means to enable cooperation among diverse sets of agents. As such, we conduct investigations into the fundamental challenges of automatically establishing a common language from scratch for agents to effectively communicate. Collectively, this work serves as a first step towards exploiting the potential of cooperative DMARL for industrial applications and provides paths towards their realisation within real-world settings.

Dedication and acknowledgements

Without the support of those that I hold dear, I almost certainly would not have made it this far. There are numerous peoples whose support, guidance and empathy has been crucial in making it here. These include my colleagues, friends, supervisors, family and most importantly my partner Caity.

I'd like to begin by thanking my partner Caity. Her unwavering support and belief were crucial and compensated for times when my own resolve was lacking. I am incredibly lucky to have her alongside me and look forward to our next adventures together.

My family have provided a much-needed anchor throughout this period. The support of my mum has been particularly notable, her willingness to be a guinea pig for my presentations was very much appreciated.

My supervisors, Rob and Raul, have been instrumental in this entire process. They have helped to guide me towards interesting topics and encouraged me to explore novel questions and ideas. Alongside my supervisors, I would like to thank other members of our group, especially Xiaoyang who I've been fortunate enough to have alongside me for the entirety of my PhD.

Many other people and organisations have played a part in my PhD. I'd like to thank (or maybe blame) Rhys for encouraging me to pursue a PhD. The NG-CDI consortium provided access to a wide variety of experts whom I've been lucky enough to share many interesting conversations with. The University of Bristol, especially Suzanne Binding, Mark Beach and Oliver Johnson, have been incredibly supportive.

This work is funded by the Next-Generation Converged Digital Infrastructure (NG-CDI) Project, supported by BT and Engineering and Physical Sciences Research Council (EPSRC), Grant ref. EP/R004935/1.

AUTHOR'S DECLARATION

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: ..... [redacted] ..... DATE: 27/09/2023

Table of Contents

List of Tables

Table **Page**

List of Figures

Acronyms

3GPP The 3rd Generation Partnership Project. 35

4G 4th Generation. 26, 38

5G 5th Generation. 26, 27, 38

6G 6th Generation. 26

A2C Advantage Actor-Critic. 48, 49

AC Actor-Critic. 18, 20, 24, 26, 98

BBU Baseband Unit. 35

BPP Bin Packing Problem. xi, 5, 29, 35, 36, 38, 39

BS Base Station. 3, 27–29, 50, 52

CBRICP Condition-Based Component Replacement and Inventory Control Policy. 44

CMDP Constrained Markov Decision Process. 23, 100

CNN Convolutional Neural Networks. 13

COMA Counterfactual Multi-Agent Policy Gradient. 2, 18, 20

commMAAC Communicative Multi-Agent Actor-Critic. xi, 54–56

convMAAC Convolutional Multi-Agent Actor-Critic. 5, 34, 48, 51–53, 56, 57, 77, 98

CPU Centralised Processing Unit. 41

CTDE Centralised Training for Decentralised Execution. 5, 19, 20, 24, 48, 98

CU Central Unit. 35

DDPG Deep Deterministic Policy Gradient. 20, 48

DEC-POMDP Decentralised Partially Observable Markov Decision Process. 16, 18, 23

DL Deep Learning. 1, 13, 24, 55

DMARL Deep Multi-Agent Reinforcement Learning. i, xi, 1–6, 9, 10, 16, 18–31, 33, 34, 41–43, 48, 53, 54, 56, 57, 59, 77, 78, 94, 95, 97–100

DNN Deep Neural Network. 13, 14, 23, 24, 38, 42, 81, 98

DP Dynamic Programming. 11, 31, 43

DQN Deep Q-Networks. 9, 13–15, 18–20, 25, 28, 29, 44

DRL Deep Reinforcement Learning. i, xi, 1, 2, 5, 9, 10, 13, 16, 17, 22–27, 29–31, 33, 34, 41, 42, 47, 48, 57, 79, 97, 99, 100

DSA Dynamic Spectrum Access. 28

DU Distributed Unit. 34–36, 39, 41, 42

EC Emergent Communication. 21, 78, 79

FCN Fully Connected Network. 24, 52, 57, 98

FM Follow Me. 73, 98

GAE Generalised Advantage Estimation. 6, 24, 60

GCN Graph Convolutional Network. 49, 52, 56, 57

GNN Graph Neural Networks. 5, 13, 24, 26, 28, 30, 34, 98

GPU Graphic Processing Unit. 41

HRL Hierarchical Reinforcement Learning. 24, 98

HSNAC Hierarchical Shared Network Actor-Critic. xi, 6, 74, 77, 98

HVRAA Heuristic Virtual Resource Allocation Algorithm. 29, 38

IAC Independent Actor-Critic. 18, 52, 53, 56

IL Independent Learners. 2, 18, 20, 57, 77, 79, 81

IoT Internet of Things. 28

IQL Independent Q-Learning. 18, 19, 28–30

KPI Key Performance Indicators. 22

LSTM Long Short-Term Memory. 23, 24, 28, 52

MAAC Multi-Agent Actor-Critic. xi, 43, 48–56

MADDPG Multi-Agent Deep Deterministic Policy Gradient. 2, 9, 19, 20, 34, 42, 48

MAPPO Multi-Agent Proximal Policy Optimisation. 20

MARL Multi-Agent Reinforcement Learning. xi, 9, 16–18, 22, 29, 44, 45, 51

MAS Multi-Agent Systems. xi, 1–3, 16

MCTS Monte-Carlo Tree Search. 29, 37–39

MDP Markov Decision Process. xi, 5, 9, 10, 12, 13, 16, 23, 28, 31, 34, 36, 37, 43–45

MFRL Mean-Field Reinforcement Learning. 19

MHDPA Multi-Headed Dot Product Attention. 54, 55

ML Machine Learning. 13, 33, 77, 81, 100

ML-Ops Machine Learning Operations. 25, 100

O-RAN Open Radio Access Network. xi, 29, 33–35, 38, 57, 97

PDM Pick Don't Move. 73, 98

PG Policy Gradient. 16, 18, 20

PI Policy Iteration. 11

POMDP Partially Observable Markov Decision Process. 12, 13, 23, 28, 44

POSG Partially Observable Stochastic Game. 5, 16, 18, 23, 30, 34, 43, 45, 56

PPO Proximal Policy Optimisation. 16, 20, 26

QoE Quality of Experience. 35

QoS Quality of Service. 3, 33

R2 Ranked Reward. 35, 37

RAM Random Access Memory. 41

RAN Radio Access Network. 5, 29, 30, 34, 35, 42, 43, 50, 56, 57, 98

REINFORCE REward Increment = Nonnegative Factor × Offset Reinforcement × Characteristic Eligibility. 9, 16, 26, 29, 79, 80

RG Referential Games. 78, 79, 84, 87, 88, 91, 94

RIC Radio Intelligent Controller. 33

RL Reinforcement Learning. xi, 1, 3, 9–13, 16, 18, 22–25, 29, 43, 44

RNN Recurrent Neural Networks. 13

RU Remote Unit. 34–39, 41, 42

RUL Remaining Useful Life. 52, 53

SEAC Shared Experience Actor-Critic. 20, 30

SG Stochastic Game. 9, 16–18, 23

SNAC Shared Network Actor-Critic. 74

SOTA State of the Art. 1, 80, 88

TRPO Trust Region Policy Optimisation. 16

UE User Equipment. 27

VDN Value Decomposition Networks. 9, 19, 20, 28

ZSC Zero-Shot Coordination. 18, 22, 24, 100

CHAPTER 1

INTRODUCTION

Through a trial-and-error like process, Reinforcement Learning (RL) offers the ability to automatically learn near-optimal[1] behaviours in sequential decision-making tasks through maximisation of a reward signal [1]. The adoption of Deep Learning (DL) methodologies within this paradigm has proved pivotal in its extension to high dimensional problems [2]. This combination is referred to as Deep Reinforcement Learning (DRL) and has enabled significant performance improvements over previously established baselines (be that heuristics or human experts) in a number of complex domains. These include image-based game-play [3], two-player adversarial games like Go [4] and robotics [5]. In conjunction with the increasing availability and affordability of compute, this forms the necessary foundations for DRL to revolutionise industrial control paving the way towards greater productivity. We are beginning to see examples of DRL agents approach the peripheries of commercial viability within applications like stratospheric balloon navigation for telecommunications [6] and chip design [7], to name a few. Although promising, the field is still in its infancy and there is still much work to do. Many challenges remain including generalisation, explainability and improving sample efficiency.

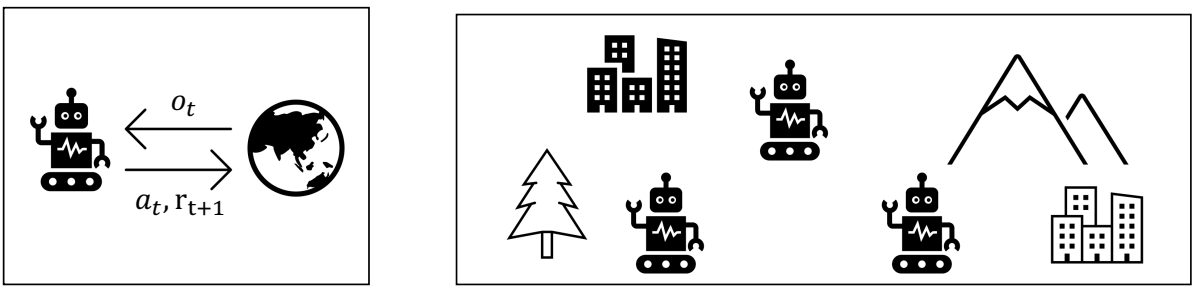
We explore an alternative challenge, namely the extension of DRL methodologies beyond the canonical single-agent (and two-player adversarial settings) towards the more generally applicable N-player setting. In many real-world applications DRL agents will not operate alone, they will find themselves within larger Multi-Agent Systems (MAS) comprised of numerous agents who are also empowered to make decisions and may be cooperative, self-interested, or competitive. In these more complex settings, agents may exert influence on one another which could impact each other's efficacy. Deep Multi-Agent Reinforcement Learning (DMARL) extends DRL to MAS and intends to develop methods which consider the implications of interactions

[1] Optimality can be guaranteed in some cases, however, this is not typical in SOTA RL methods.

on agents' learning process. They range in complexity from Independent Learners (IL) [8], where other agents are viewed as simply being part of the environment, to methods like Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [9] and Counterfactual Multi-Agent Policy Gradient (COMA) [10], which propose modifications to address issues like non-stationarity and credit assignment. The introduction of multiple agents also introduces new opportunities, for example, communication can be introduced to promote cooperative behaviours [11] and experience can be shared among agents [8].

In many cases, the decisions to instantiate a problem as an MAS may be conscious and intended to reflect the realities and characteristics of the underlying system. Examples can be found in many industrial sectors, including autonomous vehicles, economics, telecommunications and logistics. Through making this choice improved scalability and tractability [12] can be provided over monolithic representations. In many cases, these problems involve the optimisation of a system or organisational-level objective. Approaching the control problem with a singular centralised DRL is only feasible when the number of agents is small, as the asset number grows the number of actions tends to scale exponentially. For even modest numbers of agents, this quickly becomes intractable with standard DRL techniques. Alternatively, we can apply DMARL approaches to learn coordinated policies whilst providing the improved scalability we desire. Here, in this thesis, we explore the challenges and opportunities that arise from the application of cooperative DMARL to problems of this type.

1.1 Illustrative Example



(a) Agent (b) Environment

Figure 1.1: Figure 1.1a shows an agent who can observe, interact and receive a reward from an environment. On the right Figure 1.1b an environment which contains 3 agents is illustrated.

Our major motivator for moving from single-agent DRL to DMARL, is driven by the observation that many real-world applications are best represented as MAS. In our case, we consider the control of distributed assets through the use of DMARL. Let us consider the abstract example depicted in Figure 1.1. In Figure 1.1a, we show a systematic representation of an asset which is able to observe the environment through observation o_t, is able to interact with the world through actions a_t and receives a reward r_t. Figure 1.1b shows an environment with 3 agents. In

the context of telecommunications, an asset could be a Base Station (BS), router or other network infrastructure. In each case, it would make an observation of sensor values and be able to enact configuration changes through actions. The reward function may be a function of the network Quality of Service (QoS) or another performance measure and may even be shared among agents.

There is a spectrum of plausible solutions which are shown in Figure 1.2. Within the diagrams, the brain represents a decision-making functionality (or policy in the language of RL) and the lines represent their association with assets. In the case of the single-agent approach, we have greyed out the adjacent assets to indicate that they aren't represented in the training procedure. We now consider the implication of training these variations to maximise reception of r_t.

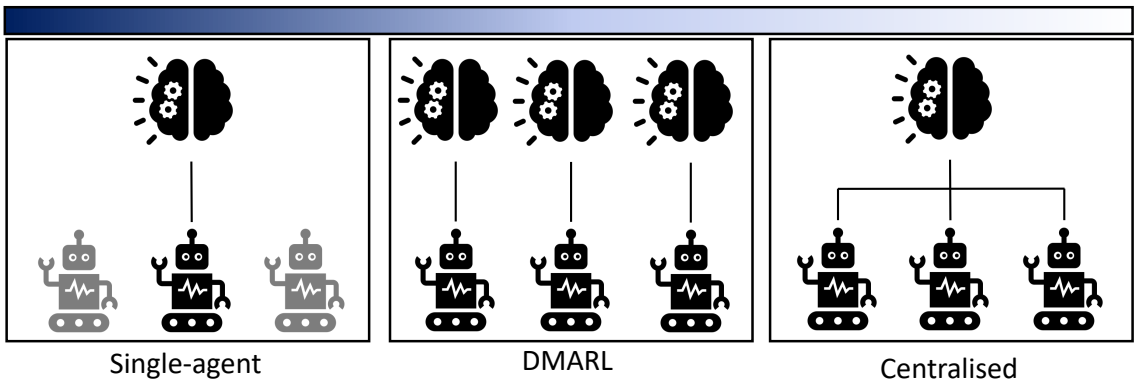


Figure 1.2: A spectrum of solutions for a MAS. On the left, we show what we refer to as a single-agent method where the presence of other agents is not modelled in training. In the middle, a DMARL solution is shown where all agents are provided with their own decision-making functionality and are trained collectively. On the right, all decision-making and training are conducted in a centralised manner.

Through the single-agent method, we can learn a policy that can control the required asset but it may have a range of limitations. In applications where agents co-exist in a shared environment (like the one shown), it will not be aware of the implications of decisions taken by other agents. This may result in the derived policy being sub-optimal. For example, it will be unable to learn cooperative behaviours or understand the impact of adversarial actions. We acknowledge that this may seem naive, but it is common in the applied literature [13, 14].

The centralised approach will enable us to learn these coordinated behaviours. There is however typically an exponential relationship between the number of actions and the number of agents [12]. For many industrial sectors that include distributed assets, this relationship can quickly become prohibitive from a computational perspective. It also requires the communication of observational data to a centralised location, which may be impractical due to communication constraints (i.e throughput, latency, privacy).

DMARL assigns each asset with a policy. It is inherently scalable and can address the aforementioned problems. However, the decomposition of the problem introduces new issues. For example, learning coordinated behaviours is harder than in the centralised settings, and issues

which we shall introduce in Chapter 2 like non-stationarity, credit assignment and partial observability can introduce challenges. Although problematic, these issues are not insurmountable and a range of algorithmic innovations allow for their mitigation [9, 10, 15]. This fundamentally is the focus of the contributions within this thesis.

1.2 Research Objectives

Through this research, we hope to understand the advantages that DMARL can provide within domains usually undertaken through heuristics and/or human experts. We hope to distil both industry-specific and general insights and recommendations which may be helpful for future practitioners and researchers alike. To achieve this, we take an empirical approach and detail our efforts in developing DMARL methodologies for two particularly compelling industrial domains. The domains we identify are telecommunications and logistics, which we choose for several reasons. Firstly, the domains are societally important, as both provide essential infrastructure to move commodities to consumers in order to satisfy various economic, educational, and cultural functions. Secondly, their infrastructure is typically characterised by large numbers of interconnected and interdependent assets making them good candidates for the application of DMARL. Thirdly, and finally, we believe our connections with relevant domain experts place us in a unique position to derive valuable insights.

Our investigations will focus on the development and evaluation of novel DMARL algorithms which address challenges within the candidate domains. In the list below we provide technical objectives and a description of what they shall entail.

1. **Problem Identification** In conjunction with subject matter experts in the aforementioned industries, we will identify suitable problems which require interaction and coordination between distributed assets. In these types of domains, we expect DMARL to offer the potential to improve decision-making and coordination, thereby improving performance. Furthermore, this activity shall focus on understanding the innate qualities of the problem and the heuristics that are typically employed.

2. **Benefits and Limitations of DMARL** Central to this research is the establishment of the challenges and limitations which MARL may provide over conventional methodologies (e.g. heuristics) within distributed control problems. Our experimental method shall primarily comprise of quantitative comparison utilising both DMARL and domain-specific metrics. However, where necessary we will evaluate qualitative aspects which could arise as a result of the training methodologies which we will employ for our DMARL algorithms. Through these comparisons, we can derive insights into the potential of DMARL within industrial applications and identify obstacles preventing their widespread adoption.

3. **Adapting and Design of DMARL approaches** Many existing innovations within DMARL have been demonstrated and applied within game-based domains [9, 10]. Our objective is to apply these advances to our identified industrial domains and to exploit their innate characteristics which may allow for the derivation of improved policies. We anticipate opportunity to utilise methodologies like CTDE [9], parameter sharing [8] and communication [11] to improve the coordination and decision-making of our agents.

4. **Inducing Cooperation** Our focus is on cooperative domains where distributed decision-makers must work together. Inherently, there is a fundamental requirement for cooperation and our research intends to focus on methods which support its emergence. For example, inter-agent communication [11] can enable cooperative behaviours and we shall explore its application.

1.3 Contributions

Collectively, this thesis provides contributions that further research in DMARL. We demonstrate its application in two notable industrial domains and comment on the path towards real-world deployed applications. Motivated by our experience of communication in applied DMARL, our final contribution focuses on the emergent properties of communication as a means to promote cooperative behaviours. Our contributions are broken down into three separate chapters, and they are described in the following subsections.

1.3.1 Telecommunications

In Chapter 3, we introduce our exploration of the application of DMARL within telecommunications. In order to motivate the necessity for DMARL, we begin by considering the suitability and limitations of DRL for a resource assignment task. As we will explain in Section 3.1, Radio Access Network (RAN) disaggregation introduces the necessity for resource assignment within future networks. This problem is modelled as an Bin Packing Problem (BPP) and then formulated as a MDP An AlphaGo Zero [4] inspired DRL approach is applied to this challenging task. Empirically, it is demonstrated to outperform our strongest baseline enabling a 5.7% improvement in resource utilisation. This concerns work detailed in [16], where the author's contributions are within the problem formulation and methodology. Through further analysis of this methodology, we suggest that extension to large-scale networks may unearth limitations which inherently distributed solutions like DMARL may not be susceptible to. We then explore the application of DMARL within an alternative setting. The setting we consider is network maintenance planning, where a finite amount of maintenance resource must be assigned among a set of distributed network elements. This problem is modelled as an POSG and a novel DMARL approach is developed which utilises a GNN-empowered centralised critic to learn coordinated policies. Our best performing algorithm, convMAAC, improves network availability by 3.49% and 6.13% in two different network

topologies over standard baselines. These concerns work in [17], where the author contributed to all parts of the research project. From this work, we establish problems that prohibit the deployment of DMARL solutions, including challenges moving from simulated domains (which are necessary for training) to real deployments.

1.3.2 Logistics

Within Chapter 4, we consider the *order-picking problem* where humans and robots work together to retrieve required items from commercial warehouses. This section details collective efforts developing a DMARL approach to handling their coordination in order to maximise order throughput. To this end, a feudal DMARL algorithm called HSNAC and a bespoke simulator were developed which improved order throughput over an established industry heuristic by 23.2%. The author's contributions concerned further optimisation of the simulator and algorithm to enable faster execution times, optimisations of the performance of the existing HSNAC algorithm through the introduction of techniques like GAE [18], development of policy visualisation tools and the identification of limitations with the approach. This section details work within [19].

1.3.3 Emergent Communication for Cooperation

As demonstrated within Chapter 3 and 4, there are a multitude of ways through which cooperation can be achieved. In our telecommunications work on distributed network maintenance planning, we expected to be able to make use of communication to enable cooperative behaviours. However, empirically we were not able to demonstrate any advantage in doing so. This led to a fundamental interest in the properties of communication, and through its study it was hoped that progress could be made in developing practically useful communicative DMARL solutions. Here, we focus on identifying and understanding the limitations in existing solutions concerning their vulnerability to partner population dynamics and the impact of vocabulary size on language acquisition. These problems are addressed through novel algorithmic innovations.

1.4 Publications

- **Thomas, J. D.**, Hernández, M. P., Parlikad, A. K., & Piechocki, R. (2021, October). Network Maintenance Planning Via Multi-Agent Reinforcement Learning. In 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (pp. 2289-2295). IEEE.

- Wang, X., **Thomas, J. D.**, Piechocki, R. J., Kapoor, S., Santos-Rodríguez, R., & Parekh, A. (2022). Self-play learning strategies for resource assignment in Open-RAN networks. Computer Networks, 206.

- **Thomas, J. D.**, Santos-Rodríguez, R., Piechocki, R. J. (2022, December). Understanding Redundancy in Multi-Agent Communication. In NeurIPS Language and Reinforcement Learning Workshop.
- Krnjaic, A., **Thomas, J. D.**, Papoudakis, G., Schäfer, L., Börsting, P., Albrecht, S. V., (2022, December). Scalable Multi-Agent Reinforcement Learning for Warehouse Logistics with Robotic and Human Co-Workers. In Arxiv.
- **Thomas, J. D.**, Santos-Rodríguez, R., Piechocki, R., & Anca, M. (2023). Multi-lingual agents through multi-headed neural networks. In Northern Lights Deep Learning Conference 2023.

CHAPTER 2

BACKGROUND AND RELATED WORK

As previously argued, many industrial settings depend on distributed and interdependent assets. The primary goal of this thesis is to explore the utilisation of DMARL as a solution to these problems. Here, we provide an overview of relevant literature and examples of its application. This chapter is structured as follows:

(2.1) Reinforcement Learning We observe that many of the ideas from MARL, build upon those from RL. Hence, we begin with an introduction to RL, introducing fundamental concepts like the Markov Decision Process (MDP), policies, value functions, and reviewing some of the general challenges that exist within this setting like exploration versus exploitation and credit assignment. From here, we review model-free Deep Reinforcement Learning (DRL) algorithms like Deep Q-Networks (DQN) [3] and REward Increment = Nonnegative Factor × Offset Reinforcement × Characteristic Eligibility (REINFORCE) [20].

(2.2) Multi-Agent Reinforcement Learning With foundational ideas from RL and DRL established, we then transition into DMARL. We will introduce the SG which generalises the MDP to the multi-agent setting and analyse the complexities this introduces. As before, we will then review model-free DMARL algorithms like Value Decomposition Networks (VDN) [21] and Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [9] which build upon DQN and REINFORCE, respectively.

(2.3) A guide to applying DRL and DMARL Application of DRL and DMARL requires navigation of a multitude of decisions. Here, we provide a distillation of the literature into an actionable sequential process. We illuminate many of the challenges that exist in a typical development cycle.

(2.4) Examples of Application Numerous examples have been made at applying DRL and DMARL to real industrial problems. In this section, we review them.

2.1 Reinforcement Learning

RL considers sequential decision-making problems where an agent is presented with an unknown environment and is tasked with learning how to behave such that the reception of a reward signal is maximised [1]. The aim of the following section is to introduce the typical formulation of this problem and the algorithms which are typically employed in an attempt to solve them. This section takes inspiration from Sutton and Barto [1].

2.1.1 Markov Decision Processes

The MDP formulates the sequential decision-making problem [1]. It can be described by the 5-tuple, $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$. $\mathcal{S}$ is the set of environment states. $\mathcal{A}$ is the set of actions. $\mathcal{R}$ is the reward function which is defined as follows $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. $\mathcal{P}$ is the transition function which is defined as $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$. Where $\Delta(\mathcal{S})$ denotes a set of probability distributions over $\mathcal{S}$ and represents a stochastic state transition function. Finally, $\gamma \in [0, 1\}$ is the discount factor which impacts the time horizon over which the reward function should be optimised.

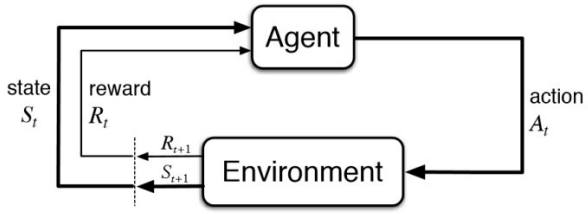


Figure 2.1: Agent-Environment interaction in an MDP [1]

This can be represented by Figure 2.1 which explicitly shows the interface between an agent and an environment. For completeness, we talk through this interactive discrete-time process. At $t = 0$, the agent begins in $s_t \in \mathcal{S}$, it then selects an action a_t computed from a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$. It then experiences a state transition defined by $\mathcal{P}$ to s_{t+1} and receive a reward, r_{t+1} given by $\mathcal{R}$. The agent is now in $s_{t=1}$ and again must select an action, after which it will experience a state transition and receive a reward. This process typically continues until the agent reaches a terminal state, $\mathcal{S}^+ \in \mathcal{S}$ which occurs at $t = T$ in the episodic case or alternatively may be continuous.

2.1.2 Policies and Value Functions

Previously we have informally stated that the objective of RL is to learn behaviours that maximise the reception of reward signal. These behaviours are more formally referred to as policies

$\pi : \mathcal{S} \rightarrow \mathcal{A}$. The reward signal which we are trying to maximise is the *return*, G, and is defined in Equation (2.1). Our objective can be concisely defined by Equation (2.2).

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k \tag{2.1}$$

$$\max_{\pi} \mathbb{E}[G_t] \tag{2.2}$$

Up until now, we have mostly overlooked the discount factor, γ. It comes into play in Equation (2.1) and impacts an agent's perception of the trade-off between short-term and long-term cumulative rewards. Setting $\gamma = 0$, results in an agent who is myopic whereas $\gamma \rightarrow 1$ will optimise long-term reward. It is common for this value to be set to between 0.95 and 0.99.

Two related and fundamental concepts are the *state-value function* and the *action-value function*. The *state-value function* indicates the expected return which can be achieved from the respective state under the current policy. The *action-value function* indicates the expected return of a particular action in a particular state as shown in Equation 2.4. Both equations are based on the Bellman equation [22] which is recursive and serves to compute the expected value of a state or state-action pair based upon subsequent state or state-action pairs.

$$\begin{aligned} v_\pi(s) &= \mathbb{E}[G_t | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma \mathbb{E}[G_{t+1} | S_{t+1} = s']] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \end{aligned} \tag{2.3}$$

$$\begin{aligned} q_\pi(s, a) &= \mathbb{E}_\pi[G_t | S_t = s, A_t = a] \\ &= \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \end{aligned} \tag{2.4}$$

If the transition probabilities and the reward function are available, $\mathcal{P}$ and $\mathcal{R}$, the optimal policy, π^* can be derived through Dynamic Programming (DP) by methods like Policy Iteration (PI) [1]. Unfortunately, these are not always known or in some cases, the dimensionality of a problem may make DP problems intractable.

2.1.3 The role of Reinforcement Learning

RL allows for consideration of problems where $\mathcal{P}$ and/or $\mathcal{R}$ are not explicitly known, but samples can be obtained through interaction with the environment. There are a variety of methods for

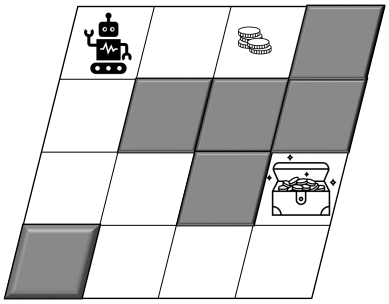


Figure 2.2: Exploration vs. Exploitation: In the grid-world shown, the episode ends upon the reception of a reward. It is highly likely that an agent will find the smaller reward (depicted by the coins) first. If we exploit this information, we will not be able to find the larger reward.

approaching this, including model-free methods like Q-Learning [23] and Policy gradient [20] which attempt to approximate the optimal action-value function and policy respectively, and model-based methods like PILCO [24] which attempt to learn $\mathcal{P}$ and $\mathcal{R}$[1]. Learning through RL methods can raise a number of challenges including *exploration vs exploitation* and *credit assignment*. There may also be specific environment qualities which introduce other problems like *non-stationarity* and *partial observability*. We discuss these below.

Exploration vs Exploitation Given an unknown environment, our objective is to learn π^*. To do so, we must explore and collect samples which enable us to build an understanding of both $\mathcal{P}$ and $\mathcal{R}$. Over time we will build understanding and may find a region of the MDP with a relatively high reward. Do we exploit this knowledge and continue to visit this region, or do we continue to explore and maybe find a higher reward? Fundamentally, this is the trade-off and is illustrated in Figure 2.2. There are many ways to handle this trade-off including including epsilon-greedy, count-based exploration incentives and intrinsic rewards for exploration [25].

Credit Assignment Many environments require agents to contend with delayed reward. In these settings, actions at previous time-steps may be critical in obtaining reward however there is a challenge identifying their contribution which is obsfucated by the passage of time. This is known as the temporal credit assignment problem [26].

Partial Observability In many cases, the true environment state, $s \in \mathcal{S}$, may not be fully observable. There are a multitude of reasons why this might be the case, including, noisy sensor values or an agent observing a limited field of view. In this more complex setting, the environment state cannot be known with certainty and the Markovian underpinning of the MDP is violated [1]. This typically manifests as "perceptual aliasing", where observed sensor values may be indistinguishable whilst actually requiring different actions from the agent [27]. In this case, the MDP is no longer suitable and the problem is typically represented as a Partially Observable

[1] In the case of PILCO, this is explicitly known but in general $\mathcal{R}$ must be learned.

Markov Decision Process (POMDP). This formulation captures uncertainty encompassed within an agents observation of the state. In this more complex domain, it is common use the notation $o \in \mathcal{O}$ to refer to agent observations.

Non-stationarity In non-stationary environments, the dynamics of the MDP can change over time. These changes may impact the transition dynamics, reward function or both and they may change gradually or suddenly. In either case, these variations fundamentally change the problem and impact the manner in which an agent should behave.

2.1.4 Model-Free Deep RL

In recent years, ML has been revolutionised through Deep Neural Network (DNN) which enables automatic feature extraction and universal function approximation [28]. Collectively, this is referred to as Deep Learning (DL) and has left very few areas of ML untouched. Some examples of the key breakthroughs within DL include CNN for image classification [29–31], RNN for sequential data [32, 33], and GNN for graph-based data [34–36]. Advances in DL have also been deeply impactful within RL, where the intersection is often referred to as Deep Reinforcement Learning (DRL). The combination of the CNN and RL was critical in achieving human-level performance in image-based sequential decision-making within Atari by Mnih et al. [3]. This advancement allowed for extension to large state spaces where applications of tabular methods like Q-Learning [23]. This can be overcome through the employment of function approximation techniques like DNN.

Within RL, there are *model-free* and *model-based* methods. Model-free methods learn a policy directly through interaction with the environment and can be further broken down into Deep Q-Networks (DQN) and policy optimisation methods. Model-based methods typically learn a model of the environment and apply planning or dynamic programming techniques to derive an optimal policy. In the literature, model-free approaches have tended to see more application and success within complex domain. The difference in performance between these two paradigms is often attributed to the implications of modelling inaccuracies within model-based approaches [24, 37]. Henceforth, we focus our review on model-free DRL methods.

2.1.4.1 Deep Q-Networks

Deep Q-Networks (DQN) extends Q-Learning through introducing DNN-based function approximation. In the interest of identifying the algorithmic innovations that were necessary to extend Q-Learning, we begin with introducing the original tabular algorithm in Algorithm 1. Q-Learning is a method which learns to approximates the optimal action-value function, q^*. It represents it as a table of dimension $|S| \times |A|$, where it updates the appropriate action-value after every environment interaction. As previously mentioned, the size of the table can become prohibitive as the environment scales. DQN [3] utilises a DNN to parameterise q_θ allowing for an improvement

in scalability. The algorithm for DQN is shown in Algorithm 2. The introduction of the DNN complicates the training process and additional apparatus is necessary to stabilise training. The two most common are (1) an experience replay buffer D and (2) a target network q_ϕ.

Algorithm 1: Q-Learning using ϵ-greedy exploration

Input: env, α, γ
Output: Q^*
$Q \in \mathbf{0}^{|S|\times|A|}$;
for *episodes* **do**
  done = False ;
  s = env.reset();
  while *not done* **do**
    $x \sim$U(0,1) **if** $x > \epsilon$ **then**
      $a \sim \text{argmax}_a(Q(s))$;
    else
      $a \sim A$;
    end
    $s', r,$ done = env.step(a_t)
    if *not done* **then**
      $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'}(Q(s',a')) - Q(s,a)]$;
    else
      $Q(s,a) \leftarrow Q(s,a) + \alpha[r - Q(s,a)]$;
    end
    $s \leftarrow s'$;
  end
end

The experience replay buffer, D, acts as a store for previous agent experience. By sampling from this rather than training directly on incoming data, it is possible to move closer towards independent and identically distributed data which facilitates the mitigation of temporal correlations which can be problematic in training. The target network, q_ϕ, acts to reduce instability which would be present in training as the target value would be constantly changing. Typically this is changed after a certain number of episodes or may be gradually changed.

Since the initial development of this algorithm, several modifications and additions have been proposed including Prioritised Experience Replay [38] which preferentially samples rare experiences from the buffer with higher probability, Double Q-Learning [39] which helps with the overestimation which Q-Learning is known to suffer from and many others. Where these and others were combined in Rainbow DQN [40] which achieved improved performance over the individual parts.

Algorithm 2: Deep Q-Networks (DQN) algorithm

Input: env, α, γ
Output: Q

Initialise Q_θ, Q_ϕ ;
Initialise $D = []$ ▷ Replay buffer of finite capacity ;
for *episodes* **do**
  done = False ;
  s = env.reset();
  while *not done* **do**
    $x \sim$U(0,1) **if** $x > \epsilon$ **then**
      $a \sim \text{argmax}_a(Q(s))$;
    else
      $a \sim A$;
    end
    $s', r,$ done = env.step(a_t)
    D.append($(s, a, r, s',$ done));
    B = D.sample() ▷ Sample random batch of transitions;
    loss = 0 ;
    for $i = 1$ **to** $|B|$ **do**
      $s_i, a_i, r_i, s'_i,$ done$_i = B_i$;
      if *not done*$_i$ **then**
        $y_i = [r + \gamma \max_{a'}(Q_\phi(s', a'))$;
      else
        $y_i = r$;
      end
      $loss \leftarrow \frac{1}{|B|}(y_i - Q_\theta(s'_i, a))^2$;
    end
    loss.backward() ;
    loss.step() ;
    $s \leftarrow s'$;
  end
end

2.1.4.2 Policy Optimisation Methods

Another highly successful class of algorithms are Policy Gradient (PG) methods. These are derivatives of REINFORCE [20] which is based on the PG theorem which is shown in Equation 2.5. Where $J(\theta)$ is a scalar performance measure (G_t for example) and $\mu(s)$ is the state distribution which would be experienced when following π_θ. From Equation 2.5, we can obtain the REINFORCE shown in Equation 2.6 through the steps depicted.

$$\begin{aligned}
\nabla J(\theta) &\propto \sum_s \mu(s) \sum_a \nabla_\theta \pi_\theta(a|s) q_\pi(s,a) \qquad (2.5)\\
&= \mathbb{E}_\pi \left[\sum_a q_\pi(S_t,a) \nabla_\theta \pi_\theta(a|S_t) \right] \\
&= \mathbb{E}_\pi \left[\sum_a \pi_\theta(a|S_t) q_\pi(S_t,a) \frac{\nabla_\theta \pi_\theta(a|S_t)}{\pi_\theta(a|S_t)} \right] \\
&= \mathbb{E}_\pi \left[q_\pi(S_t,A_t) \frac{\nabla_\theta \pi_\theta(A_t|S_t)}{\pi_\theta(A_t|S_t)} \right] \\
&= \mathbb{E}_\pi [G_t \nabla_\theta \log \pi_\theta(A_t|S_t)] \qquad (2.6)
\end{aligned}$$

This expectation can be estimated by Monte Carlo evaluations of the current policy and through doing this it can be improved. Monte Carlo methods typically suffer from high variance which can result in slow convergence [1]. Modifications which aim to reduce this variance include using a baseline, where G_t is replaced with $G_t - v_\pi(s)$. This requires learning v_π but typically improves performance overall. Actor-Critic methods further build on this and replace the G_t in the baseline version with $R_t + \gamma v_\pi(s')$ which now allows for temporal difference updates of π. Further modifications to this, include Trust Region Policy Optimisation (TRPO) [41] which limits the size of the weight update by constraining it by the KL-divergence, PPO [42] which approximates PPO update constraint, and Soft Actor-Critic [43] which utilises entropy regularisation and trains the policy in an off-policy manner.

2.2 Multi-Agent Reinforcement Learning

DMARL is a natural extension of DRL to MAS. Many of the issues that we previously introduced still exist, but new ones also manifest. As a necessary step before discussing these, we introduce the typical problem formulation within MARL. The MDP which is central to RL is no longer sufficient to describe MARL problems. A common mathematical framework for this domain is the Stochastic Game (SG) [44] which we introduce below. Other related settings include POSG [45], Markov games [46] and the DEC-POMDP. This section provides an overview of DMARL, for a more comprehensive review please refer to [47].

2.2.1 Stochastic Game

We can model the MARL setting as an N-player stochastic game [44]. This is defined by the tuple $G = \{\mathcal{I}, \mathcal{S}, \{A^i\}, \{O^i\}, \mathcal{P}, \Omega, \{R_i\}\}$. Where, $\mathcal{I}$ is a finite set of agents indexed by $\{i, \ldots, N\}$, $\mathcal{S}$ is the state space, A^i represents the finite set of actions that are available to agent i. We refer to the joint action as $\mathcal{A} = A^1 \times \cdots \times A^n$. At each time-step agent i, receives a partial observation $o^i \in O^i$ which is defined by the observation function $\Omega : \mathcal{S} \times \mathcal{A} \to O$. Agents use this information to select an action $a^i \in A^i$ according to $\pi_{i,a}$. They then experience a state transition $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ and receive a reward $R^i : S, \mathcal{A} \to \mathcal{R}$. Agents are tasked with finding action policies $\pi_{i,a} : O_i \to A_i$ which maximise the cumulative discounted reward they receive. We refer to the set of policies as $\Pi = \{\pi_1, \ldots \pi_N\}$. It is common to discuss agents converging to a Nash Equilibrium, where this indicates that the behaviour of each policy is the best response to all other policies [47].

2.2.1.1 Communication in Stochastic Games

In some cases, as in Chapter 5, we may provide agents with the opportunity to communicate. To reflect this change in agent capability the stochastic game is typically modified such that $G = \{\mathcal{I}, \mathcal{S}, \{A^i\}, \{M^i\}, \{O^i\}, \mathcal{P}, \Omega, \{R_i\}\}$. We refer to this in the wider text as a stochastic game with communication. Where M^i refers to the message set and an agent is able to select a message $m^i \in M^i$. We refer to the joint message as $\mathcal{M} = M^1 \times \cdots \times M^n$. We also introduce $\mathcal{M}^{-i} = \mathcal{M} \backslash M^i$ to refer to the set of joints messages, without i. Functionally, this is utilised to refer to the messages that would be received by agent i. The message is selected according to $\pi_{i,m} : O^i, M^{-i} \to M^i$. Unlike an action, a message has no direct impact on the physical environment and as such the transition function and reward function maintain the same form. The impact is indirect, and it provides an opportunity to influence other agents' selection of actions. We can modify the agent's action policies such that $\pi_{i,a} : O^i, \mathcal{M}^{-i} \to A_i$.

2.2.2 How do multiple agents change the problem?

Instead of just learning a singular policy, we are now trying to learn all policies in the set Π. This additional complexity manifests as the problems like *non-stationarity*, *multi-agent credit assignment*, *partial observability* and questions pertaining to *cooperation* and *competivity*.

Non-stationarity The policies of the agents in Π are not static throughout training - as the agents acquire experience they'll update and likely change their policies. If we refer back to the definition of a SG, we observe that both $\mathcal{P}$ and $\mathcal{R}$ are a function of all agents' actions and therefore their policies. Herein lies the problem, from the perspective of a single agent both $\mathcal{P}$ and $\mathcal{R}$ will appear to change [48]. This makes estimation of the expected environment return, G_t, difficult. As we previously noted, this is present in DRL too. However, in this case, it is a direct cause of the learning paradigm rather than environmental factors.

Credit Assignment In the standard RL setting determining the criticality of individual actions in return acquisition is complex. The MARL setting scales this problem as we now typically have considerably more actions. The question is typically posed as who and what resulted in the reward we obtained. Within DMARL there are methods like COMA [10] which propose the modification of the advantage calculation within AC methodologies to allow for a counterfactual estimate of an agent's contribution to the observed return. Empirically, this is not typically found to be effective though [49]. Other related approaches include Shapley values as in [50].

Partial Observability A primary motivator for DMARL is the ability to perform fully decentralised control. As such, we are often considering applications where agents only have the capacity to observe their immediate surroundings. This results in situations where agents have to undertake decisions without being privy to the true state of the environment, so may make sub-optimal decisions. Rather than a SG, the problem is then typically represented as a POSG [45] or DEC-POMDP.

Cooperative vs Competitive Introducing multiple agents raises questions about their goals and how they align. It also raises possibilities regarding the training methodology for the agent. Many of the high-profile successes of DMARL to date (e.g Backgammon, AlphaGo, Chess) have mostly been within the competitive two-player zero-sum setting. These settings enable the utilisation of training paradigms like self-play [51], where a copy of the agent is used as an opponent. Intuitively, this continually presents the agent with an adversary of equivalent strength and theoretically converges to a Nash equilibrium allowing for policy generalisation to any adversary. Empirically the derived policies can be brittle and the utilisation of population-based self-play methods is typical [52]. Cooperative games are typically not amenable to the naive application of self-play. This is as the objective mandates coordination between players which typically gives rise to idiosyncratic conventions which do not extend to arbitrary partners. Methods like Other-Play [53] have been proposed within the Zero-Shot Coordination (ZSC) to extend self-play to the cooperative domain. It is, however, much more typical to train directly for reward maximisation in cooperative settings without consideration for these types of methodologies.

2.2.3 Model-free Deep Multi-Agent Reinforcement Learning

As we stated in Chapter 1, the simplest approach is that of IL and involves direct application of the same single agent approaches introduced in Section 2.1.4. The most common of which are Independent Q-Learning (IQL) [8] and Independent Actor-Critic (IAC) [10]. However, typically these do not perform well. The majority of advances within DMARL focus on extending single-agent methodologies to handle the aforementioned complexities and additional opportunities which multiple agents introduce. Along the same lines as Section 2.1.4, we break these advances down dependent on whether they are derivatives of DQN or PG approaches. However, we do

acknowledge that there are concepts which do not fit neatly into either bracket. For example, Centralised Training for Decentralised Execution (CTDE) [54] and parameter sharing [8]. As such, we introduce it before delving into DQN and policy optimisation derivatives.

Centralised Training Decentralised Execution Permits asymmetry between the training and execution phases with the intention of improving training performance. The central idea is that restrictions which exist at execution time (when the algorithm is deployed) do not have to exist within the training stage. We may introduce apparatus that smooths training as long as this is not required at execution time. It is common to relax restrictions on partial observability and develop centralised models which are decomposable on a per-agent basis enabling decentralisation. Examples include MADDPG [9] and QMIX [55].

Parameter Sharing This refers to the utilisation of the same function approximation parameters by all agents. This was first proposed within [8] as a method to enable experience sharing and improved sample efficiency. Within [49], it is demonstrated to be advantageous over algorithms and over the majority of DMARL environments. This is a common practice and one where agent observations are often further extended through inclusion of an indicator in the observation such that the agents can learn heterogeneous behaviours [56].

2.2.3.1 Deep Q-Network derivatives

In addition to naive IQL [8], a range of adaptations have been proposed. In [57] the impact of non-stationarity on the relevance of data in the experience replay buffer is tackled. They propose two solutions based on importance sampling and an approach similar to Hyper-Q [58] where an additional conditional argument is introduced to enable estimation of environment behaviour at the particular sample point. This general issue of instability of experience replay in DMARL has also been noted within [59]. Other approaches include Mean-Field Reinforcement Learning (MFRL), which provides a scalable approach based on the mean-field approximation. They utilise the average action of neighbouring agents as a means to estimate population-level behaviours and condition each agent's policy on this in addition to its observation[2].

$$Q_{joint} = \sum_{i}^{N} Q_i \tag{2.7}$$

Value decomposition methods are perhaps the most common in this family. Where the most popular are, VDN [21], QMIX [55] and QTRAN [60]. These algorithms consider environments with shared cooperative reward $R_1 = \cdots = R_N$ and decomposition of the joint action-value function Q_{joint} into an individual action-value function for each agent, Q_i. Where the major distinction

[2]We place this under "Deep Q-Network Derivates" as this was their best-performing variant. However, it is implementable within policy optimisation approaches too.

between these methodologies is their apparatus for achieving this decomposition. [21] motivate this line of work as a means to overcome the limitations of centralised and fully-decentralised implementations of DQN. They observe that centralised methodologies can fall victim to the *"lazy agent"* problem, whereby, the success of one agent may impede the derivation of useful behaviours by a second agent. In the IL domain they cite the well-known issues of non-stationarity and partial observability as problematic for learning. Conceptually, these value decomposition methods exist between these extremes and leverage the CTDE paradigm. VDN proposes a linear decomposition of the joint action-value function into individual action-value functions, where this is trained end-to-end. QMIX [55] identifies limitations of VDN based on its inability to represent certain classes of centralised action-value functions. Rather than a linear combination they use a monotonic function, and propose a non-linear combination achieved through mixing networks. QTRAN [60] further identifies limitations of VDN and QMIX in their capacity to factorise tasks which are factorisable. The central idea is the transformation of Q_{joint} into a form which is more easily factorisable. There are various other extensions upon these methodologies including methodologies like Multi-Agent Variational Exploration [61] and Multi-Agent Exploration [62] which specifically consider the challenge of exploration in DMARL.

2.2.4 Policy Optimisation derivatives

As is the case with DQN, many of the PG methods have been extended to the DMARL setting. These include the likes of Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [9], Multi-Agent Proximal Policy Optimisation (MAPPO) [63] and Shared Experience Actor-Critic (SEAC) [64] to name but a few.

Here, we introduce those that can be classified as CTDE methods which are typically variants of AC algorithms where the critic is centralised. MADDPG [9] extends the off-policy actor-critic algorithm DDPG [65] to the multi-agent domain. It utilises a centralised action-value critic which is conditioned on all agents' observations and actions where they suggest that this improves the consistency of gradient signals as it is not so susceptible to partial observability and non-stationarity. Iqbal and Sha [66] extend this approach through the utilisation of an attention mechanism to aggregate agent observations which results in a more scalable centralised critic than the fully-connected network utilised in MADDPG. COMA [10] propose a centralised action-value critic as a means to tackle multi-agent credit assignment. To do so, they propose the utilisation of a counterfactual baseline to estimate the average return that would be expected from each agent's individual action-value function. SEAC [64] proposes an alternative approach to MADDPG which does not require a shared critic. They instead allow experience sharing through importance sampling which allows agents to learn from other experiences. MAPPO [63] extend the highly-popular PPO to multi-agent environments and demonstrate its effectiveness in a range of cooperative tasks and configurations. Interestingly, they are able to demonstrate that independent PPO is an effective baseline in a wide range of DMARL environments.

2.2.5 Communication in MARL

In Chapter 5, we study communication as a means to promote cooperation. In Section 2.2.1.1, we discussed and provided the modifications necessary to introduce communication into a stochastic game. Here, we review advances in DMARL. For a comprehensive overview of communication in DMARL, we refer interested readers to [67, 68].

By facilitating communication between multiple agents, we may promote cooperative behaviours and enable the acquisition of higher episode returns. To this end, many methods have been proposed including DIAL and RIAL [11], CommNet [69], DGN [70] and TARMAC [71]. These methods focus on cheap talk, where there is no cost for utilising communication, and allow for communication to be established as an emergent phenomenon. We consider there to be two major classes of algorithms within this sub-topic of DMARL, those that allow gradients to flow in learning and those that do not and just learn through environment reward. Allowing gradient propagation across communications links significantly improves sample efficiency as it provides a much richer learning signal than just utilising environment reward. Gradient-facilitated communication learning is by far the most common and is utilised in DIAL [11], CommNet [69], DGN [70] and TARMAC [71]. These typically utilise continuous communications channels as they are more permissible to backpropagation, however, options for discrete channels exist as seen in DIAL [11] or through the Gumbel-Softmax relaxation used in [72]. Purely environment reward-based communicative algorithms are utilised in RIAL [11] and in work in [73, 74]. Although less efficient they easily permit extension to learning in the case of non-differentiable partners which may be the case in certain human-centric applications or when a model of a communicative partner is unavailable.

A common issue that is worth noting in DMARL is deducing the value of communication. In modestly complex temporally extended environments analysis of protocols can be prohibitively difficult. Within [75], quantitive measures are suggested that attempt to disentangle the benefit of communications. Work by Jaques et al. [74] [3] and Eccles et al. [73] introduces modifications which attempt to promote communication in the environment reward-based domain through maximisation of related metrics.

A related area is Emergent Communication (EC), which studies agents' capacity to automatically derive language [67]. Much of the work focuses upon the Referential Game (sometimes referred to as a Lewis Game [76]) which mandates language formation in order to solve a cooperative task. In many ways, this setting is a microcosm of the larger class of games utilised in MARL which typically allows for the study of language formation in isolation. The typical setting involves two agents who are referred to as a speaker and listener and communicate utilising discrete symbols [72, 77].

[3] We acknowledge this work happened simultaneously to [75]

Our contribution In Chapter 5, we undertake investigations into the challenges of intra-agent communication for cooperation. We do so within a referential game (or lewis signalling game [76]) which allows for the study of the complexity of developing communications protocols for cooperative tasks. Our work builds upon that by [11, 73, 74], exploring the complexities of developing communications protocols in non-differentiable channels where agents are trained purely through environment reward. We extend work in this area in two ways. Firstly, we investigate the impact of population dynamics and find that periodic partner changes can result in catastrophic forgetting of existing protocols. A related area is ZSC [53] which considers the challenge of learning policies to interact with previously unseen partners in cooperative settings. [78] explicitly consider communication in this domain. Our work differs from this as we allow multiple unique languages to arise and propose a method which allows agents to learn and maintain multiple languages. Secondly, we explore the impact of message dimensionality within discrete communication and experiment with methodologies to reduce excessive redundancy which is missing from the literature.

2.3 A rough guide to applying DRL and DMARL

Many papers within the literature apply DRL and DMARL approaches to real-world industrial problems. Before delving into these, it is worth considering the typical academic and engineering challenges which are common. Within Figure 2.3, we have represented a simplified workflow which we will now walk through. We note that these blocks typically do not happen sequentially, and the design and development process is typically more agile and responsive. In this section, we try to maintain generality and will discuss specific challenges in the Section 2.4.

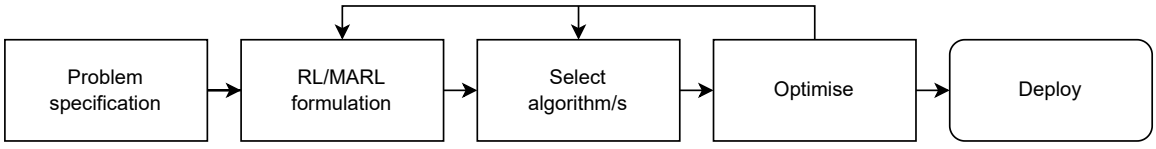


Figure 2.3: Generalised RL and MARL workflow

2.3.1 Problem Specification

Many complex industrial domains are dependent on human expertise and/or heuristic algorithms for decision-making tasks. However, these can be sub-optimal. In these domains, it may be possible to leverage methodologies from DRL or DMARL to improve system performance. If not already established, the first objective should be to identify the specifics of the system, the relevant metrics (or KPI) for optimisation, and the various constraints that may exist on the system which may be operational, regulatory, performance and/or systematic in nature. Relevant details to establish may include the availability of computational resources, interdependencies that may

exist with other systems and information regarding system operation from historical data or domain experts. With this established, a major consideration is the availability of simulators. The poor sample efficiency of DRL and DMARL techniques [79] and safety concerns mandate high-performance simulators for training. This process of simulator design and optimisation is discussed in Chapter 4. We note that the aforementioned heuristics may be useful for testing and troubleshooting of developed simulators and strongly recommend this step. Skipping this step and directly applying DRL to DMARL methodologies often leads to a significant increase in debugging complexity.

2.3.2 RL/MARL formulation

In many common DRL and DMARL research environments like OpenAI Gym [80], the observation (or state), action and reward are predefined. However, in application-focused domains, this is a design decision which is critical in achieving a well-functioning solution. Through careful consideration of the problem specification, we may be able to derive insight into the correct definitions. Existing heuristics may give insight into system sensor values which are pertinent. The form and function of controllable parameters may also naturally define the action space. As we have introduced there are a wide range of possible choices in the formation of this as an RL problem. From single-agent formulations like the MDP, POMDP and even the Constrained Markov Decision Process (CMDP) [81] to multi-agent formulations like the SG, POSG, DEC-POMDP. Identification of the correct formulation will be dependent on the specifics of the underlying problem. For example, some systems may be more amenable or require representation as multi-agent variants, whereas systems, where observations are obfuscated by noise, maybe more appropriately modelled as a partially observable variant. In many cases, this labelling will help later in the identification of suitable classes of algorithms. A particularly challenging aspect in all formulations is the challenge of reward definition which requires the translation of a system-level objective into a scalar. Techniques like reward shaping [82, 83] can be used to improve learning performance by giving additional rewards to guide an agent/s towards optimal behaviours but can be susceptible to reward hacking [84]. In multi-agent variants, like the SG and POSG, there is an additional decision between global or local rewards. We note that local rewards are typically more scalable [59, 85] but can result in competitive behaviours.

2.3.3 Select algorithm/s

As we have introduced in Section 2.1 and 4.5, there are various algorithmic approaches to DRL and DMARL. Our previously given representation will be particularly important in identifying appropriate choices. For example, multi-agent variants will mandate the selection of DMARL algorithm, and a continuous action space typically restricts the selection to a policy gradient approach. The form and property of the observation space also may make a compelling case for the utilisation of particular DNN architectures. An LSTM may be introduced into the network to

handle issues pertaining to partial observability [86]. Alternatively, graph-based observations may be best handled by GNN as used in [70, 87], however, we note that attention should be paid to the correctness of biases in graphical structures [88]. System constraints may also be addressed here Degrave et al. [89] consider an application which requires very low inference times and utilise an AC approach which exploits ideas similar to those from CTDE. In their case, their actor takes the form of a simple FCN, whereas, their critic is a much larger network comprised of an LSTM. This allows for fast inference times as the critic is not required at deployment. Other choices that are afforded to a designer include options like, action-masking to remove dangerous, non-valid or non-sensical actions [90] and utilisation of methods like GAE [18]. The objective of this step is to identify solutions which may be useful to our agent/s for optimisation in the next step. However, we acknowledge that many other decisions are plausible. In many cases, this will be highly dependent on the specifics of the task. It may, for example, be worth reviewing literature from areas like ZSC if the agent is expected to have to cooperate with unknown partners, ideas from multi-task RL if the agent is expected to undertake several tasks, or even HRL if temporal or spatial decompositions of the problem are possible.

2.3.4 Optimisation

In addition to the multitude of manners in which a problem may be represented and approached, DRL and DMARL algorithms typically have numerous hyperparameters. As reported by Henderson et al. [91], algorithm hyperparameters can be highly impactful on performance. Optimisation of these parameters typically involves the utilisation of grid-search or Bayesian optimisation which are often built into DL experimental platforms like WandB [92]. The objective is typically to maximise the expected return with the intention of outperforming existing heuristics utilised in the target problem domain. The procedure for this optimisation is usually iterative and may involve different problem representations, algorithms and DNN architectures. We stress that caution should be exercised when utilising expected return (or some derivative) when interpreting algorithm performance. Expected return does not give a direct insight into agent behaviour. It only gives a measure of efficacy, however, as mentioned earlier this can be susceptible to reward hacking. At a minimum, performances should be compared across a set of domain-specific metrics. In many cases, it may also be possible to physically inspect agent decisions through the analysis of trajectories. However, the complexity of environments often introduces significant complexity into this form of analysis.

2.3.5 Deploy

Although beyond the vast majority of research papers, the ultimate goal of a method is deployment. In many cases, assumptions which underpin research papers preclude deployment. This is not unexpected, considering the relative immaturity of DRL and DMARL respectively. As we have previously discussed, simulators are a fundamental part of the training process. We therefore

must deal with the challenge of transferring trained policies to a real environment. This is known in the literature as *Sim-to-Real* [93] which attempts to handle disparities between the respective environments through techniques like dynamics randomisation [94]. Additional challenges include *non-stationarity*, which was previously introduced and may result in reductions in performance over time. The tendency of dynamics and objectives to undergo temporal perturbations requires equipment to monitor application performance. For this, ideas from ML-Ops may be useful [95, 96][4]. Through their inclusion, we may be able to create automated pipelines which mitigate issues with non-stationarity.

2.4 Applications of DRL and DMARL

DRL and DMARL algorithms have demonstrated impressive performance across a wide range of challenging sequential decision-making domains. Many of the most high-profile of these have been in game-based domains as they provide a well-defined, controllable and safe environment for algorithm development [97]. Notable breakthroughs include human-level performance in Atari through DQN [3, 98], superhuman performance in two-player zero-sum board games like Go through AlphaGo variants [4, 99], grandmaster level control in StarCraft 2 through ideas at the intersection of imitation learning, DRL and DMARL [52], and most recently, Diplomacy through an algorithm called Cicero which integrates ideas from natural language processing and DRL [100]. In all cases, the employment of ideas from DRL and DMARL has been paramount in the development of policies which match or exceed the performance of human experts.

These high-profile innovations showcase the possibilities and potential of DRL/DMARL techniques within complex domains. Accordingly, there has been a plethora of work which builds upon these innovations with intended application to real-world tasks. Interest across the industry has been ubiquitous with numerous surveys exploring applications within areas like telecommunications [101], logistics and supply chain management [102], autonomous vehicles [103], intelligent transportation systems [104], and economics [105]. Within all of these domains, DRL and DMARL is seen as a tool to handle the significant complexity presented by modern industrial equipment. In general, the vast majority of research papers do not successfully deploy approaches.

Particularly notable examples of a successful application to real-world systems include station-keeping of stratospheric balloons for telecommunications [106], plasma control for nuclear fusion [89], microchip design [7], large scale recommender systems [107] and object manipulation for robotics [108]. We highlight these examples as they have been demonstrated within real-world deployments[5]. Bellemare et al. [106] develop a flight controller for a stratospheric balloon application which provides cellular coverage to ground-based user devices. They utilise Quantile-Regression DQN [109] and contend with partial observability through the incorporation of

[4]These may be useful in the wider context of applied RL too.

[5]Work by Degrave et al. [89] was on an experimental platform, whereas all other examples achieved deployment

uncertainty in wind measurements in the agent's observation. The agent was trained in a bespoke simulator which utilised historic wind data in combination with data augmentation techniques to provide large quantities of realistic training scenarios. These efforts culminated in a successful 39-day deployment and outperformed a highly optimised heuristic algorithm. Degrave et al. [89] consider magnetic confinement of a plasma through an AC algorithm called Maximum a Posteriori Policy Optimisation algorithm [110] which is trained in simulation. As previously stated, the approach leveraged asymmetry between the critic and actor to overcome high control frequency and achieved improved generality and flexibility over standard approaches within testing on an experimental platform. An interesting observation is how they overcome a limitation of their experimental platform's power supplies through reward shaping. Mirhoseini et al. [7] use a GNN-based encoder and PPO [42] for microchip floor-planning which was leveraged within recent tensor processing units. The approach exceeds human-level performance, a challenge that has previously proved intractable for conventional approaches due to its huge state-action dimensionality. Chen et al. [107] contend with large action spaces and learning directly from historic interaction data obtained from an alternative policy. They develop a scalable off-policy REINFORCE approach which is successfully deployed in YouTube's production systems. Finally, OpenAI et al. [108] consider a complex manipulation task requiring the solving of a Rubik's cube with a humanoid hand which is further complicated by a requirement for zero-shot Sim-to-Real transfer. To do so, they utilise a combination of PPO [42] in combination with a novel approach called automatic domain randomisation. Their novel approach learns an automatic curriculum which improves Sim-to-Real transfer. As we have established, highly distributed and interconnected systems are commonplace in many industrial settings. It is therefore notable that there are limited examples of DMARL being successfully deployed into commercial systems. It seems likely that there is still much work to do in bridging the gap between the research and applied streams within DMARL.

As we stated within Chapter 1, in this thesis we focus on two notable industrial domains which are telecommunications and logistics. We consider these sectors to be particularly intriguing due to their criticality in modern society. In both cases, they enable the transit of essential commodities to consumers through highly complicated and interconnected systems. The distributed nature of these domains makes the utilisation of DMARL particularly compelling. Furthermore, established connections with relevant industrial partners place the author in a unique position to explore these domains. Henceforth, we limit our review of applied DRL and DMARL to these domains. These are covered in Section 2.4.1 and 2.4.2 respectively.

2.4.1 Telecommunications

DRL and DMARL are seen as a key technology within 5G and future 6G communications networks [111, 112] due to their capacity to manage the significant complexity which is expected in future networks [113]. In comparison to the previous 4th Generation (4G) system, 5G networks involve

the adoption of a multitude of new technologies and ideas. These include the likes of mmWave to overcome radio spectrum scarcity [114], self-organising networks which enable automatic management (configuration, optimisation and repair) of operational networks [115, 116], and increasing network densification to provide higher capacities [117]. Through these technologies and others, 5G aims to provide data rates in excess of 100Mbps and peak data rates in excess of 10Gbps and latencies of 1ms to users [118] – it eclipses the capabilities of 4G, which has a peak data rate of 150Mbps and latencies of 10ms [113].

In order to ground our conversation of DRL and DMARL algorithms within this setting, we provide a brief description of a simplified cellular network. A cellular network consists of a set of BS which will exchange data with dynamic User Equipment (UE) through wireless communications links. Where UE will typically be allocated to the BS which can provide them with the best wireless links. BS are interconnected through a core network which enables information to be exchanged within the network and also onwards with external services. There is a wide range of problems that can arise even in this highly simplified example. For example, there may be interference issues between BS or UE equipment that are utilising the same frequency bands, allocation of radio resources among users and routing through the core network. Real-world communications networks are significantly more complicated and employment of intelligent DRL and DMARL agents is seen as a solution to these types of problems.

As we have alluded to there has been significant work applying DRL and DMARL within telecommunications, and many problems have been tackled including interference management [119, 120], self-organising networks (e.g antenna parameter optimisation) [121–124], resource management [125–127], fault management [128] and routing [87, 129]. Where we note that in recent years there has been an increased focus on DMARL [112] due to its suitability for distributed decision-making which is attractive in this domain. In the paragraphs below, we expand upon these and detail their contributions.

Antenna Tilt Optimisation BS typically comprise several directional antennas which are configured to provide a trade-off between maximisation of received signal strength for UE they are serving whilst minimising inter-cell interference. Many distributed approaches have been proposed for this challenging optimisation problem, and a multitude of DMARL methods have been proposed to enable coordination. In all cases, the following examples are motivated by the poor performance of single-agent approaches. Balevi and Andrews [121] consider optimisation of antenna tilt angle, and the vertical and horizontal half-power beam widths in order to maximise weighted sum rate within a heterogeneous network. Their training methodologies approximate the interference of nearby cells through a mean-field DMARL Yang et al. [130] and then the policy is derived through a linear function approximation based Q-Learning. The approach is demonstrated empirically within a simulated two-tier heterogeneous network which serves 400 UE. Bouton et al. [122] propose a general method for BS optimisation which is applied to antenna tilting within a traditional cellular network. They consider BS comprised of multiple cells (or

sectors) which are empowered to select their antenna tilts. The method makes use of coordination graphs where they utilise edges to indicate interference between BS sectors. The value of the edge is learned through parameter-sharing DQN, where the reward is defined as a function of the average throughput of each user in its cell. Computation of the maximum obtainable action value requires finding the max across all edges, this is achieved message parsing algorithm. This approach is demonstrated to be scalable to a 207 cell/sector network. Jin et al. [124] propose a similar approach, whereby, DQN is empowered with a graph attention network [36] which allows for local parsing of neighbouring cells configurations through message parsing. In contrast to Bouton et al. [122], they assume a fixed number of neighbours. The approach is trained on a 57 cell network and is impressively demonstrated to generalise to a larger network involving 111 cells. Bouton et al. [123] consider both antenna tilt control and combined antenna tilt and power control. In a similar way to [122, 124] they utilise a combination of graphs and DQN. In this case, they state that they are learning the credit assignment and coordination problem through a method akin to VDN [21] with a GNN. In contrast to [122, 124], they optimise a joint reward. They compare with results from Jin et al. [124] and demonstrate improved performance whilst maintaining the capacity for generalisation.

Dynamic Spectrum Access In contrast to mmWave technologies which make new spectrum available, DSA is a technique attempt to better utilise existing licensed radio bands which often go underutilised for significant portions of time [131]. These exist within the context of cognitive radio networks and involve the proposal of algorithmic methodologies that handle opportunistic access to radio channels. Wang et al. [13] model the problem as a POMDP. The agent's observation is comprised of the agent's belief of the state of a set of available channels which it is able to update by attempting to access the channel. If the agent successfully accesses a channel it receives a reward of $+1$, and otherwise -1. They solve this through the application of DQN which they demonstrate within simple experimental settings. They further extend this to a 2 to 4 user setting but acknowledge difficulties associated with the exponential growth of their action space. Zhu et al. [14] consider a variation on this setting where the network is formed of IoT terminals which introduce packets into a buffer for a relay device to transmit. The relay is assumed to be able to observe both buffer and channel states. The task is modelled as an MDP and the reward is formulated such that a trade-off is found between buffer pressure and transmission power requirements. They apply a Q-Learning variant that utilises an autoencoder to reduce the dimensionality of their state space. Extension to the multi-relay setting is not considered, however, it is identified as future work. Naparstek and Cohen [132] consider an N-user version of the setting setting considered by Wang et al. [13]. They propose a DMARL approach, namely, each user is equipped with an IQL with parameter sharing and an LSTM to handle partial observability. Their IQL is trained for all users in an offline and centralised fashion. Similar to [57] they observe issues with the utilisation of experience replay, and in their case omit it. The approach is demonstrated to on a 100-user 50 channel network effectively demonstrating the

potential of DMARL in a large-scale telecommunications task.

Resource Allocation and Management Many problems in telecommunications networks and computer systems involve questions of resource allocation and management. Mao et al. [125] propose REINFORCE to a multi-resource cluster management task. At any time step, the agent is able to permit multiple buffered jobs to the cluster and is tasked with selecting jobs such that it minimises the average slowdown of computational jobs. They demonstrate that their algorithm outperforms a wide range of established benchmarks. Zeng et al. [133] consider the challenge of edge computing as a resource management task. This involves servers which are located at the network edge and provide services to mobile user equipment. Through intelligently moving the server between edge compute (which is located at BS) in response to user behaviours communication overheads can be reduced but this comes with a migration cost. To find a balance between these opposing objectives they utilise DQN[6]. It is demonstrated on a 50 server, multi-BS network and outperforms a number of domain-specific baselines. However, they do not consider any constraints on edge computing capacity at BS. Li et al. [134] study network slicing which involves the reservation of network resources (i.e. capacity) for customers. They apply DQN and consider sequentially allocating resources for a 3 slice network.

Our contribution Our work within telecommunications involves two problems. Firstly, we consider the application of DRL to resource assignment within an O-RAN instance. Various papers have studied problems at the intersection of DRL and resource assignment [125, 133, 134]. In contrast to these, we study the resource assignment problem in an O-RAN instance. Resource assignment in the related cloud RAN setting has previously been considered by [135–139] who propose a range of exact and heuristic methods. We explore a different approach and cast the problem as a 2-dimensional Bin Packing Problem (BPP). To which, we apply a novel DRL methodology which takes inspiration from AlphaGo Zero [4]. We compare our algorithm to three strong baselines (Heuristic Virtual Resource Allocation Algorithm (HVRAA) [140], Lego [141] and MCTS) and demonstrate a 5.7% improvement in resource utilisation. This contribution is detailed in [16], where the author's contribution are within the problem formulation and methodology. Secondly, we develop a solution for network maintenance planning which we apply to a RAN. Asset maintenance planning itself is a general problem which has seen RL/DRL being applied in numerous scenarios like military settings [142], and industry sectors like energy [143] and manufacturing [144]. MARL/DMARL is less exploited and has been applied to a simple system consisting of two machines and a buffer within [145], a parallel production system [146], a truss bridge which is modelled as a multi-component system [147] and a manufacturing system [148]. These examples can be grouped as small-scale multi-asset systems [145, 146, 148] and larger multi-component systems [147]. We explore the challenge of developing maintenance policies

[6]It is not entirely clear if the implementation is on a per-server basis. If so, this would be parameter sharing IQL.

for large-scale multi-asset multi-component systems and formalise the problem as an POSG. We develop a novel approach which utilises a GNN-based centralised critic. Our approach is demonstrated in two RAN architectures and is shown to improve network availability over heuristic methods by 3.49% and 6.13% whilst not incurring any additional maintenance costs. This contribution is detailed in [17], where the author led on all parts. The second author contributed expertise in asset modelling and provided the foundations of the simulation engine.

2.4.2 Logistics

Logistics as a sector considers the challenge and opportunity which comes from provisioning and managing the movement of goods and services from a point of origin to a consumer [102]. Similarly to what we observed in telecommunications, logistics has explored the possibility of utilising DRL and DMARL to handle the complexities presented by domain-specific challenges. Some of these challenges include the likes of vehicle routing, inventory management and facility location for example [149]. Many of these problems are highly dimensional and are often combinatoric in nature. A classical example is the travelling salesman problem which has been widely studied utilising DRL [150]. In the following text, we consider the direct application of DRL and DMARL to problems within logistics.

Papoudakis et al. [49] introduced an environment called the Multi-Robot Warehouse that simulated a common warehouse problem. This problem is known as the order-picking problem, and they consider a particular instantiation which involves the movement of goods from warehouse locations to packaging locations. This is known as the pick-to-picker paradigm and they consider N-robots collecting requested items from discrete warehouse locations. A common problem in applied research is the relative scarcity of simulators that are both computationally efficient and reflect the salient features of industrial tasks. Christianos et al. [64] demonstrate an approach called SEAC which was introduced in Section 2.2.3 and was demonstrated to outperform other common algorithms in the Multi-Robot Warehouse environment. Kim et al. [151] consider an alternative challenge where an N-grid sortation system must arrange the routing of parcels correctly and quickly towards their relevant destinations. Each grid in the sortation system is instantiated as an agent and a cooperative objective is solved through the application of IQL. Ngu et al. [152] tackle a variant of dynamic vehicle routing problem which they call the same-day delivery problem - this is a common problem and requires the delivery of items from warehouses to customers in a timely fashion. They instantiate the problem as a grid world where orders are received and accepted by a singular vehicle within the environment. They then collect the item from the depot and deliver it to the customer. Agents are incentivised to complete the orders within a defined amount of time by the reward function and are represented algorithmically as a parameter sharing IQL. They compare their approach for a number of vehicle numbers against a mixed-integer programming solution and demonstrate that the relative performance of IQL improves as the number of agents increases whilst improving execution time. Madeka

et al. [153] consider inventory management where the task is to calculate the optimal stock level for a retailer's products within a warehouse. They model the problem as an MDP and apply a model-based DRL approach which overcomes difficulties experienced by DP methods.

Our contribution Our work differs from existing research in logistics, exploring an emerging application with commercial warehouses. The challenge we consider is the order-picking problem, where robot and human pickers must work together to coordinate the retrieval of items from a commercial warehouse. As we detail in Chapter 4, the same setting has been studied in the literature within [154–156] who propose a range of methods have been applied including heuristics and dynamic programming. They place a number of restrictions on how robots and human pickers may interact which we relax. Our contribution within [19] is to the best of our knowledge the first application of DMARL to this problem. The author built upon existing innovations which included further improving a novel simulator, algorithm and developing a policy visualisation methodology.

CHAPTER 3

DEEP REINFORCEMENT LEARNING IN TELECOMMUNICATIONS

Communications networks provide the ubiquitous interconnectivity that is essential to our modern lives. Their necessity for social and economic functions has led to numerous calls for its reclassification as an essential utility. Even still, our dependence continues to grow with emerging applications like autonomous vehicles and virtual reality coming to the forefront [157]. In order to support our ever-increasing demands and expectations we will require increasingly innovative technologies.

ML is envisaged as a key component to enable extraction of higher utilities and to support new applications in future networks and has attracted significant attention in the telecommunications research community [101, 111]. Through the integration of ML methods, we can improve network performance through maximisation of relevant QoS metrics in response to changing network state. As these methodologies begin to approach maturity, considerations for ML applications are becoming increasingly common in standard specifications. For example, Open Radio Access Network (O-RAN) provide provisions and interfaces for ML applications within the Radio Intelligent Controller (RIC) [158].

This combination of societal importance and the anticipated centrality of ML methods in future networks creates a compelling motivation for their study. Communications networks are typically characterised by their highly distributed nature, comprising a vast network of interconnected network elements that enable widespread access to communication services. This chapter intends to explore how DRL methods may be applied to enable the optimisation and management of these types of systems. As in Chapter 1, we argue that their scale and interconnected nature make the application of centralised and naive single methodologies impractical. Therefore, we argue for distributed control algorithms and turn to DMARL for this.

This chapter is split into four main parts, (1) "Resource Assignment in O-RAN" (Section 3.1),

(2) "Why Multi-Agent Reinforcement learning?" (Section 3.2), (3) "Network Maintenance Planning" (Section 3.3) and (4) "Conclusions and Future Work" (Section 3.4). Below, we introduce the relevant sections and provide a synopsis detailing their objectives, contributions and relationships with the wider thesis. Unless otherwise stated, all contributions presented henceforth are the author's.

(3.1) "Resource Assignment in O-RAN" RAN disaggregation is a key part of O-RAN, but introduces challenges for effective resource allocation from RU to DU. This problem is represented as an MDP and an DRL approach inspired by AlphaGo Zero [4] is demonstrated. The developed approach is demonstrated to improve resource utilisation by 5.70% over the strongest baseline. The author's contribution to this work was within the problem formulation and methodology. A major issue which we identify with this approach is the extension to larger communications networks.

(3.2) "Why Multi-Agent Reinforcement learning?" As we have discussed, the inherent characteristics of communications networks make distributed decision-making an attractive solution. We examine the implication of scalability on the resource allocation challenge considered in Section 3.1. We suggest that both centralisation and naive application of single-agent approaches present their own issues. There is a requirement for algorithmic approaches which support distribution, to that end we consider DMARL. Furthermore, we emphasise the generality of this observation which is pervasive within many industrial domains.

(3.3) "Network Maintenance Planning" As a case study for DMARL, we consider learning distributed maintenance policies for communications networks. The abundance and interdependence between network elements make this a complex task. A communication system is considered as an example of the more general class of multi-asset, multi-component systems and the maintenance problem is modelled as an POSG. For this domain, we propose convMAAC which utilises a centralised critic (as in MADDPG) empowered by a GNN to improve learning efficiency and scalability. In a simulated RAN convMAAC improves network availability over standard baselines by 3.49% and 6.13% in two different network topologies.

(3.4) "Conclusions and Future Work" Despite these contributions, the path towards deployment DMARL (and DRL more generally) is fraught with challenges. There are common challenges like sim-to-real, environmental non-stationarity and regulatory considerations. As we will learn in Chapter 4, many of these challenges are commonplace across other industrial domains.

3.1 Resource Assignment in O-RAN

Alongside virtualisation, RAN disaggregation is seen as a key in enabling more flexible deployments [159]. RAN disaggregation advocates for the decoupling of the Baseband Unit (BBU) into three elements, the Remote Unit (RU), the Distributed Unit (DU) and the Central Unit (CU) which serve differing requirements. This idea was standardised by The 3rd Generation Partnership Project (3GPP) in Release 14 [160]. Through its introduction, a number of key benefits can be realised, including, (1) higher network utilisation and efficiency [161, 162], (2) enhance network optimisation and improved network Quality of Experience (QoE) [163], and (3) conducting baseband processing for multiple RU at a DU would reduce computational resource expenditure. An illustration of the envisaged architecture is shown in Figure 3.1. It is seen as a key part of O-RAN which is a consortium of academic and industry partners who endeavour to create a more intelligent, open, virtualized and interoperable RAN which can better support the RAN of the future [158].

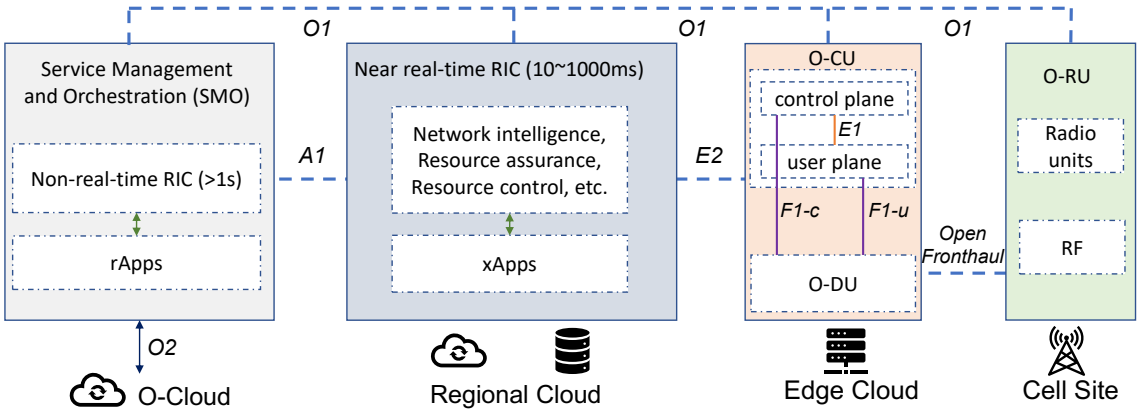


Figure 3.1: O-RAN architecture. Adapted from figure within [96].

This modification requires the DU to be equipped with computational resources which raise issues pertaining to effective resource allocation. Through intelligent resource allocation of RU baseband processing requirements at DU a number of benefits may be achieved. For example, (1) free resources may be reserved for other applications, and (2) inactive computational resources may be powered down to save energy. Resource allocation in RAN has previously been studied and a variety of methods have been proposed including stochastic mixed-integer nonlinear program which is solved by successive convex approximation [135], heuristic methods [136–138] and [139] studies both exact and heuristic methods.

We take an alternative approach, we begin by proposing the modelling of the RU-DU resource assignment problem in O-RAN as a Bin Packing Problem (BPP). To solve the combinatorial optimisation problem, we apply an adaptation of AlphaGo Zero [4] with Ranked Reward (R2) [164] which allows for an approximation of self-play to be introduced into the 1-player BPP. The proposed method achieves a performance gain of 5.70% over the best-performing baseline.

This section provides an extended summary of the work detailed in [16] in which the author contributed to the methodology.

3.1.1 Problem Formulation

We consider the system depicted in Figure 3.1. It consists of a single DU which we model as a computational cluster with capacity C. The DU is connected through fronthaul interfaces to a set of N RUs represented by $K = \{1, \dots N\}$. After a system-defined time-interval, each $k \in K$ raises a request for resource given by the tuple (c_k, d_k) where c_k and d_k represent the required computational resource and the estimated computation time. Collectively, we refer to the set of requests as $\mathcal{I} = \{((c_k, d_k)\}_{k=1}^{N}$. All requests in $\mathcal{I}$ must be served by the DU for the network to be operational.

This problem is represented as a BPP. $\mathcal{I}$ represents our blocks of height c_k and width d_k. The bin is represented by $H = \{\tilde{C}, \tilde{D}\}$, where $\tilde{C}$ is upper bounded by the cluster capacity C and $\tilde{D}$ is upper bounded by a constant D which is the maximum delay permissible for the execution of the jobs in $\mathcal{I}$. As long as the jobs are executed within D, no penalty is received by the system. Therefore, the objective can be formalised as the placement of $\mathcal{I}$ such that the utilised capacity $\tilde{C}$ is minimised. We further note two placement rules, (1) blocks are non-rotatable as axes are not interchangeable, and (2) we permit non-contiguous block placements as long as the required amount of resources are reserved for the same time period. This work considers a 2-dimensional resource problem, the challenge of multi-dimensional resource assignment is left to future work.

3.1.1.1 Markov Decision Process

We convert the formulation given in Section 3.1.1 into the form and terminology that is typical in MDP as introduced in Section 2.1.1. Below, we provide our definition of the state, action and reward.

State Given the bin, H, and the items, I, a state can be created which encompasses all information necessary to learn an effective policy. This is defined as the cartesian product of the set of possible bin configurations $\mathcal{H}$ and the set of possible item lists $\mathcal{I}$ as shown in Equation (3.1). Practically, we encode the bin configuration H and the items $(c_k, d_k) \in I$ as 2-D binary occupancy grids. In the case of the bin, the integer 1 is utilised to indicate the reservation of a resource, and 0 is used to represent an unreserved resource. The representation of items is similar, however, all items' positioning is standardised and they are situated in the bottom left corner of the grid. In the case of item placement, all binary occupancy grid elements are set to 0. The bin and item binary occupancy grids are stacked and passed to the agent as an image.

$$S = \{\mathcal{H} \times \mathcal{I}\} \tag{3.1}$$

Action Items are placed sequentially in an arbitrary order. We note that replacement and removal are not allowed which ensures the item placement process is time-finite, helping to find feasible solutions. At time-step t, we decide on the placement location (x_i, y_i) of the request k_i. Where the location refers to the bottom left corner of the block. An item can only be placed at the edge of the bin or adjacent to another item in both dimensions. This reduces the size of searching space in this problem. If necessary, we allow for the placement of non-contiguous resources, which is handled by a simple heuristic method. This process is Markovian.

Reward The reward is defined in Equation (3.2). Where, C^* is the height of the optimal solution and $\tilde{C}$ is the maximum height of the provided solution. In practise, C^* is not known and it is estimated via $H^* = \max(\frac{\sum_{i=1}^{N} c_i * d_i}{D^*}, \max_i c_i)$.

$$r = \begin{cases} \frac{H^*}{\tilde{H}} & \text{if } s_t = s^T \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

3.1.2 Methodology

Given the MDP in Section 3.1.1.1, we now propose a method which can learn a policy π that maximises the discounted cumulative reward across the distribution of RU requests. Our method is based upon AlphaGo Zero [4] and Expert Iteration [165] which have leveraged the combination of policy iteration and MCTS. For each state, M MCTS rollouts are performed which are guided by a multi-headed neural network $(\pi_\theta, v_\theta) = f_\theta(s)$. The MCTS outputs a refined action probability $\hat{\pi}(a|s)^\intercal$ and a state-value estimation $\hat{v}(s)$ which are used to train f_θ via the loss function given in Equation (3.3).

$$l = (v_\theta(s) - \hat{v}(s))^2 - \hat{\pi}(a|s)^\intercal \log \pi_\theta(a|s) \tag{3.3}$$

We utilise R2 [164] as a reward shaping method. Through its introduction, we provide an approximation to self-play which explicitly couples the agent's reception of reward to its past performance. The R2 reward is defined by Equation (3.4), where r_α is the α percentile of the finite reward buffer, $\mathcal{B}$. The reward buffer, $\mathcal{B}$, stores the last 100 episode returns and is utilised to estimate the current performance of the agent.

$$z = \begin{cases} 1 & \text{if } r > r_\alpha \text{ or } r = r_{\max} \\ -1 & \text{if } r < r_\alpha \\ random(1, -1) & \text{if } r = r_\alpha \text{ and } r < r_{\max} \end{cases} \tag{3.4}$$

Parameter	Value
(C', D')	$(15, 15)$
D^*	15
C^*	Between 2 to 15
Number of items	10
Number of bins	1
Reward buffer length	100
Number of training iterations	300
Number of self-play per iteration	20
Number of MCTS simulations	200
α in ranked reward	75
Batch size	64
Learning rate	0.001

Table 3.1: Resource Assignment experimental parameters

3.1.3 Experiments and Results

In order to assess our method, we apply it within a 2-D BPP. The BPP is configured with the parameters shown in Table 3.1. These are selected to be representative of an anticipated O-RAN deployment in the city centre of Bristol, UK. Our DNN, $f_\theta(s)$, follows the same architecture as in [166] and is first trained on synthetic data. Synthetic data is randomly generated from the slicing bins of height $C^* = U(2, 15)$ into 10 blocks thereby guaranteeing that an optimal solution exists. Training takes 7 hours on a GeForce RTX 2080 Ti and we conduct 5 random seeds. We compare our model against a number of heuristics including Heuristic Virtual Resource Allocation Algorithm (HVRAA) [140], Lego heuristic [141] and MCTS. The performances of baselines are obtained from 100 instances of the same BPP.

$$= \begin{cases} 0.1, & \text{if } i \text{ completed pick at step } t \\ -0.05, & \text{otherwise} \end{cases} \tag{3.5}$$

The training performance is shown in Figure 3.2 and it is compared to baselines in Table 3.2. Our approach achieves a 0.028 improvement in reward, $\bar{r}$, over the baselines. In addition to the reward, we compare the methods in terms of the optimality ratio which is defined as the probability of the solution being optimal and is denoted as $P(\tilde{C} = C^*)$. For our method, $P(\tilde{C} = C^*)$ is 94% which is 29% higher than the closest baseline. This demonstrates how our proposed method is much more likely to produce an optimal solution, demonstrating its superiority over baseline methods. Examples of solutions provided by the methods are shown in Figure 3.3, where Figure 3.3d is our proposed method.

The current network in Bristol is not an O-RAN instance. However, we can use current baseband processing data from existing 4G and 5G sites to estimate realistic RU requirements. Thereby, enabling us to evaluate the performance of our solution within a realistic scenario where

(a) Mean reward. (b) Optimality ratio.

Figure 3.2: Mean rewards and optimality ratio of 2D BPP during 300 training iterations.

	$\bar{r}$	σ_r	$P(\tilde{C} = C^*)$
HVRAA [140]	0.896	0.239	0.65
Lego heuristics [141]	0.737	0.349	0.44
MCTS	0.936	**0.097**	0.62
Self-play learning method	**0.964**	0.160	**0.94**

Table 3.2: Performance on 2D BPP for the proposed self-play learning method and other baseline methods. The metrics from left to right are the mean reward ($\bar{r}$), reward standard deviation (σ_r) and optimality ratio $P(\tilde{C} = C^*)$.

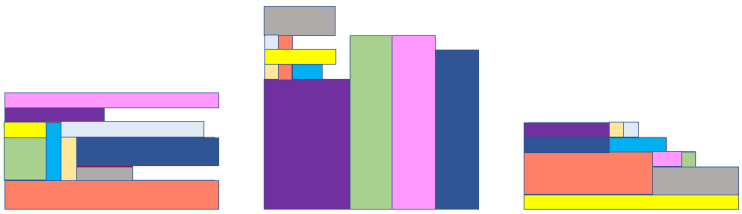
our BPP problem generation method may not hold true. A number of sites within Bristol have been identified as possible locations for DU, we sample 2 at random and then assign the nearest 10 RU sites to them. Our obtained results are shown in Table 3.3. Rather than optimality, we introduce a new metric which we refer to as resource utilisation which is defined in Equation (3.6).

$$U = \frac{\sum_{i=1}^{N} c_i * d_i}{\tilde{C} * \tilde{D}} \tag{3.6}$$

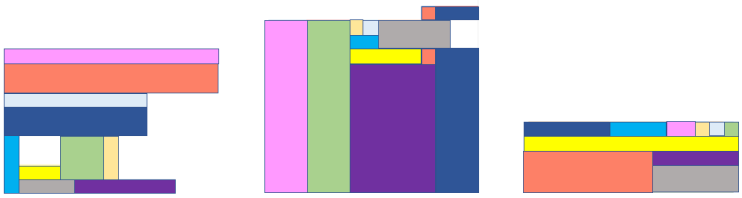
	DU1		DU2	
	$\bar{r}$	$\bar{U}$	$\bar{r}$	$\bar{U}$
HVRAA	0.879	0.809	0.938	0.810
Lego heuristics	0.841	0.774	0.811	0.700
MCTS	0.847	0.781	0.847	0.732
Self-play learning method	**0.943**	**0.869**	**1.000**	**0.864**

Table 3.3: Resource assignment performance on real sites. The metrics used are the mean reward ($\bar{r}$) and mean resource utilisation ($\bar{U}$) for DU1 and DU2.

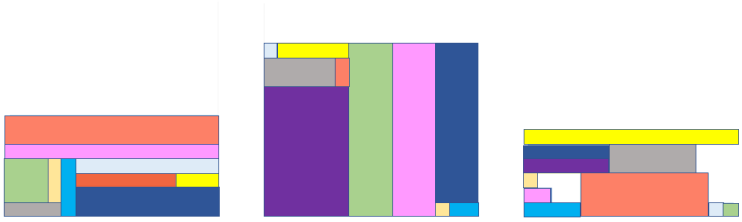
We additionally present the inference times required by the multitude of methods which we utilise in Figure 3.4. We find that our method allows significant speed-ups over MCTS but takes longer to compute compared with the negligible times required by the heuristic methods.



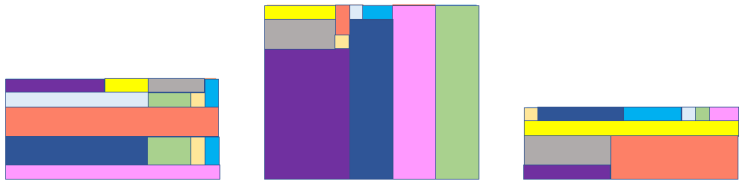
(a) Results of the HVRAA method.



(b) Results of the Lego heuristic method.



(c) Results of the MCTS method.



(d) Results of the self-play learning method.

Figure 3.3: Results of three 2D BPP instances using baseline methods and the proposed self-play learning strategy. For each instance, the same colour represents the same item.

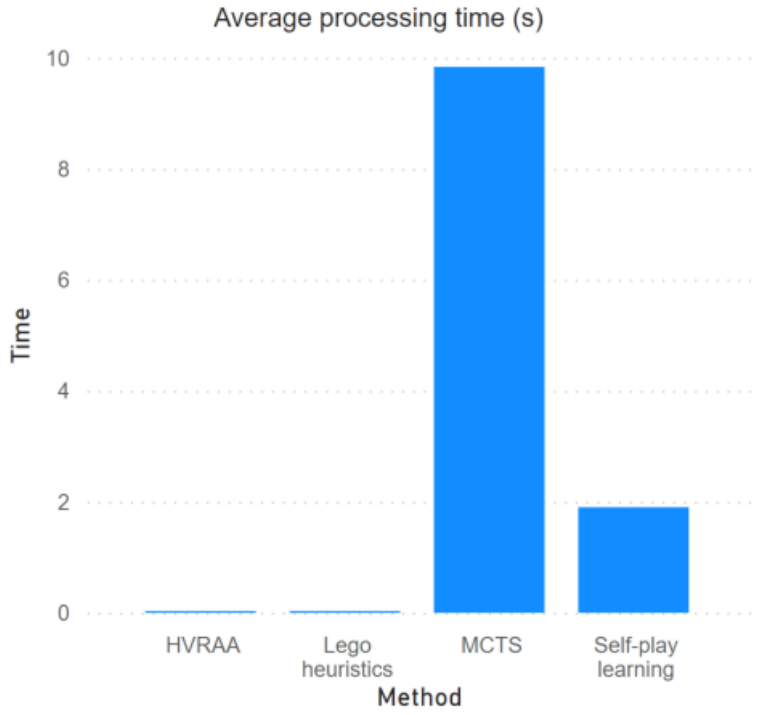


Figure 3.4: Execution time of different methods in seconds.

3.1.4 Summary

Within Table 3.2 and 3.3 we have demonstrated that our method can consistently outperform the performance of a range of strong baselines. The ability to train on synthetic data and maintain performance when applied zero-shot to realistic data demonstrates strong generalisation which is a desirable characteristic.

We do however note a range of limitations. Firstly, the markedly higher inference times presented in Figure 3.4 which may be problematic for realistic systems and its reduction would form a considerable part of productionisation. A second related point, is the suitability of this method as the network scales. This shall form the basis of the rest of this section, where we will consider the suitability of DMARL for this in the next section.

3.2 Why Multi-Agent Reinforcement Learning?

In Section 3.1, we demonstrated the application of single-agent DRL to a resource allocation problem. The experimentation includes a simplified communications network consisting of N RU, a single DU and only a single type of computational resource. Realistic communications networks are likely to be much larger, spatially distributed and utilise multiple types of computational resources (typically CPU, RAM and GPU). Extending the proposed method to realistic scenarios requires tackling a much more complex problem.

Let us consider the implication of utilising a fully centralised model in a more realistic resource allocation context. A large network will be comprised of significantly more RU and DU. Both the action space and observation space of the model will scale accordingly as there will be a larger number of requests and positions in which each item can be placed. The required DNN will grow accordingly, for a small increase this will result in longer inference times but for larger increases, it may be prohibitively large. Furthermore, as observed by [21], the derivation of centralised policies can be difficult. There are also implications for the physical network, for example, as distances grow latencies and network traffic overheads increase and there may also be concerns for network robustness as a centralised model presents a single failure point.

In Section 3.1.3, we handle a small part of a realistic network by dividing the network into segments with equal numbers of RU. This is a plausible solution, however, it operates on the principle of a fixed mapping of RU to DU derived from a distance-based heuristic. This is an improvement on the naive application of single-agent methods which were discussed in Chapter 1 as it ensures that agents interact in worlds that are effectively factorised into spheres of influence. However, it inherently limits us to a non-optimal solution as it does not optimise global resource utilisation.

The aforementioned difficulties with centralised methodologies and the direct application of single-agent are problematic. In telecommunications, these problems are particularly acute as the sector's business requires providing communications services to geographically distributed users. This mandates the distributed deployment of communications equipment. Approaching network optimisation through centralised methodologies is likely infeasible due to the scale, and single-agent style approaches may leave untapped performance available. We turn to DMARL with the intention of deriving solutions that are able to handle these complexities. More widely, distributed problems are pervasive within many industry applications [12] and hence we need solutions which address these challenges.

Through the distribution of DRL agents throughout the network, we may alleviate these problems. However, there is no free lunch and this does give rise to new problems. In the next section, we consider the challenge of defining a maintenance strategy for a network. We take inspiration from MADDPG [9] and other approaches described within Chapter 2 and demonstrate an application to a small RAN. Although, we make advances challenges still remain and we expand upon these in Section 5.4. In later sections, we consider the potential of DMARL in domains like logistics in Chapter 4.

3.3 Network Maintenance Planning

Communications network users expect the service they receive to be reliable and fast. However, equipment failures can degrade this, potentially resulting in significant reductions in the quality of service for users and costly emergency maintenance expenditures for network providers.

Through proactive maintenance planning, network providers can mitigate the potential impacts of equipment failures. Unfortunately, maintenance planning in this domain is complex due to a multitude of factors. Firstly, an individual asset (e.g base stations) may be comprised of numerous operationally critical modular components which have their own expected service life and degradation profiles that in turn influence the asset's operation. Secondly, there can be operational interdependencies between assets within the network. These factors become relevant when designing a network-wide maintenance policy that considers not only the maintenance costs (e.g. labour and parts) but also the costs of unplanned downtime within the network.

Numerous studies have investigated the challenges of maintenance planning in a variety of domains. A common choice is to model the maintenance planning problem as an MDP and to solve it utilising methodologies like RL [144, 167] or if the dimensionality allows it DP [168]. However, when large numbers of elements are considered their application typically becomes infeasible due to the *curse of dimensionality* [1]. DMARL methods provide a possible solution for this problem and their application has been explored within multi-component systems [147] and multi-asset systems [145, 146]. The work described in this section is motivated by application to communications networks which are typically multi-asset multi-component systems which is yet to be explored in the literature.

The contributions of this section are threefold: 1) Definition of the networked asset maintenance problem as a Multi-Agent problem 2) An approach to solve this problem based on MAAC and 3) Demonstration of the proposed approach in a simulated environment within the context of RAN. This work involved collaboration between the author and various other persons within the NG-CDI consortium. The author was the majority contributor to this work and was involved in all stages discussed henceforth. The work featured in this section is based upon contributions in [17] and further expands on those results presenting results obtained utilising communicative DMARL.

This section is organised as follows. First, 3.3.1, present the relevant works in group asset maintenance planning. Next, Section 3.3.3 introduces the formulation of the group maintenance problem as a POSG. Section 3.3.4 describes the algorithmic aspects of the proposed approach. The results and discussion of the evaluation, based on a simulated case study, are presented in Section 3.3.5. Conclusions and future work are presented in Section 3.3.9.

3.3.1 Asset Maintenance Planning

This section reviews relevant efforts addressing the multi-asset maintenance planning and also applications of RL for asset management.

3.3.1.1 Maintenance of Network of Assets

Markov decision processes have been widely used to model asset deterioration and to develop maintenance plans. The problem of finding the optimum maintenance policy for a component

is formulated as an MDP in [168]. The policy for determining when the component should be maintained is obtained via uni-chain policy iteration algorithms. Authors of [169] use POMDP as the basis of a methodology for assessing the benefits of condition monitoring systems.

While these works address mainly single-asset scenarios, MDP have also supported models of multi-asset systems. A model to optimize the predictive maintenance policy for a multi-system multi-component network of assets is presented in [170]. The policy is obtained using a genetic algorithm that enable the exploitation of the benefits of grouping system interventions and overlapping downtime in a two-bridge network. An MDP is also used to formulate a joint Condition-Based Component Replacement and Inventory Control Policy (CBRICP) in [171]. Dynamic programming is used to solve the CBRICP, determining the optimal maintenance and inventory actions for *k-out-n:F* systems.

3.3.2 Reinforcement Learning for AM

Several works have addressed the use of single-agent RL for asset maintenance planning. An RL model for determining the optimal timing for maintenance while maintaining the reliability of the system is presented in [167]. The model was shown to effectively optimise maintenance scheduling considering expected reward and mean time between failures (MTBF). In [142], a Monte Carlo RL approach is used to find the optimal preventive maintenance strategy for multiple components of a group of assets, with application to a fleet of military trucks. The authors showed that Monte Carlo RL enabled learning of the transition probabilities of the underlying MDP while optimising at the asset system level rather than locally. A method combining a Q-learning algorithm and an ensemble of Artificial Neural Networks (ANN) enables learning of the optimal operation and maintenance policy for a power grid [143]. Despite the reported high computational time required, a distinct feature of this method is the ability to potentially consider large systems with high dimensional state-action spaces. The benefits of neural networks with RL are also exploited in [144], where Double DQN [39] is applied to determine the preventive maintenance policies that reduce the cost in a serial production line system.

The MARL framework has been less exploited for addressing asset maintenance planning problems. A small system of two machines and one buffer is considered in [145], where authors propose a cost-sharing-RL algorithm that outperforms single-agent RL when comparing the system average costs. Likewise, authors of [146] show the benefits of a multi-agent configuration where the RL framework enables the reduction of downtime and maintenance costs compared to other available strategies in a parallel production line system. Deep Centralised Multi-Agent Actor-Critic (DCMAC) is proposed for obtaining the maintenance policy of multi-component systems, with application to a truss bridge [147]. Authors of [148] combine Genetic Algorithms (GA) and MARL to maximise the average revenue rate of a manufacturing system with intermediate buffers. This revenue considers the cost of corrective and preventive maintenance of the entire system as part of the agent's reward function. The GA assumes global visibility of

the system and periodically signals agents to adjust their actions. While there have been efforts to address cases of larger multi-component systems [147] and small-scale multi-asset systems [145, 146] separately, the use of MARL frameworks to address the complexity of maintenance policy computation in networks of multi-assets multi-component systems is still to be explored.

3.3.3 Problem Formulation

The motivation for this work is to automatically derive maintenance policies for a communications network comprised of multiple network elements. As previously stated, each network element may be comprised of several modular components. This system can be modelled as a network comprised of N assets defined by the set $Z = \{z_1, \ldots, z_N\}$. Each asset z_i may be comprised of n components, where n may be variable. Each component c follows cycles of deterioration and maintenance which may have an impact on the asset or network level. The full set of components is referred to as C. The key elements of the problem are further described below.

3.3.3.1 Component Level

Each component, $c \in C$, is modelled as an MDP as illustrated in Figure 3.5. The state space $S_{c_j} = \{s_0, s_1, ..., s_k, s_{k+1}, ..., s_{b-1}, s_b\}$ represents the states of the component until breakdown (s_b). Where each state is collated from observation of the component's sensors which may be noisy. The deterioration rate λ_k determines the change in condition from state s_k to state s_{k+1}, where the probability of the component transitioning directly to s_b increases as the component matures.

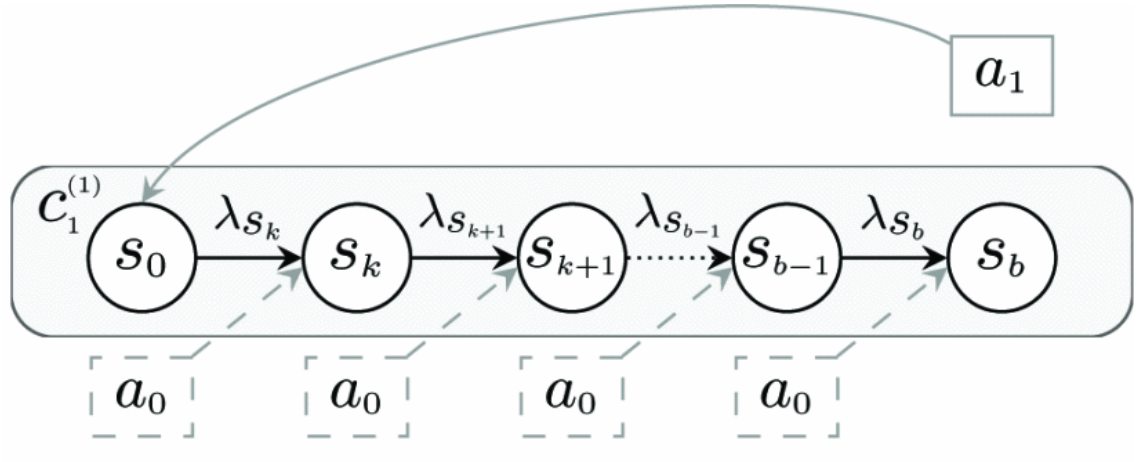
At every state, a fixed set of domain-dependent actions are assumed to be available. An example set could be: *Do nothing* (a_0), *Component Replacement* (a_1) and *Component Repair* (a_2). a_1 results in the state of the component being returned to s_0 representing replacement. a_2 restores the state of the component, however as the component is restored multiple times, the effect is reduced. This is modelled by ϵ which is a function of the repair count. This is shown in Fig. 3.5a.

3.3.3.2 Asset Level

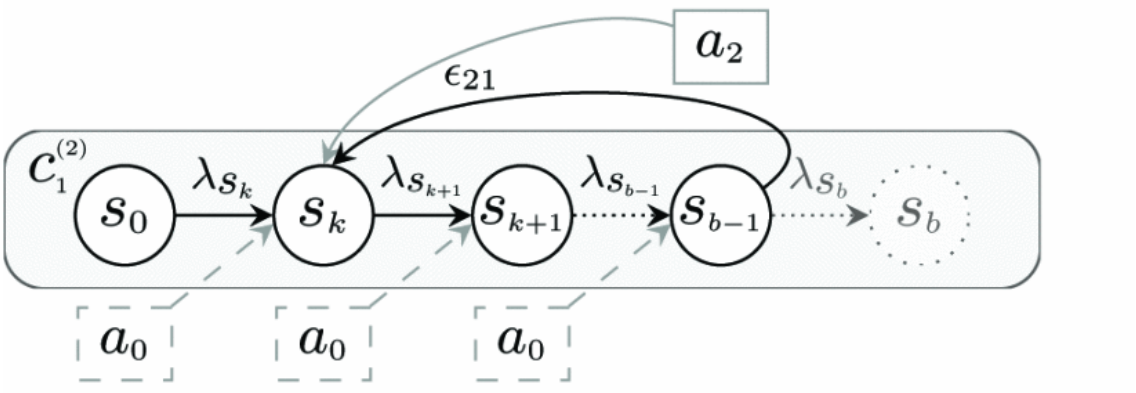
An asset z_i model can be obtained from a combination of its n components' MDP and modelling of their operational interdependencies. Here, the state space of an asset is defined as $S_{z_i} = \{S_{c_1} \times \cdots \times S_{c_n} \times f\}$, where f indicates the operational status of the asset which is defined according to component dependencies. The action space is defined as $A_{z_i} = \{A_{c_1} \times \cdots \times A_{c_n}\}$.

3.3.3.3 System Level

The larger network maintenance problem can then be represented as an N-player POSG [45], where each asset is considered as an agent. This was previously defined in Section 2.2.1 and all



(a) Replacement action a_1 taken at s_b



(b) Component Repair action a_2 with effectiveness ϵ

Figure 3.5: MDP showing two actions taken for components $c_1^{(1)}$ and $c_1^{(2)}$ of assets 1 and 2 respectively. State transitions are paired with the actions a that lead to the next state. λ represents the deterioration rate. s_b represents a component breakdown. *"Do nothing"* actions are dashed.

values maintain the same values as defined previously. The objective is to find the set of asset-level maintenance policies (Π^*) which maximises the expected cumulative discounted reward (r). An abstract illustration is provided in Figure 3.6. The exact definitions of observations, actions and the reward function are left to Section 3.3.4 as they are domain dependent.

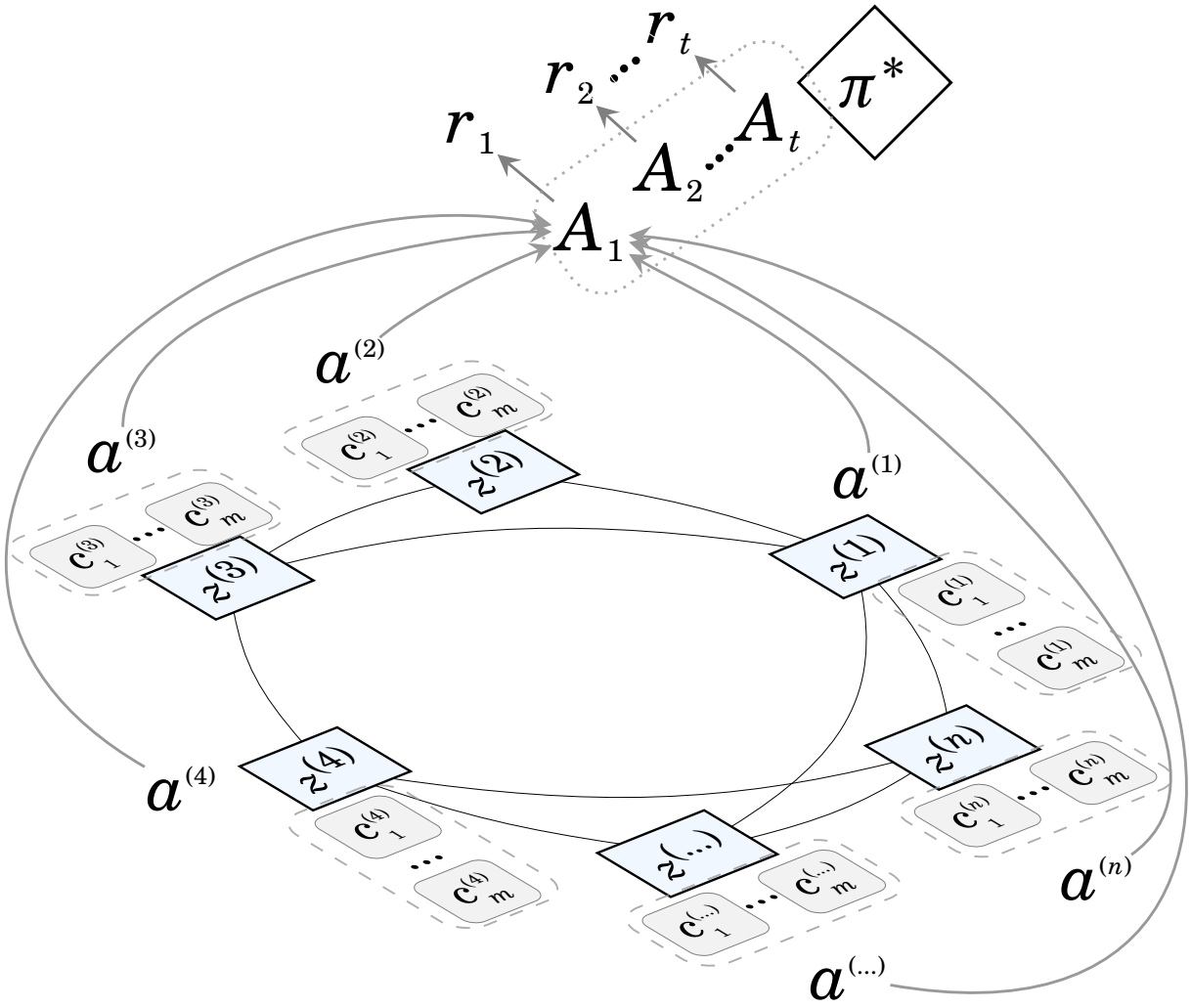


Figure 3.6: Maintenance actions a are taken in a network of n assets Z, each one with m components c. For a given time t, a joint action $a^{(k)}$ is taken for each asset, contributing to a network action A_t. The goal is to find the policy π that maximises the cumulative joint reward r.

3.3.4 MARL for asset maintenance

The target task could be approached in a centralised manner, where a single agent is responsible for learning and executing a maintenance policy for all assets. As discussed by [147], single agent DRL can struggle with large discrete action spaces. In the problem definition provided in Section 3.3.3.3, the action space may be very large if there are significant numbers of assets as

the action space scales exponentially. This is to be expected in domains with similar properties to communications networks. A more practical solution and one considered here is a decentralised approach where techniques from DMARL are applied to improve training efficiency and facilitate learning cooperative policies. By decentralising, there are also other benefits including a reduction in communication overheads, system robustness (as a centralised controller presents a single point of failure), and the framework can support the introduction of new assets [12].

DMARL approaches take considerable time to train and, as is the case in DRL, require environmental exploration to derive near-optimal behaviours. Therefore, training within simulation is considered to be the most viable path to deployment as it affords opportunity to train at speeds well in excess of real-time and without risking equipment failure. A recent example of this strategy is the work undertaken by [6] where they train a policy within a simulator to control a stratospheric balloon for telecommunications applications and then successfully deploy it in the real world. The case study undertaken in Section 3.3.5 utilises a custom simulator, but more generally a Digital Twin [172] is a viable option for algorithm training where asset models could be refined through the integration of real-world data. The use of a simulated environment also provides the opportunity to investigate system response in rare circumstances, which may offer benefits from a system resilience perspective. However, there are issues that present concerning the challenge of moving from simulation to real-world deployments which are often studied under *Sim-to-Real* [93].

The utilisation of a simulation to train DMARL also brings algorithmic benefits like CTDE [54]. This common MARL paradigm facilitates the introduction of extra information within training to improve training efficiency as long as this information is not required when the model is deployed [9]. A number of approaches exist including Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [9] and Q-MIX [55], for example.

MADDPG considers a modification to Actor-Critic (AC) [1] for the MARL domain. Within single-agent RL AC (or Independent AC in [10]) the actor is responsible for selection of actions and is conditioned on the observation of an individual agent and the critic provides feedback to the agent about the return associated with its selected action and is conditioned on the action and state of the agent. When naively applied to a DMARL problem, this can encounter issues with non-stationarity as any estimate of the value which the critic produces is dependent on the state and policies of other agents. In MADDPG, the critic is centralised and is conditioned on the observations of all agents. By doing so, the problems aforementioned can be reduced where Lowe et al. [9] suggests that it improves the consistency of gradient signals. Notably, only the actor is required for inference so the centralised critic can be discarded upon deployment.

In the remainder of this section, two algorithms referred to as Multi-Agent Actor-Critic (MAAC) and Convolutional Multi-Agent Actor-Critic (convMAAC) are introduced. Both of which adapt MADDPG to the preventive maintenance task, but utilise different critic architectures. They build upon A2C due to DDPG not natively supporting discrete action spaces. Other algo-

rithms like choices are possible, but as demonstrated in Section 3.3.5, A2C is effective.

3.3.4.1 MAAC

Algorithm 3 presents the MAAC base workflow used in this paper. Where, T and E represent the number of time steps in an episode and the number of episodes, respectively. The algorithm calculates the Advantage A, Policy gradient $\nabla_\theta J(\theta)$ and Value function gradient $\nabla_\phi J(\phi)$ with the functions detailed in Equations 3.8, 3.9 and 3.10 respectively.

Algorithm 3: Multi-Agent Actor-Critic algorithm

Input: G, T, E
Output: Trained $\pi_{1,\theta}, ..., \pi_{N,\theta}$
Initialise agent policies $\pi_{1,\theta}, ..., \pi_{N,\theta}$;
Initialise centralised critic v_ϕ ;
for e = *1, ..., E* **do**
  Initialise episode buffer $\mathcal{D}$;
  for t = *0, ..., T* **do**
    Receive $o_t = \{o_{i,t}, ..., o_{N,t}\}$;
    $a_t = \{\}$;
    for i = *0, ..., N* **do**
      Sample $a_{i,t} \sim \pi_{i,\theta}(\cdot|o_{i,t})$;
      $a_t = a_t \bigcup \{a_{i,t}\}$
    end
    Store (o_t, o_{t+1}, a_t, r_t) in D;
  end
  For all samples in D;
  Compute Advantage using *Equation 3.8*;
  Compute Policy Gradient using *Equation 3.9*;
  Compute Value Gradient using *Equation 3.10*;
  $\theta \leftarrow \theta - \alpha_\theta \nabla_\theta J_\pi(\theta)$;
  $\phi \leftarrow \theta - \alpha_\phi \nabla_\phi J\phi(\phi)$;
end

3.3.4.2 convMAAC

In situations where there is a high number of agents, as there will be in any realistic network maintenance operation the utilisation of a centralised critic may invoke significant computational expense due to it being parameterised by a fully connected network. To aid in model scalability, the problem is cast as a Graph Regression task. Thus Graph Convolutional Network (GCN) [35] are used to learn useful node embeddings which capture topological information and then a mean operation on the set of node embeddings is used to reduce the graph to a fixed length vector. Where the associated graph $G = \{V, E\}$ is defined by the connections between elements in the network. Equation 3.7 defines the operation of the GCN, where $\mathbf{h}_i^k$ refers to the encoding of agent

i after k convolutions and is initialised to the respective agents observation, o_i, at $k = 0$, $\mathbf{W}^k$ is a trainable weight matrix, and $\mathcal{N}\{i\}$ represents the set of neighbours of agent i. This results in a model with significantly less parameters than the fully connected equivalent, MAAC.

$$\mathbf{h}_i^{(k)} = \sigma\left(\mathbf{W}^k \sum_{v \in \mathcal{N}(i) \cup \{i\}} \frac{\mathbf{h}_v}{\sqrt{|\mathcal{N}(i)||\mathcal{N}(v)|}}\right) \tag{3.7}$$

$$A(O, r) = r + \gamma V(O') - V(O) \tag{3.8}$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_{i,\theta}}[\nabla_\theta \log \pi_{i,\theta}(o, a) A(O, r)] \tag{3.9}$$

$$\nabla_\phi J(\phi) = \mathbb{E}_{\Pi_\theta}[\nabla_\theta (r + \gamma V(O') - V(O))] \tag{3.10}$$

3.3.5 Simulated Case Study: Radio Access Networks

The proposed method is applied to the maintenance of telecommunications infrastructure deployed within a region of interest. This section focuses on two challenges, each with increasing difficulty. The first challenge involves an RAN consisting of N-BS. In this case, there are no interdependencies between assets and the failure of a singular BS only impacts the area that it is serving. The second challenge is an RAN comprising of $N - 1$ BS which are interconnected by a central router. This is more intricate as the central router represents a single point of failure hence it requires additional precautions. The cases are referred to as *Complete Network* and *Star Network* henceforth.

The observation, action and reward functions are specific to the particular domain and are defined here. The assets (BS and routers) are defined as 2-component systems, where their states are derived from continuous sensors which allow for inference of component state and the anticipated time til failure. It is assumed that both components are critical to normal asset operation and hence the failure of either component results in complete failure. This is represented by f which is 1 if the asset fails and 0 otherwise. The available actions for each component are *Do nothing* (a_0), *Component Replacement* (a_1) and *Component Repair* (a_2). As before, asset-level actions are defined as the Cartesian product of its components. In all cases, a maintenance action will take 1 time-step for completion. However, it is assumed that there is a finite amount of maintenance resources available at each time step reflecting a finite workforce. If this is exceeded, a subset of the requested resources will be completed.

$$r = -\alpha \sum_{i \in n} m_i - \beta \sum_{i \in n} f_i \tag{3.11}$$

As discussed in Section 2.3, in MARL a decision must be made between a local and global reward function. In this domain, a global reward function is utilised as it more easily allows for encoding of the network-level problem than a local reward. The formulation is given in Equation 3.11. m_i represents the cost of maintenance operation scheduled by asset i and f_i is a binary variable referring to the asset's operational status (1 if the asset is unavailable and 0 otherwise). This formulation captures the trade-off between maintenance and network availability which is used as a proxy for profitability. The importance of these diametrically opposing objectives can be controlled through the appropriate selection of α and β.

Table 3.4: Experiment parameters, where the numbers associated with actions u_x are their costs and the components RUL are initialised randomly according to a uniform distribution. Note that the the actions u_x which apply to both components are proportionally cheaper, this assumption is based on an operations engineer travel time being reduced relative to the equivalent actions concurrently.

Parameter	Value
Number of assets	9
Max maintenance operations per timestep	2
Episodes	3000
Episode length	100
a^0, Do nothing	0
a^1, Replace c_1	0.3
a^2, Repair c_2	0.1
a^3, Replace c_2	0.3
a^4, Replace c_1 and c_2	0.55
a^5, Repair c_1	0.1
a^6, Repair c_1 and c_2	0.175
Component RUL	$\mathcal{U}(95, 105)$
α	1
β	1
ϵ	$0.7^{\text{Number component repairs}}$

The parameters used in the experiments are presented in Table 3.4. These and the case study are based on discussions with partners of the NG-CDI project[1] and intend to reflect the realities of real scenarios. For example, note that execution of maintenance request u_4 which involves the replacement of components c_1 and c_2 is proportionally cheaper than the replacement of them individually which reflects a reduction in operational engineer travel time.

3.3.5.1 Implementation

The algorithms introduced in Section 3.3.4 are implemented in PyTorch [173]. In both MAAC and convMAAC agent networks utilise parameter sharing [8] and the architecture consists of a

[1] https://www.ng-cdi.org/

2-layer FCN followed by a 2-layer LSTM. In order to ensure distinct behaviours can be derived, an asset identifier is concatenated onto the asset observations. The inclusion of the LSTM allows for capture of temporal correlations which may go unseen if just the FCN was utilised. The major difference between MAAC and convMAAC is the architecture of the shared critic network. MAAC critic comprises of approximately $1.2M$ parameters and utilises a 2-layer FCN followed by a 2-layer LSTM which takes as input the concatenation of all agents' observations and outputs the state value. The critic of convMAAC comprises of approximately $610K$ parameters and utilises a 2-layer FCN which encodes the observation of all agents separately, and then all encodings are passed into a 2-layer GCN, the output is then averaged node-wise and passed into a 2-layer LSTM which then outputs the state value. The model is optimised using a grid search across parameters, where the hyperparameters utilised within the experiments are depicted in Table 3.5. Training is performed on an Nvidia RTX 2080Ti and takes 30 minutes to converge. WandB [92] is used for experimental logging and the GCN is implemented using DGL [174].

Table 3.5: Network Maintenance Planning model hyperparameters

Model	Actor LR	Critic LR	Gradient Clipping
MAAC	0.0001	0.0005	1.0
convMAAC	0.0001	0.0005	1.0
IAC	0.00005	0.0005	0.5

3.3.6 Experiments

Although the task is continual it is represented as episodes consisting of 100-time steps which correspond to 5 full life cycles of an average component. At the beginning of an episode, assets are assumed to be in factory condition where there is some variance in condition as defined in Table 3.4. All algorithms are allowed 3000 episodes of training and results are averaged over 3 random seeds. The derived policies are compared to two maintenance baselines which are a corrective maintenance policy and a preventive maintenance policy. The corrective policy requests for component replacement upon failure. The preventive policy takes the form of a distributed oracle which can observe the Remaining Useful Life (RUL) of the asset's component and requests maintenance resource when the RUL drops below 20. To demonstrate the benefit of critic centralisation IAC [10] agent is implemented too, where the architecture maintains a similar structure to convMAAC and MAAC.

3.3.7 Results and Discussion

Table 3.6 shows the converged models of MAAC, convMAAC and IAC to heuristic baselines corrective policy and preventive policy. It provides a comparison in terms of network availability and maintenance frequency. Network availability is defined as the average amount of the BS that are usable by users over the course of an episode. Maintenance frequency is the average

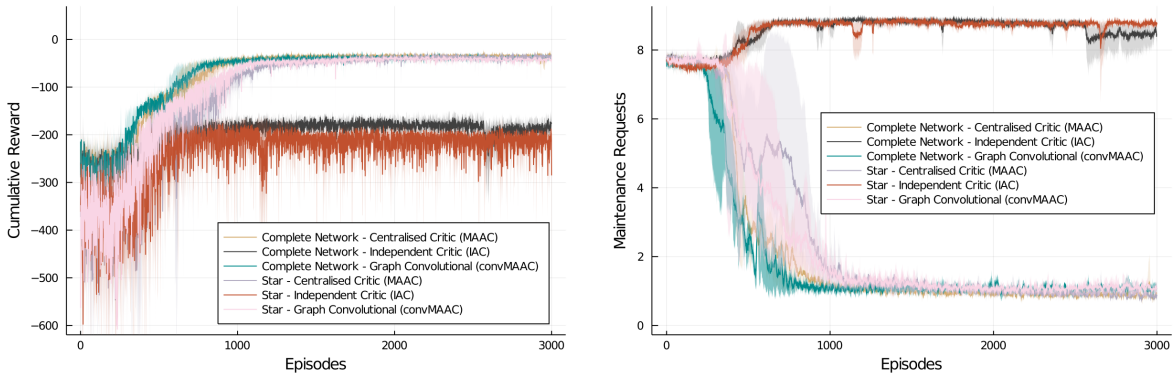
Table 3.6: Final Algorithm performances, where the first number is the mean and the second number the standard deviation.

Model	**Complete Network**		**Star Network**	
	Net. Availability	Main. Frequency	Net. Availability	Main. Frequency
MAAC	0.9972, 0.0035	0.8981,0.1356	0.998, 0.0019	0.8864, 0.07219
convMAAC	0.9997, 0.0005	1.1001, 0.1525	0.9997, 0.0012	1.092, 0.1678
IAC	0.9803, 0.0088	7.4490, 1.4560	0.97, 0.008	8.4631, 0.0699
Corrective	0.8750, 0.0084	1.1118, 0.0744	0.7911, 0.0181	1.111, 0.0769
Preventive	0.9638, 0.0100	1.0863, 0.1169	0.9384, 0.0331	1.0738, 0.1039

number of repair or replacement operations which are scheduled at every time step. Both MAAC and convMAAC achieve similar performance levels in both the *Complete Network* and *Star Network*. There is a heuristics and there is a measurable decrease in performance between the two topologies. This demonstrates that MAAC and convMAAC are able to adapt to variations in network topology which are impactful for more naive approaches.

Comparisons of the algorithms' training performances for the two cases analysed can be found in Figure 4.2. Figure 3.7a shows the cumulative reward obtained as the agents learn their policies. In both cases, MAAC and convMAAC converge to comparable cumulative rewards, where convergence is faster in the complete network case. This is likely due to the difference in complexity which makes learning in the star network slower. It appears that convMAAC has a slight edge in convergence speed over MAAC. IAC can encounter difficulties learning effective policies, where this policy was found to be approximately uniform random regardless of the case. The differences in DMARL learning algorithms performance demonstrate the importance of the centralised critic in training, as without it, it is difficult to estimate the true state value function within the environment as it is partially observable from the perspective of a single agent. When considering an extension to larger deployments it is arguable that the smaller critic size required by convMAAC may give it an additional edge. Although not required at deployment time, practically, a smaller model presents smaller computational requirements.

Through observation, the policies derived by the convMAAC and MAAC can be found to alternate between full system repair and replacement operations. This suggests that, given the experiment parameters, the agents understand that performing repair is a preferable operation to replacement and that coordinating jobs at a single site is cost-effective. An annotated example trajectory for an individual asset and the actions taken by the corresponding MAAC agent can be found in Fig. 3.8. It shows that the agent is risk-averse and tends to perform preventive maintenance of the asset when the components have a RUL of approximately 40. Where this is the point in an asset's degradation where the probability of failure for an asset starts to increase more rapidly. This is seemingly a pragmatic and reasonable decision if the possibility of contention among agents for the limited maintenance resource available is considered. At the global-level agents tend to interleave their maintenance decisions tending to schedule maintenance at regular



(a) (b)

Figure 3.7: Key MARL parameters throughout training. (a) Cumulative reward and (b) Maintenance request per time step.

intervals.

3.3.8 Further Experimentation

In the original publication [17] the potential for communicative DMARL to further improve performance was suggested. It was believed that communication may enable agents to better understand the state of the wider network and to be able to better allocate resources. Here, those ideas are explored. MAAC is extended to allow communication among agents utilising a method similar to that proposed within DGN [70]. This method is referred to as Communicative Multi-Agent Actor-Critic (commMAAC) and is further expanded upon below.

3.3.8.1 CommMAAC

As in DGN [70], Multi-Headed Dot Product Attention (MHDPA) [175] is utilised to facilitate information exchange between agents. MHDPA provide a method to compute a weighted aggregation based on perceived importance of neighbouring agents' observations which are then utilised in the respective agent's policy. This is achieved through the application of Equation (3.12) and Equation (3.13). Together they provide mechanism to support communication of relevant information between agents, in the following text we explain their operation.

$$\alpha_{i,j}^{m} = \frac{exp(\tau \mathbf{W}_Q^m h_i \cdot (\mathbf{W}_K^m h_j)^T)}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} exp(\tau \mathbf{W}_Q^m h_i \cdot (\mathbf{W}_K^m h_k)^T)} \tag{3.12}$$

$$h_i^{'} = \sigma(concatenate[\sum_{\sum_{j \in \mathcal{N}(j) \cup \{j\}}} \alpha_{ij}^m \mathbf{W}_V^m h_j, \forall m \in M)] \tag{3.13}$$

Equation (3.12) serves to provide a learnable mechanism to calculate the relative importance of the information contained within agent j's encoding, h_j to agent i with respect to its set of

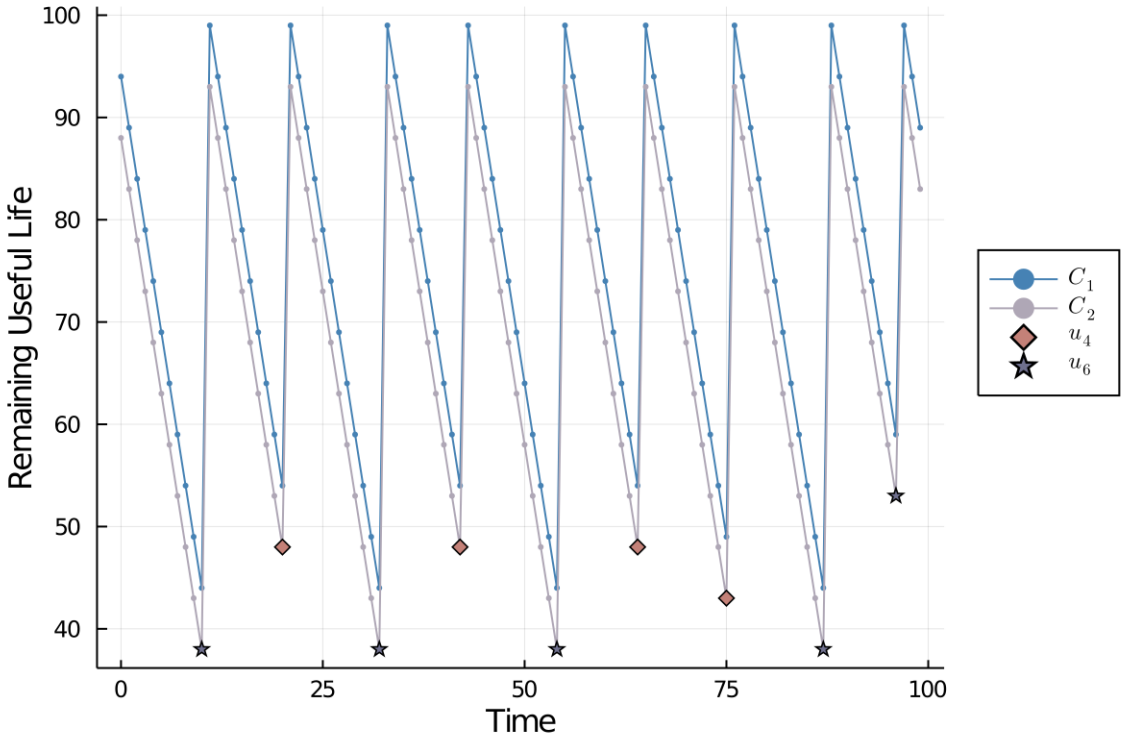


Figure 3.8: Example degradation of an asset maintained by MARL agent. C1 and C2 represent components and u_4 and u_6 represent actions 4 and 6 respectively. Where u_4 is a joint replacement and u_6 is a joint repair of both components simultaneously.

neighbours (including itself). This equation takes the familiar form of the softmax function which is commonplace across DL in situations where a categorical probability distribution is required. Here, the inputs into the exponential functions are the dot product similarity between the linear transformation of the agent's encodings by weight matrices $\mathbf{W}_Q$ and $\mathbf{W}_K$. In the numerator and denominator, the variable τ is present and is typically referred to as the temperature co-efficient. It provides a method to control the distribution of probability mass where a higher value results in more probability mass (and therefore attention) being assigned to the higher dot product similarities.

The attention weights, $\alpha_{i,j}$, are then utilised to perform a weighted aggregation of the linear transformation of the neighbouring agents' observations in Equation (3.13). Functionally, the agent's policy could then be considered as $\pi_i : O_i, H_i \rightarrow A_i$. Where H'_i is the set of possible messages derived from Equation (3.13).

3.3.8.2 Results: Comparison with MAAC

In Figure Figure 3.9 a comparison between commMAAC and MAAC is provided within the complete network topology. Despite the introduction of the MHDPA, there is no observable improvement in reward acquisition. This was initially surprising considering the significant

partial observability within the environment. However, it is perhaps plausible that the interleaving policy acquired in the previous experimentation is significantly easier to derive rather than a communicative policy. The challenges of communicative DMARL are further explored in Chapter 5.

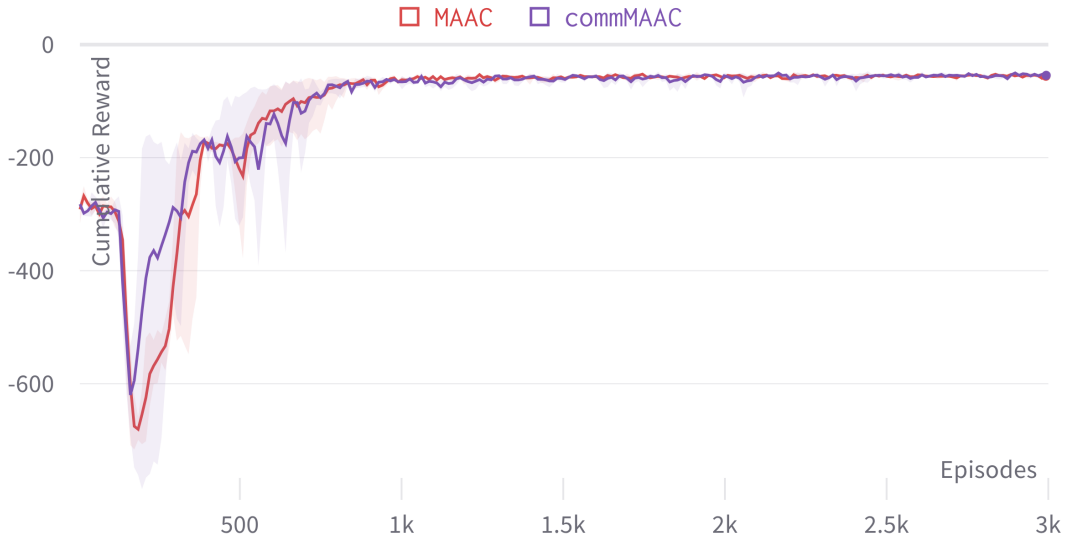


Figure 3.9: Cumulative reward comparison between MAAC and commMAAC.

3.3.9 Summary

This work models the challenge of finding a multi-asset, multi-component maintenance policy as a POSG and proposes a number of DMARL algorithms to solve it. The developed methods, MAAC and convMAAC took advantage of critic centralisation to improve training efficacy. They were demonstrated to outperform heuristics and an IAC baseline in a representative RAN maintenance planning cases study. The best-performing method, convMAAC, achieved a network availability increase of 3.49% and 6.13% in the complete and star topologies over a preventive maintenance baseline whilst maintaining similar levels of network maintenance. The utilisation of a GCN within the critic of convMAAC significantly reduces the number of parameters necessary in the critic when compared to MAAC. It is argued that this would be practically relevant from a computational perspective when extending to larger networks.

Moving this work towards deployment requires tackling a number of problems. The most pressing of which is the availability of a simulator for training. Acquisition of historical data from equipment vendors may be difficult due to its commercial sensitivity. There are also numerous opportunities to explore various algorithmic innovations to deal with particular challenges which this domain offers. Currently, policies are learned at the asset level. Methods that enable decomposition into component-level representations may be preferable if the particular domain

involves assets which are constructed from a finite number of modular components. It may also be possible to integrate a supervisory agent which improves upon the random allocation of repair resources currently utilised.

3.4 Conclusions

This chapter considered the application of DRL within telecommunications networks. Observing the highly distributed nature and scale of the underlying system, the suitability of DMARL was proposed. It was envisaged that algorithms which support decentralised learning, like DMARL, would be better suited towards the innate characteristics of telecommunications networks than centralised and naive single-agent approaches.

We began by exploring the application of single-agent DRL to "Resource Assignment in O-RAN" and demonstrated a novel solution inspired by AlphaGo Zero [4]. The derived solution improves resource utilisation by 5.7% over our strongest baseline. However, as described within Section 3.2, issues arise when this methodology is extended to large-scale communications networks. As introduced in Chapter 1, there is a spectrum of potential solutions including centralised DRL, naive single-agent DRL and DMARL. We argue that both centralised and single-agent DRL are impractical due to their challenges with dimensionality and understanding interdependencies between agents, respectively. Employment of DMARL overcomes these issues and we demonstrate its application to "Network Maintenance Planning". We introduce an GCN-based centralised critic within our algorithm convMAAC which improves training performance and scalability over IL methods and standard FCN-based centralised critics. This method was demonstrated on two simulated RAN networks and was shown to improve network availability by 3.49% and 6.13% over a heuristic baseline.

Our analysis supports the hypothesis that DMARL can facilitate the derivation of cooperative policies in challenging high dimension problems. However, there are many challenges which remain on the path towards a practically deployable method in telecommunications. Standard DMARL (and DRL, more generally), require simulators for training. The disparity between simulation and real-world environments is problematic and is the focus of research within areas like Sim-to-Real. There is also the natural propensity of the real world to change and evolve over time. This is known as environmental non-stationarity (or concept drift) and is a fundamental challenge which must be overcome. As we will learn in Chapter 4, these challenges are general across many industrial domains and their eventual mitigation is essential.

CHAPTER 4

MULTI-AGENT REINFORCEMENT LEARNING IN LOGISITICS

In this chapter we consider our second domain, logistics. Through doing so, we hope to develop an insight into the challenges of application within this domain and to enable us to distil more general insights regarding the applicability of DMARL. Herein, we consider the prospect of leveraging advances in DMARL to improve the efficiency and flexibility of order-picking systems for commercial warehouses. We envision a warehouse of the future in which dozens of mobile robots and human pickers work together to collect and deliver items within the warehouse. The fundamental problem we tackle, called the *order-picking problem*, is how these worker agents must coordinate their movement and actions in the warehouse to maximise performance (e.g. order throughput) under given resource constraints. Established industry methods using heuristic approaches require large engineering efforts to optimise for innately variable warehouse configurations. In contrast, the MARL framework can be flexibly applied to any warehouse configuration (e.g. size, layout, number/types of workers, item replenishment frequency) and the agents learn via a process of trial-and-error how to cooperate with one another. This chapter details the current status of the R&D effort initiated by Dematic[1] and the University of Edinburgh towards a general-purpose and scalable MARL solution for the order-picking problem in realistic warehouses.

The author helped to further advance the state-of-the-art in this challenging domain. The project had been in progress for approximately 3 years and had seen the development of a warehouse simulator and a hierarchical shared network actor-critic algorithm. The author built upon these existing breakthroughs aiming to further improve the competitivity of the DMARL approach. The author's contributions include, (1) further reducing the execution time of the algorithm and simulator which is necessary for experimental iteration, (2) model optimisations

[1] Dematic is a multinational company specialising in materials handling systems and logistics automation.

including introducing GAE, network architecture optimisations, and reward shaping (3) development of a policy visualisation based upon similarity to existing heuristics, and, (4) identification of the limitations of the existing approach. The text in this chapter is an adaptation of the work within [19].

4.1 Introduction

An order received by a commercial warehouse operator may comprise of several order-lines which specify a required item and a quantity. *Order-picking* is the process of retrieving these items from the warehouse and delivering them to a target location within the warehouse for further handling [176]. In its most basic form, a human worker will receive an order and travel around the warehouse with a cart and pick the required items manually. The general objective is to minimise the time required for order completion. This process typically represents a significant proportion of a warehouse's operational costs, where figures in the region of 55% are commonly quoted [177]. As such, order-picking has attracted significant automation efforts in pursuit of reducing operational costs and thereby improving commercial competitiveness.

Automation efforts have generally focused on the *pick-to-picker* paradigm, in which large-scale autonomous systems move items to pickers for dispatch to customers. There are numerous examples of these types of systems, including the Dematic Multishuttle[2], Autostore[3], and KIVA [178]. Typically, these approaches require significant capital investment and can be costly to adjust to varying warehouse capacity and consumer demand. For these reasons, adoption is generally limited to larger operations. An alternative and more common approach is the *picker-to-pick* paradigm, in which pickers move to item locations within the warehouse and directly retrieve them for dispatch. This paradigm accounts for the majority of warehouse operations, where [179] estimated that 80% of warehouses in Western Europe followed this paradigm. Despite its prevalence, automation is comparatively less mature in this domain.

In this work, we consider the augmentation of the picker-to-pick paradigm with robotic vehicles such as automated guided vehicles (AGVs) and autonomous mobile robots (AMRs). We believe their increasing affordability and capability provides an opportunity to develop an incremental pathway towards higher productivity and full warehouse automation. We refer to AGVs and AMRs interchangeably, as our solution is not strictly dependent on the type of vehicle being used. The general idea of AGV-assisted order-picking has begun to receive attention in the academic literature [154–156]. Typically, this involves the decoupling of a traditional picker's role into order transportation and item picking, where transportation is handled by the AGV and picking is handled by a human worker or robotic picker. This approach has multiple benefits: (1) pickers do not need to travel back to the depot to complete an order, thereby minimising unproductive walking time, (2) the integration of AGV technologies into existing warehouses

[2] https://www.dematic.com/en-au/products/storage
[3] https://www.dematic.com/en-au/products/storage/autostore

requires minimal modification to existing infrastructure, and (3) the system can support scaling with varying demand by changing the number of AGVs and pickers.

Simulation experts at Dematic[4] have practical experience in developing various order-picking strategies in the *picker-to-pick* paradigm through heuristic methods. They find that their application and optimisation for innately variable warehouse configurations and optimisation targets require significant engineering effort. Ideally, the derivation of optimal methods for worker control should be an automatic process. Reinforcement learning (RL) offers this capability, having achieved notable successes in a number of complex real-world domains [89, 106]. Order-picking by its very nature is a multi-agent problem, thus we leverage multi-agent RL (MARL) which extends RL to multi-agent systems [48]. An important benefit of MARL is its flexibility to operate with diverse warehouse and worker specifications as well as optimisation objectives (e.g. order throughput, battery usage, travel distance, traffic and congestion, pallet stability, labour cost), where existing heuristic approaches would require significant engineering effort to fit different specifications.

This chapter details the current status of the R&D effort initiated by Dematic and the University of Edinburgh towards a general-purpose and scalable MARL solution for the order-picking problem. We developed a high-performance simulator which is capable of representing real-world customer operations and is optimised for efficient RL training and testing. We further developed a MARL approach, Hierarchical Shared Network Actor-Critic (HSNAC), which improves sample efficiency over Shared Network Actor-Critic [64] through enabling decomposition of the large action space via a multi-layer hierarchy. HSNAC outperforms a well-established industry heuristic, achieving a 23.2% improvement in order-lines per hour while being generally applicable to different warehouse specifications. We outline a path towards the deployment of our solution, identifying a number of limitations and promising methodologies to further improve the performance and realism of our solution, and consideration for its integration into a comprehensive machine learning pipeline.

4.2 Related Literature

AGV-assisted order-picking [154] model the order-picking problem as a queuing network and explore the impact of different zoning strategies (no zoning and progressive zoning). They then further extend their method by representing it as a Markov decision process and consider dynamic switching based on the order-profile using dynamic programming. [155] consider an AGV-assisted picker and provide an exact polynomial time routing algorithm for a single-block parallel-aisle warehouses. [156] consider a warehouse partitioned into disjoint picking zones, where AGVs meet pickers at handover zones to transport the orders back to the depot. They propose a heuristic for effective order-batching to reduce tardiness. We additionally acknowledge

[4]Dematic is a multinational company specialising in materials handling systems and logistics automation.

pick-to-conveyor solutions[5] which we consider as a special case of AGV-assisted order-picking. Our work differs from existing work in this area as it does not place any restrictions on how workers may cooperate. To the best of our knowledge, our work is the first application of MARL to order-picking in the picker-to-pick context.

Multi-Agent Pickup and Delivery Problem MAPD problems [180] consider a set of agents that are sequentially assigned tasks. Each task requires the agent to visit a pickup location and a delivery location. The locations of agents and tasks are typically presented as nodes in an undirected connected graph. The agents move between locations via the graph's edges, where the edge capacity is restricted so that only one agent may move along an edge. The objective is to minimise the time duration required for task completion, which may be further broken down into task assignment and the planning of collision free paths. This formulation has been applied to several problems in warehouse logistics [180–182]. The decoupling of workers within our approach introduces complex interdependencies between the paths of different worker types which significantly complicates the problem. [183] have introduced the Cooperative multi-agent path finding problem which is applicable within this domain but requires explicit specification of the workers required to cooperate and does not allow for optimisation over extended temporal periods. We consider MARL as an alternative due to its generality and flexibility.

Multi-agent reinforcement learning MARL algorithms are designed to train coordinated agent policies for multiple autonomous agents, and have received much attention in recent years with the introduction of deep learning techniques into MARL [48]. MARL has previously seen application to various warehousing problems, including Shared Experience Actor-Critic to pick-to-picker systems [49, 64], and a deep Q-network variant for sortation control [184]. For the specific complexities of the order-picking problem, we consider methods at the intersection of MARL and hierarchical RL (HRL) to enable action space decomposition and temporal abstraction. This combination has been studied by [185] who derive MARL algorithms for macro-actions under partial observability, and [15] who propose Feudal Multi-Agent Hierarchies (FMH) which extends Feudal RL [186] to the cooperative MARL domain. We apply a 3-layer adaptation of FMH to a partially observable stochastic game with individual agent reward functions.

4.3 Background

We consider a scenario in which a warehouse operator (customer) seeks to improve the efficiency of their warehouse, $\mathcal{W}$, through automation of order-picking. The task requires optimal utilisation of the customer's resources to maximally improve warehouse operations, measured by key performance indicators such as pick rate (order-lines/hour), order lead time (seconds), and distance travelled (metres).

[5] www.dematic.com/en-gb/products/case-and-piece-picking

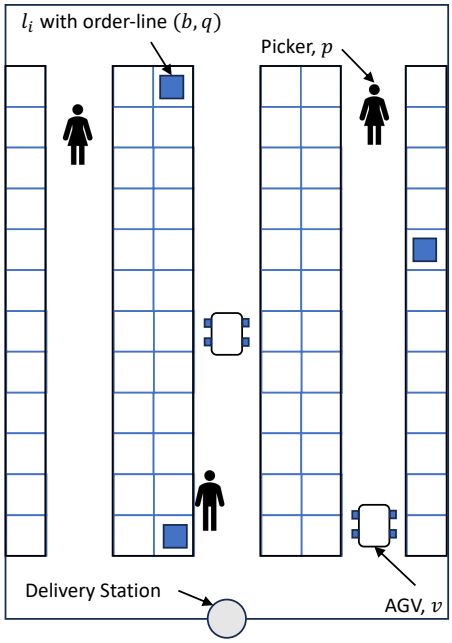


Figure 4.1: Simple illustration of warehouse with labels.

4.3.1 Warehouse Definition

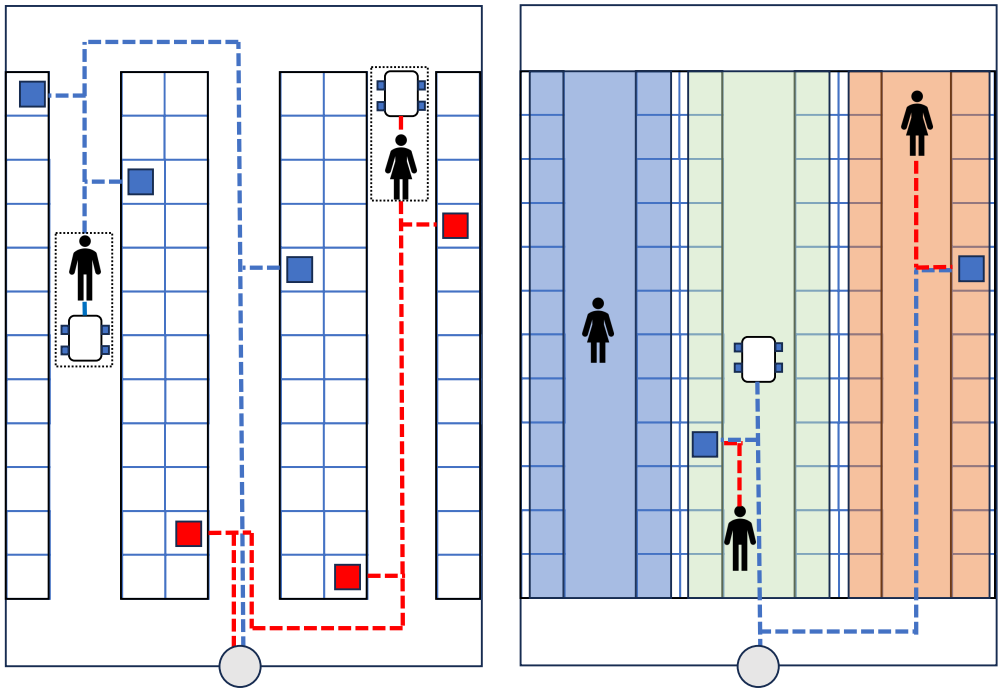
We consider a customer warehouse defined by the 3-tuple $\mathcal{W} = \{L, Z, W\}$. An illustration is provided in Figure 4.1, where key items are labelled.

- L refers to the set of spatially distributed locations within the warehouse and can be further broken down into $L = L_i \cup L_t$, where L_i and L_t refer to the set of locations with stored items and other locations (such as idle locations and order delivery stations), respectively.

- Z defines the order distribution which is dependent on the warehouse's supplier and customer behaviour and is assumed to be known. An order $z = \{(b_0, q_0), \ldots, (b_n, q_n)\}$ is sampled from Z. Each pair (b, q) represents an order-line, where b represents the item and q specifies the required quantity.

- W is the set of workers and is comprised of AGVs, V, and pickers, P, where the workers in each set are homogeneous. AGVs are assigned orders sampled from Z with z^v denoting the current order of AGV $v \in V$. Successful picking of an order-line requires coordination between AGVs and pickers.

For a given warehouse, we seek to derive a joint policy π which defines the behaviour of all workers in W such that we maximise the average pick rate K, formally denoted with $\pi = \arg\max_\pi K(W, \pi)$. A key desideratum of our solution is to automatically learn optimal policies for any given warehouse configuration and order profile.



(a) (b)

Figure 4.2: Diagrams illustrating warehouse heuristics behaviours. (a) shows Follow Me, and (b) shows Pick Don't Move.

4.3.2 Heuristic Solutions

Two heuristics which Dematic use within this setting are *Follow Me* (FM), and *Pick, Don't Move* (PDM).

Follow Me A number of AGVs are assigned to each picker and will follow them through the warehouse, the case of a single picker being assigned to an AGV is shown in Figure 4.2a. Each AGV's order is concatenated and the travelling salesman problem (TSP) solution is generated to determine the order in which the items will be picked. This improves efficiency over a traditional picker with order cart approach as the AGV can leave the picker and deliver the completed order to the packing area. FM minimises idle time for pickers, as it ensures that they are always travelling or picking, but also leads to more travelling of pickers than needed.

Pick, Don't Move Pickers are allocated to zones (such as a picker per aisle) in the warehouse which they are responsible for, while AGVs are allowed to travel throughout the entirety of the warehouse. This is depicted in Figure 4.2b, where the zones are indicated by coloured strips within the warehouse.The AGVs travel through the list of locations they need to visit to complete their order using a TSP solution. Once an AGV goes into a picker's zone, the picker is responsible for meeting the AGV at the item location, and picking relevant items to the relevant orders onto

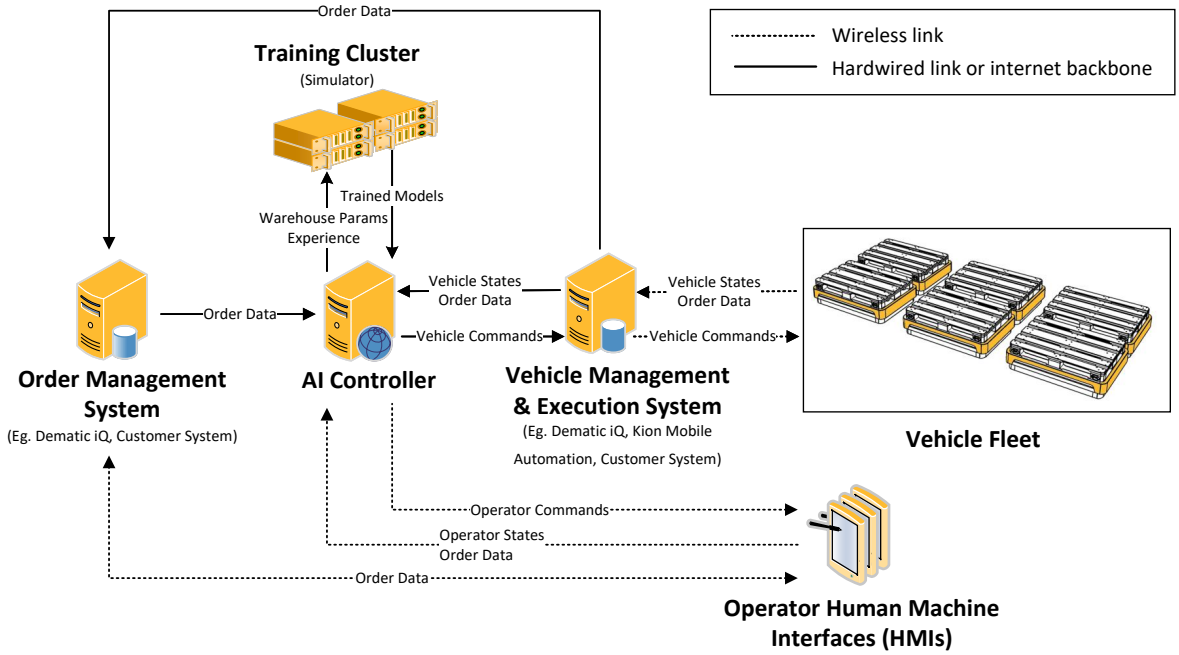


Figure 4.3: Systems architecture for our solution.

the AGV. Pickers prioritise service of AGVs by the relative proximity of the AGV and picker to their target locations. PDM minimises travel distance for pickers, allowing them to spend more time picking, and less time moving. However, it may result in under-utilisation of pickers in case there are few items within current orders in their operating zones.

4.3.3 Key Challenges

The efficacy of optimised heuristic methods is context and customer-dependent, requiring consideration of many factors such as the warehouse item clustering strategy, order profile, order prioritisation mechanism, changes in demand and supply, changes in labour and workforce conditions, and regulatory factors to name but a few. A heuristic strategy needs to be selected and repeatedly tuned with this ever-changing context. The required iterative process necessitates a regular engineering effort in consultation with a customer to review, analyse and tune performance of heuristics and parameters. This engineering burden motivates the automation of worker policies for the order-picking problem under the consideration of the following challenges:

Scalability We desire a general-purpose solution which can handle variations in multiple dimensions, including the number of total item locations $|L|$, the order distribution Z and the number of workers $|V| + |P|$. Controlling all workers with a single decision-making entity quickly becomes infeasible due to the joint action space growing exponentially with the number of workers.

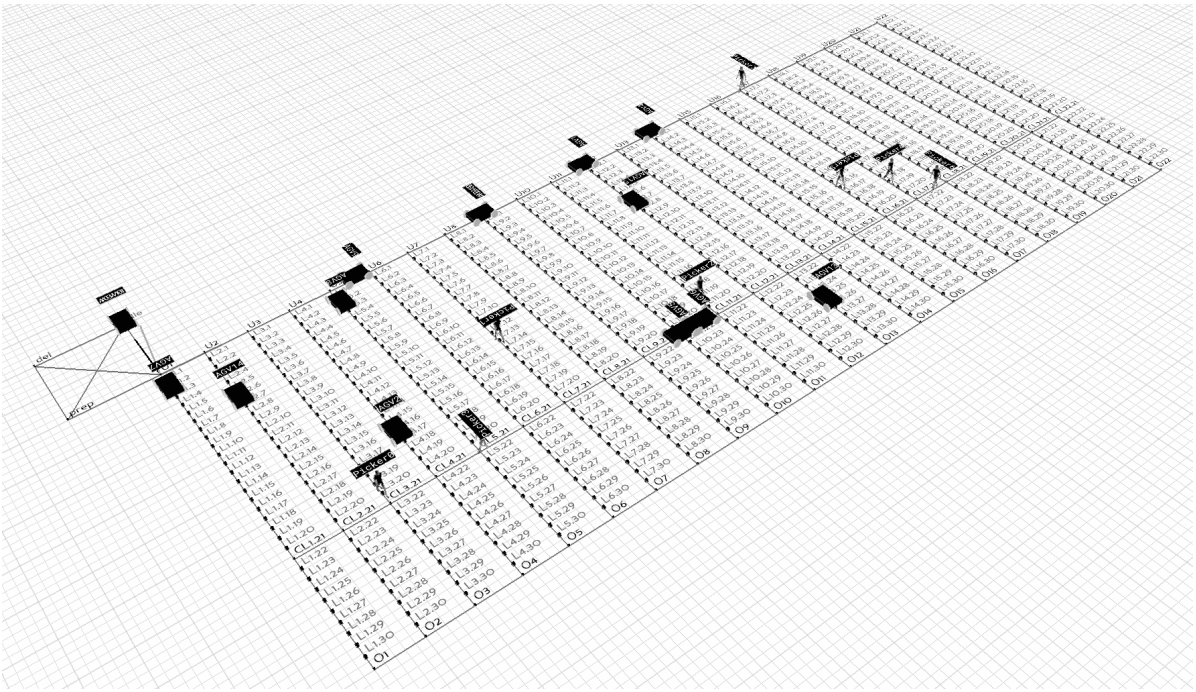


Figure 4.4: 3-D warehouse simulator. Snapshot showing human and AGV workers moving along aisles.

Hence, we consider MARL approaches in which pickers and AGVs are modelled as individual agents.

Development environment MARL algorithms require significant numbers of exploratory interactions within the environment [49], involving prolonged periods of sub-optimal behaviours within the target warehouse configuration. This is unacceptable for warehouse operators and, therefore, mandates the utilisation of a high-performance simulation platform as an essential part of the development pipeline.

Productionisation Any derived solution does not exist in isolation, but as part of a pipeline and a larger warehouse system. As such, we propose a minimal viable system architecture in Figure 4.3. The diagram identifies how we envisage our systems interacting. MARL algorithms will be trained on a training cluster. Algorithm inference runs on an AI controller, which must be deployed on-premise, as warehouse environments cannot tolerate operational downtime. Generated commands are transmitted to workers through an on-premise wireless communications network. For a robotic vehicle, this command is given via the vehicle management system and executed by the vehicle, and for a human worker, this command is provided on a mobile device or headset. Feedback is provided to the systems by the picker upon successful pick via voice or scanned barcode.

4.4 Warehouse Simulator

To facilitate efficient MARL training, experts at Dematic developed a high-performance warehouse simulator which is capable of representing real-world customer warehouses and includes implementations of the baselines described in Section 4.3.2. Figure 4.4 shows an example snapshot of a simulated warehouse used in our experiments.

4.4.1 Implementation

The simulator was developed in Python 3.9 using the Panda3D game engine to enable visualisation and implements the OpenAI gym interface [80]. We introduce several assumptions to reduce the simulator's computational requirements. These are listed below, and will be relaxed in future work.

- **Warehouse scaling** We model the warehouse at a 1:3 scale and all workers travel at a speed of 1.66 meters per second. By scaling the distances, we enable faster transit and consequently faster training times.
- **Worker commitment** When a worker chooses to travel to a location, it commits to its execution until arrival.
- **Collisions** We do not model worker collisions, assuming that they are able to move past each other without any delay or penalty. We give the vehicle management and execution system the responsibility of collision avoidance.
- **Automatic loading** Picking of items from a picker onto an AGV is done automatically and instantaneously once both workers are at the respective location. The impact of the quantity in an order-line is considered to be negligible and as such we set it to 1.
- **Fixed workers** The number of workers within the system is fixed and unchanging. Workers do not experience failures, nor does their productivity decrease over time.
- **FIFO order assignment** Orders are assigned to AGVs following a first-in-first-out queue system. Optimisation of order assignment is left for future work.

4.4.2 Simulator Optimisation

Major effort was directed towards improving the execution speed of the simulator. Through optimisations, we achieved a speed-up of 13021x (from 0.003643 steps per second to 47.51 steps per second collected over 300 samples), for a warehouse consisting of 1276 item locations, 24 agents, and running 4 environments in parallel. The most impactful modifications we made are detailed below.

Time progression in game engine We generate frames at 0.2 FPS (frames per second), such that 5 seconds in game time passes on each game engine step (and equivalently, each algorithm action step). The relatively low FPS rate impacts our worker action frequency, resulting in some minor performance degradation, but was chosen to make RL training within a reasonable time-frame tractable.

Precalculation and caching shortest paths As we shall describe within Section 4.5.2, agents select their target locations as actions. Following such a movement decision, we calculate the shortest path between the agent's current and target location. To facilitate this computation, a warehouse is represented as a graph and shortest paths computed using the Dijkstra's algorithm [187] are cached in a table. This table uses significant memory for reasonable warehouses (in the order of 500MB for 1200 locations), but path determination becomes a hash-table lookup of $O(1)$.

Boosting execution time for distance calculations Calculating distances from (x, y) coordinates to graph nodes is a frequent computation which scales with $O(|W||L|)$. To improve this operation, we use the following elements: (1) a graph coordinate hash-table for exact matches, (2) a compiled vectorised function which translates the operation to machine code, and (3) a KD-tree [188] of the graph coordinates. We see a significant improvement in execution time for an insignificant amount of memory usage.

4.5 Multi-Agent Reinforcement Learning

We use MARL to train coordinated agent policies in simulation. This section details our model and algorithm design.

4.5.1 Problem Modelling

4.5.1.1 Partially observable stochastic game

We model the multi-agent interaction as a partially observable stochastic game (POSG) for N agents [45]. A POSG is defined by the tuple $(\mathcal{I}, \mathcal{S}, \{\mathcal{O}^i\}_{i\in\mathcal{I}}, \{\mathcal{A}^i\}_{i\in\mathcal{I}}, \mathcal{P}, \Omega, \{\mathcal{R}^i\}_{i\in\mathcal{I}})$, with agents $i \in \mathcal{I} = \{1, \ldots, N\}$, state space $\mathcal{S}$, and joint action space $\mathcal{A} = \mathcal{A}^1 \times \ldots \times \mathcal{A}^N$. Each agent i only perceives its local observations $o^i \in \mathcal{O}^i$ given by the observation function $\Omega : \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{O})$ with joint observation space $\mathcal{O} = \mathcal{O}^1 \times \ldots \times \mathcal{O}^N$. The transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{S})$ returns a distribution over successor states given a state and a joint action. Agent i receives a reward r^i_t at each step t defined by its individual reward function $\mathcal{R}^i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$. The goal is to learn a joint policy $\pi = (\pi_1, \ldots, \pi_N)$ to maximise the discounted return $G^i = \sum_{t=1}^{T} \gamma^{t-1} r^i_t$ of each agent i with respect to the policies of other agents; formally, $\forall i \in \mathcal{I} : \pi_i \in \arg\max_{\pi'_i} \mathbb{E}\left[G^i \mid \pi'_i, \pi_{-i}\right]$ where $\pi_{-i} = \pi \setminus \{\pi_i\}$, and γ and T denoting the discount factor and episode length, respectively.

4.5.1.2 Action space

The completion of orders requires pickers to be able to visit all item locations $l \in L_i$ and for AGVs to be able to visit all locations $l \in L$. We enable this by defining the action space of pickers and AGVs as $\mathcal{A}_p = L_i$ and $\mathcal{A}_v = L$, respectively. This results in a large action space and an action duration that is dependent on the distance between an agents current and target location. These issues are addressed in Section 4.5.2.

4.5.1.3 Observation space

The availability of communication links between workers and the central servers affords a high degree of flexibility in modelling the information observed by agents. We aim to provide both classes of agents with sufficient information to make optimal decisions whilst pruning out unnecessary variables. Picker and AGV observations are defined in Equation (4.1) and Equation (4.2) with $\oplus$ denoting the concatenation operator. Pickers and AGVs observe the current and target locations of all agents, denoted $l_c^i \in L$ and $l_t^i \in L$ for agent i. Additionally, AGVs only observe their own order while pickers observe all orders.

$$O_p = \{(l_c^i, l_t^i) \mid i \in \mathcal{I}\} \oplus \{z^v \mid v \in V\} \tag{4.1}$$

$$O_v = \{(l_c^i, l_t^i) \mid i \in \mathcal{I}\} \oplus z^v \tag{4.2}$$

4.5.1.4 Reward function

Agents are rewarded for behaviour which is aligned with the objective stated in Section 4.3.1. The reward function for agent i as a picker and AGV, respectively, are given in Equations (4.3) and (4.4).

$$r_t^i = \begin{cases} 0.1, & \text{if } i \text{ completed pick at step } t \\ -0.05, & \text{otherwise} \end{cases} \tag{4.3}$$

$$r_t^i = \begin{cases} 0.1, & \text{if } i \text{ received picked item at step } t \\ 0.1, & \text{if } i \text{ completed order at step } t \\ -0.05, & \text{otherwise} \end{cases} \tag{4.4}$$

4.5.2 Hierarchical MARL for Order-Picking

In the current formulation, each agent's action space is very large with $|\mathcal{A}^i| \approx |L|$ and the actions may take different durations to terminate. To address these challenges and simplify the training

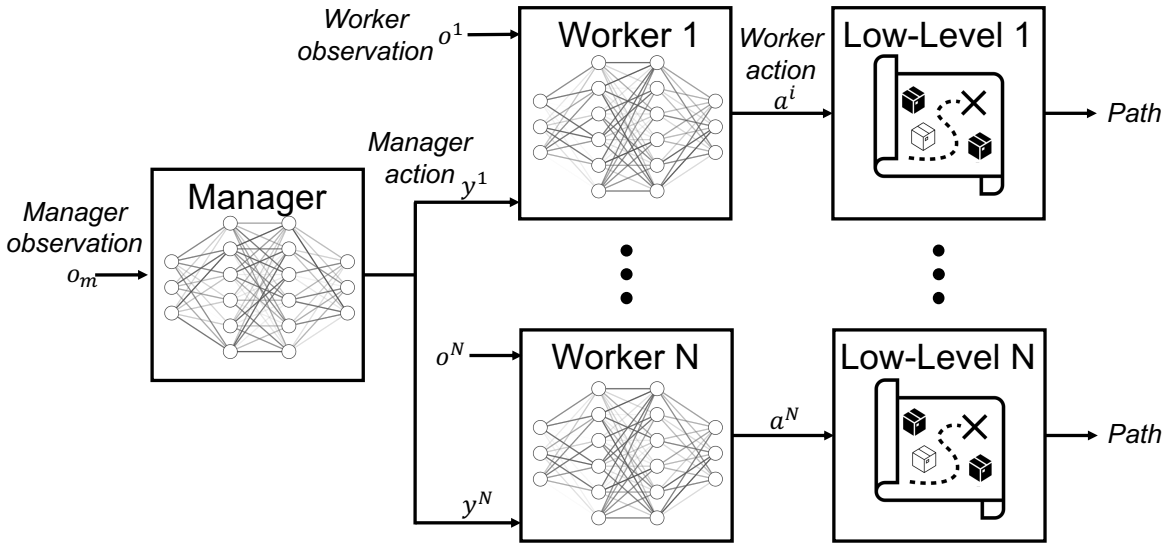


Figure 4.5: Diagram of 3-layer Feudal Hierarchy.

of agents, we utilise an adaptation of FMH [15], which involves the introduction of a manager that produces goals for subordinates to satisfy, shown in Figure 4.5. We apply this concept to partition the locations within the warehouse into a set of disjoint sectors Y, formally $L = \bigcup_{y \in Y} y$. The manager observes the current and target locations of all agents as well as the current orders assigned to each AGV, and determines a sector y^i for each agent $i \in \mathcal{I}$ to move to. Given the assigned sector, agent i's policy π_i selects its new target location $l_t^i \in y^i$ within its assigned sector. This decomposition greatly reduces the effective action space of each agent's policy which is now given by $\max_{y \in Y} |y| \ll |L|$. Once the target location of each agent is determined, a lower-level controller will then calculate a path from its current location and execute the necessary sequence of primitive actions.

We further reduce the size of the effective action space of agents through action-masking, based on the observation that each AGV $v \in V$ only needs to collect the items within their current order z^v. Therefore, AGVs should only move to locations within the warehouse which contain these items. This is achieved by masking out actions which refer to locations without items in the current order of an AGV v. Given that in expectation, $|z^v| \ll |L|$, this significantly reduces the number of actions for each AGV. For pickers, a reasonable action-masking approach is less clear cut. Empirically, we found that only considering picker actions which refer to current and target locations of all AGVs was effective.

The policy and value network of the manager are given by a multi-headed neural network comprising of three fully-connected layers consisting of 128 neurons each with ReLU activations, and a policy and value head for each agent. Each agent is parameterised by a value and critic network represented by two fully connected layer of 64 neurons with ReLU activations. This hierarchical model is trained using the Shared Network Actor-Critic (SNAC) algorithm [64]

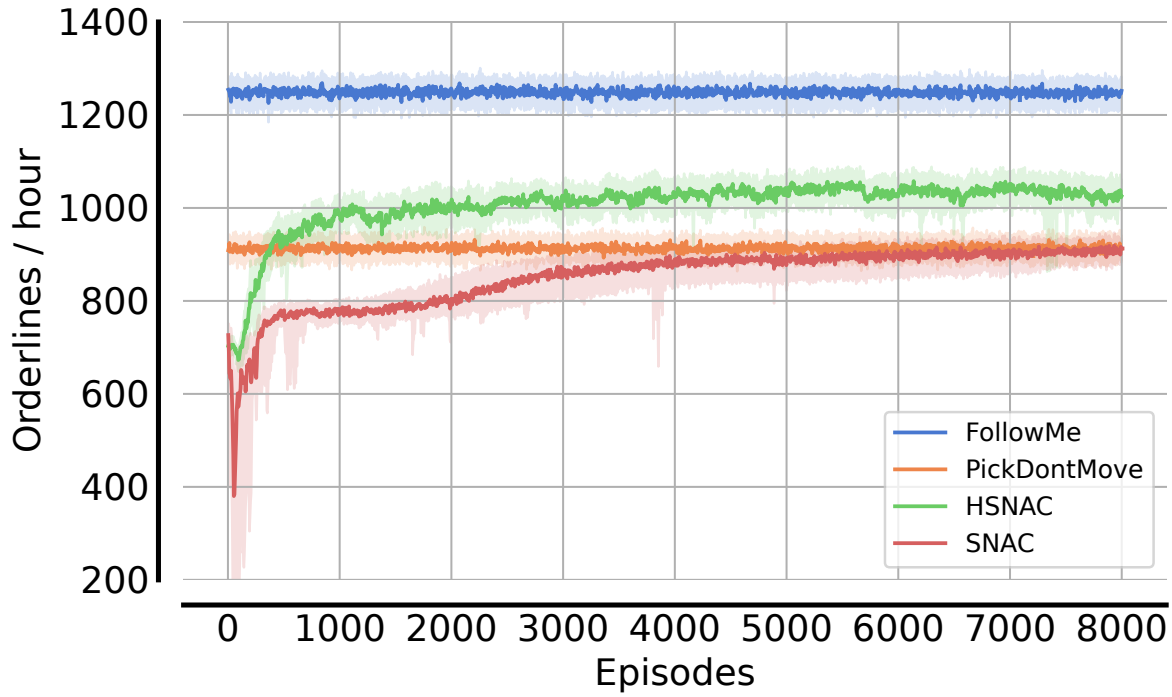


Figure 4.6: Average order-lines per hour of HSNAC, SNAC, and heuristics. Shaded area shows 95% confidence interval.

	AGVs			Pickers		
	Distance (m)	Idle Time (s)	Pick Rate	Distance (m)	Idle Time (s)	Pick Rate
HSNAC	1887 ± 3	779.48 ± 1.91	69.99 ± 0.10	2299 ± 6	688.44 ± 1.73	139.98 ± 0.19
SNAC	1975 ± 3	952.57 ± 2.10	63.80 ± 0.08	3678 ± 9	585.99 ± 1.28	127.59 ± 0.16
PDM	1529 ± 4	1028.55 ± 7.11	56.80 ± 0.21	1104 ± 5	1084.53 ± 7.36	113.61 ± 0.42
FM	1970 ± 5	514.13 ± 2.33	78.78 ± 0.25	1620 ± 6	582.49 ± 2.40	157.57 ± 0.49

Table 4.1: Performance comparisons between HSNAC (ours), SNAC, PDM and FM. Results show mean ± 95% confidence interval for distance travelled, idle time, and pick rate for both AGVs and pickers.

in which we share networks across pickers and AGVs, respectively, to improve the efficiency of the training process. The discount factor, γ, is set to 0.99 for all agents. We found that we can further improve performance by applying generalised advantage estimation (GAE) [18] and standardising the advantage estimate in each training batch to have zero mean and unit variance.

4.6 Empirical Evaluation

We test our approach in a warehouse comprising of 1276 item locations, with 16 AGVs and 8 pickers which is shown in Figure 4.4. The workers are presented with an episodic task consisting of 80 orders with an average length of 5 order-lines which are randomly distributed within L_i. We implement HSNAC in PyTorch [189] and allow it to train for 8000 episodes and 8 random

seeds. The partitioning, Y, is achieved through division of the warehouse into 22 equal-sized sections. We compare HSNAC against the PDM and FM heuristics (Section 4.3.2) and SNAC [64] which uses the same neural network architecture as the HSNAC worker agents. We also tried shared experience actor critic [64] and independent actor critic but omit them from our figures as they were not as effective as SNAC. We use pick rate in order-lines per hour as our primary performance measure, indicating the average frequency of picks in the episode. Experimentation was performed on three Ubuntu servers, two with 128 core CPUs and 256GB RAM, and one with 32 core CPUs and 128GB RAM, respectively, and experimental tracking was performed using WandB [92]. Full evaluation details with additional results and episode video recordings are available in an extended report [6].

Figure 4.6 shows the pick rate in order-lines per hour of HSNAC and all baselines across training. We observe that HSNAC achieves noticeably higher order-lines per hour compared to PDM, but is still under-performing compared to FM. Comparing HSNAC to SNAC demonstrates the advantage of the hierarchical architecture, with HSNAC converging significantly faster.

In Table 4.1, we show selected aggregated characteristics for 50 episodes for PDM and FM, and across all trained HSNAC and SNAC seeds. We find that HSNAC's improvement in pick rate over PDM is largely due to a reduction in idle time by 27.55% for AGVs and 44.68% for pickers, however it also incurs an increased travel distance of 20.96% for AGVs and 70.23% for pickers over the course of an episode. Upon closer inspection of agent behaviours, we were able to observe situations in which two or more pickers raced towards AGVs, which appears to demonstrate the emergence of competitive behaviours.

Analysis of video provides an insight into agent behaviours, but it is hard to appreciate statistical qualities of the agents' behaviours and also can be susceptible to bias. With this in mind, we propose a measure motivated by the diverse manner in which our heuristics employ pickers. We collect all pickers' order-line completion locations and then average and normalise them into a vector F. We can then compute the cosine similarity between all pickers' which we choose to represent as a heatmap. In Figure 4.7, we show how this looks for our heuristics. As expected, PDM exhibits very little overlap whereas FM order-completion locations appear to be much more uniform. We show plots for multiple seeds of HSNAC and SNAC in Figure 4.8. We observe a correlation between pickers which indicates that a variety of zone allocations may be emerging. In contrast to PDM, allocated zones are not served by a singular picker. HSNAC appears to allow a greater degree of flexibility, where two or more agents may serve the same zone of the warehouse, but may also move to other areas. SNAC does not derive this same behaviour.

[6] sites.google.com/view/scalablemarlwarehouse/

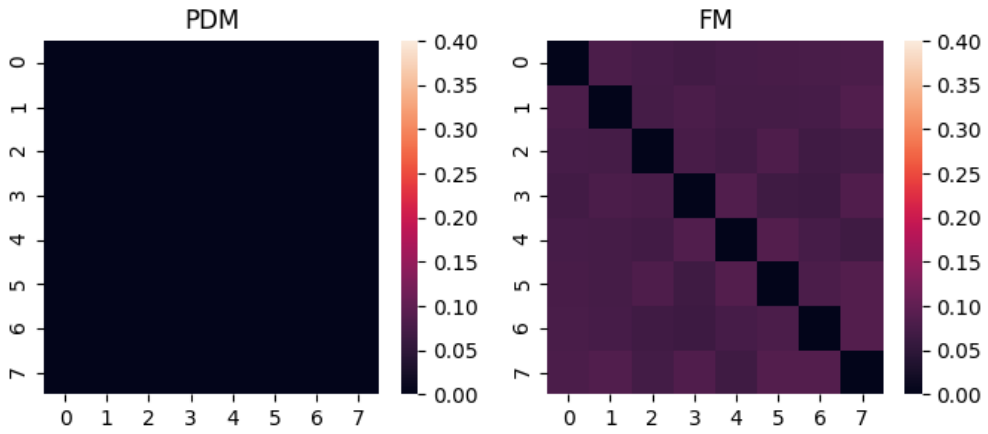


Figure 4.7: A figure showing the cosine similarity between average picker order-picking locations for the two heuristics, PDM and FM. It is clear from the figure that PDM agents do not share locations, whereas FM agents' are free to visit any location.

4.7 Conclusions

Our results support our hypothesis that MARL agents can derive effective solutions for the order-picking problem. It plausibly provides a mechanism by which we can avoid costly heuristic optimisation in the future by supporting learning directly through interaction within a simulated environment. This was the primary objective of the efforts detailed within this chapter. We achieved this through employing a number of methodologies, where the hierarchical decomposition of the action space had a large impact which we attribute to it being able to provide a solution to the order-picking problem at a lower spatial resolution than SNAC. Through HSNAC, we were able to outperform PDM which is a well-established industry heuristic. However, the manner in which it achieves this is not without fault. The increase in travel distance for pickers shown in Table 4.1 likely represents a notable increase in maintenance costs for robotic pickers and possible concerns for human picker welfare. This is likely attributable to the competitive behaviours we observed. We believe that these observations, in combination with the current supremacy of FM, demonstrate that whilst we are already in a region of commercial viability as we greatly exceed heuristic flexibility, there is more work to be done to achieve performance parity with the best performing heuristic.

The simulator (Section 4.4) forms the foundation of our approach, and provides a necessary trade-off between accuracy and execution speed in order to make training in a reasonable time plausible. In real-world settings, many of our assumptions may be challenged. For example, (1) loading may take a variable amount of time dependent on the required quantity or item type, (2) the number of workers may vary with shift patterns and (3) agent policy decisions may be able to happen quicker than the decision interval allowed. Despite this, even with relaxation of all our assumptions, transferring policies trained in simulation to the real world bring additional

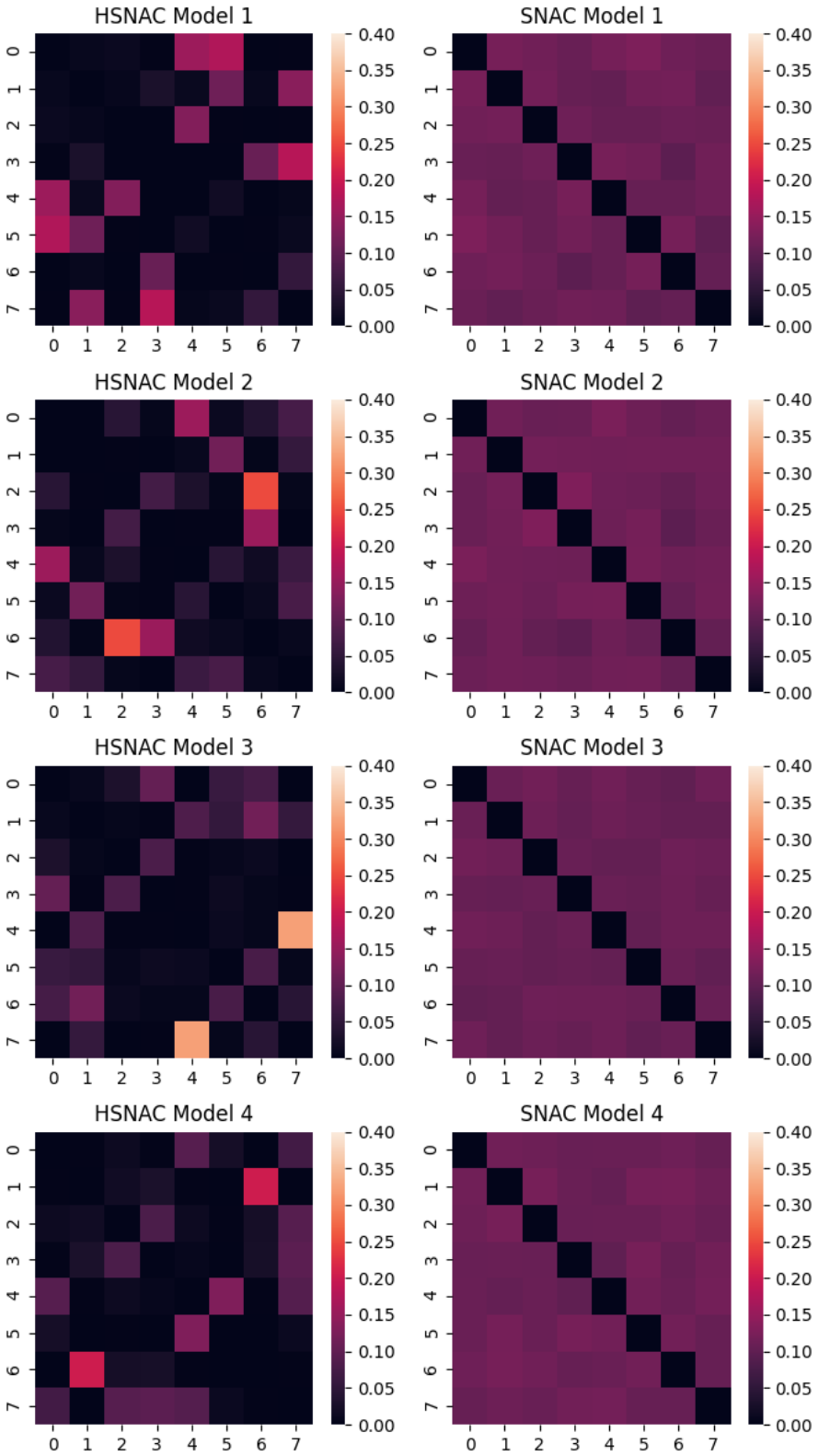


Figure 4.8: This figure shows the cosine similarity between the average order-picking location for HSNAC and SNAC. HSNAC exhibits behaviours which indicate that agents are collectively implementing zone allocations. This structure does not emerge for SNAC.

challenges. This is a well-known challenge within the RL community known as *sim-to-real* [93]. In addition to this, a warehouse does not exist in a vacuum and it needs to be adaptive to pressures that are placed on it by its supply chain. This is an issue of *concept drift* [190] and being able to handle its implication is essential.

Moving our solution towards production requires comprehensive analysis and mitigation of the limitations which we have identified. We plan to investigate methods to reduce picker competitivity, where we hypothesise that additional factors such as energy usage penalty and agent-to-agent communication may inhibit the emergence of these behaviours. We have begun some limited investigations into curriculum learning [191] as a method to allow for tightening of assumptions such as our decision interval, and plan to continue with this work and expand the scope to consider more variables. We have also begun to consider challenges associated with deployment. For example, upon our warehouse's order profile or other parameters drifting, we envisage opportunity for re-training within simulation. With suitable demand prediction models, we may even be able to undertake this proactively. A real-world proof-of-concept is within our project roadmap, and addressing hardware interoperability and sim-to-real challenges is critical for commercial viability.

CHAPTER 5

EMERGENT COMMUNICATION FOR COOPERATION

As we have demonstrated within this thesis, there are numerous ways to achieve cooperation among agents. In Chapter 3, we explored a network maintenance planning problem and found that convMAAC could achieve effective maintenance policies through repeated interaction. Chapter 4 proposed the application of DMARL to a warehouse order picking problem and demonstrated HSNAC which is a feudal method. Despite their demonstrable competence, these methods do not make use of communicative policies of a form comparable to those we introduced in Section 2.2.5. Within Chapter 3, we expected to be able to exploit communication to learn better cooperative policies. However, we found that there was no observable performance improvement over their non-communicative counterparts. This was unexpected considering the significant partial observability present in that environment. Its ineffectiveness raised fundamental questions for the author about the emergent qualities of inter-agent communications, which we explore in the following chapter.

Communication can provide a mechanism to promote cooperation, and its benefits are clearly observable in numerous biological systems [192, 193]. Its evidential importance in these systems creates a compelling narrative for its study within ML as a means to develop better cooperative systems [67]. As we observed within Section 2.2.5, numerous methodologies have been proposed which equip DMARL agents which the necessary apparatus to facilitate communication. In that section, we propose a dichotomy that divides the approaches into two categories based on their formulation's allowance for gradient propagation across the communications channel.

Within this chapter, we delve into the more restrictive non-differentiable setting which has been previously explored within [11, 73, 74]. This setting typically makes use of discrete communications channels and IL [8] which makes learning more challenging than their differentiable counterparts. However, it easily permits extension to situations where a model of an agent we

are communicating with is not available. This may be the case in human-centric settings and in industrial applications which require coordination between distinct commercial entities. This chapter intends to serve as an investigation into questions within inter-agent communications. There are numerous questions that can be asked. For example, (1) "When should we communicate?", (2) "Why do we communicate?", (3) "How should we communicate?" and (4) "Who are we going to be communicating with?". This is a mostly random set, and there are numerous other examples. All of which could themselves be formed into a full thesis. In this chapter, we provide investigations into (3) and (4), where we provide an overview of the work we undertake in the synopses below.

(5.2) Who are we going to be communicating with? When considering EC, or any communicative setting, it is important to identify whom we are communicating with. Is our conversational partner someone new or someone we are familiar with? We explore the second possibility through periodic interactions within populations of conversational agents. We demonstrate that catastrophic forgetting may be problematic with conventional approaches and propose an architectural modification which enables its mitigation. The section is titled "Who - Periodic Interactions in Populations of Agents" and is an adaptation of the work within [194].

(5.3) How should we communicate? Understanding redundancy To communicate, agents must have some functionality to transmit messages. Often for discrete communications, the number of messages made available to an agent is a design decision. This sub-chapter explores the impact this has on sample efficiency and generalisation. This section is titled "How - Impact of message dimension" is based upon contributions within [195].

This chapter is structured as follows. In Section 5.1 we introduce the main concepts, definitions and notation. Section 5.2 and Section 5.3 provide the results of our investigations into the questions of *who* and *how*. Finally, Section 5.4 serves as a concluding section and identifies future work which would may be interesting to undertake.

5.1 Preliminaries

Within this section, we introduce nomenclature, concepts and notations from DMARL and EC. We refer back to Section 2.2.1.1 which provides the necessary formulation to discuss communication within a stochastic game. We note that we may use s or l in place of the agent index i to refer to agents' which are speakers and listeners. For example, π_s and π_l are used to refer to the policy of a speaker or listener. Here, we will introduce the Referential Games (RG) in Section 5.1.1, and provide more details on learning to communicate in Section 5.1.2.

5.1.1 Referential Games

Referential Games (RG) [76] are common-place when studying EC [196]. They provide an environment where the agents' are required to learn how to communicate without additional environmental distractors. Within more complex environments, issues can arise ascertaining the benefit of communications [75] and hence we focus on this more simple and interpretable setting for our experimentation.

An RG involves two agents who are typically called the speaker, s, and the listener l. Each agent receives an observation $o^s \in O^s$ and $o^l \in O^l$ which is typically sampled from some dataset, D. The agents undertake a cooperative task that requires the listener, l, to output an action $a^l \in A^l$ which is dependent on both O^l and O^s. The only way to achieve this is by effectively utilising a (typically) discrete communications channel which permits the transmission of a message $m \in M^s$ from s to l. If the listener chooses the correct action both agents receive a reward of 1 and otherwise, they receive a reward of -1. An illustration of a RG which we utilise extensively in this chapter is given in Figure 5.1 which requires the listener to add together images from MNIST [197].

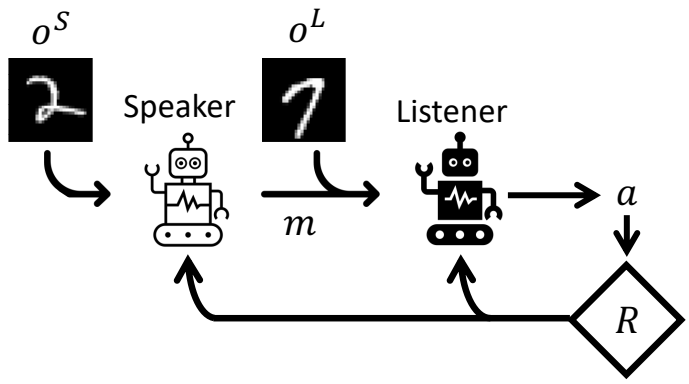


FIGURE 5.1. A Referential Game comprising of two agents, a speaker and a listener. This variant includes an MNIST-based game in which the listener must correctly add the observations of both s and l. As utilised within [73].

5.1.2 Learning to Communicate

Deriving communications protocols among *tabula rasa* IL with DRL is exceedingly difficult. The simplest objective function we can use is REINFORCE [20] shown in Equation (5.1). However, this does not typically work well due to the complexity of the communicative task [73]. In the context of the RG in Figure 5.1, it requires the speaker to provide salient information in the conveyed message and for the listener to correctly use it. If either of these mechanisms fails, no useful feedback is provided to either agent.

$$\nabla J(i,\theta) = \mathbb{E}_{\pi_{i,\theta}}\left[G_t \log \pi_{i,\theta}(A_t^i | O_t^i, \mathcal{M}_t^{-i})\right] \tag{5.1}$$

In order to improve the efficacy of communicative policies a range of methods have been developed [73, 74]. Within this chapter, we use a method developed within [73] which achieves SOTA performance in this setting as our base algorithm. In the following text, we provide an overview of its operation. In addition to utilising REINFORCE[20] to train the speaker and listener, biases are introduced into the agents' loss functions which promote *positive signalling* and *positive listening*. These were first introduced in [75] to measure communication and are defined in Definition 5.1 and 5.2 respectively.

Definition 5.1 (Positive Signalling)**.** An agent exhibits positive signalling if its messages are statistically dependent on either its actions or observations.

Definition 5.2 (Positive Listening)**.** An agent exhibits positive listening if its policy is dependent on the actions of another agent.

[73] adapted these ideas into explicit biases for the speaker and listener which are shown in Equation (5.2) and Equation (5.3) respectively. These can be shown to improve the emergent qualities of communication in this restrictive setting through improved coupling between the constituent elements of the system. Equation (5.2) is a more computationally stable version of mutual information. Where, $\overline{\pi_s}$ refers to the average speaker policy and is estimated empirically over a batch of experience, λ and $\mathcal{H}_{target}$ are hyperparameters. From an intuitive perspective, this loss encourages the speaker to learn a policy where observations and messages are correlated. Equation (5.3) intends to capture the concept of causal influence of communication as presented in [75]. Here, $\bar{\pi}_l$ refers to a modification of the listener policy which is not conditioned on the speaker's message and is trained to optimise the supervised cross-entropy loss given in Equation (5.4). This formulation encourages the listener to selection actions which are dependent on the received message.

$$L_{ps}(\pi_s^i, m_t^s, o_t^s) = -\mathbb{E}(\lambda \mathcal{H}(\overline{\pi_s}) - (\mathcal{H}(m_t^s | o_t^s) - \mathcal{H}_{target})^2) \tag{5.2}$$

$$L_{pl}(o_t^l, m_t^s) = -\sum_{a \in A^l} |\pi_l(a | o_t^l, m_t^s) - \bar{\pi}_l(a | o_t)| \tag{5.3}$$

$$L_{ce}(o_t^l, m_t^s) = -\sum_{a \in A^l} \pi_A^i(a | o_t^l, m_t^s) \log\left(\bar{\pi}_A^i(a | o_t^l)\right) \tag{5.4}$$

All agent policies within this chapter are optimised according to the methodology described within this subsection.

5.2 Who are we going to be communicating with?

On an average day, we may have the opportunity (or requirement) to converse with multiple different people. We note that the way in which we utilise language is highly dependent on our familiarity with our conversational partner. With a friend or family member who we may have a high degree of familiarity with, we are likely to have previously established language and conventions. This is an example of our capability to make use of information acquired within previous interactions to improve conversational efficacy in later interactions. Here, we note that much of the experimentation on communication in the IL setting has been restricted to small static populations of agents, where this typically involves 2-agents [11, 73, 74]. This restricted experimental setting does not allow for the study of more complex characteristics like those we have discussed in the text above.

In this more complex setting which we approximate as a population of agents who interact pair-wise periodically, we expect conventional methodologies to encounter difficulties. As these methodologies do not make use of parameter-sharing networks [8], it is conceivable that multiple unique languages may arise where these languages are unlikely to be compatible. As a result of this, any interaction with a new agent mandates the learning of a shared language. Without specific modifications to the agent's architecture, this new language will overwrite the previous one as a consequence of a known phenomenon within ML named catastrophic forgetting [198]. As the previous language has been lost, any interaction with the associated conversational partner will require re-training. We provide an informal illustration of this concept within Figure 5.2. Here, in order to address this issue, architectural modifications inspired by the Continual Learning literature [199] are used to extend the algorithm proposed by Eccles et al. [73]. Namely, multi-headed neural networks are used where a different head is maintained for each language. This paper formalises this concept and demonstrates it within a novel environment called *Communication Carousel* which extends a referential game to facilitate the study of this problem.

This section continues by providing a formal problem statement within Section 5.2.1. We then solution based upon a multi-headed DNN architecture in Section 5.2.2 and a novel experimentation platform within Section 5.2.3. Results and discussion are then provided in Section 5.2.4.

5.2.1 Periodic Interactions in Populations of Agents

Let us consider the existence of two sets of agents, where these are referred to as speakers $T_x = \{\pi_{s,0}, ...\pi_{s,n}\}$ and listeners $R_x = \{\pi_{l,0}, ...\pi_{l,n}\}$, respectively[1]. For some pairing of T_x to R_x, the agents' capacity to effectively convey information will be limited by their ability to understand one another. Over time, the agents can adapt to each other and arrive at an emergent protocol which maximises task reward.

[1]To avoid clashes with standard RL notation, the speakers and listeners sets have symbols consistent with transmitter and receiver.

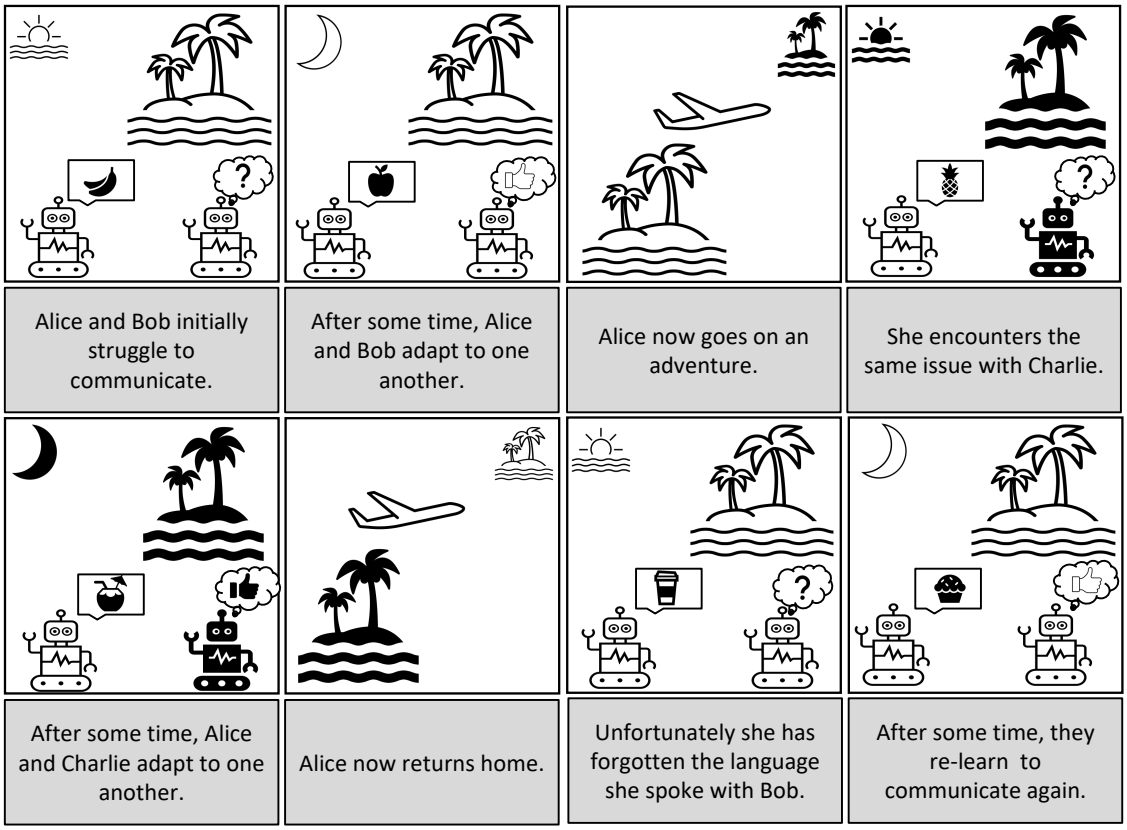


FIGURE 5.2. In larger populations of agents which interact periodically there is potential for many unique languages to emerge. When agents interact with speakers of a different language, they are likely to overwrite their previous language.

The first question this work intends to delve into is, what happens to their established emergent protocol when an agent (be that the speaker or the listener) interacts with a new partner? More formally, when the mapping from T_x to R_x is randomised and a period of training is allowed, how does this impact the agent's capacity for conversation with its previous partner? This problem exists within the continual learning setting, where Catastrophic Forgetting is known to be an issue [198]. It should be expected that as a pair of agents build up familiarity with one another, their previous languages will drift.

The fundamental issue with this mode of operation is that it always requires an agent to re-train upon interacting with a different partner even if they had previously arrived at an efficient protocol. Ideally, this should be avoided as this period of adaptation is costly. Naturally, the second question is simply, how can we mitigate this issue?

5.2.2 Multi-headed Neural Networks

As mentioned above, this primary issue in our scenario is Catastrophic Forgetting [198]. Following the naming convention from [199], our approach considers a simple parameter isolation method,

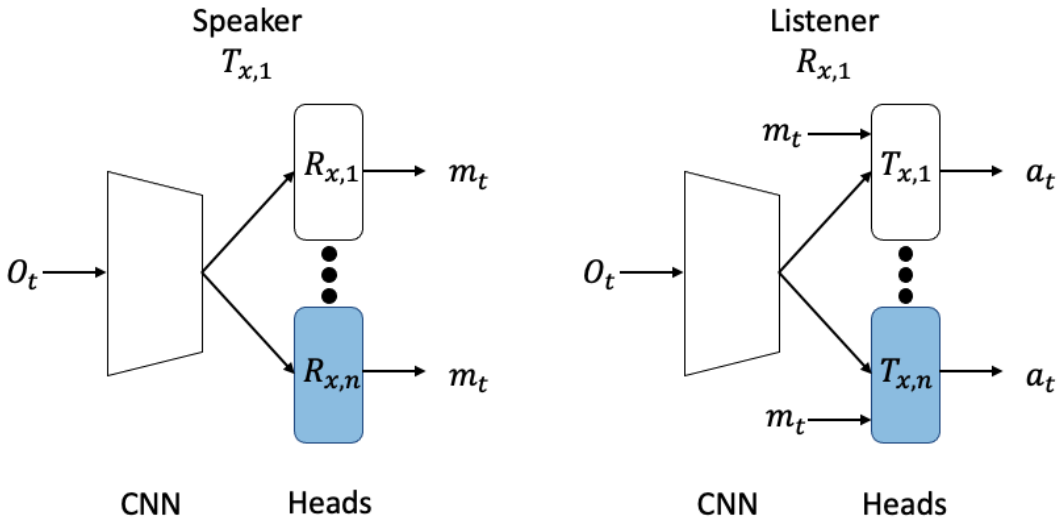


FIGURE 5.3. DNN architecture of speaker and listener shown on the left and right, respectively. The networks maintain a separate head for each possible partner, where the label indicates the conversational partner that it refers to. The white and blue colouring is representative of how gradients are allowed to propagate through the network. In both cases the CNN and first head are trained together, whereas the alternative heads are trained separately.

where each speaker and listener maintains a separate output head for each possible partner. This idea is based on [200]. It is assumed that the identity of each potential partner is observable and therefore the correct head can be chosen.

The architecture is presented in Figure 5.3, where the CNN for both the speaker and listener are only trained with the first partner. This decision is justified by the assumption that, in most cases, languages consider mappings from a similar set of concepts to different words or phrases and, as such, the features learned by the CNN for one language should be transferable. An additional variant upon this model is proposed in which the weights of the non-primary heads are pre-initialised with those of the primary head upon establishment of the first language. This can be demonstrated to improve sample efficiency when compared to random initialisations.

5.2.3 Communication Carousel

The intention of this work is to investigate the agents' capacity to maintain emergent languages after interacting with new partners. To achieve this, N-parallel referential games are instantiated and speakers and listeners are afforded E episodes with their initially assigned partner. After the initial E episodes, the agents are rotated and allowed the same number of episodes to interact with their new partner. This is illustrated in Figure 5.4. After a number of partner changes, ω,

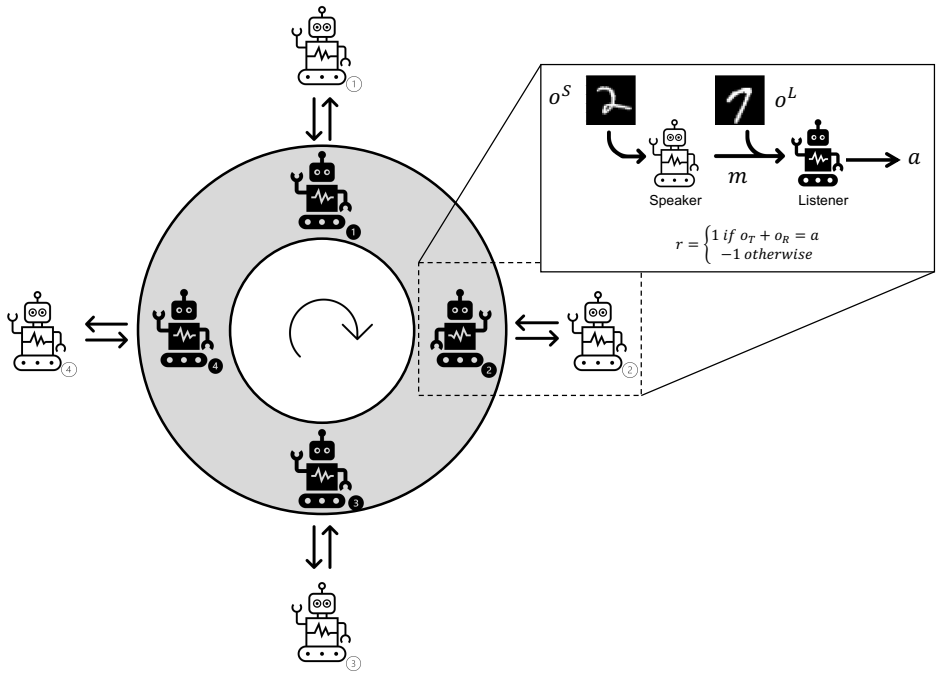


FIGURE 5.4. An environment to facilitate simulation of periodic interactions between agents. In this version, the agents' play a Referential Games (RG), but inclusion of other environments is straightforward.

the speakers and listeners are returned to their initial partner and afforded a further E episodes to reconverge. All experimental parameters are introduced in Table 5.1. This environment formulation provides a simple and interpretable test-bed for studying agent adaptation where the complexity can be easily controlled through appropriate selection of the referential game.

Table 5.1: Environment parameters in Communications Carousel.

Symbol	Value
N	4
E	75000
ω	1

All code is implemented in Pytorch [173] according to the methodology described in Section 5.2.2[2]. As previously introduced, the implementation of the speaker and listener follows the methodology described by [73], where we train agents independently utilising REINFORCE and utilise the same hyperparameters. As we were unable to achieve convergence with the defined architecture we made one modification. We introduced an extra layer into the DNN which alleviated this issue. This minor modification to the method proposed by [73] without the multi-headed output is utilised as a baseline within our experimentation.

[2]Code available at https://github.com/Jon17591/multi-lingual-agents

5.2.4 Results and Discussion

The results obtained support the hypothesis that the Multi-headed methods defined within Section 5.2.2 results in better maintenance of multiple emergent languages.

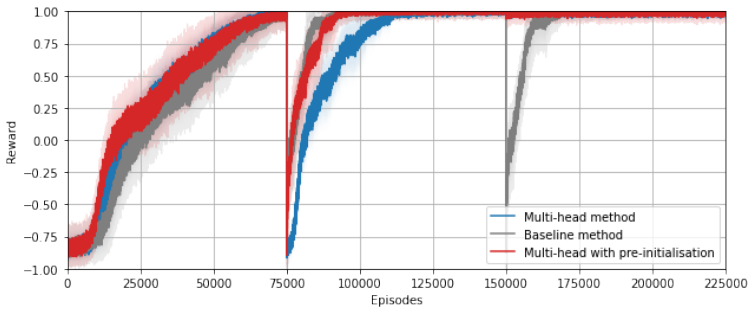
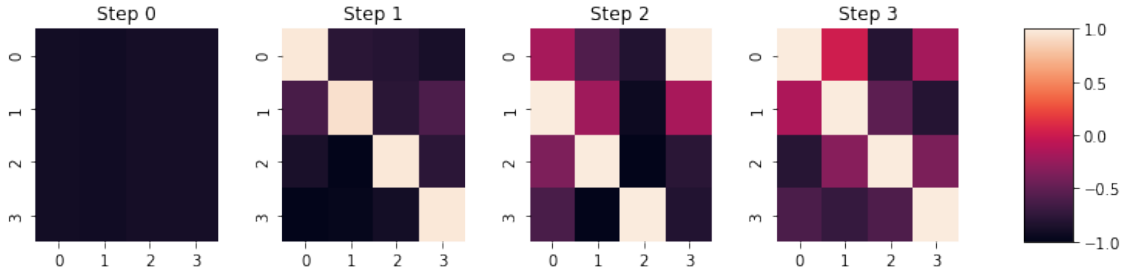


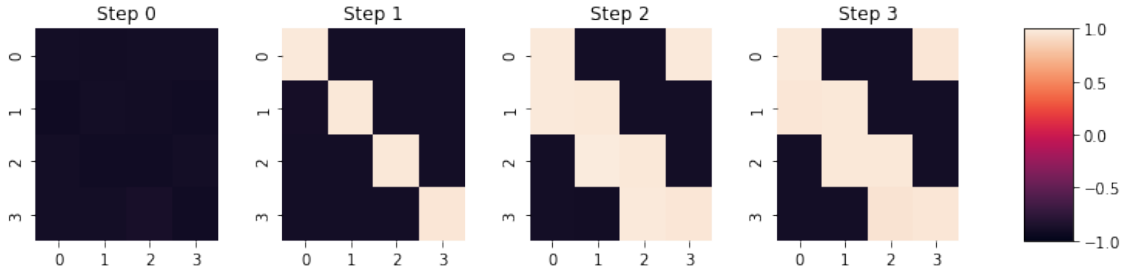
Figure 5.5: Average reward obtained by all 4-agents with their current partner. Partner is changed to a new partner at 75000 episodes and then to the original partner at 150000 episodes.

Figure 5.5 demonstrates the average reward which agents receive with their current conversational partner for the baseline, Multi-headed method and the Multi-headed method with pre-initialisation of the non-primary heads. The most notable observation to draw from this Figure is that the reward for the baseline method reduces substantially when it returns to the initial conversational partner at $150k$ episodes, this reduction is not present in either of the Multi-headed method. This would suggest that Catastrophic Forgetting has been avoided. This claim is further supported by Figure 5.6a, 5.6b and 5.6c. These figures show the average reward obtained by all pairings of speakers and listeners in the form of a heatmap. The steps refer to the beginning of training, after every partner switch and at the end of training where this corresponds to episodes 0, 75k, 150k and 225k in Figure 5.5. Note that the baseline method experiences a significant reduction in reward acquisition once it has trained with a new partner whereas this is not present in either of the Multi-headed methods.

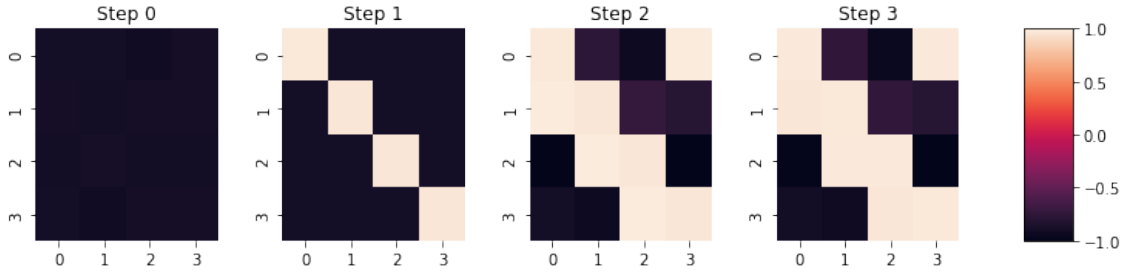
A drawback of the standard Multi-headed method appears to be the reduction in sample efficiency present when switching to the second partner (75k episodes) in Figure 5.5. It seems that the Multi-headed method takes longer to acquire the second language. This is as the additional heads are untrained and comprise of randomly initialised weights. The baseline method represents a policy that has converged to a solution. The entropy of both sets of speaker policies (shown in Figure 5.7) gives an indication as to why this occurs. It is clear that the Multi-headed method begins with significantly higher entropy. The introduction of this extra stochasticity may make the arrival at a common protocol more time intensive as there is less determinism to the respective messages and, as such, it is more difficult to achieve synchronisation between the agents. This can be overcome by pre-initialising the weights of each head with the solution of the primary head, thereby achieving comparable convergence speeds to the baseline.



(a) Heatmap for baseline method evaluated for all pairings at episodes=0, 75000, 150000 and 225000. Scale represents the average reward which is obtained over 100 episodes.



(b) Heatmap for Multi-head method evaluated for all pairings at episodes=0, 75k, 150k and 225k. Scale represents the average reward which is obtained over 100 episodes.



(c) Heatmap for Multi-head method evaluated for all pairings at episodes=0, 75k, 150k and 225k. The scale represents the average reward which is obtained over 100 episodes.

Figure 5.6: Pair-wise conversational partner efficacy for various methods.

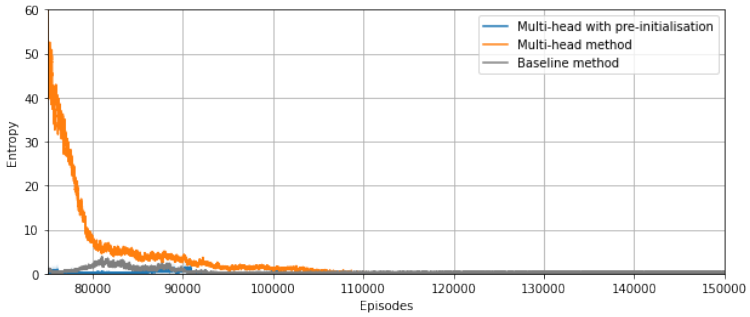


Figure 5.7: Entropy of speakers between 75k and 150k episodes.

5.2.5 Further Extensions

A current limitation of our method which we hope to address is that all derived languages are unique. The resulting multi-agent system has a quadratic relationship between the number of languages and the number of speakers/listeners. This is not the case in natural systems with the number of distinct languages being somewhat restricted. An interesting avenue to explore could consider methodologies which restrict the number of languages that may emerge thereby aiming to improve zero-shot performance.

Furthermore, although we focus on emergent communication in this work, we believe the results presented apply more generally to cooperative games. The idiosyncratic conventions which cooperative agents' can develop are equivalent to the languages which arise in referential games. In future work, we intend to expand our analysis to consider a broad set of cooperative problems where periodic interactions may arise. We believe this mode of operation and methodology may be applicable in a range of human-centric tasks where the personalisation of policies may be desirable.

5.3 How are we going to communicate? Understanding redundancy

Our investigations of "Who are we going to be communicating with?" in Section 5.2, serve to explore one dimension of variability which we may expect. As we have discussed, there are many other dimensions which can be explored. In this section, we change our focus to the parameterisation of the discrete communications channel. Within games which allow communication for communication (like RG), there is typically an opportunity to choose the parameterisation of the message set. Within previous works [11, 73, 74], little attention has been paid to how this may impact the messaging protocol, which could lead to unexpected behaviours.

At a fundamental level, we should expect there to be a relationship between the communicative requirements of a task and the provisioning of the message set. If the set is too small, we will limit the performance achievable by our agents. We note that it is less clear what happens at

equality and values exceeding that. Here, we attempt to establish this. Investigation of this in temporally-extended environments is difficult, as it is often not clear what the communicative requirements of tasks are. We, therefore, conduct our investigations within RG which allows for the relationship to be known.

As we will go on to demonstrate in Section 5.3.1, the selection of the message set size has implications for both sample efficiency and generalisation. We find that when utilising the SOTA method introduced by Eccles et al. [73] on the MNIST-based RG (see Section 5.1.1) that over-allocation improves sample-efficiency on training data in comparison to equality. Interestingly the derived policies tend to retain a high degree of redundancy and typically utilise multiple messages to refer to the same digit. Where we observe a relationship between the redundancy and reward acquisition on test data.

We further build upon this result and investigate methods to reduce the redundancy within the protocol. By doing so, we hope to build more robust representations that better capture distinct concepts that a speaker (or agent more generally) may wish to communicate. We achieve this through the introduction of a linearly-scheduled entropy regulariser into the speaker's loss function. At the beginning of training, this term acts to maximise the utilisation of the message set which has previously been demonstrated to improve performance in this setting [73]. As training progresses, we linearly reduce this incentive until it is zeroed out. The speaker then tends towards a policy which reduces the utilisation of the message set. This tendency of emergent languages to reduce their entropy has previously been observed within [201] and we make use of this property here. We empirically validate our approach on MNIST [197] and then extend our analysis to KMNIST [202] and Fashion-MNIST [203]. In all cases, we observe improvements in sample efficiency and generalisation performance over the original method introduced by Eccles et al. [73].

The following section is structured as follows. We demonstrate our problem in Section 5.3.1. Observing the aforementioned characteristics, we introduce our entropy scheduling method in Section 5.3.2 . Our experimentation setup and results are introduced in Section 5.3.3. We then discuss our results, their implications and outline a direction for future work in Section 4.7.

5.3.1 Problem Demonstration

The aim of this work is to investigate the impact of message set size in MARL, in this section, we demonstrate that over-provisioning M results in improved sample efficiency, but that the policies tend to maintain a higher than necessary level of redundancy in their messaging protocols. We conduct experiments with 10, 20, 30 and 40 messages each repeated with 10 random seeds. We split the dataset into a $55k$ training set, and a $5k$ validation set. We use the same hyperparamaters defined within [73] and implement our code within PyTorch Lightning [204].

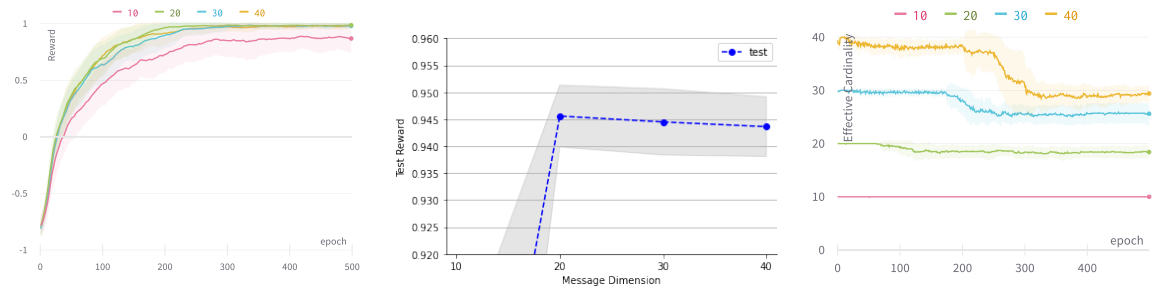
The performance of different cardinalities of M are shown in Figure 5.8. It is apparent from Figure 5.8a that over-provisioning of M results in improved sample efficiency when compared

to $|M| = 10$. Over-allocation of M may afford a degree of flexibility within learning and reduce the difficulty of the task. The extra capacity could allow the speaker to represent more distinct members of an image class as separate messages, whereas $M = 10$ requires the speaker to learn that these images represent the same concept. This is supported by Figure 5.8c, which shows the *effective cardinality* of the message protocol. We define *effective cardinality* in Equation (5.5) which tells us which messages are being used by the speaker on a large batch of validation data, B. It illustrates how speakers tend to converge to messaging protocols which utilise a subset of M which is notably larger than the minimum viable size of 10. We expected that this may manifest in a reduction in test performance – this is supported by Figure 5.8b which shows a negative correlation from $|M| \geq 20$ between test performance and the remaining cardinality. Noting, the significant drop in test performance present between message set cardinalities of 10 and 20, we conduct a targeted parameter sweep within this range which is shown in Figure 5.9. We find that cardinalities of 16 to 18 can improve test performance compared to $|M| = 20$. However, this appears to come with a greater degree of variance. Values less than 16 experience significant reductions in performance. Henceforth, we conduct experiments with $M = 20$ as it provides a trade-off between variance and mean performance.

$$c(B) = \sum_{m \in M} \mathbf{1}[\text{iff} m \in B] \tag{5.5}$$

A question that immediately presents itself is whether these characteristics are just a manifestation of the speaker side biases within [73]. In their standard form, these act to incentivise a speaker to maximise the mutual information between messages and observations. As we saw previously this is implemented into the speaker's loss through Equation (5.2). Where $\overline{\pi^i_M}$ is the average of the message policy over a batch of experience, and, λ and $\mathcal{H}_{target}$ are both hyperparameters. This encourages the speaker to output messages with uniform probability on average, but are not random with respect to an observation.

It would seem plausible that our observations from Section 5.3.1 may be caused by the



(a) Performance on training data (b) Performance on test data (c) Effective Cardinality of speaker

Figure 5.8: Figures showing the impact of multiple message set cardinalities. Plotted results show mean and 95% confidence interval.

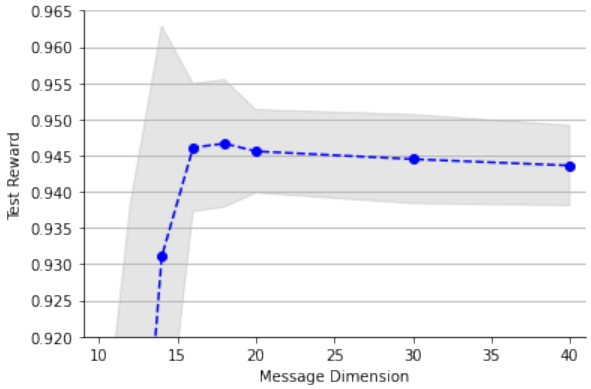
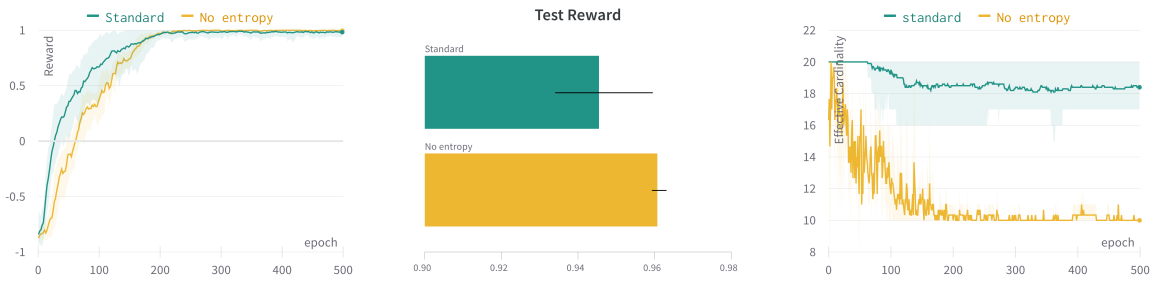


Figure 5.9: Figure showing the test performance as in Figure 5.8b with additional message cardinalities $(12, 14, 16, 18)$. Plotted results show mean and 95% confidence interval.

maximisation of $\mathcal{H}(\overline{\pi_M^i})$. This term may encourage the speaker to maintain excessive redundancy. To evaluate the importance of this term, we perform an ablation in which we remove it from the loss function. These are shown in Figure 5.10. Its removal appears to reduce learning speed as demonstrated in Figure 5.10a. However, both the cardinality and test performance of the converged policies are preferable. A natural question that arises from here is whether it is possible to obtain the initial sample efficiency which is offered through the inclusion of the entropy maximisation term and the generalisation performance that is given by its exclusion? Effectively, we are looking for a method that allows for the manipulation of effective cardinality at different stages throughout the training procedure.

5.3.2 Scheduling Entropy Regularisation

We have demonstrated that maximisation of $\mathcal{H}(\overline{\pi_M^i})$ encourages the maintenance of additional redundancy. However it does seem to offer improved sample efficiency. We suggest that it may enable this by providing a mechanism which encourages the speaker to represent more distinct



(a) Performance on training data (b) Performance on test data (c) Effective Cardinality of speaker

Figure 5.10: Figures showing the impact of entropy maximisation. Plotted results show mean and 95% confidence interval for message dimension of 20.

members of an image class as separate messages thereby reducing task complexity and aiding sample efficiency. As training progresses and the speaker's competence begins to increase, it would seem to be desirable to encourage it to minimise redundancy and hopefully improving generalisation. Here, we propose a methodology to enable this. We achieve this by modifying the behaviour of λ and making it a function of the speaker's experience as shown in Equation (5.6). Where t is specified in epochs and c is a hyperparameter which controls the point at which we switch from entropy maximisation to minimisation.

$$\lambda(t) = \lambda \times \max\left(\left[1 - \left(\frac{t}{c}\right), 0\right]\right) \tag{5.6}$$

Intuitively, $\lambda(t)$ acts to reduce the influence of the $\mathcal{H}(\overline{\pi^i_M})$ as the agent gathers experience. Once $\lambda(t) = 0$, we end up in the mode investigated in Section 5.3.1 and will tend towards the reduction of effective cardinality.

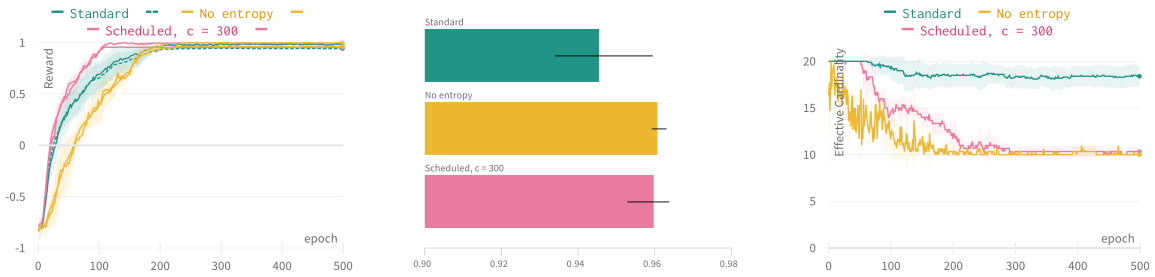
5.3.3 Experiments

We demonstrate entropy scheduling through an extension on the experimental protocol utilised in Section 5.3.1. In order to understand the implication of differing observational distributions and task complexities we introduce two further datasets. In addition to MNIST [197], we utilise KMNIST [202] and Fashion-MNIST [203]. These were selected as they do not require modification to the underlying neural network allowing for the avoidance of a potential source of variability in our analysis. Despite the change in dataset, the underpinning RG remains unchanged still requiring the speaker and listener to add together digits. Although the task may initially seem nonsensical as it now requires adding together items of clothing (or Japanese letters), we highlight that the task actually requires the agents to add together the labels of the images, **not** the actual images. This is to say that the semantic meaning of the images is unimportant.

The algorithmic implementation remains the same as in Section 5.3.1, with minor modifications to facilitate the scheduling given in Equation (5.6). We run a grid-search across $c = [100, 200, 300, 400, 500]$. We report training reward, validation reward and test performance effective cardinality for our best performing c. The obtained values are averaged across 3 random seeds trained for 500 epochs each. As baselines, we include both results obtained in Section 5.3.1, where these are the standard implementation of [73] and the modification without the maximisation of the average message entropy.

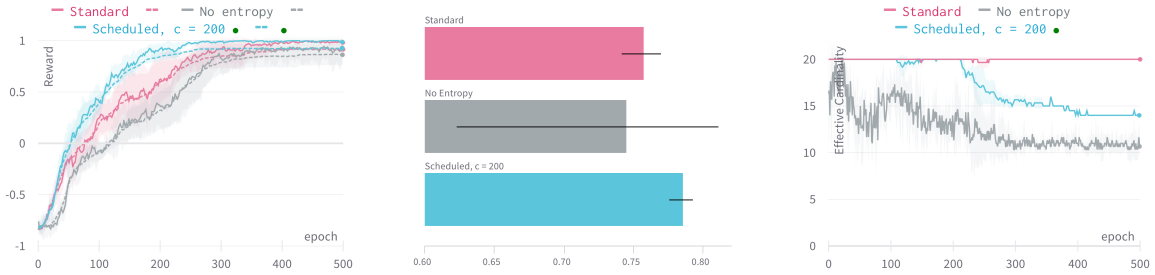
5.3.4 Results and Discussion

Our results for MNIST, KMNIST and Fashion-MNIST are shown in Figure 5.11, 5.12 and 5.13, respectively. These results support our hypothesis that addressing excessive redundancy can improve generalisation performance in comparison to the standard implementation of [73] as



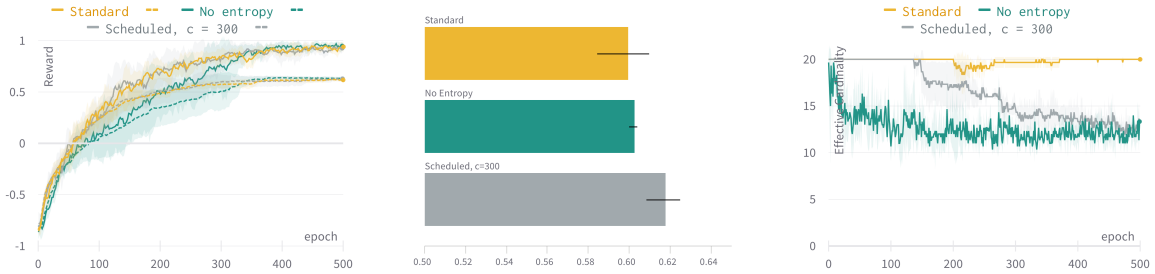
(a) Performance on training data (b) Performance on Test data (c) Effective Cardinality of speaker

Figure 5.11: Performance comparison of entropy scheduling, no entropy and baseline on MNIST. Plotted results show mean and 95% confidence interval.



(a) Performance on training data (b) Performance on Test data (c) Effective Cardinality of speaker

Figure 5.12: Performance comparison of entropy scheduling, no entropy and baseline with KMNIST. Plotted results show mean and 95% confidence interval.



(a) Performance on training data (b) Performance on Test data (c) Effective Cardinality of speaker

Figure 5.13: Performance omparison of entropy scheduling, no entropy and baseline with MNIST. Plotted results show mean and 95% confidence interval.

shown in Figure 5.11b, 5.12b and 5.13b. The scheduled entropy regularisation, $\lambda(t)$, facilitates this and can be shown to significantly reduce the excess redundancy in the protocol as shown in Figure 5.11c, 5.12c and 5.13c.

In comparison to the "No entropy" variant, the scheduled entropy regularisation method achieves competitive test performance and improves the mean performance on Fashion-MNIST and KMNIST (Figure 5.12b and 5.13b) but slightly underperforms on MNIST (Figure 5.11b). On MNIST and Fashion-MNIST, the scheduled entropy regulariser converges to similar values of effective cardinality. On KMNIST it appears to converge to a value of 14, where the reason for this is unclear but we expect that if we increase the training duration that this would converge towards 10 as the "No entropy" variant does. We do note, that even with this higher effective cardinality the test performance has significantly lower variance and a higher mean value which may attest to this being a peculiarity of the dataset.

A notable difference can be observed in the sample efficiency of the scheduled entropy regularisation. On MNIST and KMNIST (Figure 5.11a and 5.12a) it converges more quickly and on Fashion-MNIST, it achieves comparable performance to the standard implementation. This is a somewhat unexpected but welcome result. To explain this, we refer to the baselines within Figure 5.11a and observe that the initial performance of the standard implementation is superior to the "No entropy" variant but that the models converge at approximately the same point. This along with the dominance of the scheduled entropy regulariser may imply that entropy maximisation is only beneficial at the beginning of training. We hypothesise that later in the training procedure it may act as a source of noise which obfuscates the training objective.

5.3.5 Future Work

We believe that this method represents an interesting step towards generalisation within communicating MARL. However, there are a number of limitations which we aim to address in future work.

Firstly, our current investigations were restricted to referential games. This was due to their amenability to analysis. Most usefully, it allowed us to know the minimal number of messages required for the task and for the agents to concentrate on communication without distraction by other environmental tasks. In future work, we intend to extend our experimentation to more complex temporally-extended environments where communications protocols may be more abstract. These types of environments may mandate the utilisation agent architectures involving recurrent neural networks which will add other complexities.

Secondly, we consider a restricted class of scheduling functions. In this work, we only evaluated linear schedulers which are a function of training time. There are many other plausible functions, for example, a step or exponential function. It is plausible that the correct choice may be dependent on exact environmental characteristics. It may also be advantageous to use alternative values to condition the scheduler on other than training time which may better capture an agent's

understanding of the environment.

5.4 Conclusion

Communication is a method by which agents can cooperate, and it is commonplace in many real-world systems. Learning how to communicate, however, is exceedingly difficult and raises significant challenges for DMARL agents. This chapter considered the challenge of learning how to communicate and conducted fundamental research exploring a multitude of challenges. We focused on two distinct questions, firstly, "Who are we going to be communicating with?", and secondly, "How are we going to be communicating?". In doing so, we hoped to understand the limitations and qualities of communicative protocols.

In our analysis of "Who are we going to be communicating with?" in Section 5.2, we considered the implications of periodic interactions within populations of agents. We observed how multiple languages can arise, and how partner changes can result in catastrophic forgetting of established protocols. In order to overcome this, we considered the development of agents which can maintain multiple languages. This work builds upon that by [73] and introduces a parameter isolation method into their neural network in order to mitigate the aforementioned issues. The modification involves the utilisation of a multi-headed output network, where each head is utilised for a specific language. This approach was validated empirically within a novel referential game formulation which facilitated the evaluation of language maintenance through interactions with multiple unique agents and can serve as a simple test-bed for future work. The results demonstrate that the proposed method effectively avoids catastrophic forgetting when compared to the standard implementation of [73]. Future work could consider this methodology within more complex domains and zero-shot scenarios.

In Section 5.3 we explored our second question - "How are we going to be communicating?". Here, we considered the impact of the discrete message set size on the emergent protocol. Investigations were conducted within an RG which allowed for the communicative requirements of the task to be known. When utilising the method introduced by [73], we found that equality between the message set size and the task's requirements led to noticeably reduced sample efficiency and absolute performance in comparison to over-parameterisation. However, the derived policies tended to maintain high degrees of redundancy. Further analysis established that the route course of the redundancy was an entropy maximisation term within [73]. Its omission translated to reduced redundancy which improved generalisation, but at the cost of sample efficiency. Through the introduction of a linearly scheduled decay into the entropy maximisation term, we were able to achieve the best of both variants. The proposed approach was empirically validated on MNIST, KMNIST and Fashion-MNIST. On all datasets we observed reduced redundancy, improved test performance and greater than or equivalent sample efficiency to [73]. On MNIST and KMNIST, we observed improvements in sample efficiency which was attributed to the entropy maximisation

acting as a source of noise later in later stages of training. We believe this method presents an interesting insight into a commonly overlooked hyperparameter and its connections to sample efficiency and generalisation. Future work could estimate communications requirements of temporally extended tasks which would enable the translation of the observations provided here into actionable insights.

By developing a deeper understanding of questions like those posed here we hope to better understand the nature of communications. Through doing so, we may be able to equip agents with apparatus which facilitates better cooperative behaviours. At the very least, we expect that work of this type may enable us to develop better solutions to complex tasks. More aspirationally, we anticipate that future industrial systems will be heavily dependent on inter-agent communication in order to facilitate the organisation of complex heterogeneous systems. Advantage may be extracted by enabling DMARL agents to partake in these conversations. As we have identified, there are numerous opportunities for future work, from a task-centric perceptive identification capability to quantify task dependence on communication would be advantageous.

CHAPTER 6

CONCLUSIONS

Many industry's operations are dependent on infrastructure comprised of vast numbers of interacting and interdependent assets. The resultant system is sufficiently intricate and high dimensional that it typically prohibits the direct application of conventional DRL methodologies. This thesis explored the suitability of DMARL as a solution to this problem. As we have learned, DMARL is tailored towards distributed control and encompasses a range of algorithmic innovations which provide tractability and suitability within this challenging and common domain. Our analysis was directed towards two compelling industrial sectors which have considerable societal importance. These are telecommunications and logistics, both of which provide essential services. Furthermore, we also explored the role of communication in cooperative DMARL, which we anticipate as a key method in future methods which may need to support cooperation within hybrid human-robot teams. In the text below, we summarise our contributions.

The initial focus of our work within telecommunications was to examine the limitations of conventional DRL methodologies within what is typically a highly distributed system. We investigated the potential of utilising single-agent DRL to optimise resource allocation in an O-RAN system. An effective solution was developed which improved resource utilisation by 5.7% over our strongest baseline. Despite the observed improvements, scalability limitations within the proposed solution emerged upon closer inspection. While analysing the feasibility of centralised and single-agent methods, we observed that centralised approaches often struggle with the curse of dimensionality, limiting their scalability to realistic networks. Conversely, single-agent methods may overlook the implications of agent interactions, which can reduce their effectiveness. The aforementioned difficulties can be remedied through the adoption of DMARL. The viability of DMARL was demonstrated in a network maintenance task which required assets to be able to make decentralised maintenance decisions. To improve learning efficiency while maintaining full

decentralisation at deployment, we incorporated concepts from CTDE. Notably, we integrated critic centralisation into our AC algorithms during training, following a common practice in CTDE. The best-performing solution, convMAAC, utilised an GNN-based centralised critic and was shown to improve network availability by 3.49% and 6.13% over a heuristic baseline within two simulated RAN. Notably, the employment of an GNN-based critic greatly reduced the required parameters within the DNN in comparison to conventional FCN alternatives providing improved scalability. Key limitations of the model were identified, where these included Sim-to-Real and environment non-stationarity.

Within the logistics work, we focused on the extension and improvement of an existing solution for an order-picking problem with application to commercial warehouses. The best-performing solution, HSNAC, was developed in collaboration with domain experts and harnessed various ideas from across DMARL, most notably including methodologies at its intersection with HRL. HSNAC was demonstrated to achieve competitive performance with established heuristics, outperforming PDM whilst falling short of FM. However, unlike the heuristic methods, the derived approach can learn effective behaviours directly from interaction and does not require manual optimisation significantly reducing the human effort required. Our contributions were multi-faceted, including enhancing the computational efficiency of the DMARL training pipeline, model optimisations, policy visualisations and identification of current limitations. Similar to the telecommunications work, the expected limitations of the logistics solution are foreseen to bear a striking resemblance, with a particular emphasis on addressing Sim-to-Real transfer and challenges associated with environmental non-stationarity. Fundamentally, this is attributable to underlying limitations of DMARL models and the dynamic nature of real systems.

Both convMAAC and HSNAC facilitated learning of cooperative policies within their target industrial tasks. They made use of varied algorithmic mechanisms to foster cooperative behaviours attesting to the diversity of methodologies which are available and may be utilised within differing applications. In our telecommunications work, we anticipated an opportunity further to improve performance through the facilitation of inter-agent communication. However, empirically, we were unable to demonstrate this. This led to an interest in the fundamental properties of emergent communication as a method for cooperation. As a microcosm of the problem of learning to communicate, we conducted a series of experiments within referential games. We established that periodic interactions in populations of agents can lead to catastrophic forgetting of established protocols and how vocabulary size and utilisation can impact learning efficiency. In both cases, we demonstrated methodologies which enable the mitigation of these observations. Although, more abstract than other concepts within the thesis, we believe communication will likely be an integral part of future systems as a method to enable cooperation within heterogeneous teams which may involve human-robot collaboration. It is therefore essential that fundamental work like this is continued such that the limitations of conventional methods can be established.

In conclusion, this thesis served to further research in cooperative DMARL and demonstrated

application within two challenging industrial domains. Our contributions highlight the potential of DMARL to revolutionise industrial control and pave the way towards greater productivity. Furthermore, they record the process, challenges and considerations which are common and must be overcome in order to successfully develop viable DMARL solutions. Despite our efforts, many obstacles still remain before these methodologies can be successfully deployed. Notably, we have not grappled with the complexities of deployment or those that will be raised by integration with real systems. In the following section, we provide recommendations for how future work in this area may proceed.

6.1 Future Work

As the methodologies described within this thesis and the literature more widely begin to approach maturity, there will be an increasing propensity towards deployment within real-world applications. This represents a significant domain shift, from the sanitised and often simplified experimental environments commonplace in academia to the highly dynamic and complex components that constitute production systems. Furthermore, employed solutions will have to operate within the constraints of societal bureaucracy and contractual obligations. Here, we identify compelling opportunities for further work.

Sim-to-Real The infamously low sample efficiency of DMARL and DRL and dangers of learning within the real-world mandates training within simulation. The simulators are typically required to be fast and parallelisable in order to reduce the effective training time. As we have mentioned previously, these simulators are typically approximations of relevant processes and the quality of this approximation is often referred to qualitatively as the Sim-to-Real gap. The mitigation of this would seem to be important in the pursuit of practically applicable systems. There would, however, appear to be practical trade-offs which exist between simulator accuracy and computational efficiency which require investigation. It may be expected that certain classes of tasks or perhaps even representations may be less susceptible to the challenges of Sim-to-Real. For example, it may be expected that digitally native applications that are somewhat disconnected from physical processes may be easier to move towards deployment.

Non-stationary environments The underlying behaviour of industrial systems varies over time, the reasons for which are numerous and may include changes in the workforce, technology, user demands or global events (e.g. pandemics). These changes can be abrupt or they may be gradual, but no matter how they occur, adaption is typically critical for commercial longevity. There are numerous opportunities for future work and we explore a subset of them below. Here, we explore variation in workforce composition and the challenge of ensuring system-level goals are achieved. At a high level, we desire a workforce that can work together seamlessly. Cooperative DMARL tends to develop idiosyncratic conventions which may not be compatible with other

agents (e.g. human employees) who were not part of the training procedure. Exploration of methodologies like ZSC within industrial settings is a worthwhile endeavour. Furthermore, its intersection with ideas presented in Chapter 5 regarding periodic interactions and personalisation may enable incrementally improvement of human experience.

Risk and Governance DRL and DMARL are on their way to becoming a common part of toolboxes which organisations can use to solve decision-making processes. However, their introduction requires an understanding of their alignment with organisational values, objectives, risk appetites and legality. In their current form, there would seem to be numerous issues that have to be overcome. For example, DRL and DMARL do not typically come with performance guarantees, comprehensive testing is impractical for developed applications is difficult, and the explainability of decisions can be difficult to achieve. Methodologies like CMDP and further consideration of ML-Ops are likely essential. This is especially important in regulated industries like telecommunications and finance in the United Kingdom.

Auto ML DMARL and DRL promise the capacity to automatically learn to solve tasks through interaction. This glosses over the significant engineering efforts that are currently critical for its successful application. As we captured in Section 2.3, there is a vast array of decisions which must be made and navigation of these requires significant intuition, determination and most critically luck. Overcoming significant sensitivity to formulation, methodology and algorithmic hyperparameters is typically an innate part of any development cycle. This challenge is exacerbated by the speed at which DRL research community is progressing. Ever-increasing the number of decisions that are available for engineering teams. Effective distillation of these ideas into actionable insights is necessary if we are to see widespread adoption of these promising methodologies within commercial applications.

BIBLIOGRAPHY

[1] Richard S Sutton and Andrew G Barto.
Reinforcement learning: An introduction.
MIT press, 2018.

[2] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath.
Deep reinforcement learning: A brief survey.
IEEE Signal Processing Magazine, 34(6):26–38, 2017.
doi: 10.1109/MSP.2017.2743240.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al.
Human-level control through deep reinforcement learning.
nature, 518(7540):529–533, 2015.

[4] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al.
Mastering the game of Go without human knowledge.
nature, 550(7676):354–359, 2017.

[5] Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine.
How to train your robot with deep reinforcement learning: lessons we have learned.
The International Journal of Robotics Research, 40:698 – 721, 2021.

[6] Marc G Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C Machado, Subhodeep Moitra, Sameera S Ponda, and Ziyu Wang.
Autonomous navigation of stratospheric balloons using reinforcement learning.
Nature, 588(7836):77–82, 2020.
ISSN 1476-4687.
doi: 10.1038/s41586-020-2939-8.

[7] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, et al.
A graph placement methodology for fast chip design.

Nature, 594(7862):207–212, 2021.

[8] Ming Tan.
Multi-agent reinforcement learning: Independent vs. cooperative agents.
In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.

[9] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch.
Multi-agent actor-critic for mixed cooperative-competitive environments.
In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6382–6393, Red Hook, NY, USA, 2017. Curran Associates Inc.
ISBN 9781510860964.

[10] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson.
Counterfactual Multi-Agent Policy Gradients.
32nd AAAI Conference on Artificial Intelligence, AAAI 2018, pages 2974–2982, may 2017.

[11] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson.
Learning to communicate with deep multi-agent reinforcement learning.
In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 2145–2153, Red Hook, NY, USA, 2016. Curran Associates Inc.
ISBN 9781510838819.

[12] Lucian Buşoniu, Robert Babuška, and Bart De Schutter.
Multi-agent reinforcement learning: An overview.
Innovations in multi-agent systems and applications-1, pages 183–221, 2010.

[13] Shangxing Wang, Hanpeng Liu, Pedro Henrique Gomes, and Bhaskar Krishnamachari.
Deep reinforcement learning for dynamic multichannel access in wireless networks.
IEEE Transactions on Cognitive Communications and Networking, 4(2):257–265, 2018.
doi: 10.1109/TCCN.2018.2809722.

[14] Jiang Zhu, Yonghui Song, Dingde Jiang, and Houbing Song.
A new deep-q-learning-based transmission scheduling mechanism for the cognitive internet of things.
IEEE Internet of Things Journal, 5(4):2375–2385, 2018.
doi: 10.1109/JIOT.2017.2759728.

[15] Sanjeevan Ahilan and Peter Dayan.
Feudal multi-agent hierarchies for cooperative reinforcement learning.
arXiv preprint arXiv:1901.08492, 2019.

[16] Xiaoyang Wang, Jonathan D Thomas, Robert J Piechocki, Shipra Kapoor, Raúl Santos-Rodríguez, and Arjun Parekh.
Self-play learning strategies for resource assignment in open-ran networks.
Computer Networks, 206:108682, 2022.

[17] Jonathan Thomas, Marco Pérez Hernández, Ajith Kumar Parlikad, and Robert Piechocki.
Network maintenance planning via multi-agent reinforcement learning.
In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2289–2295, 2021.
doi: 10.1109/SMC52423.2021.9659150.

[18] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel.
High-dimensional continuous control using generalized advantage estimation.
In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[19] Aleksander Krnjaic, Jonathan D Thomas, Georgios Papoudakis, Lukas Schäfer, Peter Börsting, and Stefano V. Albrecht.
Scalable multi-agent reinforcement learning for warehouse logistics with robotic and human co-workers.
In *UNDERREVIEW*, 2022.

[20] Ronald J. Williams.
Simple statistical gradient-following algorithms for connectionist reinforcement learning.
Machine Learning, 8(3-4):229–256, 5 1992.
ISSN 0885-6125.
doi: 10.1007/BF00992696.
URL http://link.springer.com/10.1007/BF00992696.

[21] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel.
Value-decomposition networks for cooperative multi-agent learning based on team reward.
In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, page 2085–2087, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

[22] Richard Bellman.
Dynamic Programming.
Princeton University Press, 1957.

[23] Chris Watkins.
Learning from Delayed Reward.

PhD thesis, King's College, 1989.
URL http://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf.

[24] Marc Deisenroth and Carl E Rasmussen.
Pilco: A model-based and data-efficient approach to policy search.
In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472. Citeseer, 2011.

[25] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell.
Curiosity-driven exploration by self-supervised prediction.
In *ICML*, 2017.

[26] Marvin Minsky.
Steps toward artificial intelligence.
Proceedings of the IRE, 49(1):8–30, 1961.

[27] Marco Wiering and Martijn van Otterlo.
Reinforcement Learning: State-of-the-Art.
Springer Publishing Company, Incorporated, 2014.
ISBN 364244685X.

[28] Ian Goodfellow, Yoshua Bengio, and Aaron Courville.
Deep Learning.
MIT Press, 2016.
http://www.deeplearningbook.org.

[29] Kunihiko Fukushima and Sei Miyake.
Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition.
In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[30] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner.
Gradient-based learning applied to document recognition.
Proceedings of the IEEE, 86(11):2278–2324, 1998.

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton.
ImageNet Classification with Deep Convolutional Neural Networks.
In F Pereira, C J Burges, L Bottou, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[32] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams.
Learning representations by back-propagating errors.
Nature, 323(6088):533–536, 1986.
ISSN 1476-4687.
doi: 10.1038/323533a0.
URL https://doi.org/10.1038/323533a0.

[33] Sepp Hochreiter and Jürgen Schmidhuber.
Long short-term memory.
Neural computation, 9(8):1735–1780, 1997.

[34] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini.
The graph neural network model.
IEEE Transactions on Neural Networks, 20(1):61–80, 2009.
doi: 10.1109/TNN.2008.2005605.

[35] Thomas N. Kipf and Max Welling.
Semi-supervised classification with graph convolutional networks.
In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[36] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio.
Graph attention networks.
In *International Conference on Learning Representations*, 2018.
URL https://openreview.net/forum?id=rJXMpikCZ.

[37] Ignasi Clavera, Jonas Rothfuss, John Schulman, Yasuhiro Fujita, Tamim Asfour, and Pieter Abbeel.
Model-based reinforcement learning via meta-policy optimization.
In *Conference on Robot Learning*, pages 617–629. PMLR, 2018.

[38] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver.
Prioritized experience replay.
In *ICLR (Poster)*, 2016.

[39] Hado Van Hasselt, Arthur Guez, and David Silver.
Deep reinforcement learning with double q-learning.
In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[40] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver.

Rainbow: Combining improvements in deep reinforcement learning.
In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018.
ISBN 978-1-57735-800-8.

[41] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz.
Trust region policy optimization.
In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 07–09 Jul 2015. PMLR.
URL https://proceedings.mlr.press/v37/schulman15.html.

[42] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov.
Proximal policy optimization algorithms.
ArXiv, abs/1707.06347, 2017.

[43] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine.
Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.
In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018.
URL https://proceedings.mlr.press/v80/haarnoja18b.html.

[44] Lloyd S Shapley.
Stochastic games.
Proceedings of the national academy of sciences, 39(10):1095–1100, 1953.

[45] Eric A Hansen, Daniel S Bernstein, and Shlomo Zilberstein.
Dynamic programming for partially observable stochastic games.
In *AAAI Conference on Artificial Intelligence*, volume 4, pages 709–715, 2004.

[46] Michael L Littman.
Markov Games as a Framework for Multi-Agent Reinforcement Learning.
In *Machine Learning Proceedings*, pages 157–163. 1994.
doi: 10.1016/b978-1-55860-335-6.50027-1.

[47] Sven Gronauer and Klaus Diepold.
Multi-agent deep reinforcement learning: a survey.
Artificial Intelligence Review, 55(2):895–943, 2022.

ISSN 1573-7462.
doi: 10.1007/s10462-021-09996-w.
URL https://doi.org/10.1007/s10462-021-09996-w.

[48] Georgios Papoudakis, Filippos Christianos, Arrasy Rahman, and Stefano V Albrecht.
Dealing with non-stationarity in multi-agent deep reinforcement learning.
arXiv preprint arXiv:1906.04737, 2019.

[49] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht.
Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks.
In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*, 2021.
URL http://arxiv.org/abs/2006.07869.

[50] Sergiu Hart.
Shapley value.
Springer, 1989.

[51] Gerald Tesauro.
Td-gammon, a self-teaching backgammon program, achieves master-level play.
Neural computation, 6(2):215–219, 1994.

[52] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P Agapiou, Max Jaderberg, Alexander S Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver.
Grandmaster level in StarCraft II using multi-agent reinforcement learning.
Nature, 575(7782):350–354, 2019.
ISSN 1476-4687.
doi: 10.1038/s41586-019-1724-z.

[53] Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster.
"other-play" for zero-shot coordination.
In *International Conference on Machine Learning*, pages 4399–4410. PMLR, 2020.

[54] Landon Kraemer and Bikramjit Banerjee.
Multi-agent reinforcement learning as a rehearsal for decentralized planning.
Neurocomputing, 190:82–94, 2016.

ISSN 0925-2312.
doi: https://doi.org/10.1016/j.neucom.2016.01.031.
URL https://www.sciencedirect.com/science/article/pii/S0925231216000783.

[55] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson.
QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning.
In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4295–4304. PMLR, 10–15 Jul 2018.

[56] Justin K Terry, Nathaniel Grammel, Ananth Hari, Luis Santos, and Benjamin Black.
Revisiting parameter sharing in multi-agent deep reinforcement learning.
arXiv preprint arXiv:2005.13625, 2020.

[57] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip HS Torr, Pushmeet Kohli, and Shimon Whiteson.
Stabilising experience replay for deep multi-agent reinforcement learning.
In *International conference on machine learning*, pages 1146–1155. PMLR, 2017.

[58] Gerald Tesauro.
Extending Q-Learning to General Adaptive Multi-Agent Systems.
In S Thrun, L Saul, and B Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2003.
URL https://proceedings.neurips.cc/paper/2003/file/e71e5cd119bbc5797164fb0cd7fd94a4-Paper.pdf.

[59] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer.
Cooperative multi-agent control using deep reinforcement learning.
In *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers 16*, pages 66–83. Springer, 2017.

[60] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi.
Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning.
In *International conference on machine learning*, pages 5887–5896. PMLR, 2019.

[61] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson.
MAVEN: Multi-Agent Variational Exploration.
In H Wallach, H Larochelle, A Beygelzimer, F d'Alché-Buc, E Fox, and R Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

URL https://proceedings.neurips.cc/paper/2019/file/f816dc0acface7498e10496222e9db10-Paper.pdf.

[62] Lukas Schäfer, Oliver Slumbers, Stephen McAleer, Yali Du, Stefano V Albrecht, and David Mguni.
Ensemble value functions for efficient exploration in multi-agent reinforcement learning.
arXiv preprint arXiv:2302.03439, 2023.

[63] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu.
The surprising effectiveness of PPO in cooperative multi-agent games.
In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.
URL https://openreview.net/forum?id=YVXaxB6L2Pl.

[64] Filippos Christianos, Lukas Schäfer, and Stefano Albrecht.
Shared experience actor-critic for multi-agent reinforcement learning.
In *Advances in Neural Information Processing Systems*, volume 33, pages 10707–10717. Curran Associates, Inc., 2020.
URL https://proceedings.neurips.cc/paper/2020/file/7967cc8e3ab559e68cc944c44b1cf3e8-Paper.pdf.

[65] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra.
Continuous control with deep reinforcement learning.
In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2016.
URL http://dblp.uni-trier.de/db/conf/iclr/iclr2016.html#LillicrapHPHETS15.

[66] Shariq Iqbal and Fei Sha.
Actor-attention-critic for multi-agent reinforcement learning.
In *International conference on machine learning*, pages 2961–2970. PMLR, 2019.

[67] Angeliki Lazaridou and Marco Baroni.
Emergent multi-agent communication in the deep learning era.
arXiv preprint arXiv:2006.02419, 2020.

[68] Marco Baroni, Roberto Dessi, and Angeliki Lazaridou.
Emergent language-based coordination in deep multi-agent systems.
In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*, pages 11–16, Abu Dubai, UAE, December 2022. Association for Computational Linguistics.
URL https://aclanthology.org/2022.emnlp-tutorials.3.

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

[70] Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu.
Graph convolutional reinforcement learning.
In *International Conference on Learning Representations*, 2020.
URL https://openreview.net/forum?id=HkxdQkSYDB.

[71] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau.
Tarmac: Targeted multi-agent communication.
In *International Conference on Machine Learning*, pages 1538–1546. PMLR, 2019.

[72] Roberto Dessi, Eugene Kharitonov, and Marco Baroni.
Interpretable agent communication from scratch (with a generic visual processor emerging on the side).
In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.
URL https://openreview.net/forum?id=1AvtkM4H-y7.

[73] Tom Eccles, Yoram Bachrach, Guy Lever, Angeliki Lazaridou, and Thore Graepel.
Biases for emergent communication in multi-agent reinforcement learning.
In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 13111–13121, 2019.

[74] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, DJ Strouse, Joel Z Leibo, and Nando De Freitas.
Social influence as intrinsic motivation for multi-agent deep reinforcement learning.
In *International Conference on Machine Learning*, pages 3040–3049. PMLR, 2019.

[75] Ryan Lowe, Jakob Foerster, Y-Lan Boureau, Joelle Pineau, and Yann Dauphin.
On the pitfalls of measuring emergent communication.
In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2019.

[76] David Kellogg Lewis.
Convention: A Philosophical Study.
Cambridge, MA, USA: Wiley-Blackwell, 1969.

[77] Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni.
Multi-agent cooperation and the emergence of (natural) language.
In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
URL https://openreview.net/forum?id=Hk8N3Sclg.

[78] Kalesha Bullard, Douwe Kiela, Franziska Meier, Joelle Pineau, and Jakob Foerster.
Quasi-equivalence discovery for zero-shot emergent communication.
arXiv preprint arXiv:2103.08067, 2021.

[79] William H Guss, Cayden Codel, Katja Hofmann, Brandon Houghton, Noboru Sean Kuno, Stephanie Milani, Sharada Mohanty, Diego Perez Liebana, Ruslan Salakhutdinov, Nicholay Topin, Manuela Veloso, and Philip Wang.
The MineRL Competition on Sample Efficient Reinforcement Learning using Human Priors.
In *Thirty-third Conference on Neural Information Processing Systems (NeurIPS) Competition track*, dec 2019.
URL https://www.microsoft.com/en-us/research/publication/the-minerl-competition-on-sample-efficient-reinforcement-learning-using-human-priors/.

[80] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba.
Openai gym, 2016.

[81] Eitan Altman.
Constrained Markov decision processes: stochastic modeling.
Routledge, 1999.

[82] V. Gullapalli and A.G. Barto.
Shaping as a method for accelerating reinforcement learning.
In *Proceedings of the 1992 IEEE International Symposium on Intelligent Control*, pages 554–559, 1992.
doi: 10.1109/ISIC.1992.225046.

[83] Andrew Y Ng, Daishi Harada, and Stuart Russell.
Policy invariance under reward transformations: Theory and application to reward shaping.
In *Icml*, volume 99, pages 278–287. Citeseer, 1999.

[84] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané.
Concrete problems in ai safety, 2016.
URL https://arxiv.org/abs/1606.06565.

[85] Drew Bagnell and Andrew Ng.
On Local Rewards and Scaling Distributed Reinforcement Learning.
In Y Weiss, B Schölkopf, and J Platt, editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005.
URL https://proceedings.neurips.cc/paper/2005/file/02180771a9b609a26dcea07f272e141f-Paper.pdf.

[86] Xiujun Li, Lihong Li, Jianfeng Gao, Xiaodong He, Jianshu Chen, Li Deng, and Ji He.
Recurrent reinforcement learning: a hybrid approach.
arXiv preprint arXiv:1509.03044, 2015.

[87] Paul Almasan, José Suárez-Varela, Krzysztof Rusek, Pere Barlet-Ros, and Albert Cabellos-Aparicio.
Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case.
Computer Communications, 196:184–194, 2022.
ISSN 0140-3664.
doi: https://doi.org/10.1016/j.comcom.2022.09.029.
URL https://www.sciencedirect.com/science/article/pii/S0140366422003784.

[88] Vitaly Kurin, Maximilian Igl, Tim Rocktäschel, Wendelin Boehmer, and Shimon Whiteson.
My body is a cage: the role of morphology in graph-based incompatible control.
In *International Conference on Learning Representations*, 2021.
URL https://openreview.net/forum?id=N3zUDGN5lO.

[89] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al.
Magnetic control of tokamak plasmas through deep reinforcement learning.
Nature, 602(7897):414–419, 2022.

[90] Anssi Kanervisto, Christian Scheller, and Ville Hautamäki.
Action space shaping in deep reinforcement learning.
In *2020 IEEE Conference on Games (CoG)*, pages 479–486. IEEE, 2020.

[91] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger.
Deep reinforcement learning that matters.
In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018.
ISBN 978-1-57735-800-8.

[92] Lukas Biewald.
Experiment tracking with weights and biases, 2020.
URL https://www.wandb.com/.
Software available from wandb.com.

[93] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund.
Sim-to-real transfer in deep reinforcement learning for robotics: a survey.
In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744, 2020.
doi: 10.1109/SSCI47803.2020.9308468.

[94] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel.
Sim-to-real transfer of robotic control with dynamics randomization.
In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3803–3810, 2018.
doi: 10.1109/ICRA.2018.8460528.

[95] Cedric Renggli, Luka Rimanic, Nezihe Merve Gurel, Bojan Karlas, Wentao Wu, and Ce Zhang.
A Data Quality-Driven View of MLOps.
IEEE Data Engineering Bulletin, 44(1):11–23, mar 2021.
URL https://www.microsoft.com/en-us/research/publication/a-data-quality-driven-view-of-mlops/.

[96] Peizheng Li, Jonathan Thomas, Xiaoyang Wang, Ahmed Khalil, Abdelrahim Ahmad, Rui Inacio, Shipra Kapoor, Arjun Parekh, Angela Doufexi, Arman Shojaeifard, and Robert J. Piechocki.
Rlops: Development life-cycle of reinforcement learning aided open ran.
IEEE Access, 10:113808–113826, 2022.
doi: 10.1109/ACCESS.2022.3217511.

[97] Kun Shao, Zhentao Tang, Yuanheng Zhu, Nannan Li, and Dongbin Zhao.
A survey of deep reinforcement learning in video games.
arXiv preprint arXiv:1912.10944, 2019.

[98] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller.
Playing atari with deep reinforcement learning.
arXiv preprint arXiv:1312.5602, 2013.

[99] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver.
Mastering Atari, Go, chess and shogi by planning with a learned model.
Nature, 588(7839):604–609, 2020.
ISSN 1476-4687.
doi: 10.1038/s41586-020-03051-4.
URL https://doi.org/10.1038/s41586-020-03051-4.

[100] Null null, Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, Athul Paul Jacob, Mojtaba Komeili, Karthik Konath, Minae Kwon, Adam Lerer, Mike Lewis, Alexander H Miller, Sasha Mitts, Adithya Renduchintala, Stephen Roller, Dirk Rowe, Weiyan Shi, Joe Spisak, Alexander Wei, David Wu, Hugh Zhang, and Markus Zijlstra.
Human-level play in the game of Diplomacy by combining language models with strategic reasoning.
Science, 378(6624):1067–1074, 2022.
doi: 10.1126/science.ade9097.
URL https://www.science.org/doi/abs/10.1126/science.ade9097.

[101] Nguyen Cong Luong, Dinh Thai Hoang, Shimin Gong, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim.
Applications of deep reinforcement learning in communications and networking: A survey.
IEEE Communications Surveys Tutorials, 21(4):3133–3174, 2019.
doi: 10.1109/COMST.2019.2916583.

[102] Yimo Yan, Andy H F Chow, Chin Pang Ho, Yong-Hong Kuo, Qihao Wu, and Chengshuo Ying.
Reinforcement learning for logistics and supply chain management: Methodologies, state of the art, and future opportunities.
Transportation Research Part E: Logistics and Transportation Review, 162:102712, 2022.
ISSN 1366-5545.
doi: https://doi.org/10.1016/j.tre.2022.102712.
URL https://www.sciencedirect.com/science/article/pii/S136655452200103X.

[103] Sampo Kuutti, Richard Bowden, Yaochu Jin, Phil Barber, and Saber Fallah.
A survey of deep learning applications to autonomous vehicle control.
IEEE Transactions on Intelligent Transportation Systems, 22(2):712–733, 2021.
doi: 10.1109/TITS.2019.2962338.

[104] Ammar Haydari and Yasin Yılmaz.

Deep reinforcement learning for intelligent transportation systems: A survey.
IEEE Transactions on Intelligent Transportation Systems, 23(1):11–32, 2022.
doi: 10.1109/TITS.2020.3008612.

[105] Amirhosein Mosavi, Yaser Faghan, Pedram Ghamisi, Puhong Duan, Sina Faizollahzadeh Ardabili, Ely Salwana, and Shahab S Band.
Comprehensive review of deep reinforcement learning methods and applications in economics.
Mathematics, 8(10):1640, 2020.

[106] Marc G Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C Machado, Subhodeep Moitra, Sameera S Ponda, and Ziyu Wang.
Autonomous navigation of stratospheric balloons using reinforcement learning.
Nature, 588(7836):77–82, 2020.

[107] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi.
Top-k off-policy correction for a reinforce recommender system.
In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 456–464, 2019.

[108] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang.
Solving Rubik's Cube with a Robot Hand.
arXiv preprint, 2019.

[109] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos.
Distributional reinforcement learning with quantile regression.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[110] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller.
Maximum a posteriori policy optimisation.
In *International Conference on Learning Representations*, 2018.
URL https://openreview.net/forum?id=S1ANxQW0b.

[111] Rongpeng Li, Zhifeng Zhao, Xuan Zhou, Guoru Ding, Yan Chen, Zhongyao Wang, and Honggang Zhang.
Intelligent 5G: When Cellular Networks Meet Artificial Intelligence.
IEEE Wireless Communications, 24(5):175–183, 10 2017.
ISSN 1536-1284.

doi: 10.1109/MWC.2017.1600304WC.
URL http://ieeexplore.ieee.org/document/7886994/.

[112] Amal Feriani and Ekram Hossain.
Single and multi-agent deep reinforcement learning for ai-enabled wireless networks: A tutorial.
IEEE Communications Surveys Tutorials, 23(2):1226–1252, 2021.
doi: 10.1109/COMST.2021.3063822.

[113] Mamta Agiwal, Abhishek Roy, and Navrati Saxena.
Next Generation 5G Wireless Networks: A Comprehensive Survey.
IEEE Communications Surveys & Tutorials, 18(3):1617–1655, 2016.
ISSN 1553-877X.
doi: 10.1109/COMST.2016.2532458.
URL http://ieeexplore.ieee.org/document/7414384/.

[114] Maria Rita Palattella, Mischa Dohler, Alfredo Grieco, Gianluca Rizzo, Johan Torsner, Thomas Engel, and Latif Ladid.
Internet of Things in the 5G Era: Enablers, Architecture, and Business Models.
IEEE Journal on Selected Areas in Communications, 34(3):510–527, mar 2016.
ISSN 0733-8716.
doi: 10.1109/JSAC.2016.2525418.
URL http://ieeexplore.ieee.org/document/7397856/.

[115] Nicola Marchetti, Neeli Rashmi Prasad, Johan Johansson, and Tao Cai.
Self-Organizing Networks: State-of-the-art, challenges and perspectives.
In *2010 8th International Conference on Communications*, pages 503–508. IEEE, jun 2010.
ISBN 978-1-4244-6360-2.
doi: 10.1109/ICCOMM.2010.5509022.
URL http://ieeexplore.ieee.org/document/5509022/.

[116] Mugen Peng, Dong Liang, Yao Wei, Jian Li, and Hsiao-Hwa Chen.
Self-configuration and self-optimization in LTE-advanced heterogeneous networks.
IEEE Communications Magazine, 51(5):36–45, may 2013.
ISSN 0163-6804.
doi: 10.1109/MCOM.2013.6515045.
URL http://ieeexplore.ieee.org/document/6515045/.

[117] Naga Bhushan, Junyi Li, Durga Malladi, Rob Gilmore, Dean Brenner, Aleksandar Damnjanovic, Ravi Sukhavasi, Chirag Patel, and Stefan Geirhofer.
Network densification: the dominant theme for wireless evolution into 5G.

IEEE Communications Magazine, 52(2):82–89, feb 2014.
ISSN 0163-6804.
doi: 10.1109/MCOM.2014.6736747.
URL http://ieeexplore.ieee.org/document/6736747/.

[118] Jeffrey G. Andrews, Stefano Buzzi, Wan Choi, Stephen V. Hanly, Angel Lozano, Anthony C. K. Soong, and Jianzhong Charlie Zhang.
What Will 5G Be?
IEEE Journal on Selected Areas in Communications, 32(6):1065–1082, jun 2014.
ISSN 0733-8716.
doi: 10.1109/JSAC.2014.2328098.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6824752.

[119] Hao Ye, Geoffrey Ye Li, and Biing-Hwang Fred Juang.
Deep reinforcement learning based resource allocation for v2v communications.
IEEE Transactions on Vehicular Technology, 68(4):3163–3173, 2019.
doi: 10.1109/TVT.2019.2897134.

[120] Yasar Sinan Nasir and Dongning Guo.
Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks.
IEEE Journal on Selected Areas in Communications, 37(10):2239–2250, 2019.
doi: 10.1109/JSAC.2019.2933973.

[121] Eren Balevi and Jeffrey G Andrews.
Online antenna tuning in heterogeneous cellular networks with deep reinforcement learning.
IEEE Transactions on Cognitive Communications and Networking, 5(4):1113–1124, 2019.

[122] Maxime Bouton, Hasan Farooq, Julien Forgeat, Shruti Bothe, Meral Shirazipour, and Per Karlsson.
Coordinated reinforcement learning for optimizing mobile networks.
arXiv preprint arXiv:2109.15175, 2021.

[123] Maxime Bouton, Jaeseong Jeong, Jose Outes, Adriano Mendo, and Alexandros Nikou.
Multi-agent reinforcement learning with graph q-networks for antenna tuning.
arXiv preprint arXiv:2302.01199, 2023.

[124] Yifei Jin, Filippo Vannella, Maxime Bouton, Jaeseong Jeong, and Ezeddin Al Hakim.
A graph attention learning approach to antenna tilt optimization.
In *2022 1st International Conference on 6G Networking (6GNet)*, pages 1–5, 2022.
doi: 10.1109/6GNet54646.2022.9830258.

[125] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula.
Resource management with deep reinforcement learning.
In *Proceedings of the 15th ACM workshop on hot topics in networks*, pages 50–56, 2016.

[126] Weichao Mao, Haoran Qiu, Chen Wang, Hubertus Franke, Zbigniew Kalbarczyk, Ravi Iyer, and Tamer Basar.
A mean-field game approach to cloud resource management with function approximation.
In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.
URL https://openreview.net/forum?id=E3LgJdPEkP.

[127] Wenting Wei, Huaxi Gu, Kun Wang, Jianjia Li, Xuan Zhang, and Ning Wang.
Multi-dimensional resource allocation in distributed data centers using deep reinforcement learning.
IEEE Transactions on Network and Service Management, pages 1–1, 2022.
doi: 10.1109/TNSM.2022.3213575.

[128] Faris B. Mismar and Brian L. Evans.
Deep q-learning for self-organizing networks fault management and radio performance improvement.
In *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, pages 1457–1461, 2018.
doi: 10.1109/ACSSC.2018.8645083.

[129] Zili Ning, Ning Wang, and Rahim Tafazolli.
Deep reinforcement learning for nfv-based service function chaining in multi-service networks : Invited paper.
In *2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR)*, pages 1–6, 2020.
doi: 10.1109/HPSR48589.2020.9098994.

[130] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang.
Mean field multi-agent reinforcement learning.
In *International conference on machine learning*, pages 5571–5580. PMLR, 2018.

[131] Cheng-Xiang Wang, Fourat Haider, Xiqi Gao, Xiao-Hu You, Yang Yang, Dongfeng Yuan, Hadi M. Aggoune, Harald Haas, Simon Fletcher, and Erol Hepsaydir.
Cellular architecture and key technologies for 5g wireless communication networks.
IEEE Communications Magazine, 52(2):122–130, 2014.
doi: 10.1109/MCOM.2014.6736752.

[132] Oshri Naparstek and Kobi Cohen.

Deep multi-user reinforcement learning for distributed dynamic spectrum access.
IEEE Transactions on Wireless Communications, 18(1):310–323, 2019.
doi: 10.1109/TWC.2018.2879433.

[133] Deze Zeng, Lin Gu, Shengli Pan, Jingjing Cai, and Song Guo.
Resource management at the network edge: A deep reinforcement learning approach.
IEEE Network, 33(3):26–33, 2019.
doi: 10.1109/MNET.2019.1800386.

[134] Rongpeng Li, Zhifeng Zhao, Qi Sun, Chih-Lin I, Chenyang Yang, Xianfu Chen, Minjian Zhao, and Honggang Zhang.
Deep reinforcement learning for resource management in network slicing.
IEEE Access, 6:74429–74441, 2018.
doi: 10.1109/ACCESS.2018.2881964.

[135] Jianhua Tang, Tony QS Quek, Tsung-Hui Chang, and Byonghyo Shim.
Systematic resource allocation in cloud RAN with caching as a service under two timescales.
IEEE Transactions on Communications, 67(11):7755–7770, 2019.

[136] Wang Anchun, Xiao Liang, Zhou Shidong, Xu Xibin, and Yao Yan.
Dynamic resource management in the fourth generation wireless systems.
In *International Conference on Communication Technology Proceedings, 2003. ICCT 2003.*, volume 2, pages 1095–1098. IEEE, 2003.

[137] Song Yang, Nan He, Fan Li, Stojan Trajanovski, Xu Chen, Yu Wang, and Xiaoming Fu.
Survivable task allocation in cloud radio access networks with mobile edge computing.
IEEE Internet of Things Journal, 2020.

[138] Wenchao Xia, Tony QS Quek, Jun Zhang, Shi Jin, and Hongbo Zhu.
Programmable hierarchical C-RAN: From task scheduling to resource allocation.
IEEE Transactions on Wireless Communications, 18(3):2003–2016, 2019.

[139] Niezi Mharsi.
Cloud-Radio Access Networks: design, optimization and algorithms.
PhD thesis, 2019.

[140] Wei Zhu, Yi Zhuang, and Long Zhang.
A three-dimensional virtual resource scheduling method for energy saving in cloud computing.
Future Generation Computer Systems, 69:66–74, 2017.

[141] Haoyuan Hu, Lu Duan, Xiaodong Zhang, Yinghui Xu, and Jiangwen Wei.
A multi-task selected learning approach for solving new type 3d bin packing problem.

arXiv preprint arXiv:1804.06896, 2018.

[142] Stephane RA Barde, Soumaya Yacout, and Hayong Shin.
Optimal preventive maintenance policy based on reinforcement learning of a fleet of military trucks.
Journal of Intelligent Manufacturing, 30(1):147–161, 2019.

[143] Roberto Rocchetta, L Bellani, M Compare, Enrico Zio, and E Patelli.
A reinforcement learning framework for optimal operation and maintenance of power grids.
Applied energy, 241:291–301, 2019.

[144] Jing Huang, Qing Chang, and Jorge Arinez.
Deep reinforcement learning based preventive maintenance policy for serial production lines.
Expert Systems with Applications, 160:113701, 2020.
ISSN 09574174.
doi: 10.1016/j.eswa.2020.113701.

[145] Xiao Wang, Hongwei Wang, and Chao Qi.
Multi-agent reinforcement learning based maintenance policy for a resource constrained flow line system.
Journal of Intelligent Manufacturing, 27(2):325–333, 2016.
ISSN 15728145.
doi: 10.1007/s10845-013-0864-5.

[146] Andreas Kuhnle, Johannes Jakubik, and Gisela Lanza.
Reinforcement learning for opportunistic maintenance optimization.
Production Engineering, 13(1):33–41, 2019.
ISSN 18637353.
doi: 10.1007/s11740-018-0855-7.

[147] C. P. Andriotis and K. G. Papakonstantinou.
Managing engineering systems with large state and action spaces through deep reinforcement learning.
Reliability Engineering and System Safety, 191(April):106483, 2019.
ISSN 09518320.
doi: 10.1016/j.ress.2019.04.036.

[148] Bangcheng Li and Yifan Zhou.
Multi-component Maintenance Optimization: An Approach combining Genetic Algorithm and Multiagent Reinforcement Learning.

2020 Global Reliability and Prognostics and Health Management, PHM-Shanghai 2020, 2020.
doi: 10.1109/PHM-Shanghai49105.2020.9280997.

[149] Mark S Daskin.
Logistics: An overview of the state of the art and perspectives on future research.
Transportation Research Part A: General, 19(5):383–398, 1985.
ISSN 0191-2607.
doi: https://doi.org/10.1016/0191-2607(85)90036-6.
URL https://www.sciencedirect.com/science/article/pii/0191260785900366.

[150] Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev.
Reinforcement learning for combinatorial optimization: A survey.
Computers & Operations Research, 134:105400, 2021.

[151] Ju-Bong Kim, Ho-Bin Choi, Gyu-Young Hwang, Kwihoon Kim, Yong-Geun Hong, and Youn-Hee Han.
Sortation control using multi-agent deep reinforcement learning in n-grid sortation system.
Sensors, 20(12):3401, 2020.

[152] Elvin Ngu, Leandro Parada, Jose Javier Escribano Macias, and Panagiotis Angeloudis.
Decentralised Multi-Agent Reinforcement Learning Approach for the Same-Day Delivery Problem.
Transportation Research Record, 0(0):03611981221093324.
doi: 10.1177/03611981221093324.
URL https://doi.org/10.1177/03611981221093324.

[153] Dhruv Madeka, Kari Torkkola, Carson Eisenach, Dean Foster, and Anna Luo.
Deep inventory management.
arXiv preprint arXiv:2210.03137, 2022.

[154] Kaveh Azadeh, Debjit Roy, and MBM De Koster.
Dynamic human-robot collaborative picking strategies.
Available at SSRN 3585396, 2020.

[155] Maximilian Löffler, Nils Boysen, and Michael Schneider.
Picker routing in agv-assisted order picking systems.
INFORMS Journal on Computing, 34(1):440–462, 2022.

[156] Ivan Žulj, Hagen Salewski, Dominik Goeke, and Michael Schneider.
Order batching and batch sequencing in an amr-assisted picker-to-parts system.
European Journal of Operational Research, 298(1):182–201, 2022.

[157] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare.
Deep reinforcement learning at the edge of the statistical precipice.
Advances in neural information processing systems, 34:29304–29320, 2021.

[158] Open RAN - The open road to 5G, 2017, SAMSUNG. Available at: https://image-us.samsung.com/Samsungus/samsungbusiness/pdfs/Open-RAN-The-Open-Road-to-5G.pdf.

[159] Michele Polese, Leonardo Bonati, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia.
Understanding o-ran: Architecture, interfaces, algorithms, security, and research challenges.
IEEE Communications Surveys Tutorials, pages 1–1, 2023.
doi: 10.1109/COMST.2023.3239220.

[160] Technical specification group radio access network: Study on new radio access technology: Radio access architecture and interfaces (release 14), 3GPP TR 38.801 v14.0.0, 2017, March 2017.

[161] Bo Yi, Xingwei Wang, Keqin Li, Min Huang, et al.
A comprehensive survey of network function virtualization.
Computer Networks, 133:212–262, 2018.

[162] Jun Wu, Zhifeng Zhang, Yu Hong, and Yonggang Wen.
Cloud radio access network (C-RAN): a primer.
IEEE Network, 29(1):35–41, 2015.

[163] Parallel wireless creates OpenRAN "All G" radio access network architecture, June 2020, Parallel wireless white paper. Availabel at: https://www.parallelwireless.com/wp-content/uploads/parallel-wireless-creates-openran-all-g-radio-access-network.pdf.

[164] Alexandre Laterre, Yunguan Fu, Mohamed Khalil Jabri, Alain-Sam Cohen, David Kas, Karl Hajjar, Torbjorn S. Dahl, Amine Kerkeni, and Karim Beguir.
Ranked reward: Enabling self-play reinforcement learning for combinatorial optimization.
2018.
doi: 10.48550/ARXIV.1807.01672.
URL https://arxiv.org/abs/1807.01672.

[165] Thomas Anthony, Zheng Tian, and David Barber.
Thinking fast and slow with deep learning and tree search.
Advances in Neural Information Processing Systems, 30, 2017.

[166] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu.
IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures.
In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1407–1416. PMLR, 10–15 Jul 2018.
URL https://proceedings.mlr.press/v80/espeholt18a.html.

[167] Michael Knowles, David Baglee, and Stefan Wermter.
Reinforcement learning for scheduling of maintenance.
Res. and Dev. in Intelligent Syst. XXVII: Incorporating Applications and Innovations in Intel. Sys. XVIII - AI 2010, 30th SGAI Int. Conf. on Innovative Techniques and Applications of Artificial Intel., pages 409–422, 2011.
doi: 10.1007/978-0-85729-130-1_31.

[168] G. K. Chan and S. Asgarpoor.
Optimum maintenance policy with Markov processes.
Electric Power Systems Research, 76(6-7):452–456, 2006.
ISSN 03787796.
doi: 10.1016/j.epsr.2005.09.010.

[169] Rengarajan Srinivasan and Ajith Kumar Parlikad.
Value of condition monitoring in infrastructure maintenance.
Computers & Industrial Engineering, 66(2):233–241, 2013.

[170] Zhenglin Liang and Ajith Kumar Parlikad.
Predictive group maintenance for multi-system multi-component networks.
Reliability Engineering and System Safety, 195(October 2019):106704, 2020.
ISSN 09518320.
doi: 10.1016/j.ress.2019.106704.

[171] Jun Wang and Xiaoyan Zhu.
Joint optimization of condition-based maintenance and inventory control for a k-out-of-n:F system of multi-state degrading components.
European Journal of Operational Research, 290(2):514–529, 2021.
ISSN 03772217.
doi: 10.1016/j.ejor.2020.08.016.

[172] Michael Grieves.

Virtually Perfect: Driving Innovative and Lean Products through Product Lifecycle Management.
11 2011.
ISBN 0982138008.

[173] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala.
Pytorch: An imperative style, high-performance deep learning library.
In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.
URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance pdf.

[174] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang.
Deep graph library: A graph-centric, highly-performant package for graph neural networks.
arXiv preprint arXiv:1909.01315, 2019.

[175] Ashish Vaswani, Google Brain, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin.
Attention Is All You Need.
Technical report.

[176] Charles G Petersen and Roger W Schmenner.
An evaluation of routing and volume-based storage policies in an order picking operation.
Decision Sciences, 30(2):481–501, 1999.

[177] Jolyon Drury.
Towards more efficient order picking.
IMM monograph, 1(1):1–69, 1988.

[178] Peter R Wurman, Raffaello D'Andrea, and Mick Mountz.
Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses.
AI Magazine, 29(1):9, 2008.
doi: 10.1609/aimag.v29i1.2082.
URL https://ojs.aaai.org/index.php/aimagazine/article/view/2082.

[179] René De Koster, Tho Le-Duc, and Kees Jan Roodbergen.

Design and control of warehouse order picking: A literature review.
European journal of operational research, 182(2):481–501, 2007.

[180] Hang Ma, Jiaoyang Li, T K Satish Kumar, and Sven Koenig.
Lifelong Multi-Agent Path Finding for Online Pickup and Delivery Tasks.
In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2017.

[181] Hang Ma, Wolfgang Hönig, TK Satish Kumar, Nora Ayanian, and Sven Koenig.
Lifelong path planning with kinematic constraints for multi-agent pickup and delivery.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7651–7658, 2019.

[182] Qinghong Xu, Jiaoyang Li, Sven Koenig, and Hang Ma.
Multi-goal multi-agent pickup and delivery.
In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[183] Nir Greshler, Ofir Gordon, Oren Salzman, and Nahum Shimkin.
Cooperative multi-agent path finding: Beyond path planning and collision avoidance.
In *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pages 20–28, 2021.
doi: 10.1109/MRS50823.2021.9620590.

[184] Ju-Bong Kim, Ho-Bin Choi, Gyu-Young Hwang, Kwihoon Kim, Yong-Geun Hong, and Youn-Hee Han.
Sortation Control Using Multi-Agent Deep Reinforcement Learning in N-Grid Sortation System.
Sensors, 20(12), 2020.
ISSN 1424-8220.
doi: 10.3390/s20123401.
URL https://www.mdpi.com/1424-8220/20/12/3401.

[185] Yuchen Xiao, Joshua Hoffman, and Christopher Amato.
Macro-action-based deep multi-agent reinforcement learning.
In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 1146–1161. PMLR, 30 Oct–01 Nov 2020.
URL https://proceedings.mlr.press/v100/xiao20a.html.

[186] Peter Dayan and Geoffrey E Hinton.
Feudal reinforcement learning.
Advances in Neural Information Processing Systems, 5, 1992.

[187] Edsger W Dijkstra.
A note on two problems in connexion with graphs.
Numerische Mathematik, 1(1):269–271, 1959.

[188] Jon Louis Bentley.
Multidimensional binary search trees used for associative searching.
Commun. ACM, 18(9):509–517, sep 1975.
ISSN 0001-0782.
doi: 10.1145/361002.361007.
URL https://doi.org/10.1145/361002.361007.

[189] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala.
Pytorch: An imperative style, high-performance deep learning library.
In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.
URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance pdf.

[190] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang.
Learning under concept drift: A review.
IEEE Transactions on Knowledge and Data Engineering, 31(12):2346–2363, 2018.

[191] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone.
Curriculum learning for reinforcement learning domains: A framework and survey.
Journal of Machine Learning Research, 21(181):1–50, 2020.
URL http://jmlr.org/papers/v21/20-212.html.

[192] Karl Von Frisch.
The dance language and orientation of bees.
In *The dance language and orientation of bees*. Harvard University Press, 2013.

[193] Charles F Hockett and Charles D Hockett.
The origin of speech.
Scientific American, 203(3):88–97, 1960.

[194] Jonathan D. Thomas, Raul Santos-Rodriguez, Robert Piechocki, and Mihai Anca.
Multi-lingual agents through multi-headed neural networks.

In *Thirty-Fifth Conference on Neural Information Processing Systems, Cooperative AI workshop*, 2021.
URL https://drive.google.com/file/d/1QGZ-4hpavjSnTHDXIRlEYW5N12v7dIfs/view.

[195] Jonathan David Thomas, Raul Santos-Rodriguez, and Robert Piechocki.
Understanding redundancy in discrete multi-agent communication.
In *Second Workshop on Language and Reinforcement Learning*, 2022.
URL https://openreview.net/forum?id=cbsgHxnjV6.

[196] Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni.
Multi-agent cooperation and the emergence of (natural) language.
In *International Conference on Learning Representations*, 2017.
URL https://openreview.net/forum?id=Hk8N3Sclg.

[197] Yann LeCun and Corinna Cortes.
MNIST handwritten digit database.
2010.
URL http://yann.lecun.com/exdb/mnist/.

[198] Robert M French.
Catastrophic forgetting in connectionist networks.
Trends in Cognitive Sciences, 3(4):128–135, 1999.
ISSN 1364-6613.
doi: https://doi.org/10.1016/S1364-6613(99)01294-2.

[199] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars.
A continual learning survey: Defying forgetting in classification tasks.
IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 1–1, 2021.
doi: 10.1109/TPAMI.2021.3057446.

[200] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell.
Decaf: A deep convolutional activation feature for generic visual recognition.
In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 647–655, Bejing, China, Jun 2014. PMLR.

[201] Eugene Kharitonov, Rahma Chaabouni, Diane Bouchacourt, and Marco Baroni.
Entropy minimization in emergent languages.

In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[202] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha.
Deep learning for classical japanese literature, 2018.
URL http://arxiv.org/abs/1812.01718.
cite arxiv:1812.01718Comment: To appear at Neural Information Processing Systems 2018 Workshop on Machine Learning for Creativity and Design.

[203] Han Xiao, Kashif Rasul, and Roland Vollgraf.
Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[204] William Falcon et al.
Pytorch lightning.
GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning, 3, 2019.