Intelligent Resource Management with Deep Reinforcement Learning in Device-to-Device Communication

by

David Cotton

Thesis submitted in fulfilment of the requirements for the degree of

Master of Analytics (Research)

Under the supervision of

Zenon Chaczko, Doan Hoang & Massimo Piccardi

at

School Electrical and Data Engineering
Faculty Engineering and Information Technology
University of Technology Sydney
NSW, 2007, Australia

March 2022

Certificate of original authorship

I, *David Cotton* declare that this thesis, is submitted in fulfilment of the requirements for the award of *Master of Analytics (Research)*, in the *School of Electrical and Data Engineering*, *Faculty Engineering and Information Technology* at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise reference or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

SIGNATURE: _____

[David Cotton]

DATE: 1st March, 2022

PLACE: Sydney, Australia

Acknowledgements

I would like to thank the many people who have helped me complete this thesis. Thanks to my supervisors Dr. Zenon Chackzo, Prof. Doan Hoang and Prof. Massimo Piccardi for all the support and insightful comments that have immeasurably improved this work. Zenon, thank you for all your help and friendship. You helped me find my academic feet when I was lost and build my confidence to publish our research. Doan, thank you for your challenging questions and stimulating discussions. Massimo, thank you for your fantastic advice and support. Whenever there was a challenge, you were always two-steps ahead with timely, clear feedback and a path forward.
I would also like to acknowledge Prof. Richard Xu for providing me the opportunity to undertake this degree, sharing your deep machine learning knowledge and pushing my work to a higher level. Thank you Dr. Jason Traish for your mentorship and supporting the development of my ideas. You were always extremely generous with your time and knowledge.
I would also like to thank my friends in academia Sam Hartridge and Prof. Brendan Mulhern. Even though you specialise in fields very different from mine, listening to me whinge and providing me your helpful insight has helped me untangle byzantine academic process. Also, to friends outside academia with no interest in machine learning that allowed for an escape from this thesis when needed.
I would also like to thank my kids, James and Ellie. I am sorry that completing this degree has taken so much of our time together and I promise to spend more time playing with you from now on. Lastly, and most importantly, my beautiful wife Clare. This thesis deserves to have your name on the cover for all the support your have provided. When your friends and family said I was crazy for leaving my job for the destitute student world, you backed me. Through the highs and many lows of this degree you have helped me vent, unpack and calculate the best response. Despite having no interest in maths or computers, you have probably proofread more than twenty drafts of this thesis and the research papers it contains, of impossibly dense and incredibly boring academic writing. For all this and everything I've missed, I am truly, truly grateful of your love and support.

List of Publications

Conference :

1. D. Cotton, J. Traish and Z. Chaczko, "Coevolutionary Deep Reinforcement Learning", *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, Canberra, Australia, 1–4 December 2020.

2. D. Cotton and Z. Chaczko, "GymD2D: A Device-to-Device Underlay Cellular Offload Evaluation Platform", *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, Nanjing, China, 29 March–1 April 2021.

Table of Contents

List of Figures

List of Tables

Acronyms

3GPP 3rd Generation Partnership Project. 12

5G fifth generation. 3, 12

A2C Advantage Actor-Critic. ix, 68, 71

A3C Asynchronous Advantage Actor-Critic. 33, 71

AI artificial intelligence. 37, 40

API application programming interface. 59, 65

AWGN additive white Gaussian noise. 63

BS base station. ix, 3, 8, 15, 16, 59, 62–65, 67–69, 72

CCI co-channel interference. 9, 10

CERL collaborative evolutionary reinforcement learning. 40, 43

CLaRE Coevolutionary Learning and REinforcement. 5, 6, 40, 43, 49, 51, 79

CNN convolutional neural networks. 45, 47, 65

CoEA coevolutionary algorithms. 39

CSI channel state information. 21

CUE cellular user equipment. 15–17, 20, 22, 58, 59, 62–66, 68, 69, 72–74, 76, 77

D2D device-to-device. v, 1–8, 11–23, 56–63, 68, 69, 72–79

DNN deep neural network. 3, 8, 23, 28

DQN Deep Q-Networks. ix, 23, 28–32, 43, 47–49, 68, 70, 71, 76

DRL deep reinforcement learning. viii, 1, 3–8, 22, 23, 28, 40, 43, 54, 58, 60, 65, 68–70, 72–79

DUE D2D user equipment. viii, 13, 15–18, 20, 22, 23, 58, 62–66, 68–70, 72–76

EA evolutionary algorithm. 39, 40

EIRP effective isotropic radiated power. 62

ERL evolutionary reinforcement learning. 40, 43

FSP fictitious self-play. 36

FSPL free space path loss. 63, 64

GAE Generalised Advantage Estimator. 71

HetNet heterogeneous cellular network. 10, 20, 22

IoT Internet of things. 14

LTE Long Term Evolution. 3, 65, 72

LTE Advanced Long Term Evolution Advanced. 12

M2M machine-to-machine. 14, 19, 67

MARL multi-agent reinforcement learning. 34, 35

MBS macro base station. 10, 22, 58, 62

MCTS Monte Carlo tree search. 44, 45, 49–52

MDP Markov decision process. 24–26, 36

MEC mobile edge caching. 4, 13

NR New Radio. 3, 65, 72

OFDMA orthogonal frequency division multiple access. 22, 58, 62

PLE path loss exponent. 63, 72

PU primary user. 11, 15

QoS quality of service. 23, 67

RB resource block. 20, 22, 23, 58, 62–66, 69, 71, 72, 76

ReLU Rectified Linear Unit. 48, 70

RF radio frequency. 1, 13, 18

RL reinforcement learning. v, viii, 2, 3, 5–8, 22–26, 28, 31, 34–38, 40, 43, 45, 54, 56, 58, 59, 78

RRM radio resource management. 2, 6, 7, 18, 19, 21, 56–58, 61, 64, 65, 72–79

SAC soft actor critic. ix, 68, 70, 71

SBS small base station. 10, 21, 22

SINR signal-to-interference-plus-noise ratio. 9, 10, 18, 20, 57, 59, 63, 65, 69

SNR signal-to-noise ratio. 69

SU secondary user. 11

UE user equipment. ix, 3, 8, 10, 12, 14, 17, 18, 20, 63–65, 67, 76, 79

V2V vehicle-to-vehicle. 14, 19, 67

Wi-Fi wireless fidelity. 2, 10

Abstract

Radio resource management in device-to-device cellular offload can be optimised to increase network capacity, quality of service, energy efficiency, lower latency and provide more resilient networks. However, this resource optimisation problem is both NP-Hard and required to operate at a millisecond timescale, limiting feasible solutions.

In this thesis, we investigate how deep reinforcement learning can be applied to improve resource allocation. To empirically demonstrate our approach, we develop a network simulator for device-to-device cellular offload research. We also introduce an improved self-play algorithm for training reinforcement learning without expert guidance.

We apply our self-play training algorithm to the game Connect Four. Leveraging the competitive pressures of coevolution, we improve the performance of agents trained with our method, achieving a 15% higher win rate. Furthermore, agents exhibit more stable training dynamics and suffer fewer performance regressions.

We evaluate our network simulator and demonstrate deep reinforcement learning can significantly increase network capacity. Our network simulator reduces research friction and provides an evaluation platform to compare, share and build upon results. Our toolkit is provided to other researchers as open-source software.

Chapter 1

Introduction

Rapidly increasing demand for cellular services is necessitating more efficient radio frequency (RF) spectrum utilisation. It is predicted that the average user will soon be downloading a terabyte annually [1]. In particular, data intensive applications such as video traffic and the high connectivity of smart devices, is placing greater strain on networks. Compounding the issue, the RF spectrum over which wireless devices communicate is a finite resource and the portion of the spectrum most suited for cellular networks is becoming increasingly congested. This thesis investigates the use of a subset of machine learning, deep reinforcement learning (DRL), to improve cellular network performance, by optimising the allocation radio resource amongst cellular devices.

1.1 Background

Device-to-device (D2D) communication is a method for cellular devices to communicate directly, as opposed to the normal cellular mode in which traffic makes multiple network hops through base stations, backhaul and core networks [2]. D2D communication can occur **inband** sharing licenced cellular spectrum, or **outband** on unlicensed spectrum. Advantages of D2D communication include ultra-low latency and increased spectral and energy efficiency. D2D has been proposed for several use-cases including public safety communications, communications relay, localised services, and the focus of this thesis–traffic offloading [3].

Traffic offloading is a method to improve cellular network efficiency, by

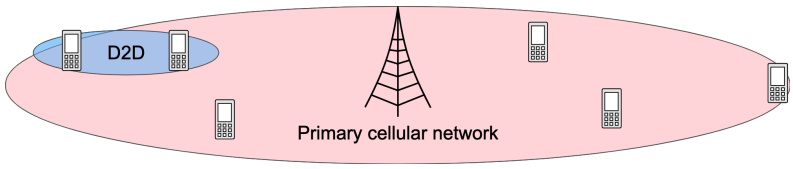

Figure 1.1: Cellular networks can offload communications that can occur directly, reducing primary network demand and increasing capacity.

communicating via alternate means. A familiar example is Wi-Fi, mobile phones are commonly configured to automatically send data traffic over Wi-Fi when available. However, there are other traffic offloading methods, including cellular densification and opportunistic communications.
It has been proposed that D2D could be used to provide extra bandwidth, by offloading communications that can be conducted directly [4]. This could occur non-orthogonally using inband underlay, in which D2D users share frequencies with the primary cellular network, as secondary users, as depicted in Fig. 1.1. Secondary users are able to share frequencies in **underlay** mode, by managing their interference to avoid interfering with primary users [5]. D2D underlay cellular offload is seen as a promising method to enhance network capacity in highly congested environments such as train stations and sports stadiums, by exploiting local geographic clustering of users to offload traffic [6].
Wireless communication systems must manage the allocation of radio resources, such as transmit power levels, frequencies, beamforming, modulation and error coding schemes, to manage interference and utilise resources as efficiently as possible. This is known as **radio resource management** (RRM). More efficient allocation strategies are able to achieve faster communication rates, greater network capacity, better quality of service, lower latency and improved energy efficiency. Existing spectrum management systems are quite inflexible and significantly underutilise available spectrum resources in order to provide reliable communications. Software-defined radio, the significant computing power now available in smartphones, and advances in machine learning and optimisation enable dynamic spectrum management, to coordinate transmissions and more efficiently utilise radio resources.
Reinforcement learning (RL) is a field of machine learning in which a software agent learns through experience to solve sequential decision-making problems. RL differs from other machine learning paradigms in that agents

are responsible for their own data generating process and must balance the competing concerns of exploration and exploitation. **Deep reinforcement learning** (DRL) is a subset of RL which uses deep neural networks (DNNs) to approximate an agent's policy and/or value function. This allows DRL to scale to more challenging environments.

1.1.1 Research Problems

(1) Developing of scalable resource allocation algorithms.

One of the main challenges in supporting D2D underlay cellular offload is interference mitigation between D2D and cellular communications. D2D user equipment (UE) need means to sense and limit their impact on the primary network. This can be achieved by using cellular base stations (BSs) to centrally coordinate the allocation of radio resources, such as transmit frequencies and power levels amongst cellular and D2D UE. However, this resource allocation problem is both NP-Hard and extremely time sensitive [7, 8]. For example, in the LTE and 5G NR standards, transmission are organised into 10 ms frames. This complexity limits the suitability of some classes of resource optimisation algorithms, which may not be able to scale to compute the resource allocation of potentially hundreds of devices in milliseconds.

One of the more promising class of algorithms for addressing this problem is DRL. There are several reasons for this. Firstly, while DNNs require a lengthy initial training period, once trained they are able to provide high quality, approximate answers in milliseconds. This training period can be conducted offline, on real or synthetic data, before being put into service. Secondly, DNNs scale better than many other optimisation or search methods, which are typically particularly computationally constrained. Thirdly, DNNs are powerful function approximators, allowing them to generalise to unseen states. This provides better flexibility to handle dynamic and variable propagation environments. Fourthly, DRL can be further trained online, allowing it to adapt to change in their environment. Lastly, through methods such as federated learning, DNNs can collaboratively learn a shared model from many devices in a network.

(2) Difficulty comparing existing research.

Another challenge is that existing D2D cellular offload research is limited in scope and generally more exploratory. As opposed to incrementing on established benchmarks, the trend has been towards exploring the utility of a diverse range of algorithmic approaches on many related, but different problems, such as improving spectral efficiency, power efficiency, quality of service, fairness, coverage, reliability and mobile edge caching. A challenge facing researchers is the lack of established problem definitions and benchmarks that facilitate reproducible and robust research.

1.1.2 Research Questions

Ultimately, we would like to answer the following questions:

1. How can we apply DRL to optimise D2D underlay cellular offload resource allocation to improve cellular network capacity?

2. How can we make it easier for D2D researchers to compare, share and build upon prior research?

1.2 Aim and Objectives

1.2.1 Aim

Radio resource management in D2D underlay cellular offload is a challenging optimisation problem in which more efficient resource allocation can improve various network efficiency measures. The aim of this thesis is to improve *network capacity* in *D2D underlay cellular offload* with the use of *deep reinforcement learning*.

1.2.2 Objectives

The following research objectives will be completed:

1. Develop a more efficient self-play training algorithm that improves the final performance, sample efficiency and training stability.

2. Develop a D2D cellular offload network simulator.

3. Develop a radio resource management system for D2D cellular offload using deep reinforcement learning.

1.3 Methodology

The primary methodology used in this thesis is empirical evaluation through simulation. When dealing with complex propagation models, analytical methods can become unacceptably complex, in which case simulation provides better flexibility.

1.3.1 Data Management Plan

Evaluation in this thesis will use data collected from the simulators and will be managed according to the following data management plan. No external data sources are required. There are two types of data that will be collected for analysis:

1. Simulator output, describing the state of the simulator after each simulation step; and

2. Algorithm performance metrics, such as neural network loss or entropy.

The resulting data will be stored in a machine readable format, compressed JSON files, in a centralised directory on the computer conducting the machine learning training. Each trial will be saved into its own directory, named with the date-time the simulation started and a descriptive name of the experiment. Each trial will record: the simulation configuration, algorithm hyperparameters, and the output from each simulation step.
It is anticipated that all research will be conducted internally within UTS, with no outside collaboration, greatly simplifying data access and ownership. By default, data access will be limited to project collaborators, but will shared on request. As the data will be synthetically generated, there are no consent, ethical and security considerations.

1.4 Results

- **Competitive RL Training Algorithm**: The coevolutionary RL training algorithm, CLaRE, demonstrated significantly improved final

performance, sample efficiency and training stability than existing self-play training methods. CLaRE achieved an almost 80% win rate, approximately 15% above the next leading method. Importantly, CLaRE showed strong convergence properties and more stable training dynamics.

- **D2D Network Simulator**: A network simulator and evaluation environment for D2D RRM research, *GymD2D*, was developed and released to the public as open-source software. GymD2D makes it easier for researchers to build, share and compare D2D resource allocation algorithms and results. GymD2D was evaluated with several high-performing DRL algorithms which demonstrated they could support 200% more devices and increase the system capacity by more than 11% with minimal impact on primary network performance.

1.5 Organisation

An overarching theme of this thesis is applied multi-agent deep reinforcement learning. An astute reader may notice a disconnect between the first and second research chapters, Chapters 3 and 4. This is due to the author changing supervisors and pivoting research focus to better align with the new supervisors. Chapter 3 investigates competitive multi-agent RL and was inspired by a previous research project developing RL agents to play real-time strategy video games. Chapter 4 leveraged lessons learnt from this work. However, when investigating the D2D RRM problem, centralised, cooperative models seemed most suitable for the task at hand.
The remainder of the thesis is organised as follows:

- **Chapter 2** provides a review of the literature on D2D communication, RRM and RL. The chapter begins with a brief background of cellular communication in order to define key concepts and terminology. The second section provides an overview of D2D communication and introduces the resource allocation problem. The third section presents the latest research into D2D RRM. The fourth section provide a formal background on RL. Finally, the last section introduces DRL and the algorithms used to optimise D2D radio resources.

- **Chapter 3** addresses the problem of training RL agents in competitive scenarios. The main contribution is a coevolutionary training algorithm

for RL that is capable of exploiting competitive pressures within a population of agents to increase final performance and sample efficiency.

- **Chapter 4** describes a network simulator and evaluation environment for D2D RRM research. The chapter establishes the need for a standardised benchmarking process for D2D RRM research, the design principles of such an environment and the describes the system model. A RRM system is developed and several leading DRL algorithms are evaluated in terms of network capacity performance.
- Finally, **Chapter 5** concludes this thesis by summarising the findings presented, discussing the research contributions and putting forward promising directions for future research.

Chapter 2

Literature Review

This chapter provides a background on the use of DRL for resource allocation optimisation in D2D underlay cellular offload. It begins with a brief review of cellular communications in order to highlight the relevant terminology and key concepts further sections will use. The second section provides an overview of D2D communications, describing its advantages, use cases and challenges. The third section provides a deep dive into the D2D resource management problem. The fourth section establishes the formal background of RL, the main optimisation method used in this thesis. Finally, the last section describes DRL, a subset of RL which uses deep neural networks to represent an agent's represent policy and/or value functions and the state-of-the-art algorithms used in the research chapters to optimise the allocation of radio resources.

2.1 Cellular Networks

Cellular networks are communication networks that facilitate portable wireless communications over large geographic areas. Cellular systems consist of **user equipment** (UE) which communicate wirelessly through **base stations** (BS). To increase network capacity and reliability, cellular networks are arranged in grids of smaller service areas. This cellular structure allows frequencies to be reused between non-neighbouring cells which has the advantage of increasing network capacity, reducing the required transmit power and enabling the networks to cover larger areas. BS are connected to the core network through backhaul links.

2.1.1 Propagation and Path Loss

The behaviour in which radio waves travel through their environment is known as **propagation** and is an important consideration in wireless communications as it impacts the received signal level. When emitted, electromagnetic radiation such as radio waves reduce in intensity proportional to the distance travelled according to the inverse-square law of geometric spreading. In addition to spreading, electromagnetic radiation is affected by the propagation phenomena of reflection, refraction, diffraction, absorption, polarisation and scattering. The frequency of emitted radiation impacts the degree of the effect the propagation phenomena. This results in some frequencies penetrating solid objects better, being more susceptible to atmospheric conditions or travelling further. The decrease in intensity of a transmitted signal due to absorption and scattering is known as **attenuation**. Obstacles between a transmitter and receiver can generate propagation effects such as **shadowing** and **multipath fading** that interfere with a signal's reception. The combination of spreading, attenuation and other propagation effects that reduce signal intensity is known as **path loss** and there exist several models for approximating path loss in different types of environments, under varying assumptions.

2.1.2 Interference

In addition to path loss, another important factor in wireless communications is interference. There are multiple sources of interference that can disrupt reception including, electromagnetic interference, co-channel interference, adjacent-channel interference, and intersymbol interference. This thesis is most concerned **co-channel interference** (CCI), which is caused by multiple transmitters using the same channel (frequency band) simultaneously and in proximity of each other. This can be modelled in orthogonal time, frequency and space dimensions.
Interference is quantified with the **signal-to-interference-plus-noise ratio** (SINR) ξ, which measures the ratio between the linear power of a target signal P from transmitter i, and the sum of interference signals I plus noise N at receiver j:

$$\xi_{i,j}[dB] = \frac{P}{I+N}\,, \tag{2.1}$$

expressed in dB.

2.1.3 Spectral Efficiency

The theoretical upper bound on the transmission rate is given by the Shannon-Hartley theorem which states that the capacity of channel C in Mbps, can be calculated using SINR $\xi_{i,j}$:

$$C_{i,j}[Mbps] = B \log_2 \left(1 + \xi_{i,j}\right), \tag{2.2}$$

where B is the channel bandwidth in MHz.
The measure of the ratio of the transmitted data rate and channel bandwidth is known as the **spectral efficiency**. In wired systems, spectral efficiency is equal to the modulation efficiency and this is measured in digital systems in bit/s/Hz. One method for improving spectral/modulation efficiency is the use of higher order modulation schemes. In wireless systems, multiple transmitters can share frequency bands and avoid CCI if geographically separated through path loss. As a consequence, in wireless systems spectral efficiency may be measured in bit/s/Hz per unit area or cellular site.

2.1.4 Traffic Offloading

Demand for wireless data is rapidly growing. Due to physical constraints, the information that can be communicated wirelessly through a single channel is limited. Researchers are developing many new techniques to increase data rates including improved encoding, multiple-input multiple-output and traffic offloading. Traffic offloading methods can be broadly grouped into:

- *Cellular densification*, extending the cellular idea to the extreme, smaller cells facilitate greater frequency reuse, lower energy consumption and facilitate the use of higher frequencies (millimeter-wave) which otherwise suffers from greater attenuation. A common method for cellular densification is the placement of small base stations (SBSs) inside the coverage zones of macro base stations (MBSs). This creates heterogeneous cellular networks (HetNets), in which UE must be assigned access points, creating new optimisation problems. Examples of SBSs include femto/pico/micro cells, which can be the size Wi-Fi routers and hidden inside everyday objects such as walls or lamp posts.

- *Dynamic spectrum management*, systems for dynamically adapting resource allocation response to demand. The portion of the radio frequency spectrum suitable for cellular communication is both scarce and underutilised

due to inflexible spectrum management policies. Dynamic spectrum management methods such as opportunistic spectrum access allow users to exploit under-utilised spectral resources. Examples of dynamic spectrum management include cognitive radio and D2D cellular offload.

2.1.5 Opportunistic Spectrum Access

Opportunistic spectrum access is a method for allowing secondary users (SU) to share radio resources with primary users (PU) to improve some network efficiency measure, usually capacity. Through the use of software defined radios, capable of sensing their environment and dynamically adjusting their communications, it is possible for SU to minimise their interference on the primary network. There are several mechanisms for sharing radio resources [5]:

- *Interweave networks* utilise spectrum vacancies, but must avoid interfering with PU. Spectrum vacancies may exist geographically or temporally. Geographic vacancies commonly occur due to the frequency bands being licensed by frequency and not by region. For example, military frequencies are typically only utilised around defence installations. Temporal vacancies exist between licensed communications and could be several hours or fractions of a second long. Interweave networks require sensing algorithms to determine suitable spectrum holes to exploit.

- *Underlay networks* in which SU can transmit simultaneously with PU but must adjust their transmissions to avoid interference. Underlay networks do not require SU to detect spectrum vacancies but must have means to monitor their interference and reduce it below an acceptable threshold. SU in underlay networks can avoid interference by reducing their transmit power, however this method can greatly limit communication distance.

- *Overlay networks* use knowledge of PUs transmissions to negate interference. This can be achieved by knowing PUs encoding scheme and modulating the SU signal with it. By using the SU to relay the message, they can transmit at any power.

2.2 D2D Communications

D2D is a broad set of methods for peer-to-peer wireless communication for cellular or other Internet connected radios. In contrast to normal cellular operation, UE utilising D2D mode communicate directly with one another instead of communicating through base stations and connected networks. The defining features of D2D communications are ultra-low latency and localised communications.
D2D was initially proposed as an opportunistic communication method using overlay networking to increase throughput [4]. It was then identified that D2D could be used to facilitate localised, rich multimedia services with the high data rates of WiMax and less user friction [9]. As opposed to alternate protocols such as WiFi, WiMax and Bluetooth, D2D would allow seamless transitions between cellular and D2D modes. This was possible by utilising underlay networking and allowing base stations to manage UE power control. D2D was incorporated by the 3GPP into the LTE Advanced release 12 to provide public safety communications. It has since been identified as a possible component of 5G with telecommunication industry bodies promoting its utility [10].

2.2.1 D2D Advantages

Advantages of D2D communication over cellular include:

- *ultra-low latency*, by transmitting directly instead of requiring multiple network hops, latency can be greatly reduced. This is particularly useful for time-sensitive applications such as industrial process automation, cooperative or autonomous vehicles, augmented reality, unmanned aerial vehicle command and control, or gaming.

- *localisation*, the direct communication and interference restrictions of D2D necessitates communicating devices be in proximity of each other. This lends D2D towards use generating restricted geographic service areas, such as geofencing, the establishment of virtual perimeters. D2D has greater range than alternative protocols such as Bluetooth or Wi-Fi and can reduce the location detection/awareness signalling required by other location-based service methods.

- *energy efficiency*, lower transmission power used by DUE to avoid interference helps reduce energy consumption. This extends the life of battery powered mobile devices, reduces energy expenditure across the network, reduces the radiation users are exposed to, and reduces noise in other RF systems.

- *network redundancy*, direct communication can be used to communicate outside cellular coverage. This can possibly be used for emergency communication in the case of network failure, or to relay communications from devices outside of coverage.

In this thesis, the advantages we are most concerned with are the increased *throughput* and *network capacity*. Cellular offload reduces the amount of traffic on primary networks and can be combined with mobile edge caching (MEC) to further reduce network traffic. It is anticipated that cellular offload will be most useful in highly congested scenarios such as train stations or sports stadiums, where networks resources are stretched thin.

2.2.2 D2D Use Cases

The aforementioned advantages of D2D, lend the technology to be used to enhance connectivity for the rapidly growing number of Internet connected devices. Possible use cases (Fig. 2.1) include:

- *cellular offload* in which traffic that can be conducted directly between devices is offloaded from the cellular network. To address rising performance demands and limited bandwidth, D2D links can be established within cellular coverage areas and share radio resources. This allows D2D to increase spectral efficiency, increase throughput and reduce latency.

- *public safety communications*, D2D was included in LTE release 12 for public safety purposes, which includes emergency services (police, fire brigade, ambulance) and other public safety agencies such as rail safety or government agencies. In this context, D2D allows public safety operators to communicate in the event of cellular network failure, outside of service areas (reducing network deployment costs), and to broadcast/multicast to groups of subscribers. Feasibly, D2D could be leveraged to allow emergency services to search for people in disaster situations.

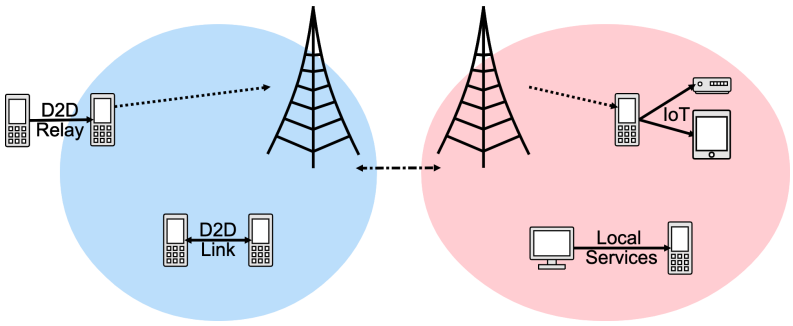

Figure 2.1: **D2D use cases** such as improving network capacity through cellular offload, network redundancy with D2D relay, localised services, and IoT enhancement.

- *communications relay*, D2D can be used to relay communications to cellular networks. This could be used to relay communications for UE outside of cellular coverage, or to assist with low battery levels by using nearby devices with more battery to perform higher powered communications.

- *localised services*, the range restrictions of D2D could be used to provide services to restricted geographic areas. These could be informational in nature, such as a virtual kiosk, museum guide, or augmented reality which creates interactive experiences that combines real world video with computer generated graphics. Alternative commercial applications include hyper-local marketing, eHealth or geofencing in which a connected device can be restricted to a limited geographic area.

- *vehicle-to-vehicle (V2V)*, communications systems that allow vehicles to share information with each other for safety or informational purposes. Ad-hoc vehicular networks allows vehicles to cooperate to a greater extent and can be used to optimise their interactions through sharing safety information, providing traffic information, reducing congestion, assisting emergency vehicles, or regulatory concerns.

- *machine-to-machine (M2M)*, communications systems that allow industrial systems to coordinate between themselves and with human operators. M2M allows industrial process to share instrumentation and sensor data and interface with controlling software.

2.2.3 D2D Taxonomy

As discussed in Sub-section 2.2.2, there exist several potential use cases beyond increasing network capacity that D2D could provide. Before examining the challenges for D2D systems, the thesis presents a taxonomy of D2D, to provide a framework to describe and analyse the differences between implementations that appear in the literature. The thesis extends upon existing D2D classification described in the literature [3, 11].

D2D Control

D2D systems can be classified by whether they are managed in a centralised or decentralised manner.

- *full control mode* in which DUE are managed by the cellular network operator. This requires BS communication over primary channels to receive this coordinating information after which devices can then communicate directly.

- *autonomous mode* in which DUE manage themselves to ensure they do not interfere with PU.

- *hybrid control mode* to combine benefits of full control and autonomous. In hybrid control the network can perform the most critical operations such as authentication and big picture resource allocation (such as a connection), with small scale resource allocation (such as individual resource blocks) managed by individual DUE.

Full control mode allows a more complete picture of the interference landscape which allows interference to be optimised to a greater extent. Full control mode allows CUE to be included in the optimisation problem, as there may optimisations possible to increase capacity with little impact to CUE by adjusting their resource allocation. Autonomous mode allows DUE to operate in periods of network failure at the cost of increased risk of interference due to their limited observability of other devices on the network. Hybrid control modes increase complexity which may introduce performance or security impacts.

D2D Coverage

D2D systems can be classified by the degree of network coverage,

- *in coverage* when DUE intend to operate in the licensed band simultaneously with CUE.
- *out of coverage* when DUE are outside cellular network coverage.
- *partial coverage* when at least one communicating DUE is inside of coverage and another out. This could be used to relay the out of coverage DUE to the cellular network.

D2D Communication Distribution

There are two communications distribution paradigms considered in D2D literature,

- *one-to-one communication*, direct communication between pairs of DUE. This is the most common routing scheme.
- *one-to-many communication*, transmitting DUE can broadcast or multicast to many DUE.

D2D Communication Area

D2D can be classified by whether communicating DUE are being served by the same BS,

- *same cell*, communicating DUE as both connected to the same BS.
- *different cell*, communicating DUE are connected to different BS.

D2D Interference Classification

There are multiple forms of interference that can occur in D2D communications. In this section we describe the interference classes, firstly to bring to the reader's attention, and secondly because their severity and mitigation strategies differ between classes. The three types of interference that need to be considered are:

- *DUE to CUE*, interference from secondary DUE to primary cellular users is the most important form of interference to consider in D2D systems. Secondary DUE must ensure primary cellular communication is not impeded.
- *CUE to DUE*, interference from primary cellular users to DUE is a second form of interference impacting D2D communications. While CUE are not required to minimise interference for secondary DUE, consideration of CUE interference can allow both systems to operate more efficiently.
- *DUE to DUE*, interference between groups of communicating DUE. Different groups of communicating DUE can impact one another, particularly if they are operating using the same radio resources and in proximity of each other.

Effective D2D management will account for these different interference scenarios. These interference scenarios highlight the difficulties autonomous mode D2D control faces, when considering the diversity of the interference picture.

2.2.4 D2D Challenges

There are several ongoing research challenges in the development of D2D systems. These challenges involve trade-offs that must be considered in context of the targeted use cases for D2D.

Peer discovery

To facilitate D2D communications, a peer discovery service is required to allow UE to become aware of neighbouring devices and select an appropriate communication strategy. This process can include, optimising multi-hop paths, time/frequency synchronisation, mode selection and how often these parameters should be reevaluated. Peer discovery can be grouped into restricted or open discovery classes. Restricted discovery requires the UE's permission to be discovered, providing greater privacy. Conversely, open discovery allows UEs to be detected whenever they are in proximity of compatible devices, potentially simplifying discovery and greatly increasing the pool of available devices. Peer discovery services can be facilitated by the network or autonomously by DUE. Network assisted discovery can improve optimisation efficiency and

security concerns at the cost of increased signalling overhead. Autonomous discovery distributes the signalling overhead to amongst individual UEs which can significantly decrease energy efficiency.

Energy consumption

D2D communications can significantly reduce energy consumption. Not only can DUE use lower transmit power levels, but system consumption can be lowered as less traffic passes through the core network. The literature on D2D energy consumption can be broken down to two overarching perspectives: analysing energy efficiency and minimising power consumption.
Energy efficiency analysis is concerned with the impacts on energy consumption of D2D enabled devices compared with typical cellular transmissions. D2D control modes impact energy efficiency, with full control systems exhibiting lower power consumption compared with autonomous mode [12]. This is in part due to full control modes greater optimisation potential and that peer discovery can be managed through normal cellular operations instead of requiring additional beacons or signalling processes.
Power consumption is typically treated as a secondary benefit of reduced SINR in D2D systems, but some research has treated power consumption as a primary objective [13]. It is useful to factor power consumption in the optimisation process to avoid D2D devices driving global power consumption higher as a result of harmful interference.
An interesting application of D2D to power consumption was the utilisation of D2D relays for cellular communication for devices with low battery levels [14]. This could be used to extend battery life of devices in enabled systems.

Radio resource management

Resource allocation is one of the more challenging problems in D2D cellular offload. To manage interference and provide the best quality of service possible for all users, radio systems aim to utilise the RF spectrum as efficiently as possible. More efficient spectral utilisation can allow more devices to communicate simultaneously, at faster speeds. In this search for spectral efficiency, cellular systems have a range of radio parameters, known as **radio resources**, which are allocated to devices and configure the manner of their communication.
The allocation of radio resources to devices is known as **radio resource**

management (RRM). Radio resources consist of parameters such as uplink and downlink frequencies, power levels, modulation scheme, error coding and antenna beamforming.
D2D RRM is one of the key challenges this thesis seeks to address. In the next section we examine the D2D RRM problem in more depth.

2.3 D2D Resource Management

The combinatorial complexity of allocating radio parameters between D2D and cellular networks, containing multiple, moving devices, while ensuring acceptable quality of service, has led to resource management solutions providing more conservative allocations. This challenge has inspired a volume of research into a wide variety of optimisation methods to improve a range of different resource allocation problems. Parallel research into other direct, wireless communication methods such as V2V and M2M investigate similar resource allocation problems, making the findings of interest to us.
One of the more common motivations in D2D RRM research is the increase of cellular network capacity. For this type of research, the problem is generally modelled using cellular offload with underlay networking. Another common motivation is improved energy efficiency. Initially, the D2D resource allocation literature focused on centralised, full control methods, however more recently decentralised solutions have garnered more attention.
To begin this section, we provide a classification to describe and compare the optimisation problems D2D resource allocation research investigates. We follow this up with an overview of the different simulation models used.

2.3.1 Optimisation Classification

D2D resource allocation is an optimisation problem. There are three key dimensions we can slice the literature across: the optimisation objective(s), the resource(s) being allocated (design variables), and the optimisation algorithm used. Specifying these dimensions provides a framework to compare D2D resource allocation research, which frequently address slightly different problems under a variety of assumptions. In the following paragraphs we provide an overview of some of the most common objective, resource and algorithm classes that appear in the literature.

Objectives

D2D resource management systems commonly investigate the following optimisation objectives:

- *Spectral efficiency* increasing spectrum reuse in terms of bit/s/Hz/area.
- *Network/cell capacity* increasing the achievable network or cell data capacity in terms of DUE and/or CUE capacity, commonly measured using the Shannon-Hartley channel capacity in bps.
- *Energy efficiency* reducing network energy consumption, particularly for battery constrained UE, which can be measured in terms of UE energy consumption in J/bit, or network energy consumption in W.
- *Quality of service* using measurements of channel quality such as the CDF of channel SINR or outage probability of cellular links.
- *Mobile edge caching* which aims improve QoE by caching popular files at the edge nodes of cellular networks, and includes problems such as determining optimal caching policies for spectral or energy efficiency.
- *HetNets* includes optimisation problems such as cell planning and coverage, traffic load balancing, and outage and network healing.

Resource classes

D2D resource management systems typically aim to optimise the allocation of one or more of the following:

- *Power level*, the maximum transmit power levels DUE should use so as to avoid interfering with other users.
- *Frequencies*, the frequencies DUE are assigned to communicate over. This includes both licensed and unlicensed frequencies, down to individual cellular RBs.
- *Communication mode*, whether UE should transmit a message via cellular or D2D mode.
- *Base station* in the case of HetNets, which access point UE should communicate through.

- *Cached files*, retrieval of cached files, such as web assets (e.g. CSS, Javascript), from edge servers in SBSs or possibly on neighbouring devices.

Algorithm classes

A variety of algorithm types have been applied to D2D resource management problems. In the literature reviewed, resource allocation algorithms could be roughly grouped into the following categories:

- *Rule-based algorithms* that assign values determined by hard-coded rules. These algorithms can be constructed to have very low computational complexity, $\approx O(1)$, using heuristics on parameters such as power levels and CSI to generate conservative approximations with very small observation spaces (e.g., single devices). It is generally seen as infeasible to expand rule-based algorithms to consider the broader network without methods to speed up computation.

- *Dynamic programming*, a recursive programming method that simplifies complex problems by reducing them down into simpler sub-problems. Dynamic programming methods can speed up computation significantly, however still may not be fast enough for real-world applications.

- *Combinatorial optimisation* methods such as integer programming. While combinatorial optimisation methods can find optimal solutions, they are generally slower and scale poorly with input size. Combinatorial optimisation methods are generally going to be most applicable to larger timescale allocation problems such as mode selection.

- *Graph theoretic* algorithms which model the problem using structures composed of pairwise relations between objects. Some RRM problems can be reduced to be solved by known graph network flow algorithms such as bipartite matching.

- *Game theoretic* solutions that model the resource allocation problem as a strategic interaction problem in either cooperative or non-cooperative scenarios. Frequently game theoretic solutions use numerical optimisation methods to find equilibrium solutions, also limiting their scalability.

- *Reinforcement learning* which approaches the resource allocation problem probabilistically in which agents aim to maximise their expected utility. Algorithms in the RL class include: classical RL, such as Q-learning; multi-armed bandits, which are a single step specialisation of RL; and DRL.

This classification is meant to be indicative of the solution mechanism used, to aid in comparison. Some of the reviewed literature combined elements of multiple algorithm classes, for example graph and dynamic programming or game theoretic and optimisation.

2.3.2 Simulation Models

Evaluation of D2D resource allocation algorithms can be conducted analytically, empirically, or empirically through simulation. Simulation is a common approach as analytical methods can become unacceptably complex when using more detailed propagation models and cost and time overheads involved with developing prototypes limit empirical evaluation.
As previously discussed, D2D can operate in both licensed and unlicensed bands. D2D can share frequencies in licensed bands, orthogonally through overlay networking, or as is more commonly studied in the literature, non-orthogonally through underlay networking. Typically, cellular networks are assumed to be based on orthogonal frequency division multiple access (OFDMA).
Simulations commonly model scenarios that restrict their network scope to a small number of MBSs, typically between one and nine, surrounded by many randomly positioned CUEs and DUEs. The most common scenario is a single MBS, which has the advantages of being more computationally efficient and abstracting away cell handover concerns. In HetNets problems, the environment also contains a number of SBSs. Typically omni-directional antenna and isotropic propagation are utilised, with path loss modelled using various log-distance models.
Generally, it is assumed that cellular systems are under full load, with one CUE assigned to each RB and all RB occupied, however there is some variation, with several works modelling traffic using random processes such as Markov chains. Simulations vary in scope and commonly features between 2–30 , 2–30 CUEs and 2–60 DUEs. It is generally assumed that DUE are paired and operate in a range between 10–30m apart.

2.3.3 RL-based D2D Resource Management

RL is a promising approach for D2D resource optimisation as it can provide high quality approximate answers, adapt to conditions, scale to large numbers of devices and provide solutions within the millisecond time constraints. RL, like all machine learning, requires an initial training period, after which the trained models can make decisions in milliseconds.

The first applications of RL for D2D resource management used classical RL algorithms such as Q-learning [15] and SARSA [16]. One of the original applications of RL in D2D used tabular Q-learning to allocate channels and transmit power levels to DUE in cellular offload [17]. However, this work was unable to empirically demonstrate scalability, the state-action space was very limited, consisting of two channels and three power levels.

Scalability is an ongoing challenge for RL D2D research, as tabular Q-learning—which uses a look-up table of Q-values—is computationally constrained as the look-up table grows exponentially with the size of the state-action space. One proposed solution was the development of a more compact state representation [18], in which a function approximator is used to more efficiently map individual Q-values via a parameter vector. Alternatively, decentralised Q-learning, in which a larger Q-table is decomposed, provided means to scale to up to 30 RB and 12 DUE [18, 19].

More recently, DRL, a subset of RL in which policies are parameterised with DNNs, has demonstrated the ability to scale to higher dimensional state-action spaces. The DQN algorithm was compared to Q-learning in an environment with time-varying channels, modelled using finite-state Markov chain and demonstrated that DRL can outperform classical RL in maximising system capacity [20]. Building upon this work, the DQN algorithm has been used for D2D resource management to develop decentralised control [21], maximise throughput while preserving QoS [22], and optimise energy efficiency in non-cooperative environments [23].

The success of DQN for D2D resource management has inspired researchers to investigate the other DRL algorithms. A multi-agent actor-critic algorithm was compared to a DQN in D2D underlay cellular offload and demonstrated that QoS and system capacity could be improved by lowering signalling overhead through decentralised control [24].

2.4 Reinforcement Learning

Reinforcement learning (RL) is a field of machine learning which aims to solve sequential decision-making problems. It learns to improve its behaviour through trial and error, maximising a scalar reward signal [16]. In contrast to other machine learning paradigms such as supervised or unsupervised learning, RL generates its own training data. This frees practitioners from the labour of curating training datasets and human biases involved in their construction. By extension, this implies that the distribution of the training dataset is likely to shift over time and that there is a feedback loop in which the actions an agent chooses, influences the training data it receives. This self-directed learning paradigm can make RL easier to employ on more complex problems as its reduces the practitioners need to prepare the agent for every scenario it could encounter. RL agents learn by making observations of the **state** of the environment $S_t \in \mathcal{S}$ at each timestep t. They use this information to select **actions** $A_t \in \mathcal{A}$ to obtain a **reward** $R_t \in \mathbb{R}$ (Figure 2.2).

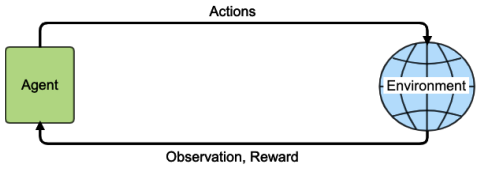

Figure 2.2: Reinforcement learning agents learn through observing their environment, taking actions based on their observations, and assessing the utility of their behaviour through the incoming reward signal.

2.4.1 Markov Decision Processes

The sequential decision-making problem RL aims to solve can be formalised by the **Markov decision process** (MDP) [25]. MDPs use the Markov property which assumes that future states are dependent only on the current state and not on the history of previous states. An MDP can be defined as a five-tuple $\langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$ in which:

- $S_t \in \mathcal{S}$ is the set of all possible states an agent can be in, known as the state-space,
- $A_t \in \mathcal{A}$ is the set of all possible actions an agent can take, known as the action-space,

- $S_{t+1} \sim p(\cdot|S_t, A_t)$ is the transition probability function which characterises the distribution of next states over state-action pairs,
- $R_t \doteq r(S_t, A_t, S_{t+1})$ is the reward function, and
- $\gamma \in [0, 1]$ is a reward discounting factor.

In episodic RL, the MDP is assumed to be finite. The agent interacts with the environment in a series of discrete timesteps $t \in \mathbb{N}_{\geq 0}$ until the terminal timestep T. A sequence of states, actions and rewards, from $t = 0$ until the terminal state S_T is known as a **trajectory** τ or an **episode**

$$\tau = (S_0, A_0, R_0, S_1, A_1, R_1, \ldots, S_T)\,. \tag{2.3}$$

An agent typically requires many episodes to learn the environment dynamics. The accumulated, discounted reward for an episode is known as the **return**

$$G(\tau) \doteq \sum_{t=0}^{T} \gamma^t R_t\,. \tag{2.4}$$

The *goal* of RL is to learn to maximise the expected return

$$\max \mathbb{E}\big[G(\tau)\big]\,. \tag{2.5}$$

2.4.2 Policies and Value Functions

An RL agent maintains a set of beliefs about its environment in a **policy** $\pi \colon \mathcal{S} \times \mathcal{A} \to [0, 1]$, which is a mapping between states and distributions over actions. Agents select actions to take by sampling from their policy.

$$A_t \sim \pi(\cdot \mid S_t) \tag{2.6}$$

Given a policy, the value of each state $v^\pi(s)$ or state-action pair $q^\pi(s, a)$ is the expected return of that policy

$$v_\pi(s) \doteq \mathbb{E}_{\tau\sim\pi}\big[G(\tau) \,\big|\, S_0 = s\big] \tag{2.7}$$

$$q_\pi(s, a) \doteq \mathbb{E}_{\tau\sim\pi}\big[G(\tau) \,\big|\, S_0 = s, A_0 = a\big]\,. \tag{2.8}$$

These are known as the **state-value** and **action-value** functions. The state and action values can be recursively calculated with the **Bellman expectation equation**,

$$q_\pi = \mathbb{E}_{s'\sim p(\cdot|s,a)}\Big[r(s, a, s') + \gamma \mathbb{E}_{a'\sim\pi(s')}\big[q_\pi(s', a')\big] \,\Big|\, s, a, \pi\Big]\,. \tag{2.9}$$

The agent's goal to maximise its expected return is achieved with the **optimal** state-value v^* or action-value q^* functions, which are the solution to the MDP

$$v_*(s) \doteq \max_\pi v_\pi(s)\,, \tag{2.10}$$

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a)\,. \tag{2.11}$$

This can be recursively calculated with the **Bellman optimality equation** [16],

$$q_*(s,a) = \mathbb{E}_{s'\sim p(\cdot|s,a)}\left[r(s,a,s') + \gamma \max_{a'\in\mathcal{A}} q_*(s',a')\,,\right] \tag{2.12}$$

if the value of all states and actions are known. In practice state and action values are typically not known a priori, and it is infeasible to iterate over all possible values in non-trivial environments. Thus, the value-functions must be estimated from experience. In model-free RL, Monte Carlo methods can be used to estimate value functions and discover optimal policies.
RL algorithms are commonly categorised as being **value-based**, **policy gradient** or a combination known as **actor-critic** methods. These differ by whether the policy is represented explicitly as in the case of policy gradient methods or implicitly as in value-function methods.

2.4.3 Value-Based Methods

Value-based methods aim to directly learn the value of actions in each state with the action-value function $q_\pi(s,a)$ and generate an implicit policy by selecting actions from this value function. These methods iteratively learn an approximate action-value function that converges to the optimal action-value function. Most commonly, actions are sampled greedily, choosing the action with the highest q-value

$$A_t = \arg\max_a q(s,a) \tag{2.13}$$

but can use stochastic methods such as softmax

$$A_t = \frac{\exp(q(s,a))}{\sum_{a'\in\mathcal{A}}\exp(q(s,a'))}\,. \tag{2.14}$$

2.4.4 Policy Gradient Methods

An alternative to value-based methods is to learn a parameterised policy instead of a value function. Learning a policy directly has several advantages. Firstly, policy gradient methods tend to have better convergence properties.

Secondly, learning a policy directly is more feasible in higher dimensional or continuous environments due to a value function's requirement to enumerate every state and maximise over all actions. Lastly, they allow an agent to learn stochastic policies. However, policy-based methods are typically less efficient and suffer from high variance in their updates [16, 26].
As previously discussed (2.5), the goal of RL is to maximise the expected return. With a parameterised policy π_θ, where $\theta \in \mathbb{R}^d$ is a parameter vector, this becomes an optimisation problem to find θ that maximises a performance measure $J(\pi_\theta)$:

$$J(\pi_\theta) = \max_\theta \mathbb{E}_{\tau\sim\pi_\theta}\big[G(\tau)\big]\,. \tag{2.15}$$

One way to optimise the policy is through gradient ascent,

$$\theta_{k+1} = \theta_k + \alpha\nabla_\theta J(\pi_\theta)\,, \tag{2.16}$$

where $\nabla_\theta J(\pi_\theta)$ is the policy gradient and α is the learning rate. That is to compute:

$$\nabla_\theta J(\theta_\pi) = \nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}\big[G(\tau)\big]\,. \tag{2.17}$$

To numerically compute the policy gradient, we first derive an analytical solution, assuming the policy is differentiable. As the environment dynamics are usually unknown, we reformulate without a dependence on the state distribution, known as the **policy gradient theorem** [27]:

$$\nabla_\theta J(\theta_\pi) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(A_t \mid S_t) q_{\pi_\theta}(S_t, A_t)\right]. \tag{2.18}$$

2.4.5 Actor-Critic Methods

Policy gradient algorithms are known to learn more slowly than value-based methods due to high variance in their estimates [16]. Actor-critic methods learn a value function in addition to a parameterised policy, to reduce the variance in gradient updates. These methods learn two models, an actor and a critic. The critic learns a value function, such as the action-value function,

$$q_w(s, a; \theta) \approx q_{\pi_\theta}(s, a)\,, \tag{2.19}$$

through bootstrapping to update the value of preceding states online. The actor seeks to maximise the approximate policy gradient,

$$\begin{aligned} \nabla_\theta J(\theta) &\approx \mathbb{E}\Big[\nabla_\theta \log \pi_\theta(s, a) q_w(s, a)\Big] \\ \Delta\theta &= \alpha\nabla_\theta \log \pi_\theta(s, a) q_w(s, a) \end{aligned} \tag{2.20}$$

guided by the critic.

2.5 Deep Reinforcement Learning

Deep reinforcement learning (DRL) is a subset of RL which uses DNNs to represent an agent's policy and/or value function. The addition of DNNs confers three main benefits over tabular RL. Firstly, deep learning automates feature discovery which reduces domain specific engineering. Secondly, function approximation allows RL to generalise to unseen states. This becomes increasingly important in larger or continuous state spaces where identical observations may never occur. Thirdly, deep learning enables RL to scale to more complex environments which otherwise become constrained by computational limits.
In this section we will review several DRL algorithms which are used in the research chapters.

2.5.1 Deep Q-Networks

This first DRL algorithm to achieve broad success, **Deep Q-Networks** (DQN), demonstrated the effectiveness of the fusion of deep learning and RL, surpassing all previous algorithms in the challenging *Atari* Arcade Learning Environment [28, 29]. DQN is a model-free, value-based, off-policy algorithm, which uses two (commonly convolutional) neural networks to represent an online θ and target θ^- value functions.
DQN's success can be attributed in part to the use of an experience replay buffer to stabilise learning. The experience replay buffer D stores incoming $\langle s, a, r, s' \rangle$ tuples and replays them randomly during learning in order to break the temporal correlations between successive frames. The online neural network is updated with the minibatches of experience sampled from the experience replay buffer. The online network's parameters are updated using the differentiable loss function,

$$\mathcal{L}_t(\theta_t) = \mathbb{E}_{(s,a,r,s')\sim U(D)} \left[\left(Y_t - q(s, a; \theta_t) \right)^2 \right] , \tag{2.21}$$

where Y_t is the approximate target Q-value,

$$Y_t = R_t + \gamma \max_{a'} q(s', a'; \theta_t^-) , \tag{2.22}$$

to minimise the mean squared (or huber) error in the Bellman equation.

The success of DQN inspired a raft of follow-on research, of which several improvements where successfully combined to create the **Rainbow DQN** [30].

2.5.2 Double DQN

DQN is affected by an overestimation bias in the calculation of expected target Q-values. It uses the same action-value to both select and evaluate an action, leading to overoptimistic estimates. **Double DQN** [31] decouples the maximisation step in (2.22), by selecting the next action that maximises the Q-value from the online network,

$$Y_t^{DoubleDQN} = R_t + \gamma \max_{a'} q(S_{t+1}, \arg\max_{a} q(S_{t+1}, a; \theta_t); \theta_t^-)\,. \tag{2.23}$$

2.5.3 Dueling DQN

Dueling DQN specifies a network architecture which factorises the action-value function into state and **advantage** a_π values,

$$q_\pi(s, a) = v^\pi(s) + a_\pi(s, a)\,, \tag{2.24}$$

and has been show to greatly improve DQN performance [32]. One explanation is that for many states it is unnecessary to estimate the value of each action, but for algorithms which use bootstrapping, such as DQN, improving the estimation of state values is of great importance. This factorisation allows the advantage α and value β networks to specialise in learning their respective functions.
A key ingredient of the dueling architecture is ensuring the expectation of the advantage is zero,

$$\mathbb{E}_{a\sim\pi(s)}[a_\pi(s, a)] = 0\,. \tag{2.25}$$

This can be implemented by subtracting the mean,

$$q(s, a; \theta, \alpha, \beta) = v(s; \theta, \beta) + a(s, a; \theta, \alpha) - \frac{1}{|\mathcal{A}|} \sum_{a'\in\mathcal{A}} a(s, a'; \theta, \alpha)\,. \tag{2.26}$$

While Dueling DQN can be implemented with two different neural networks, in practice it is often preferable to use a single network θ with an advantage α and value β heads.

2.5.4 Prioritised Replay

DQN uniformly samples experience tuples from its replay buffer, but not all transitions have the same information content to learn. This process can be improved with the use of a **Prioritised Replay** Buffer [33]. Prioritised Replay samples transitions with probability proportional to their last absolute temporal-difference (TD) error,

$$\delta_t \doteq R_t + \gamma \max_{a' \in \mathcal{A}} q(S_{t+1}, a'; \theta^-) - q(S_t, A_t; \theta) \tag{2.27}$$

$$p_t \propto |\delta_t|^\omega \,, \tag{2.28}$$

where ω is a hyperparameter to correct an importance sampling bias.
New transitions are inserted into the buffer with maximal priority to increase the probability they are sampled.

2.5.5 Multi-Step Learning

The DQN learning update samples transitions containing a single reward value and greedily selects the next action, in a single-step bootstrapping process. Bootstrapping is a process of updating estimates on the basis of other estimates, in this case the estimate of the value of states on estimates of the value of successor states. Multi-step learning, or n-step bootstrapping, increases the amount of steps forward the agent tries to predict and can improve learning efficiency by providing more information per transition [34]. Multi-step learning uses an ***n*-step return**,

$$G_{t:t+n} \doteq \sum_{k=0}^{n-1} \gamma_t^k R_{t+k} + \gamma^n V_{t+n-1}(S_{t+n}) \,, \tag{2.29}$$

where V_t is the estimate of v_π at time t.
As the DQN is an off-policy algorithm, the reward sampled from the replay buffer does not depend on the current policy. The Multi-Step DQN [35] minimises the n-step loss,

$$\mathcal{L}_t(\theta_t) = \Big(\hat{G}_{t:t+n}(\theta^-) - q(S_t, A_t; \theta)\Big)^2 , \tag{2.30}$$

where $\hat{G}_{t:t+n}(\theta^-)$ is the estimate of the multi-step return from the target network,

$$\hat{G}_{t:t+n}(\theta^-) = \sum_{k=0}^{n-1} \gamma_t^k R_{t+k} + \gamma_t^n \max_{a' \in \mathcal{A}} Q_t(S_{t+n}, a'; \theta^-) \,. \tag{2.31}$$

2.5.6 NoisyNets

A key challenge in RL is managing the exploration/exploitation tradeoff. At each timestep, agents must choose between exploring their environments—to better understand its dynamics, or exploiting them—maximising reward. DQN explores during training using an epsilon-greedy strategy, where with probability ϵ it will pick an action uniformly at random. It is commonly the case that some actions are known to perform poorly while several perform well. In these situations epsilon-greedy can disproportionately focus on the single best action. As the action-space increases, the impact of action selection combinatorially explodes and can greatly impede performance.
NoisyNets provide a mechanism to improve exploration, by adding parametric noise to neural network weights [36]. Consider the standard linear layer,

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}\,, \tag{2.32}$$

where $\mathbf{x} \in \mathbb{R}^p$ is the layer input, $\mathbf{W} \in \mathbb{R}^{q\times p}$ the weight matrix and $\mathbf{b} \in \mathbb{R}^q$ the bias. The corresponding noisy layer is,

$$\mathbf{y} = (\mathbf{W}\mathbf{x} + \mathbf{b}) + ((\mathbf{W}_{noisy} \odot \epsilon^{\omega})\mathbf{x} + \mathbf{b}_{noisy} \odot \epsilon^{b})\,, \tag{2.33}$$

where $\epsilon^{\omega} \in \mathbb{R}^{q\times p}$ and $\epsilon^{b} \in \mathbb{R}^q$ are noise random variables. These noisy linear layers replace the final one or two standard linear layers in the agent's network.

2.5.7 Categorical DQN

Value-based methods usually aim to learn the expected return of states or state-action values. An alternative approach is to learn an underlying distribution for each value, known as the **value distribution**. The value distribution z_π is random process which models the discounted return,

$$z_\pi(s,a) \doteq \sum_{t=0}^{T} \gamma^t R_t\,. \tag{2.34}$$

Similar to value functions, the value distribution can be recursively calculated with the distributional Bellman equation,

$$z_\pi(s,a) \overset{d}{=} r(s,a,s') + \gamma z_{a'\sim\pi}(s',a')\,, \tag{2.35}$$

where $\stackrel{d}{=}$ denotes distributional equality. The action-value function is the expectation of the value distribution,

$$q_\pi(s,a) = \mathbb{E}_z[z_\pi(s,a)]\,. \tag{2.36}$$

This distributional perspective was applied to the DQN algorithm, to create the **Categorical DQN**, also known as C51 or the Distributional DQN [37]. The Categorical DQN models the value distribution using a discrete parametric distribution, parameterised by $N_{atoms} \in \mathbb{N}^+$ and $v_{min}, v_{max} \in \mathbb{R}$, whose support is the vector $\mathbf{z}$,

$$\mathbf{z}_i = v_{min} + (i-1)\frac{v_{max} - v_{min}}{N_{atoms} - 1}\,, \tag{2.37}$$

for $\{i \in 1, \ldots, N_{atoms}\}$. On this support is defined an approximating distribution d_t, with probability mass $p_\theta^i(S_t, A_t)$ on each atom i, such that,

$$d_t = (\mathbf{z}, p_\theta(S_t, A_t))\,, \tag{2.38}$$

with the objective to update θ to match the actual distribution of returns. This can be achieved by minimising the KL divergence between d_t and the target distribution:

$$\begin{aligned} d_t' \doteq (R_t + \gamma_t \mathbf{z}, p_\theta^-(S_{t+1}, \arg\max_{a'} q_\theta^-(S_{t+1}, a'))), \\ \mathcal{D}_{KL}(\Phi_{\mathbf{z}} d_t' || d_t)\,, \end{aligned} \tag{2.39}$$

where $\Phi_{\mathbf{z}}$ is an L2-projection of the target distribution onto $\mathbf{z}$.

2.5.8 Rainbow DQN

The Rainbow DQN combined the six DQN extensions previously discussed, Double DQN, Dueling DQN, Prioritised Replay, multi-step learning, Categorical DQN and NoisyNets into a single integrated agent [30]. This was achieved by modifying the Categorical DQN's target distribution with a multi-step variant:

$$d_{t:t+n} = \left(\sum_{k=0}^{n-1} \gamma_t^k R_{t+k} + \gamma^n \mathbf{z}, p_\theta^- \left(S_{t+n}, \arg\max_{a'} q_\theta^-(S_{t+n}, a') \right) \right), \tag{2.40}$$

with loss,

$$\mathcal{D}_{KL}(\Phi_{\mathbf{z}} d_{t:t+n} || d_t)\,. \tag{2.41}$$

Integrating Double DQN's decoupled action selection, prioritised replay and using noisy layers.

2.5.9 Asynchronous Advantage Actor-Critic

Asynchronous Advantage Actor-Critic (A3C) is a model-free, on-policy, actor-critic algorithm [38]. It reduces variance in gradient updates with the use of many (commonly 16–128) actor workers trained asynchronously in parallel. This requires individual actor parameters to be periodically synced with shared global parameters.
A3C uses a two neural network for the policy θ and the value function θ_v. Similar to the Dueling DQN in subsection (2.5.3), it is typically preferable to implement this as a single shared network with policy and value outputs. Each worker contains its own two-headed network plus the global network. Actor networks are updated periodically with the policy gradient method:

$$\nabla_{\theta'} \log \pi(A_t|S_t;\theta') a_\pi(S_t, A_t; \theta, \theta_v)\,, \tag{2.42}$$

with the use of the k-step approximated advantage function,

$$a_\pi(S_t, A_t; \theta, \theta_v) = \sum_{i=0}^{k-1} \Big(\gamma^i R_{t+i} + \gamma^k v_\pi(S_{t+k}; \theta_v)\Big) - v_\pi(S_t; \theta_v)\,. \tag{2.43}$$

Additionally, entropy regularisation is added to improve exploration:

$$\nabla_{\theta'} \log \pi(A_t, S_t; \theta')(R_t - v_\pi(S_t; \theta_v)) + \beta \nabla_{\theta'} H(\pi(S_t; \theta'))\,, \tag{2.44}$$

where H is the entropy and β the entropy strength hyperparameter.

Chapter 3

Coevolutionary Deep Reinforcement Learning

3.1 Overview

This chapter presents an improved algorithm for training RL in competitive environments. A popular method for this, **self-play**, has not previously been studied in depth. We compare several self-play variants and find that the use of competitive pressures within a larger training population can improve performance.
This research was presented at the IEEE Symposium Series on Computational Intelligence 2020 in Canberra, Australia [39].

3.2 Preliminaries

3.2.1 Multi-Agent Reinforcement Learning

Historically, RL was developed from a single-agent perspective [40]. **Multi-agent reinforcement learning** (MARL) investigates methods for operating RL in systems with multiple rational, decision-making agents, where these other agents could be RL, other intelligent software agents, or human. Learning in multi-agent environments is challenging as an agent's optimal policy is dependent on the behaviour of the other agents in the environment. Developing successful MARL algorithms is of interest as many real world problems are multi-agent in nature, such as self-driving cars, supply chain logistics, smart grids and packet routing.

MARL is a broad and complex domain, encompassing a wide variety of problems. A useful property to identify in multi-agent system is the level of **cooperation** and **competition** between agents, as it impacts feasibility of particular classes of MARL algorithms to the problem at hand.

3.2.2 Competitive Training Methods

Fully competitive environments are in some sense more straightforward than cooperative or mixed conflict, as agent goals are generally non-aligned and instead seek to maximise their individual gains. This allows more developed single-agent RL algorithms to be employed, by treating the other agents as part of the environment.

Several methods have been proposed for training RL in competitive environments. In this section we group them into four categories: **expert opponents**, **offline reinforcement learning**, **game theoretic** and **self-play**. Prior research has combined these methods, for example, initially training with offline RL, then improving performance with self-play or game theoretic methods [41, 42].

Expert Opponents

One competitive training method is to train RL agents against an expert opponent, such as a human or software agent. The main advantage of this method is that specialised algorithms are not required. Single-agent learning algorithms and training methods can be used, by treating the opponent as part of the environment and playing repeated games against them, until learning to outperform them.

However, there are several disadvantages to this method. Firstly, in many situations expert opponents are not available or feasible. This could be because one does not exist, due to budgetary limitations, or the opponent is not able to provide sufficient training data. For example, human opponents are generally infeasible as RL can require billions or trillions of samples to learn in more complex environments [42, 43]. Secondly, researchers have observed that the performance of trained agents tends to be limited by the complexity of the environment [44, 45]. Thirdly, using this method agents may only learn a single strategy profile and struggle to generalise. Lastly, due to the

curse of dimensionality agents may struggle to learn in higher dimensional environments or against opponents that dominate them.

Offline Reinforcement Learning

Offline RL, also known as batch RL, is a method for learning from a fixed dataset, similar to supervised learning. This dataset could be constructed from expert human games [41, 42]. In contrast, "online" RL is responsible for its own data gathering process, in which it influences the data it receives by the actions it selects.

An advantage of offline RL is its sample efficiency, particularly when learning complex action sequences. This can be particularly challenging for RL, when rewards are temporally uncorrelated and combinatorial complexity grows with the sequence length. Offline RL can learn to imitate complex behaviours in the dataset directly, without needing to discover them itself.

However, there are several disadvantages. Firstly, datasets can be difficult to obtain. They might not be available, be expensive or difficult to collect, or of poor quality. Secondly, similar to the expert opponents method, the performance of agents trained by offline RL can be limited by the complexity of the dataset [44, 45]. Lastly, it is difficult to procure training datasets that enumerate all possible scenarios agents could encounter, possibly leading to overfitting and generalisation issues.

Game Theory

Much of RL is based on the MDP, of which assumes the environment is stationary. To address this shortcoming one solution is to incorporate game theory, which models the strategic interactions between multiple agents [46, 47]. A game theoretic RL training method is fictitious self-play (FSP) [48]. FSP is a sample-based implementation of generalised weakened fictitious play. It computes a policy by sampling episodes of self-play games, then at each iteration computing a mix between the best response to the uniform mixture of opponent policies, and the agent's own average strategy. The best response is computed through Fitted Q Iteration [49].

Self-Play

Self-play allows RL to be trained without external guidance by competing with itself for limited rewards. In its simplest implementation a single agent plays as both players in a two-player, zero-sum game, in which repeated games are played until the agent reaches a satisfactory performance level. Advantages include that it does not require expert knowledge, does not impose a performance ceiling and that it encourages agents to develop novel and emergent behaviours, free from human biases.
Unfortunately, self-play is not without drawbacks. It is known to be less sample efficient and suffer more unstable learning dynamics than other training methods. This is in part due to a non-stationary learning problem where agents are simultaneously acting as both student and teacher. Consequently, we look to address these issues with the use of coevolutionary algorithms.

3.3 Background

3.3.1 Self-Play

Self-play training for agents in competitive environments traces roots back to early AI research. In 1950 Shannon theorised that a computer could learn to master chess through an alternating maximisation and minimisation process whereby the program traversed the game tree and calculated optimal moves [50]. Shannon's application of the minimax procedure influenced the creation of Samuel's checker agent [51], a seminal machine learning program which learnt to play checkers to an amateur level. Samuel improved his program to use a *rote learning* procedure to search through the game tree and record the utility of game states as scored by an evaluation function. This search procedure was conducted by playing many games against another version of itself, in what can be described as prototypical self-play RL.
Several decades later, self-play was critical in the training of an RL backgammon agent, TD-Gammon [52]. TD-Gammon was a major breakthrough, achieving a grand master level of play, with minimal expert knowledge required. TD-Gammon was trained with a single agent controlling both players in the game and using the actions of both players to update its value-function. This method has since seen widespread use in the game playing AI literature and we refer to it as *single-agent self-play*. Further research into the success of TD-Gammon

discovered that the stochasticity of backgammon dice rolls may have assisted exploration and minimised the training instability of single-agent self-play [53].

More recently, improvements to self-play have been integral for some of the most impactful RL algorithms. It was discovered during the development of AlphaGo, an agent which learnt to play the board game Go at a superhuman level, that playing against a randomly selected previous iteration of the policy network stabilised training and prevented overfitting the current policy [41]. This idea of maintaining a pool of previous versions to diversify self-play training data has proved fruitful. AlphaGo Zero, an evolution of AlphaGo trained entirely from self-play, kept the current best performing agent as a champion, responsible for generating training data when trained against a pool of uniform randomly selected previous versions [44]. Agent populations have been further developed to introduce opponent sampling strategies [45]. In this paper, we refer to the method of training the current best performer against previous best performers as *champion self-play*.

Population-based methods propose to evolve a diverse pool of candidate solutions, instead of spawning from a single individual. These methods foster diversity and recognise that some strategies are more effective earlier in the learning process than later on when the meta-game may have evolved. By developing a range of policies, strategies that are initially weaker may prove more effective as new skills are mastered. Diverse opponents help ensure more robust policies and reduce overfitting. However, population-based methods can generate significant redundant computation, such as the training of suboptimal population members. An example of population-based methods, Population-based training, is a hyperparameter optimisation algorithm which focuses resources towards the best performing candidates [54]. The authors then built upon population-based training to develop the algorithm FTW, capable of learning to cooperate in a team-based first-person shooter video game [55]. FTW trained agents through self-play with a two-tier optimisation process in which the inner the loop optimised individual agents using a reward constructed to maximise team win probability. The outer loop used population-based training to optimise agent hyperparameters and update neural network weights if the agent's win probability dropped below a threshold.

Self-play has been combined with game theoretic optimisation and the grandmaster level *Starcraft 2* agent, AlphaStar, is one of the best known examples of this [42]. AlphaStar used supervised learning to initially train a diverse population

of agents with a range of different strategy profiles. After being trained to strong level, the agents then further improved their policies through a combination of population-based self-play and game theoretic learning.

3.3.2 Coevolutionary Algorithms

Evolutionary algorithms (EA) are population-based optimisation methods inspired by Darwinian evolution. EAs begin with a population of candidate solutions and a fitness measure, which is a heuristic for a candidate solutions distance to their objective. Small mutations are made to the population and candidates are evaluated using the fitness measure, with "fitter" changes preferred to become the next generation's parents. This facilitates a "survival of the fittest" evolutionary process which incrementally drives the fitness of the population towards the desired objective. A high-level outline of the procedure is provided in Algorithm 1.

Coevolutionary algorithmss (CoEA) are a subset of EAs which use subjective fitness measures. Subjective fitness is measured relative to other individuals within the population(s), in contrast to EAs objective fitness measures which are measured independent of the population. Subjective fitness measures allow CoEAs to more successfully operate in larger, higher dimensional search spaces than objective measures as they encourage candidates to focus on areas which are more individually beneficial and use competitive adaption to incrementally combine these changes to drive objective progress [56].

Algorithm 1 Abstract Evolutionary Algorithm

```
1:  procedure TRAIN
2:      Initialise population 𝒫
3:      Evaluate 𝒫
4:      while not terminated do
5:          Select parents from 𝒫
6:          Generate offspring from parents
7:          Evaluate offspring
8:          Select survivors for new 𝒫
9:      end while
10:     return 𝒫
11: end procedure
```

3.3.3 Evolutionary Reinforcement Learning

Evolutionary computing and RL are influential AI research directions that have seen significant success, particularly in open-ended learning problems, such as games. This success has lead to a recombination of ideas from these fields, creating the subfield of evolutionary reinforcement learning (ERL). Neuroevolution is a prototypical example of ERL which uses EAs instead of gradient descent to train neural networks [57]. In contrast with gradient descent, which minimises a loss function over labeled training data to learn neural network parameters, neuroevolution evolves the parameters towards a specified behaviour. This allows neuroevolution to be applied more directly to open-ended learning problems [58, 59]. Recently, ERL methods have shown great promise, outperforming gradient based algorithms on some benchmarks. For example, it was found that evolution strategies are a highly scalable alternative to gradient-based DRL that reduced the training time from hours to minutes [60]. Another notable algorithm, collaborative evolutionary reinforcement learning (CERL), demonstrated that a combination of gradient-based RL and neuroevolution EAs outperformed its constitute algorithms on the challenging Mujoco humanoid benchmark [61]. CERL utilises a population of EA actors to gather experience and a portfolio of TD3 [62] agents with distinct hyperparameters, sharing a single replay-buffer. CERL combines the advantages of gradient-based and gradient-free learning, by periodically copying the network weights from the TD3 portfolio over the weights of underperforming EA population members.

3.4 Coevolutionary Reinforcement Learning

Coevolutionary Learning and REinforcement (CLaRE), improves upon population-based approaches for training RL with the use of competitive coevolution. CLaRE does not place any limitation on the type of RL algorithm used (e.g. value/policy-based, on/off policy, etc.).
CLaRE is initialised with a population $\mathcal{P}$ of randomly initialised RL agents. The algorithm consists of an evolutionary outer loop and an inner training loop. An algorithm epoch is a single iteration of the outer loop. The inner training loop asynchronously trains a generation of RL agents for a predetermined period. In this generation, opponent pairs are sampled and play complete episodes, updating their policy and/or value functions as dictated by their individual learning algorithms. The training period acts like the mutation

and recombination function in classical evolutionary algorithms—or perhaps closer to the original biological notions—that evolutionary adaptations by one agent affect the survivability of its peers. After each generation, agent fitness f is calculated using a chosen subjective fitness measure on the results of the training. The fitness is used to probabilistically select survivors from the population. The population is updated by cloning survivors to replace underperforming agents, completing a coevolutionary epoch. The outer evolutionary loop continues until a terminating condition is met, such as total number of agent steps or an evaluation threshold. Algorithm 2 describes the approach in more detail.

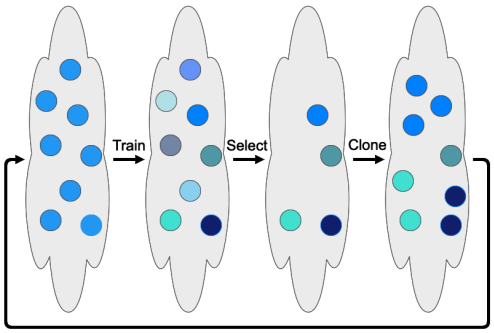

Figure 3.1: **The coevolutionary training cycle** begins with a population of reinforcement learning agents. The agents are trained in competition with themselves for a period, in which they all learn slightly different policies. The fittest individuals are selected to become the parents for the next epoch.

3.4.1 Training Procedure

Within the inner training loop, generation length and agent sampling strategy are hyperparameters which tradeoff the degree of population mutation versus evolution speed. Generation length can be measured by the number of steps experienced by agents or other measures such as population diversity via a niching or fitness sharing measure. Longer epochs provide more opportunity for individual mutation at the cost of less frequent balancing of dominant strategies. Optimal epoch length is environment specific and strikes a balance between providing sufficient individual progression and metrics to accurately differentiate fitter individuals versus sufficient evolutions to filter out weaker strategies and complexify behaviour.

Algorithm 2 Coevolutionary Learning and REinforcement

```
1: procedure TRAIN
2:     Initialise population 𝒫
3:     while not terminated do
4:         Initialise epoch metrics m
5:         while not terminated do
6:             a ← sample_agents(𝒫)
7:             update r with play_episode(a)
8:         end while
9:         f ← measure_fitness(m)
10:        s ← select_survivors(f)
11:        update 𝒫 with s
12:    end while
13:    return 𝒫
14: end procedure
```

3.4.2 Survivor Selection

Survivors are selected every generation to be cloned to replace underperforming or dominated agents. This increases competition amongst the population, creating extrinsic motivation for agents to evolve new capabilities in order to regain competitive advantage. The agent population is evaluated with a subjective fitness measure on metrics collected during agent training to provide a weakly ordered set of agent ratings. Elo ratings [63] were used as the fitness measure. Elo ratings determine the significance of a victory based on the relative strength difference of the opponent, rewarding victories over higher ranked agents more favourably than lower ranked agents. For each agent $i \in \mathcal{P}$ a rating r_i is maintained. The ratings r are initialised with an arbitrary base rating, for every $i \in \mathcal{P}, r_i \leftarrow 1200$, which is updated with the game result after each episode. To update r, the probability of player i winning is calculated by,

$$\mathbb{P}(i \text{ beats } j) = \frac{1}{1 + e^{-(r_i - r_j)/400}}\,, \tag{3.1}$$

and use the win probability to update the ratings for both players,

$$\begin{aligned} r_i &:= r_i + k(s_i - \Pr(i \text{ beats } j)) \\ r_j &:= r_j + k(s_j - (1 - \Pr(i \text{ beats } j)))\,, \end{aligned} \tag{3.2}$$

where the score s for each player is given by,

$$s_i = \begin{cases} 1 & \text{if player } i \text{ won} \\ 0.5 & \text{if draw} \\ 0 & \text{if player } i \text{ lost}\,, \end{cases}$$

and use a k-factor, $k = 32$. To select survivors, r is converted into a probability distribution and survivors sampled from it. For this process a weighted softmax function is used,

$$f(r) = \frac{e^{r_i/\tau}}{\sum_{j=1}^{k} e^{r_j/\tau}}\,, \tag{3.3}$$

with temperature τ, and sample survivors from it. We found that the stochasticity from sampling agents added stability to our algorithm.

3.4.3 Related Work

A key distinction between CLaRE from other ERL algorithms is that it only uses the constituent learning algorithm to train population members. For example, if CLaRE is parameterised with a gradient-based DRL algorithm, it will only train with gradient descent. Furthermore, CLaRE does not utilise ECs to gather experience. CLaRE treats the RL process as the evolutionary mutation process. An additional point of difference is that CLaRE uses subjective fitness measures, which allows the algorithm to be more generally applied to problems when the optimal behaviour is not known. For example, CERL uses the cumulative sum of rewards received in a rollout as the fitness measure [61]. In a zero-sum environment containing multiple agents this may not satisfactorily differentiate agent performance, such as comparing wins against a weaker agent with a win against a stronger agent.

3.5 Method

In our evaluation, all RL training methods trained the same learning algorithm, a Rainbow DQN [30]. Our implementation of Rainbow DQN differs in several ways from the reference implementation, of which the most standout changes are that we do not use the Categorical DQN [37], Noisy Networks [36] or multi-step learning [34] components. Another notable change is that we used

a weighted softmax,

$$A_t = \frac{\exp(q(s,a)/\tau)}{\sum_{a' \in \mathcal{A}} \exp(q(s,a')/\tau)}\,, \tag{3.4}$$

to probabilistically sample actions instead of epsilon-greedy,

$$A_t = \begin{cases} \arg\max_a q(s,a), & \text{if } X \sim \mathcal{U}(0,1) > \epsilon \\ A \sim \mathcal{U}(0, |\mathcal{A}|), & \text{otherwise} \end{cases}. \tag{3.5}$$

This change was primarily made as we found it performed better during our initial agent tuning. Additionally, epsilon-greedy cannot be naively implemented in environments with illegal actions (as described in the following section), due to the entire action-space not being a legal move at every timestep.

3.5.1 Evaluation Environment

Our evaluation was conducted in the two-player, fully competitive, zero-sum, perfect information board game *Connect Four*. The canonical version of Connect Four is played on a board with seven columns and six rows. Players take turns dropping coloured discs in one of the seven columns with the aim of being the first player to create a straight line of four consecutive discs of their colour in either a vertical, horizontal or diagonal direction. Despite the simple sounding rules, Connect Four has a state-space complexity of approximately $4.5 \cdot 10^{12}$, which makes learning optimal behaviour sufficiently challenging. The environment configuration parameters are listed in Table 3.1.
Connect Four was chosen as our evaluation environment for two main reasons. Firstly, to restrict our problem domain as a pedagogical step towards learning in more complex environments and conflict dynamics. Connect Four is comparatively computationally efficient and has a maximum game length is 42 turns. This made our experiments computationally feasible without access to specialised hardware.
Secondly, Connect Four was chosen as tree-based search algorithms such as MCTS perform strongly in it. The strength of tree-based solvers in our environment provides an objective, quantitative measure with which to reliably benchmark training progress. The ability to objectively analyse performance is an important consideration when operating in a subjective domain as an agent's relative strength within a population may not correlate with external notions of their strength.

Table 3.1: Environment Configuration

Parameter	Value
Board height	6
Board width	7
Win length	4

3.5.2 Observation Space

Connect Four is a two-player, alternating turn board game with perfect information. At each timestep t, for each agent i, the environment provides an observation $S_{t,i}$, including the agent not playing this turn. This implementation decision was made for three reasons. Firstly, this provides agents with a history of individual actions. Secondly, for compatibility with *RLlib* [64]. RLlib required an action, reward, terminal, next state and information values for each observation, at each timestep. As Connect Four is an alternating turn game, not providing observations for both players lead to a situation where the non-active player did not receive the final reward and terminal values. Lastly, it was more convenient to incorporate with MCTS, which uses an environment model to conduct rollouts to simulate both players. In later experiments we used our own custom RL framework and experimented with only providing observations to the active player. However, as this only provided a minor execution speed improvement and no discernible win-rate performance benefit, we remained with dual observations.

Each agent observation $S_{t,i}$ consists of the **board state** and an **action mask** which are described below.

Board State

The board state is encoded as into a board height $\times$ width two-dimensional matrix representation as can be seen in Fig. 3.2. The player's discs are encoded with a 1, the opponents with a 2 and available spaces with a 0. Traditional CNNs architectures usually expect the height and width dimensions to be equal. To achieve this, we pad the top row with 3s.

We initially experimented with a flattened representation in combination with a fully connected neural network, but it did not perform as well. We hypothesise that this is due to CNNs improved local spatial coherence.

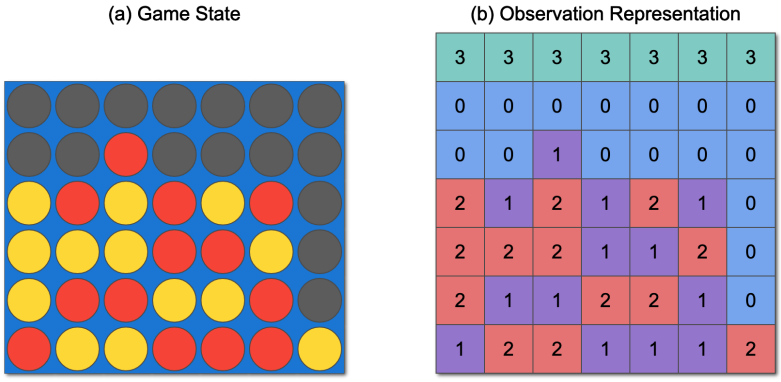

Figure 3.2: **Connect Four Observation Representation** (a) Shows the game state, with a red and yellow player, where black indicates available spaces. (b) The observation representation for the red player. The player's red discs are encoded with a 1, the opponents yellow discs with a 2, unfilled columns with a 0, and padding with a 3.

Action Mask

The Connect Four rules define illegal actions, for example discs cannot be placed in full columns. A common solution for dealing with invalid actions is to utilise action masking. An action mask is a binary vector of equal dimension to the action space, in which a one indicates a legal action and a zero an illegal action. A depiction of a possible game state and the corresponding action mask is shown in Fig. 3.3.
A trick to implement action masking in popular machine learning frameworks such as Tensorflow or Pytorch is to use their slightly non-mathematical behaviour of $\log 0 = -\infty$. This allows the action mask vector $\mathbf{m}$ to be added to the Q-values $q(s, a)$,

$$q'(s, a) = q(s, a) + \log(\mathbf{m}) \,, \tag{3.6}$$

setting masked action to $-\infty$, which when combined with softmax for action sampling, sets their $\Pr = 0$.

3.5.3 Action Space

Our implementation of Connect Four required both players $i \in \mathcal{I}$, at every timestep t, to select an action $A_{t,i}$. The global action space $\mathcal{A} = \{0 \dots c\}$, where c is the number of columns, is the set of all possible actions. At each state, only a subset of actions are legal, as per the action mask. For the

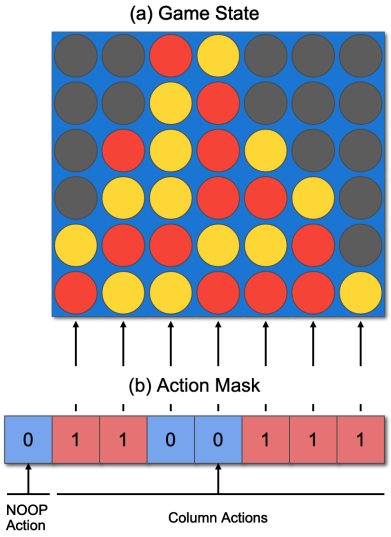

Figure 3.3: **Connect Four Action Mask** (a) Shows the game state, with a red and yellow player, where black indicates available spaces. (b) The action mask for the red player. As defined by the action space, the first action is the NOOP action, followed by the board game columns. Legal actions are encoded with a 1 and illegal actions with a 0.

inactive player, the only legal action is the "NOOP" action $A_{t,i} = 0$. For the active player, the legal actions are a subset of the indices of the columns $A_{t,i} \subseteq \{1 \dots c\}$. A diagram of the action space is shown in Fig. 3.4.

3.5.4 Reward Function

We used a sparse, ternary reward function. That is all zero, except for the terminal timestep T. At each timestep t, for each player i, the reward $R_{t,i}$ is calculated:

$$R_{t,i} = \begin{cases} 0, & t \neq T; \\ 1, & \text{if player } i \text{ won}; \\ 0, & \text{if draw}; \\ -1, & \text{if player } i \text{ lost}. \end{cases} . \tag{3.7}$$

3.5.5 Neural Network Architecture

The DQN agent used two identical CNNs for target and online networks. Each network used three convolutional layers, feeding into two dense layers.

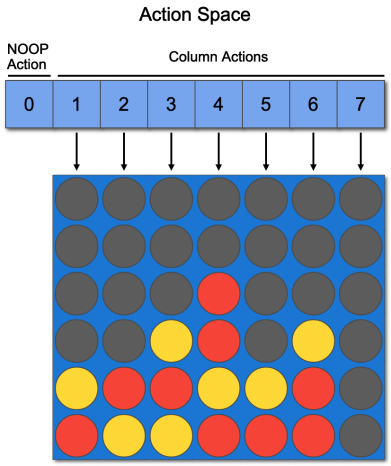

Figure 3.4: The **Connect Four Action Space** contains a NOOP action in position zero, followed by columns indices.

The first hidden layer convolves 16 2x2 filters with stride 1, the second hidden layer convolves 32 2x2 filters with stride 1 and the final hidden layer convolves 64 3x3 filters with stride 2. Valid padding and Xavier weight initialisation [65] is used for our convolutional layers. The convolutional layers feed into two linear network heads to estimate the state value function and advantage logits. Leaky ReLU activation is used between all layers. The online neural network is trained by gradient descent with the Adam optimiser using a learning rate $\alpha = 5 \cdot 10^{-5}$ which is not annealed, on a minibatch of 32 transition tuples every 4 timesteps as per the original DQN [28]. The target and online networks are synchronised every 500 timesteps. Network updates begins after 1,000 timesteps.

3.5.6 Agent Configuration

Agents use reward discounting factor of $\gamma = 0.99$. An action sampling softmax temperature $\tau = 0.02$ was used. To encourage greater exploration in early stages, the temperature parameter was linearly annealed from $1.0 \rightarrow 0.02$ over the first 10,000 timesteps, using the following piecewise linear schedule,

$$\tau = \begin{cases} \tau_{min} + (\tau_{max} - \tau_{min}) * (1 - \frac{T}{p})), & T \leq p \\ \tau_{min} & \text{otherwise} \end{cases}, \tag{3.8}$$

Table 3.2: Rainbow DQN Hyperparameters

Parameter	Value
Discounting factor γ	0.99
Learning start	$1,000$ steps
Adam learning rate α	$5 \cdot 10^{-5}$
Batch size	32
Online network update period	4 steps
Target network sync period	500 steps
Exploration softmax temperature τ	$1.0 \rightarrow 0.02$
Exploration annealing period p	$10,000$ steps
Replay buffer capacity	$50,000$
Replay buffer prioritisation exponent ω	0.6
Replay buffer importance sampling β	0.4

where $\tau_{max} = 1.0$ is the maximum temperature, $\tau_{min} = 0.02$ is the minimum temperature, T is the the agent's current total training timesteps (not including any evaluation timesteps), and $p = 10,000$ is the exploration period (in total timesteps). During evaluation, we continue to use softmax action selection with $\tau = 0.02$, instead of acting greedily.

A prioritised replay-buffer with a capacity of 50,000, prioritisation exponent $\omega = 0.6$ and an importance sampling exponent $\beta = 0.4$ was used. These hyperparameters, shown in Table 3.2, were tuned by training our DQN directly against a MCTS opponent.

3.6 Evaluation

3.6.1 Evaluated Algorithms

To evaluate CLaRE, this paper compares with two common self-play implementations in terms of sample efficiency and final performance. The first self-play algorithm we compare with, single-agent self-play, uses a single learning agent to play all players in a game. The second algorithm, champion self-play maintains a learning agent and a pool of previous champions from which it randomly samples training opponents. At the end of each epoch, if the learning agent wins more than 60% of games, it becomes a champion and a copy is added to the pool of previous champions.

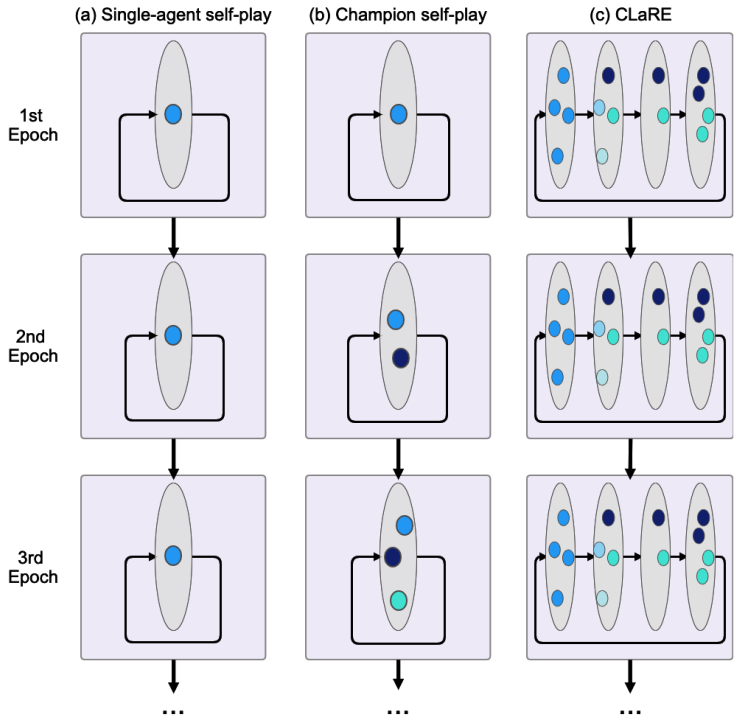

Figure 3.5: **Evaluated algorithms.** The main evaluation compared CLaRE with the two most commonly used self-play implementations. (a) Single-agent self-play trains a population of one agent. (b) Champion self-play maintains a pool of previous champions from which it randomly selects opponents. (c) CLaRE utilises a coevolutionary training mechanism which randomly selects the next generation with respect to the agents fitness.

3.6.2 Methodology

We evaluated our algorithm on the board game Connect Four against several self-play implementations with varying population sizes to assess the effect on training stability, final performance and sample efficiency. Each experiment periodically evaluated all agents being trained against Monte Carlo tree search (MCTS) [66, 67] opponents.

Each algorithm was trained for 250 epochs per experimental run. Epochs consist of a training and evaluation phase. Experience generated during evaluation phases was not used by any agents to learn from. Training phases ran for 10,000 steps per agent and evaluation phases appraised each learning

agent for 50 episodes per MCTS opponent, in order to reduce the noise in evaluations. As there is a positional advantage to the first player to make a move in Connect Four, the starting player was randomly assigned during training and evenly distributed during evaluation (25 episodes per player position). Each algorithm was evaluated with three experimental runs and mean results reported.

3.6.3 Results

Our evaluations demonstrated that CLaRE significantly outperformed both self-play implementations, both in terms of final performance and sample efficiency, achieving an almost 80% evaluation win-rate, approximately 15% above the next leading method as can be seen in Fig. 3.6. Importantly, CLaRE showed strong convergence properties as opposed to single-agent self-play which failed to converge. The occurrence of strategy cycles is a common cause of convergence failure and is likely a contributing factor here. The training curves can be seen in Figs. 3.7 to 3.9.

CLaRE showed more stable training dynamics than other self-play methods, with less variation in evaluation performance between epochs. In particular single-agent self-play appeared very unstable with large fluctuations between evaluations. This indicates that single-agent self-play was overfitting and the policy failed to generalise.

Agents trained with CLaRE present noticeably reduced maximum Q-values than the other self-play methods and greater neural network loss and temporal difference error values than champion self-play. Hypothetically this is evidence of increased competition within our algorithm's population.

3.6.4 Ablation Study

An ablation study was conducted to understand the contribution of elements the CLaRE algorithm. The ablation investigated the contribution of four different survivor selection mechanisms and four different population sizes on final evaluation win percentages. The ablations ran a single training run for 250 epochs and report the mean evaluation win rate from the final epoch. As can be seen in Fig. 3.10, the combination of Elo ratings as a fitness measure with softmax survivor sampling proved most effective. Larger population

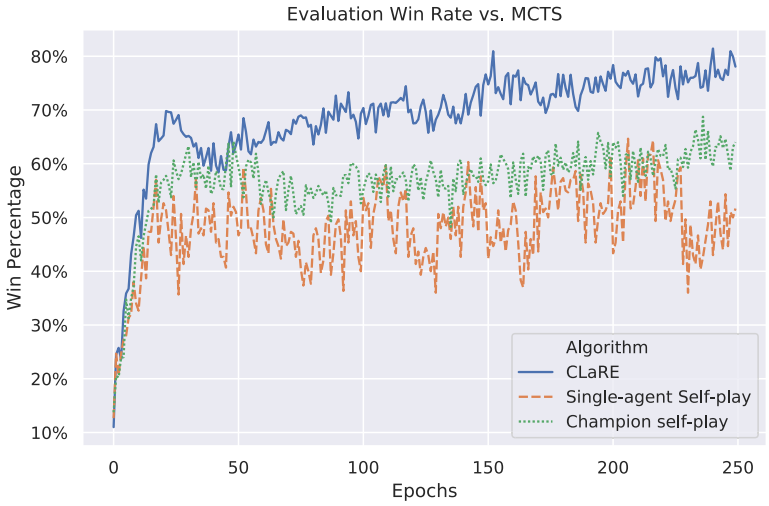

Figure 3.6: **Evaluation Win Percentage** for each training method as a function of the number of epochs. We compare our CLaRE algorithm (blue) to single-agent and champion self-play training methods and report the average of three runs. The evaluation win rate is the mean percentage of games won by the training population against MCTS opponents in the evaluation phase of each epoch.

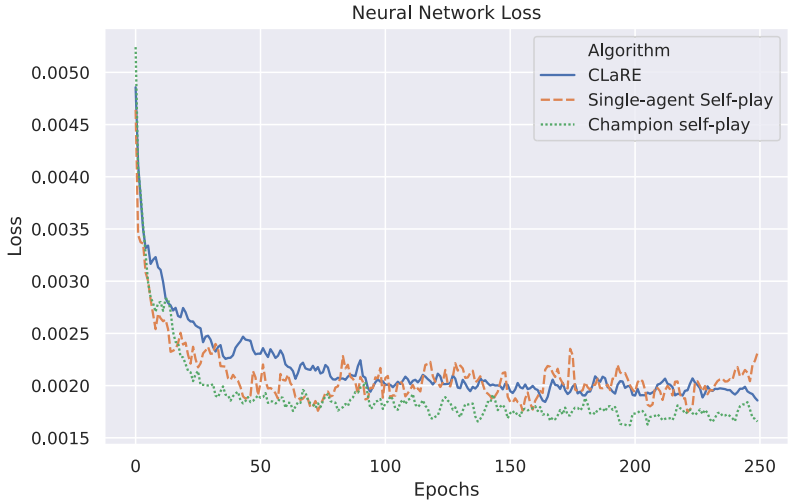

Figure 3.7: **Neural Network Loss** for each training method as a function of the number of epochs. The mean loss in updates to the online neural network during training.

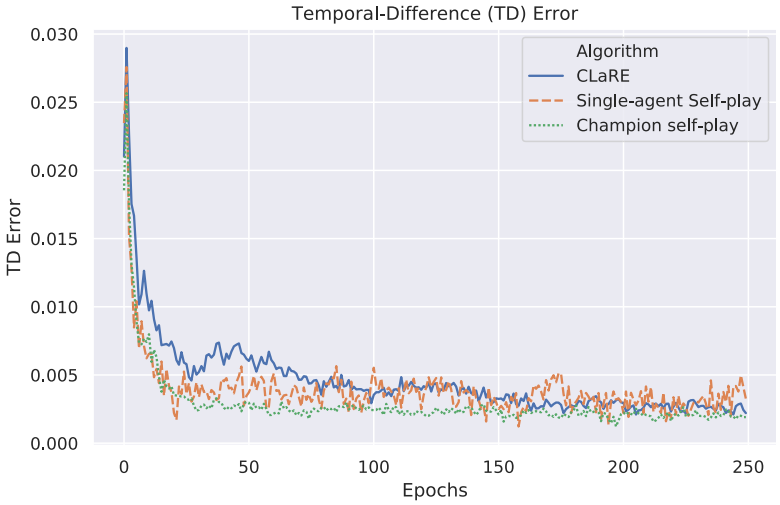

Figure 3.8: **Temporal-Difference Error** for each training method as a function of the number of epochs. The temporal-difference (TD) error in updates during the training phase. TD error indicates how "surprising" a transition is and measures the difference between the online neural network's Q-values and the target network's estimate.

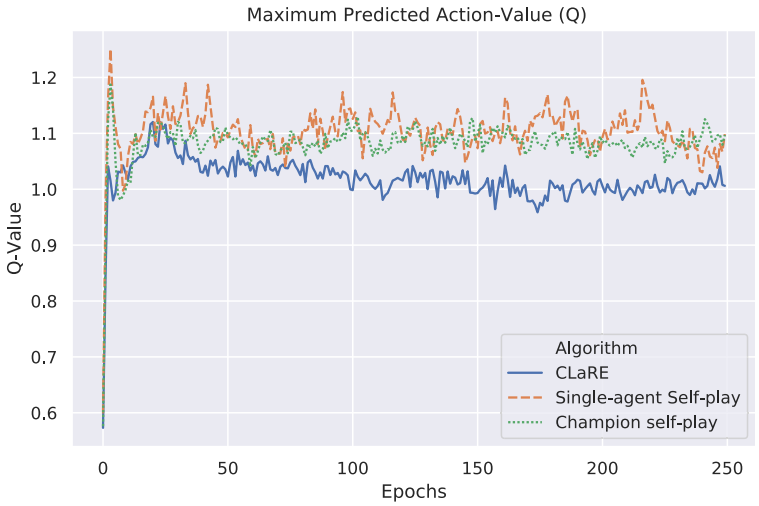

Figure 3.9: **Maximum Q-Values** for each training method as a function of the number of epochs. The maximum action-value (Q) of each transition during training. The max Q-value indicates the potential utility an agent sees in each state.

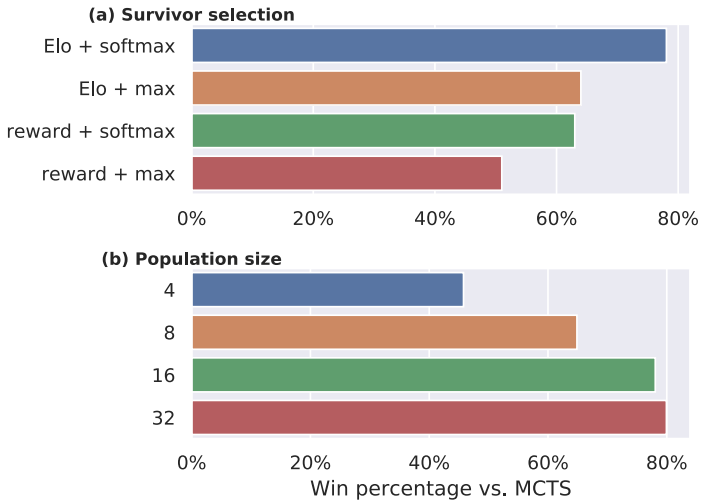

Figure 3.10: **Ablations for CLaRE.** These ablations compare the final evaluation performance of CLaRE after 250 epochs. (a) Comparing different survivor selection mechanisms. We compare the combination of Elo ratings versus training rewards mean as a heuristic for fitness in tandem with softmax survivor sampling versus simply taking the top performing agents. These experiments used a population size of 16 and sampled 8 survivors. (b) Comparing the effects of population size on win rate. These experiments used Elo + softmax survivor sampling.

sizes increased performance, however due to computational limits our main evaluation used a population size of sixteen agents.

3.7 Conclusion

This chapter has demonstrated how coevolutionary principles can be utilised to optimise the training of a population of RL agents. We have described a generalised algorithm for competitive DRL that builds upon existing population-based approaches by promoting competition within an agent population. These competitive pressures have the effect of generating a curriculum to train agents more efficiently and robustly. Our experiments have demonstrated that our coevolutionary training algorithm, CLaRE, results in higher final performance and faster convergence speed than other self-play training algorithms. This chapter draws together existing self-play research and provides a framework to describe different self-play methods. Much of the existing self-play literature

is not the main focus of the research it appears in and/or is part of a larger suite of improvements, making it difficult to ascertain the impact of different methods. In contrast, this research examines the effectiveness of different self-play algorithms in isolation.

While the experiments have focused on value-based learning algorithms, it is expected that similar gains would be observed with policy gradient methods. In the future we hope to extend our analysis to more complex multi-agent environments.

Promising future research directions include fostering greater diversity within the agent population such as with niching, fitness-sharing or quality diversity methods. It is anticipated that incorporating improvements such as these could result in powerful learning algorithms, able to teach themselves from first principles.

Chapter 4

GymD2D: A Device-to-Device Underlay Cellular Offload Evaluation Platform

This chapter presents an open-source network simulator and evaluation platform developed for D2D cellular offload RRM research. In contrast to Chapter 3, which investigated *competitive* multi-agent RL, this chapter instead approaches cellular resource management optimisation as a *cooperative* multi-agent problem. This change in direction came about due to a change in supervisors.
This research was presented at the IEEE Wireless Communications and Networking Conference 2021 in Nanjing, China [68].

4.1 Overview

Challenges facing D2D cellular offload researchers include insufficient tooling, difficulty comparing research and a lack of established benchmarks. Prior research uses a variety of different, and usually custom, network simulators and system models. This makes it difficult to compare algorithms, verify and have confidence in reported empirical results. For example, when two papers use different terminal configurations or path loss models, the results are not directly comparable and may require the reader to reimplement the algorithms and run their own comparison.
In this chapter we present *GymD2D*, a network simulator and evaluation platform for RRM in D2D underlay cellular offload. GymD2D provides convenient

abstractions to aid researchers in quick prototyping of resource allocation algorithms. The toolkit allows users to programmatically configure the environment to simulate a wide variety of scenarios. GymD2D has been designed with extensibility as a core design principle, allowing users to override and extend its behaviour to meet their research needs. It has been developed in the Python programming language to allow users to leverage its extensive ecosystem of scientific computing packages. The open-source nature of GymD2D centralises development effort and avoids the redundant work of individual researchers creating their own simulators. This puts more eyes on bug fixing, provides a more stable platform and increases confidence in reported empirical results. GymD2D reduces entry barriers for junior researchers, helps researchers from other disciplines to cross-pollinate ideas easier and more generally increases participation.
Our software package is open-source and is provided to the community under a MIT licence at https://github.com/davidcotton/gym-d2d.

4.2 Preliminaries

4.2.1 Device-to-Device Communication

In this section we provide an overview of the D2D RRM literature to situate the reader as to the requirements of the platform. Firstly, we highlight the most common optimisation problems. Secondly, we analyse key differences across simulators, paying special attention to simplifying assumptions frequently observed. Thirdly, we survey the optimisation algorithms used for resource allocation. Finally, we outline limitations of existing research, providing direction for the simulation requirements of future work.

Optimisation Problems

RRM is an optimisation problem where the objective is to utilise radio resources as efficiently as possible. D2D RRM has been proposed for improving spectral efficiency, energy efficiency and quality of service. In these uses cases, the objective can be to optimise data rates, throughput, capacity, SINR, power consumption, energy efficiency or latency [2, 69].
D2D systems can be centrally managed by the network operator, manage interference autonomously or use a hybrid control mode which aims to combine

the benefits of both. The choice of control mode limits the applicability of certain algorithms which may only be feasible in centrally managed paradigms.

Simulation Models

D2D cellular offload typically investigates networks using orthogonal frequency division multiple access (OFDMA), communicating on licensed bands using underlay networking. The most common scenario is a single MBS surrounded by many randomly positioned CUEs and DUEs. It is generally assumed that cellular systems are under full load and each RB is allocated to a CUE. In the literature, simulations vary in scope from 2–30 RBs, 2–30 CUEs and 2–60 DUEs, while MBS operate with a cell radius of 20–500 m. It is frequently assumed that DUE are already paired and operate in a range between 10–30 m apart. Typically, omni-directional antenna and isotropic propagation are utilised. Path loss is commonly modeled using log-distance models, with or without shadowing.

Optimisation Algorithms

A wide variety of optimisation methods have been investigated on a range of D2D RRM problems. Initially, D2D radio resources were proposed to be managed using existing cellular uplink power control mechanisms [9]. Consequently, it was identified that resources could be more efficiently allocated with the use of mathematical optimisation [70]. However, due to the computational complexity of these methods and the millisecond timescales involved, it may not feasible to solve to optimality. This can be addressed with the use of greedy heuristic algorithms which reduce computational complexity at the cost of global optimality [71]. Alternatively, resource allocation can be optimised graph theoretically [7], game theoretically [72], with evolutionary algorithms [73] or reinforcement learning (RL) [17]. More recently, deep reinforcement learning (DRL), a subfield of RL which uses deep neural networks to represent policy and/or value functions has demonstrated promising results [21, 24]. DRL is well suited for many D2D RRM problems as neural networks provide rich approximations, scale well and generalise to unseen data.

Research Limitations

A common limitation observed in D2D cellular offload research is BSs not enforcing uplink power control for CUE, who transmit at maximum power, a very energy inefficient approach. Another research challenge is accounting for large SINR increases on the primary network, such as could significantly impact primary network throughput or drive up CUE transmit power levels. Resource allocations algorithms need to demonstrate their effectiveness in larger search spaces that more closely reflect real world demands. Iterative learning algorithms need to be capable of generalising to out of training distribution data and be robust under diverse propagation conditions. Lastly, in our opinion, one of the greatest limitations of existing research is the lack of established benchmarks and comparison with other algorithms.

4.2.2 OpenAI Gym

OpenAI Gym is an open-source software toolkit for RL [74]. Gym provides an abstraction layer that enables a variety of tasks, known as *environments* in RL parlance, to be wrapped to present a consistent interface. The abstraction provided by Gym allows the easy interchange of algorithms and environments. This makes it is easy to test how a single algorithm generalises across a diverse set of environments or to benchmark different algorithms on a given environment. The simplicity and flexibility Gym offers has proved very popular and has lead to it becoming the de facto environment format in RL. While Gym was designed for RL research, the application programming interface (API) it provides makes it easy to apply many other algorithms types.

4.2.3 Network Simulation

Due to the highly dynamic nature of communication systems, simulation is a common experimental methodology. For research in more conceptual stages, such as D2D cellular offload, simulation is a cost-effective method to prototype ideas. Network simulators have been developed for a wide variety of communication systems and in this section we will touch upon several related platforms.
The most common simulation method in communication systems is discrete-event simulation. In this paradigm, the state of the simulation model evolves over time, but can only be updated at discrete points [75]. Discrete-event simulation

consists of an initialisation stage, an event processing loop and a final output stage.
One of the most widely used network simulators in education and research is *ns-3*. Ns-3 is an open-source, modular, discrete-event simulator for wired and wireless networks. It provides the full TCP/IP stack and wireless propagation modelling. Another popular alternative with comparable features is *OMNeT++*, while there exists similar commercial tools such as *NetSim* and *MATLAB*.
Ns-3 has been incorporated into an OpenAI Gym environment under the *ns3-gym* project [76].

4.3 GymD2D

This section describes our D2D cellular offload network simulator and evaluation framework, GymD2D.

4.3.1 Design Principles

The design of GymD2D has been inspired by the authors experience developing and comparing reinforcement learning algorithms. In our experience the following design principles stimulate experimentation and the sharing of ideas.

- **Simple**: Easy to get started with, the framework should allow researchers to be productive quickly.
- **Configurable**: The framework should be easily configured to meet the broad range of D2D cellular offload use cases. Configurability allows researchers to programmatically test algorithm generalisation and scalability.
- **Extensible**: The framework should allow users to extend the system's behaviour to meet their needs. The nature of research dictates a stream of new ideas we can't anticipate, but we can provide researchers the flexibility to adapt.
- **Scalable**: The framework should be performant and easily parallelisable. Developing new algorithms requires significant experimentation and reducing the time spent waiting for results is important for productivity. Some algorithms, such as policy gradient DRL, require parallel environments

to function. Real world solutions are often a combination of both algorithmic and architectural components.

- **Reproducible**: Experiments should be easily repeatable. To build confidence in our deductions, it is important that we can reperform experiments to ensure the observed outcomes were not statistical anomalies. Reproducibility allows researchers to share their contributions with community more easily.

4.3.2 Architecture

GymD2D consists of two main components, a network simulator and a Gym environment. The network simulator models physical layer cellular networking. The Gym environment provides an abstraction layer to allow researchers to experiment with different simulation parameters and algorithms programmatically. Users supply RRM algorithms to manage the wireless devices under simulation. GymD2D outputs metrics on the state of the simulation to the user; to allow the effectiveness of RRM algorithms to be analysed statistically and through visualisation. A high level overview of the architecture of GymD2D is depicted in Fig. 4.1.

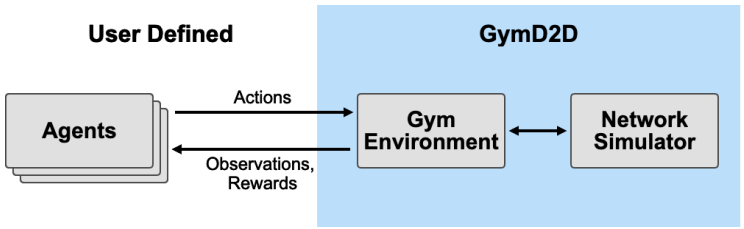

Figure 4.1: **Proposed GymD2D architecture.** GymD2D consists of a network simulator, wrapped by an OpenAI Gym environment. The user creates their own RRM algorithms to control wireless devices.

4.3.3 System Model

GymD2D is designed to study resource allocation problems in the physical layer. Data link and above layers, D2D session establishment and management concerns are considered out of scope.

The system models a single cell, employing OFDMA. It contains: a BS b, a set of CUEs $c \in \mathcal{C} = \{1, \ldots, C\}$ and a set of D2D transmitter–receiver ordered pairs $(d, e) \in \mathcal{D} = \{(d_1, e_1), (d_2, e_2), \ldots, (d_D, e_D)\}$. A set of RBs $k \in \mathcal{K}$ are available for allocation. Each RB is 180 kHz wide in frequency and contains 12 subcarriers.

An assumption is made that all devices are equipped with omni-directional antenna and transmit isotropically. Accordingly, the network resides within a circular cell of radius R, with the MBS located in the centre at position $(0, 0)$. The simulation environment contains no obstructions or outside interference. D2D communicate one-to-one and D2D relay is not supported.

We denote the effective isotropic radiated power (EIRP) E of BSs, CUEs and DUEs as E^b, E^c, E^d respectively. The EIRP of a BS is calculated,

$$E^b = P - 10log_{10}s + g_{ant} - l_{ix} - l_{cb} + g_{amp}\,, \tag{4.1}$$

and the EIRP of CUEs and DUEs,

$$E^c = E^d = P - 10log_{10}s + g_{ant} - l_{ix} - l_{bd}\,, \tag{4.2}$$

where P is the transmit power level in dBm, s is the number of subcarriers, g_{ant} is the transmitting antenna gain, l_{ix} is the interference margin loss to approximate noise from surrounding cells, l_{bd} is body loss to approximate attenuation caused by the user, l_{cb} is cable loss, and g_{amp} is amplifier gain.

We denote the received signal level R from transmitter i at receiver j of BS, CUEs and DUEs as $R^b_{i,j}$, $R^c_{i,j}$, $R^d_{i,j}$. The received signal level of BS as,

$$R^b_{i,j} = E_i - PL_{i,j} + g_{ant} - l_{cb} + g_{amp}\,, \tag{4.3}$$

and the received signal level of CUEs or DUEs,

$$R^c_{i,j} = R^d_{i,j} = E_i - PL_{i,j} + g_{ant} - l_{bd}\,, \tag{4.4}$$

where P_i is the EIRP from transmitter i and $PL_{i,j}$ is the path loss of the chosen path loss model between i and j.

We assume D2D transmissions are synchronised to cellular transmissions and occupy the same K orthogonal resources. During both uplink and downlink, co-channel interference is calculated for each receiver sharing RB k. GymD2D considers co-channel interference between:

- *D2D to cellular*, interference from secondary DUE on the primary cellular network,

Table 4.1: BS Configuration

Parameter	Value
Antenna gain g_{ant}	17.5 dBi
Thermal noise σ^2	-118.4 dBm
Interference margin l_{ix}	2.0 dB
Cable loss l_{cb}	2.0 dB
Amplifier gain g_{amp}	2.0 dB
Receiver sensitivity	-123.4 dBm

Table 4.2: UE Configuration

Parameter	Value
Antenna gain g_{ant}	0.0 dBi
Thermal noise σ^2	-104.5 dBm
Interference margin l_{ix}	3.0 dB
Body loss l_{bd}	3.0 dB
Receiver sensitivity	-107.5 dBm

- *cellular to D2D*, interference from CUE or BS to DUE, and
- *D2D to D2D*, the interference between DUE pairs sharing a RB.

Accordingly, we model the instantaneous SINR ξ of receiver j from transmitter i on RB k,

$$\xi_{i,j,k} = \frac{R_{i,j}}{\sum_{n \in \mathcal{T}_k, n \neq i} R_{n,j} + \sigma^2}\,, \tag{4.5}$$

where $\mathcal{T}_k$ is the set of transmitters allocated to RB k and σ^2 is additive white Gaussian noise (AWGN).
The capacity of channel $C_{i,j}$ can be calculated using the SINR $\xi_{i,j}$,

$$C_{i,j}[Mbps] = B \log_2(1 + \xi_{i,j})\,, \tag{4.6}$$

where B is the channel bandwidth in MHz and $\xi_{i,j}$ is the SINR in dB.
The default configuration parameters for BSs are listed in Table 4.1 and DUEs and CUEs in Table 4.2.

4.3.4 Path Loss Models

GymD2D contains several of the most common path loss models and makes it easy for users to implement their own custom models. By default, GymD2D uses the simplest model, free space path loss (FSPL),

$$FSPL(f, d)[dB] = 10 n log_{10}\Big(\frac{4\pi f d}{c}\Big)\,, \tag{4.7}$$

where $n = 2$ is the path loss exponent (PLE) in free space, f is the carrier frequency in Hz, d is the distance between the transmitter and receiver and c is the speed of light in m/s.
To simulate obstructed propagation environments it can be useful to model fading effects as random processes. One such model is the log-distance with

shadowing path loss model, which is included in GymD2D. The log-distance path loss model extends FSPL to mimic random shadowing effects, such as caused by buildings, with a log-normal distribution,

$$PL^{LD}(f,d)[dB] = FSPL(f,d_0) + 10nlog_{10}\frac{d}{d_0} + \chi_\sigma\,, \tag{4.8}$$

where d_0 is an arbitrary close-in reference distance, typically 1–100m and χ_σ is a zero-mean Gaussian with standard deviation σ in dB. Empirical measurements have shown values of $n = 2.7$ to 3.5 to be suitable to model urban environments [77].

4.3.5 Network Simulator

The network simulator models a single cellular cell which is populated with a collection of randomly placed CUE and DUE pairs. It is a configurable component which can be customised to emulate a range of cellular offload scenarios. This includes the number and configuration of BSs, CUEs and DUEs and environmental parameters such as the available RBs, cell size and path loss model.
The main components of the network simulator are: a collection of wireless devices (BSs, CUEs, DUEs), a path loss model and a traffic model as shown by the class diagram in Fig. 4.2.

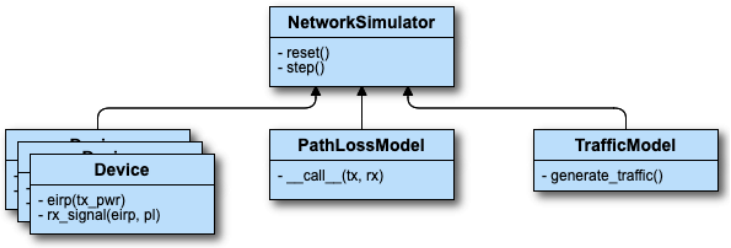

Figure 4.2: **Network simulator architecture.** The main components of the network simulator are a collection of CUEs, DUEs and BS, the path loss model and the traffic model.

Each simulation, the actions of BS and UEs within the cell can be generated internally by the *traffic model* or externally from a user defined RRM algorithm. A typical use case would be to use the internal traffic model to control BS and CUEs and the user RRM algorithm the DUEs.

GymD2D uses a discrete-event simulation model. This method is congruent with the Gym API in which the incoming actions are the *events* and the Gym *step()* method calls equate to the system update intervals and model a single LTE or NR frame. At each step, each device may transmit, receive, or take no action. An action is tuple consisting of a transmitter, receiver, communication mode, RB and transmission power. The simulator consolidates the actions from both the traffic model and the RRM algorithm, then calculates the resulting propagation and interference. After calculating propagation, metrics on the state of the network, such as SINR and throughput, are output to the Gym environment.

4.3.6 Gym Environment

The Gym environment has been designed to be configuration driven, to facilitate the programmatic scheduling and reproducibility of experiments. When instantiating a new Gym environment, configuration can be provided to specify the BSs, CUEs and DUEs that inhabit the simulation and the environmental conditions. A list of the available configuration parameters is provided in Table 4.3.
The Gym environment outputs observations, rewards and diagnostic information on the state of simulator at every timestep. In response to these inputs, RRM algorithms provide actions, comprised of the RB and power level for each UE under their control. The format these inputs and outputs can be customised to allow a variety of algorithm classes to be applied to the environment. For example, when using a DRL algorithm the observations can formatted to return a pictorial representation suitable as input to a CNN.
The diagnostic information the environment outputs is not intended to be used by RRM algorithms in their decision making, but rather for researchers to use to inspect the performance of their algorithms.

4.3.7 Capabilities and Limitations

This section provides a summary of GymD2D's capabilities and limitations.

Capabilities

- **Highly configurable**, as outlined in Table 4.3.

Table 4.3: GymD2D Environment Configuration Parameters

Parameter	Default Value
Number of resource blocks	25
Number of cellular users	25
Number of D2D pairs	25
Cell radius	500.0 m
Maximum D2D pair distance	20.0 m
Maximum CUE transmission power	23 dBm
Minimum DUE transmission power	0 dBm
Maximum DUE transmission power	20 dBm
Path loss model	FSPL
Traffic model	Uplink
Carrier frequency	2.1 GHz
Number of subcarriers	12
Subcarrier spacing	15 kHz
Channel bandwidth	5 MHz

- **Supports most optimization and control paradigms**. By allowing users to customise simulator inputs and output, GymD2D allows researchers to investigate and compare most optimisation algorithms. Additionally, as GymD2D facilitates multiple control agents, these various algorithms can be combined or compared alongside each other.

- **Calculates co-channel interference between all receivers**, including DUE to CUE, CUE to DUE and DUE to DUE.

- **Supports both power level and RB optimisation.**

- **Centralised or decentralised communications modes**, allows researchers to experiment with full control, autonomous and hybrid control modes.

- **Customisable traffic patterns**, GymD2D does not make any assumptions about where or how devices communicate. This allows the user to customise traffic patterns and investigate how control algorithms perform during periods of higher or lower demand.

- **Customisable path loss models**, GymD2D provides multiple path loss models out of the box or allows users to define their own.

- **Experiment management utilities**, GymD2D provides several experiment management utilities such as: a helper to plot UE locations; a method to save and load experiment configurations files; and diagnostic information that provides detailed feedback on the state of the environment at every timestep. These metrics can be used to troubleshoot algorithm performance and generate detailed analyses.

Limitations

- **Single base station**, GymD2D does not currently support multiple BSs. This limits the ability investigate the impacts of adjacent cell sites, BS location optimisation, heterogeneous networks, communications relay and partial coverage.
- **Cellular numerology and QoS**, and the ability for BSs and UEs to signal QoS and power level changes is not currently supported.
- **Energy consumption** of BSs and UEs, to be used to optimise energy efficiency.
- **Advanced propagation modelling**, currently GymD2D provides linear path loss models, unidirectional propagation, and does not support terrain attenuation modelling.
- **One-to-many communication**, including broadcast and multicast schemes.
- **Limited peer discovery**, could be expanded beyond initial environment configuration to model peer discovery process optimisation.
- **Overlay networking** is not currently supported.
- **Multimedia, edge networks, caching** is not currently supported.
- **V2V or M2M** is not currently supported.

The design of GymD2D does not preclude these limitations, which can be supported at a later date. Some limitations, such as multiple BSs and cellular numerology, are on our development roadmap and set to be included in future releases. Conversely, limitations such as advanced propagation modelling and V2V or M2M support are not currently in progress.

4.4 Evaluation Methods

In this section we describe the methods and the implementation details used to evaluate GymD2D with several leading DRL algorithms and provide performance baselines.
We evaluated GymD2D with four agents; three proven, high-performing DRL algorithms: Rainbow DQN [30], SAC [78], and A2C [38]; and a random agent. These algorithms encompass the three most successful model-free DRL approaches, off-policy value-based, off-policy policy gradient and on-policy policy gradient.

4.4.1 Agent Architecture

We employed a centralised control architecture to manage radio resource, in which a single "meta-agent" computes the allocations for each DUE, residing in a single cell, at every timestep t. Centralised control of a single cell can be achieved by co-locating resource management systems in cellular BSs and leveraging existing channel state information to inform decision-making. In this configuration, due to the physical separation between cells, intra-cell coordination is not necessary. Advantages of centralised control include more complete information and more efficient action coordination.
The resource management system generates an observation for each DUE using cell-wide channel state information from all CUEs and DUEs. These observations are batched up and processed one at a time by the Meta-Agent architecture, to output a batch of resource allocation actions. The resulting actions are then communicated to DUEs over existing control channels. The agent architecture is shown in Fig. 4.3.

4.4.2 Observation Space

An observation vector S_t^d is generated for each D2D link $(d, e) \in \mathcal{D}$ at every timestep, t. Each observation vector is composed of the channel state of the D2D link $S_t^{d,e}$, and the channel states of all other channels in the cell:

$$S_t^d = \{S_t^{d,e}\} \cup \{S_t^{c,b} : (c, b) \in \mathcal{C} \times \{b\}\} \cup \{S_t^{f,g} : (f, g) \in \mathcal{D}, f \neq d\}, \quad (4.9)$$

where a channel state $S_t^{i,j}$ between transmitter i and receiver j, is composed the transmitter's position, the receiver's position, the receiver's previous

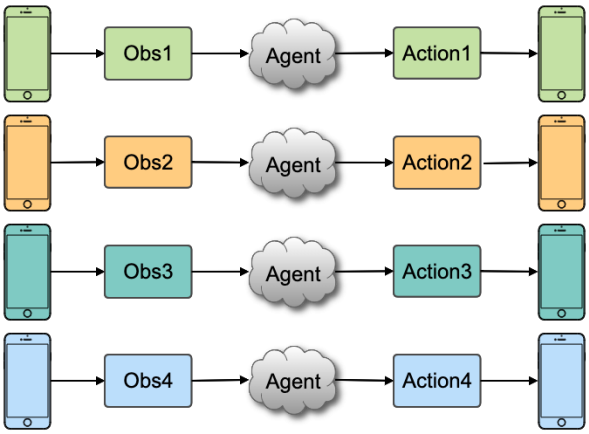

Figure 4.3: **The agent architecture employed by our evaluation.** An observation is generated for each DUE. A single meta-agent processes each observation and generates a resource allocation action, which is communicated back to DUE.

SINR and the receiver's previous SNR,

$$S_t^{i,j} = [\mathrm{Pos}_i, \mathrm{Pos}_j, \xi_{i,j}, \mathrm{SNR}_{i,j}]\,. \tag{4.10}$$

Position features are an ordered-pair of the two-dimensional cartesian coordinates relative to the BS, which is centered at $(0, 0)$.

4.4.3 Action Space

The DRL agent generates a resource allocation action for each DUE observation, at every timestep, t. Each action is a joint transmit power level and an RB allocation. D2D transmit power levels $\mathcal{P}^d = \{P^d_{min}, \ldots, P^d_{max}\}$ are a set integer power levels, between the minimum and maximum values, in dBm. The action space is modelled as the cartesian product $\mathcal{K} \times \mathcal{P}^d$, which are mapped to agent outputs.

4.4.4 Reward Function

The reward function was designed to encapsulate the true optimisation objective, to avoid it being "gamed" by the process known as reward hacking [79]. It rewards agents equally, in proportion to the total network capacity, that is the sum channel capacity of CUEs and DUEs, so long as no CUE are

significantly impacted by interference. This utilitarian reward function was designed to encourage agents to learn to cooperate to maximise system throughput while protecting primary network performance. At each timestep t, DUE reward is calculated by:

$$R_t = \begin{cases} \frac{\sum_{i,j \in \mathcal{N} \cup \mathcal{M}} C_{i,j}}{|\mathcal{N}|} & \forall m \in \mathcal{M}, C_{m,b} > 0 \\ -1 & \exists m \in \mathcal{M}, C_{m,b} = 0 \end{cases}, \tag{4.11}$$

where $C_{i,j}$ is the channel capacity between transmitter i and receiver j in Mbps as calculated in Eq. (4.12).

4.4.5 Neural Network Architecture

DRL agents used a fully connected neural network with two hidden layers trained using the Adam optimiser. Each hidden layer contained 128 units and used ReLU activation between layers. Learning rates are algorithm dependent and described below in the agent configuration section.

4.4.6 Agent Configuration

DRL agents used a reward discounting factor of $\gamma = 0.9$.

Rainbow DQN: our implementation used distributional, dueling, double-Q and noisy networks components with a prioritised replay buffer and single step returns. It's neural networks were trained with a learning rate $\alpha = 5 \cdot 10^{-4}$ on minibatches of 32 samples every 4 steps. Learning began after 1,000 steps, and the online and target networks were synchronised every 500 steps. The network used 51 distributional atoms and was bounded between the expected returns of $v_{min} = -10$ and $v_{max} = 10$. Dueling networks were implemented using a single, dense layer with 128 hidden units for both the advantage and value layers. The prioritised replay buffer had a capacity of 50,000 and used a prioritisation exponent $\omega = 0.6$, importance sampling exponent $\beta = 0.4$ which was annealed to $\beta = 0.4$ over the first 20,000 steps. DQN hyperparameters are shown in Table 4.4.

Discrete SAC: the discrete action variant was used [80]. It's Q-model used twin Q-networks, each a dense network containing two hidden layers of 128 units with ReLU activation. Similar to the DQN, it used a prioritised replay buffer with a capacity of 50,000, a prioritisation exponent $\omega = 0.6$, importance

Table 4.4: Rainbow DQN Hyperparameters

Parameter	Value
Discounting factor γ	0.9
Learning rate α	$5 \cdot 10^{-4}$
Batch size	32
Online network update period	4 steps
Learning start	$1,000$ steps
Target network sync period	500 steps
Distributional atoms	51
Distributional bounds v_{min}, v_{max}	[-10,10]
Replay buffer capacity	$50,000$
Replay buffer prioritisation exponent ω	0.6
Replay buffer importance sampling β	$0.6 \rightarrow 0.4$
Importance sampling annealing	20,000 steps

Table 4.5: SAC Hyperparameters

Parameter	Value
Discounting factor γ	0.9
Learning rate α	$3 \cdot 10^{-4}$
Batch size	256
Learning start	$1,500$ steps
Target coefficient τ	0.005
Replay buffer capacity	$50,000$
Buffer prioritisation ω	0.6
Buffer I.S. β	$0.6 \rightarrow 0.4$
Buffer I.S. annealing	20,000 steps

Table 4.6: A2C Hyperparameters

Parameter	Value
Discounting factor γ	0.9
Num workers	10
Learning rate α	$1 \cdot 10^{-4}$
Rollout length	10
Entropy coefficient β	0.01
GAE λ	1.0

sampling exponent $\beta = 0.4$ which was annealed to $\beta = 0.4$ over the first 20,000 steps. SAC hyperparameters are shown in Table 4.5.

A2C: is the synchronous version of A3C. It used Generalised Advantage Estimator (GAE) [81] with $\lambda = 1.0$. We used 10 rollout workers to gather experience for the critic. A2C hyperparameters are shown in Table 4.6.

Random Agent: the random agent used a uniform distribution to sample indices from the cartesian product of power levels and RBs.

4.5 Evaluation

4.5.1 Methodology

We evaluated GymD2D with several leading DRL algorithms to determine their efficiency allocating radio resources as D2D demand increased. The objective was to maximise the total system capacity, that is the sum data rate of all CUE and DUE, calculated for each transmitter/receiver pair i, j by:

$$C_{i,j}[Mbps] = \begin{cases} B \log_2(1 + \xi_{i,j}) & \xi_{i,j} \geq \rho_j \\ 0 & \xi_{i,j} < \rho_j \end{cases}, \tag{4.12}$$

where $B = 0.18$ is the RB bandwidth in MHz and $\rho_b = -123.4$ and $\rho_d = -107.5$ is the receiver sensitivity of a BS and DUE respectively in dBm. Our evaluation simulated a single cell under full load. The scenario contained 25 RBs and CUEs, with each CUE allocated an individual RB. We employed a centrally managed control mode in which DUE communicated in the uplink frame, with the resource allocation managed by the network operator. Each RRM algorithm was evaluated with 10, 20, 30, 40 and 50 communicating D2D pairs. Algorithms were compared by training to convergence, then evaluating for 100 episodes. For each algorithm–D2D link density comparison, we conducted ten trials, retraining from scratch and evaluating, to account for variations in performance. Each episode lasted for ten steps or equivalently ten LTE/NR frames to simulate short bursts of traffic on a busy network. In each episode all CUE and DUE remained geographically fixed, but at the end of each episode, all CUE and DUE were randomly repositioned within the cell to simulate new devices accessing the network. Wireless propagation was modelled using the Log-Distance Shadowing model Eq. (4.8) with PLE $n = 2.0$ and $\chi_\sigma = 2.7$. The simulation parameters are detailed in Table 4.7.

4.5.2 Results

Firstly, the system was evaluated with only cellular communication, to determine the baseline system total capacity. This baseline is used to compare the capacity of a system with and without D2D communication and draw conclusion about the performance impacts of different D2D RRM methods. The baseline system total capacity was 94.75 Mbps.

Table 4.7: Simulation Parameters

Parameter	Value
Cell radius	500 m
Maximum D2D pair distance	30 m
Carrier frequency	2.1 GHz
RB bandwidth	180 kHz
Number of RBs	25
Number of CUEs	25
Number of DUE pairs	$10, 20, 30, 40, 50$
CUE transmit power	23 dBm
DUE min, max transmit power	$0, 20$ dBm
Path loss model	Log-Distance Shadowing
Path loss exponent	2.0
Shadowing SD χ_σ	2.7

Next, the four RRM methods were evaluated at increasing D2D link densities, to investigate optimisation efficiency as D2D demand increased, as shown in Figs. 4.4 and 4.5. We found that all three DRL algorithms achieved a similar level of performance, increasing system total capacity over the baseline by more than 11%. Conversely, the performance of the random agent shows that without careful resource allocation, the system capacity drops sharply, as can be seen by the solid red line in Fig. 4.4. Fig. 4.6 shows the sum capacity of DUEs achieved by each DRL algorithm. This shows that system capacity increased sublinearly with D2D demand. From this, we hypothesise that there exists a saturation level at which the further active D2D links reduces network performance.

We found that despite allowing DUE to communicate up to half the power of CUE (20 vs. 23 dBm), they typically converged into operating ranges between 7 and 15 dBm, shown in Fig. 4.7. This resulted in a negligible decrease in the total CUE capacity, 1–2 Mbps or ≈1.84% below the baseline system capacity. This decrease was approximately constant across D2D density.

4.5.3 Discussion

Our results demonstrated that intelligent resource allocation strategies can significantly increase system capacity with minimal impact on primary users. From a utilitarian viewpoint, the minor primary network impact is offset by the significant system gains through the use of D2D underlay cellular offload.

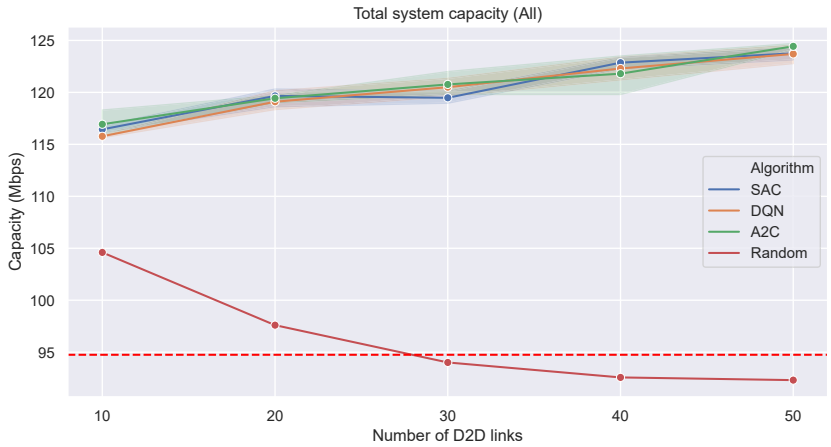

Figure 4.4: **Total system capacity of the three DRL agents and the random agent as D2D link density increases.** This figures shows the performance of the four evaluated RRM algorithms as D2D link density increases, in terms of the total system capacity. Total system capacity is the sum network capacity of all CUE and DUE traffic per evaluation timestep. The solid center lines indicate the mean algorithm performance across ten trials with the shaded area the 95% confidence interval. The dashed red line indicates the baseline total system capacity.

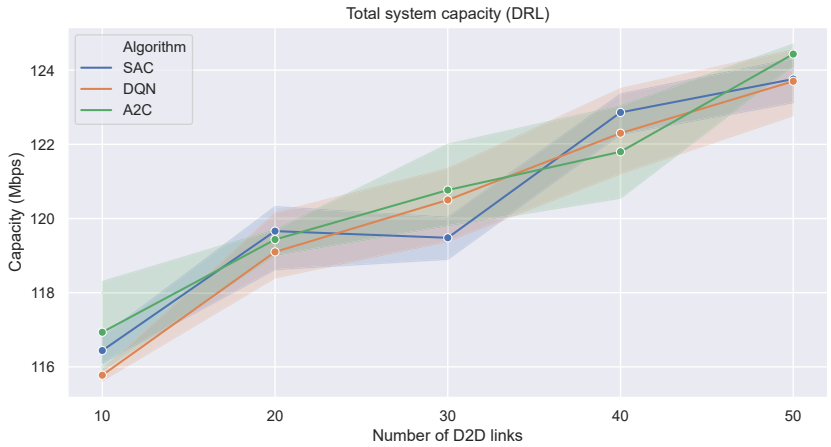

Figure 4.5: **Total system capacity of the three DRL agents.** This figure presents the same data as Fig. 4.4 without the random agent or baseline capacity to zoom in on DRL agent performance. Solid center lines indicate the mean algorithm performance across ten trials with the shaded area the 95% confidence interval.

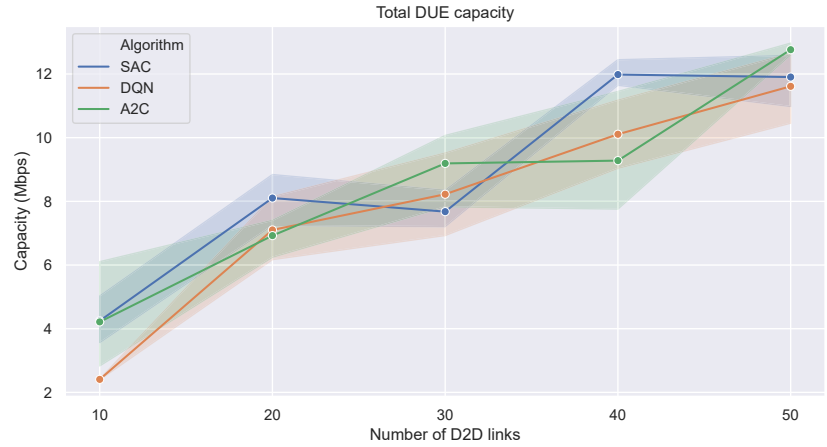

Figure 4.6: **Total DUE capacity of the three DRL agents.** Shows the sum network capacity of D2D users, achieved by each DRL RRM algorithm, as D2D link density increases. Solid center lines indicate the mean algorithm performance across ten trials with the shaded area the 95% confidence interval.

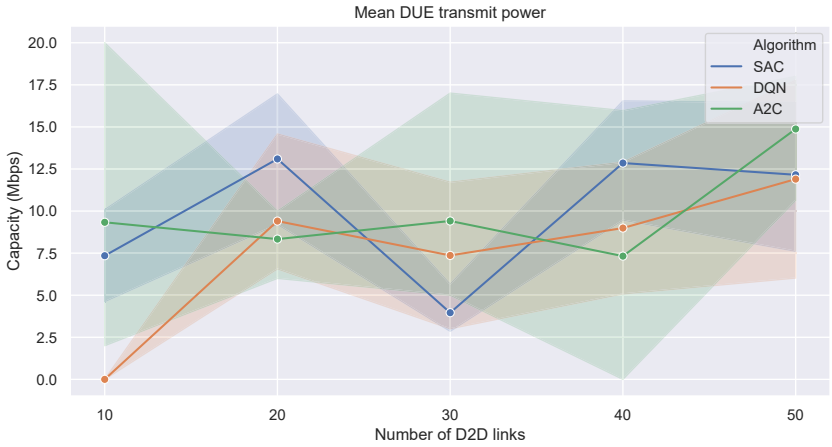

Figure 4.7: **Mean DUE transmit power levels.** The mean DUE transmit power levels, for each DRL RRM algorithm, as D2D link density increases. Solid center lines indicate the mean algorithm performance across ten trials with the shaded area the 95% confidence interval.

To further understand the behaviour of the trained DRL agents, we investigated the actions they selected. We observed that they converged to allocating all DUE onto one or two RBs. This is surprising as we had anticipated DUEs to be evenly distributed amongst RBs. Prior research has restricted their solution-space to one-to-one correspondence between DUEs and RB [82], a decision we assume is to limit the combinatorial complexity of the problem. However, these results suggest that this could be suboptimal and the solution-space should include DUEs sharing RBs.

Investigating further, we observed that over the course of a training run, the DQN converging from an even RB distribution to the focused allocation strategy. This behaviour developed in the later stages of training and only contributed modest increases to the system capacity.

As expected, the optimal strategy was to assign DUE to share RBs with the most geographically distant CUE. When combined with the focused RB allocation strategy described above, this typically resulted in the RRM algorithm choosing to allocate DUE to share with the one or two most isolated CUE.

Despite the random UE positioning, the DRL agents were able to learn policies that generalised much better than we anticipated when using fully connected neural networks. We were also surprised how quickly agents adapted during an episode, improving their performance over the course of the ten-step episode.

4.6 Conclusion

In this research we have presented GymD2D, a network simulator and evaluation platform for RRM in D2D underlay cellular offload. GymD2D makes it easy for researchers to build, benchmark and share RRM algorithms and results. Our toolkit is designed to quickly prototype physical layer resource allocation algorithms, without the complexity of higher layer protocols. GymD2D is configurable and extensible, allowing it to be employed to simulate a range of D2D research needs.

We have evaluated GymD2D with several leading DRL algorithms and demonstrated the performance gains of intelligent RRM, increasing system capacity by more than 11%. There was no clear winner amongst the DRL algorithms which performed similarly. The results also demonstrated that D2D cellular offload can significantly minimise its impact on primary networks.

In the future we plan to increase the simulation complexity in GymD2D, adding more realistic modelling. Other interesting research challenge include investigating the impacts of CUE power control on cellular offload and supporting D2D relay. We continue to use GymD2D in ongoing research, developing methods for scaling up DRL based D2D RRM.

Chapter 5

Conclusion

In this thesis we have investigated the use of DRL to optimise the allocation of radio resources in D2D underlay cellular offload to increase network capacity. We applied DRL to the RRM optimisation problem and introduced a centralised radio resource management system which was able to increase the system capacity by more than 11%, with minimal impact on primary network performance. To empirically demonstrated our approach, we developed a network simulator and evaluation platform for D2D cellular offload research, which we have released as open-source software for other researchers to use. The network simulator makes it easier for researchers to compare, share and build upon prior research.

We have also developed an improved self-play training algorithm for training RL in competitive environments. Self-play allows RL to be trained without expert guidance, however it is known to be less sample efficient and suffer more unstable learning dynamics than other training methods. To address these issues we harnessed the competitive pressures of coevolution by competing within a larger population. We compared our improved training method with several self-play variants and found our method achieved a 15% higher win rate and exhibited more stable training with fewer performance regressions.

5.1 Summary of Results

Chapter 3 investigated the problem of training RL agents in competitive environments without expert instruction. Self-play is a method for training RL in competition with itself, however it is known be less sample efficient

and suffer unstable learning dynamics. The chapter presented a population-based approach which used coevolution to generate competitive pressures to train agents more robustly and efficiently. It presented an algorithm, CLaRE, which achieved an almost 80% win-rate, approximately 15% above the next leading method. Importantly, CLaRE showed stronger convergence properties and more stable training dynamics.
Chapter 4 described a network simulator and evaluation platform for D2D RRM research. Challenges facing D2D cellular offload researchers include insufficient tooling, difficulty comparing research and a lack of established benchmarks. This makes it hard to compare algorithms, verify results and confidence in reported empirical results. Our contribution was to develop a network simulator for physical layer resource allocation research that we open-sourced and to provide to the community. We evaluated the toolkit, GymD2D, with several state-of-the-art DRL algorithms to provide benchmarks. We found that the use of DRL to optimise resource allocation was able to increase network capacity by more than 11%.

5.2 Future Work

Interesting avenues for future work would be to look at increasing the simulation complexity, such as:

- **Introducing cellular power control**, so cellular users are not always transmitting at full power as is commonly simulated, as this is highly energy inefficient and impractical in real world situations.
- **Reducing the observation fidelity**, currently it is common for RRM algorithms to use quite detailed values to inform decision-making, when cellular numerology is much less detailed.
- **Improved propagation modelling**, commonly simulations use relatively linear path loss models which are much more easily learnt than the much more dynamic real world conditions. Additionally, propagation is generally modelled using omni-directional antennas over 360 degrees, providing much more distance between UE.
- **Increasing problem spaces**, commonly simulation occurs with smaller numbers of devices and available resources than real world conditions.

Effort needs to be made to ensure algorithms are capable of scaling to real world loads.

Bibliography

[1] T. S. Rappaport, W. Roh, and K. Cheun, "Mobile's millimeter-wave makeover," *IEEE Spectrum*, vol. 51, no. 9, pp. 34–58, 2014.

[2] A. Asadi, Q. Wang, and V. Mancuso, "A survey on device-to-device communication in cellular networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 4, pp. 1801–1819, 2014.

[3] H. H. Hussein, H. A. Elsayed, and S. M. A. El-kader, "Intensive benchmarking of d2d communication over 5g cellular networks: prototype, integrated features, challenges, and main applications," *Wireless Networks*, pp. 1–20, 2019.

[4] B. Kaufman and B. Aazhang, "Cellular networks with an overlaid device to device network," in *2008 42nd Asilomar conference on signals, systems and computers*. IEEE, 2008, Conference Proceedings, pp. 1537–1541.

[5] A. Goldsmith, S. A. Jafar, I. Maric, and S. Srinivasa, "Breaking spectrum gridlock with cognitive radios: An information theoretic perspective," *Proceedings of the IEEE*, vol. 97, no. 5, pp. 894–914, 2009.

[6] X. Bao, U. Lee, I. Rimac, and R. R. Choudhury, "Dataspotting: offloading cellular traffic via managed device-to-device data transfer at data spots," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 14, no. 3, pp. 37–39, 2010.

[7] R. Zhang, X. Cheng, L. Yang, and B. Jiao, "Interference-aware graph based resource sharing for device-to-device communications underlaying cellular networks," in *2013 IEEE wireless communications and networking conference (WCNC)*. IEEE, 2013, Conference Proceedings, pp. 140–145.

[8] S. Maghsudi and S. Stańczak, "Hybrid centralized–distributed resource allocation for device-to-device communication underlaying cellular networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 4, pp. 2481–2495, 2015.

[9] P. Janis, C.-H. Yu, C. Ribeiro, C. Wijting, K. Hugl, O. Tirkkonen, and V. Koivunen, "Device-to-device communication underlaying cellular communications systems," *Int'l J. of Communications, Network and System Sciences*, vol. 2009, 2009.

[10] The 5g infrastructure public private partnership: the next generation of communication networks and services. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2015/02/5G-Vision-Brochure-v1.pdf

[11] P. Mach, Z. Becvar, and T. Vanek, "In-band device-to-device communication in ofdma cellular networks: A survey and challenges," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 1885–1922, 2015.

[12] S. Mumtaz, H. Lundqvist, K. M. S. Huq, J. Rodriguez, and A. Radwan, "Smart direct-lte communication: An energy saving perspective," *Ad Hoc Networks*, vol. 13, pp. 296–311, 2014.

[13] M. G. d. S. Rêgo, T. F. Maciel, H. de HM Barros, F. R. Cavalcanti, and G. Fodor, "Performance analysis of power control for device-to-device communication in cellular mimo systems," in *2012 International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, 2012, Conference Proceedings, pp. 336–340.

[14] T. Ta, J. S. Baras, and C. Zhu, "Improving smartphone battery life utilizing device-to-device cooperative relays underlaying lte networks," in *2014 IEEE International Conference on Communications (ICC)*. IEEE, 2014, Conference Proceedings, pp. 5263–5268.

[15] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[16] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1.

[17] Y. Luo, Z. Shi, X. Zhou, Q. Liu, and Q. Yi, "Dynamic resource allocations based on q-learning for d2d communication in cellular networks," in *2014 11th International Computer Conference on Wavelet Actiev Media Technology and Information Processing (ICCWAMTIP)*. IEEE, 2014, Conference Proceedings, pp. 385–388.

[18] X. Chen, J. Wu, Y. Cai, H. Zhang, and T. Chen, "Energy-efficiency oriented traffic offloading in wireless networks: A brief survey and a learning approach for heterogeneous cellular networks," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 4, pp. 627–640, 2015.

[19] S. Nie, Z. Fan, M. Zhao, X. Gu, and L. Zhang, "Q-learning based power control algorithm for d2d communication," in *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE, Conference Proceedings, pp. 1–6.

[20] A. Moussaid, W. Jaafar, W. Ajib, and H. Elbiaze, "Deep reinforcement learning-based data transmission for d2d communications," in *2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, Conference Proceedings, pp. 1–7.

[21] J. Tan, L. Zhang, and Y.-C. Liang, "Deep reinforcement learning for channel selection and power control in d2d networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, Conference Proceedings, pp. 1–6.

[22] S. Gengtian, T. Koshimizu, M. Saito, P. Zhenni, L. Jiang, and S. Shimamoto, "Power control based on multi-agent deep q network for d2d communication," in *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. IEEE, Conference Proceedings, pp. 257–261.

[23] K. K. Nguyen, T. Q. Duong, N. A. Vien, N.-A. Le-Khac, and M.-N. Nguyen, "Non-cooperative energy efficient power allocation game in d2d communication: A multi-agent deep reinforcement learning approach," *IEEE Access*, vol. 7, pp. 100 480–100 490, 2019.

[24] Z. Li, C. Guo, and Y. Xuan, "A multi-agent deep reinforcement learning based spectrum allocation framework for d2d communications," *arXiv preprint arXiv:1904.06615*, 2019.

[25] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley Sons, 2014.

[26] Reinforcement learning. [Online]. Available: https://www.davidsilver.uk/teaching/

[27] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, Conference Proceedings, pp. 1057–1063.

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

[29] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[30] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," *arXiv preprint arXiv:1710.02298*, 2017.

[31] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *AAAI*, vol. 2. Phoenix, AZ, 2016, Conference Proceedings, p. 5.

[32] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," *arXiv preprint arXiv:1511.06581*, 2015.

[33] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[34] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine learning*, vol. 3, no. 1, pp. 9–44, 1988.

[35] J. F. Hernandez-Garcia and R. S. Sutton, "Understanding multi-step deep reinforcement learning: a systematic study of the dqn target," *arXiv preprint arXiv:1901.07510*, 2019.

[36] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, and O. Pietquin, "Noisy networks for exploration," *arXiv preprint arXiv:1706.10295*, 2017.

[37] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," *arXiv preprint arXiv:1707.06887*, 2017.

[38] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, 2016, Conference Proceedings, pp. 1928–1937.

[39] D. Cotton, J. Traish, and Z. Chaczko, "Coevolutionary deep reinforcement learning," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020, Conference Proceedings, pp. 2600–2607.

[40] E. Alonso, M. D'inverno, D. Kudenko, M. Luck, and J. Noble, "Learning in multi-agent systems," *The Knowledge Engineering Review*, vol. 16, no. 3, pp. 277–284, 2001.

[41] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, and M. Lanctot, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[42] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, and P. Georgiev, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, pp. 1–5, 2019.

[43] Openai five. [Online]. Available: https://blog.openai.com/openai-five

[44] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, and A. Bolton, "Mastering the

game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[45] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch, "Emergent complexity via multi-agent competition," *arXiv preprint arXiv:1710.03748*, 2017.

[46] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*. MIT press, 1994.

[47] M. L. Littman, *Markov games as a framework for multi-agent reinforcement learning*. Elsevier, 1994, pp. 157–163.

[48] J. Heinrich, M. Lanctot, and D. Silver, "Fictitious self-play in extensive-form games," in *International Conference on Machine Learning*, 2015, Conference Proceedings, pp. 805–813.

[49] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *Journal of Machine Learning Research*, vol. 6, pp. 503–556, 2005.

[50] C. E. Shannon, *Programming a computer for playing chess*. Springer, 1988, pp. 2–13.

[51] A. L. Samuel, "Some studies in machine learning using the game of checkers. ii—recent progress," *IBM Journal of research and development*, vol. 11, no. 6, pp. 601–617, 1967.

[52] G. Tesauro, *Td-gammon: A self-teaching backgammon program*. Springer, 1994, pp. 267–285.

[53] J. B. Pollack and A. D. Blair, "Why did td-gammon work?" in *Advances in Neural Information Processing Systems*, 1997, Conference Proceedings, pp. 10–16.

[54] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, and K. Simonyan, "Population based training of neural networks," *arXiv preprint arXiv:1711.09846*, 2017.

[55] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castaneda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, and A. Ruderman, "Human-level performance in first-person multiplayer games with population-based deep reinforcement learning," *arXiv preprint arXiv:1807.01281*, 2018.

[56] R. P. Wiegand, "An analysis of cooperative coevolutionary algorithms," Thesis, 2003.

[57] D. J. Montana and L. Davis, "Training feedforward neural networks using genetic algorithms," in *IJCAI*, vol. 89, 1989, Conference Proceedings, pp. 762–767.

[58] D. Whitley, S. Dominic, R. Das, and C. W. Anderson, "Genetic reinforcement learning for neurocontrol problems," *Machine Learning*, vol. 13, no. 2-3, pp. 259–284, 1993.

[59] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[60] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv preprint arXiv:1703.03864*, 2017.

[61] S. Khadka, S. Majumdar, T. Nassar, Z. Dwiel, E. Tumer, S. Miret, Y. Liu, and K. Tumer, "Collaborative evolutionary reinforcement learning," *arXiv preprint arXiv:1905.00976*, 2019.

[62] S. Fujimoto, H. Van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *arXiv preprint arXiv:1802.09477*, 2018.

[63] A. E. Elo, *The rating of chessplayers, past and present.* Arco Pub., 1978.

[64] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. Jordan, and I. Stoica, "Rllib: Abstractions for distributed reinforcement learning," in *International Conference on Machine Learning*, 2018, Conference Proceedings, pp. 3053–3062.

[65] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, Conference Proceedings, pp. 249–256.

[66] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*. Springer, 2006, Conference Proceedings, pp. 282–293.

[67] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *International conference on computers and games*. Springer, 2006, Conference Proceedings, pp. 72–83.

[68] D. Cotton and Z. Chaczko, "Gymd2d: A device-to-device underlay cellular offload evaluation platform," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, Conference Proceedings, pp. 1–7.

[69] C. Chakraborty and J. J. Rodrigues, "A comprehensive review on device-to-device communication paradigm: Trends, challenges and applications," *Wireless Personal Communications*, pp. 1–23, 2020.

[70] C.-H. Yu, O. Tirkkonen, K. Doppler, and C. Ribeiro, "Power optimization of device-to-device communication underlaying cellular communication," in *2009 IEEE international conference on communications*. IEEE, 2009, Conference Proceedings, pp. 1–5.

[71] M. Zulhasnine, C. Huang, and A. Srinivasan, "Efficient resource allocation for device-to-device communication underlaying lte network," in *2010 IEEE 6th International conference on wireless and mobile computing, networking and communications*. IEEE, 2010, Conference Proceedings, pp. 368–375.

[72] C. Xu, L. Song, Z. Han, Q. Zhao, X. Wang, and B. Jiao, "Interference-aware resource allocation for device-to-device communications as an underlay using sequential second price auction," in *2012 IEEE international conference on communications (ICC)*. IEEE, 2012, Conference Proceedings, pp. 445–449.

[73] L. Su, Y. Ji, P. Wang, and F. Liu, "Resource allocation using particle swarm optimization for d2d communication underlay of cellular networks," in *2013 IEEE wireless communications and networking conference (WCNC)*. IEEE, 2013, Conference Proceedings, pp. 129–133.

[74] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[75] K. Wehrle, M. Günes, and J. Gross, *Modeling and tools for network simulation*. Springer Science Business Media, 2010.

[76] P. Gawłowicz and A. Zubow, "ns3-gym: Extending openai gym for networking research," *arXiv preprint arXiv:1810.03943*, 2018.

[77] T. S. Rappaport, *Wireless communications: principles and practice*. prentice hall PTR New Jersey, 1996, vol. 2.

[78] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning*. PMLR, 2018, Conference Proceedings, pp. 1861–1870.

[79] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[80] P. Christodoulou, "Soft actor-critic for discrete action settings," *arXiv preprint arXiv:1910.07207*, 2019.

[81] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[82] P. Janis, V. Koivunen, C. Ribeiro, J. Korhonen, K. Doppler, and K. Hugl, "Interference-aware resource allocation for device-to-device radio underlaying cellular networks," in *VTC Spring 2009-IEEE 69th Vehicular Technology Conference*. IEEE, 2009, Conference Proceedings, pp. 1–5.