arXiv:2310.07745v1 [cs.LG] 11 Oct 2023

Deep Reinforcement Learning for Autonomous Cyber Operations: A Survey

Gregory Palmer[1*], Chris Parry[1], Daniel Harrold[1] and Chris Willis[1]

[1]BAE Systems Applied Intelligence Labs, Chelmsford Office & Technology Park, Great Baddow, Chelmsford, Essex, CM2 8HN.

*Corresponding author(s). E-mail(s): gregory.palmer@baesystems.com;

Abstract

The rapid increase in the number of cyber-attacks in recent years raises the need for principled methods for defending networks against malicious actors. *Deep reinforcement learning* (DRL) has emerged as a promising approach for mitigating these attacks. However, while DRL has shown much potential for cyber defence, numerous challenges must be overcome before DRL can be applied to autonomous cyber operations (ACO) *at scale*. Principled methods are required for environments that confront learners with *very* high-dimensional state spaces, large multi-discrete action spaces, and adversarial learning. Recent works have reported success in solving these problems individually. There have also been impressive engineering efforts towards solving all three for real-time strategy games. However, applying DRL to the full ACO problem remains an open challenge. Here, we survey the relevant DRL literature and conceptualize an idealised ACO-DRL agent. We provide: i.) A summary of the domain properties that define the ACO problem; ii.) A comprehensive evaluation of the extent to which domains used for benchmarking DRL approaches are comparable to ACO; iii.) An overview of state-of-the-art approaches for scaling DRL to domains that confront learners with the curse of dimensionality, and; iv.) A survey and critique of current methods for limiting the exploitability of agents within adversarial settings from the perspective of ACO. We conclude with open research questions that we hope will motivate future directions for researchers and practitioners working on ACO.

Keywords: Autonomous Cyber Operations, Multi-agent Deep Reinforcement Learning, Adversarial Learning

1 Introduction

The rapid increase in the number of cyber-attacks in recent years has raised the need for responsive, adaptive, and scalable *autonomous cyber operations* (ACO) [148, 8]. Adaptive solutions are desirable due to cyber-criminals increasingly showing an ability to evade conventional security systems, which often lack the ability to detect new types of attacks [182]. The ACO problem can be formulated as an adversarial game involving a Blue agent tasked with defending cyber resources from a Red attacker [14]. Deep reinforcement learning (DRL) has been identified as a suitable machine learning (ML) paradigm to apply to ACO [7, 112, 123]. However, current "out of the box" DRL solutions do not scale well to many real world scenarios. This is primarily due to ACO lying at the intersection of three open problem areas for DRL, namely: i.) The efficient processing and exploration of vast high-dimensional state spaces [2]; ii.) Large combinatorial action spaces, and; iii.) Minimizing the exploitability of DRL agents in adversarial games [65] (see Figure 1).

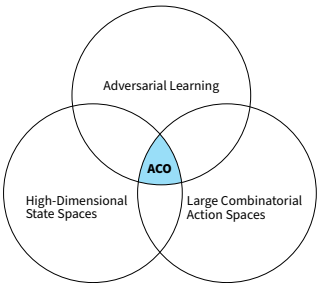


Fig. 1: Three challenges that an idealised DRL-ACO agent must conquer.

The DRL literature features a plethora of efforts addressing the above challenges individually. Here, we survey these efforts and define an idealised DRL agent for ACO. This survey provides an overview of the current state of the field, defines long term objectives, and poses research questions for DRL practitioners and ACO researchers to dig their teeth into.

Our contributions can be summarised as follows:

1.) To enable an extensive evaluation of future DRL-ACO approaches, we provide an overview of ACO benchmarking environments, as well as environments found within the DRL literature that confront learners with comparable challenges.

2.) We identify suitable methods for addressing the curse of dimensionality for ACO. This includes a summary of approaches for state-abstraction, efficient exploration and mitigating catastrophic forgetting, as well as a critical evaluation of high-dimensional action approaches.

3.) We formally define the ACO problem from the perspective of adversarial learning. Even within "simple" adversarial games, finding (near) optimal policies is

non-trivial. We therefore review principled methods for limiting exploitability, and map out paths towards scaling these approaches to the full ACO challenge.

2 Related Work

A number of surveys have been conducted in recent years that provide an overview of the different types of cyber attacks (e.g., intrusion, spam, and malware) and the ML methodologies that have been applied in response [182, 113, 167, 121, 196]. Given that ML methods themselves are susceptible to adversarial attacks, there have also been efforts towards assessing the risk posed by adversarial learning techniques for cyber security [45, 171]. However, while these works evaluate existing threats to ML models in general (e.g., white-box attacks [140] and model poisoning attacks [101]), our survey focuses on the adversarial learning process for DRL agents within ACO, desirable solution concepts, and a critical evaluation of existing techniques towards limiting exploitability.

Nguyen and Reddi [148] surveyed the DRL for cyber security literature, providing an overview of works where DRL-based security methods were applied to cyber–physical systems, autonomous intrusion detection techniques, and multi-agent DRL-based game theory simulations for defence strategies against cyber-attacks. In contrast, our work focuses on generation after next solutions; we capture the challenges posed by the ACO problem at scale and survey the DRL literature for suitable methods designed to address these challenges separately, providing the building blocks for an idealised ACO-DRL agent. Such an agent will require a suitable evaluation environment. While there have been efforts towards evaluating cyber security datasets [227, 97, 182], to the best of our knowledge we are the first to evaluate the extent to which cyber security and other benchmarking environments are representative of the full ACO problem.

3 Background & Definitions

Below we provide the definitions and notations that we will rely on throughout this survey. First, we will formally define the different types of models through which the interactions between RL agents and environments can be described. We will encounter each type in this survey (See Figure 2 for an overview).

3.1 Markov Decision Process

Markov Decision Processes (MDPs) describe a class of problems – fully observable environments – that defines the field of RL, providing a suitable model to formulate interactions between reinforcement learners and their environment [193]. Formally: An MDP is a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$, where: $\mathcal{S}$ is a finite set of states; for each state $s \in \mathcal{S}$ there exists a finite set of possible actions $\mathcal{A}$; $\mathcal{R}$ is a real-valued payoff function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S}' \to \mathbb{R}$, where $\mathcal{R}_a(s, s')$ is the expected payoff following a state transition from s to s' using action a; $\mathcal{P}$ is a state transition probability

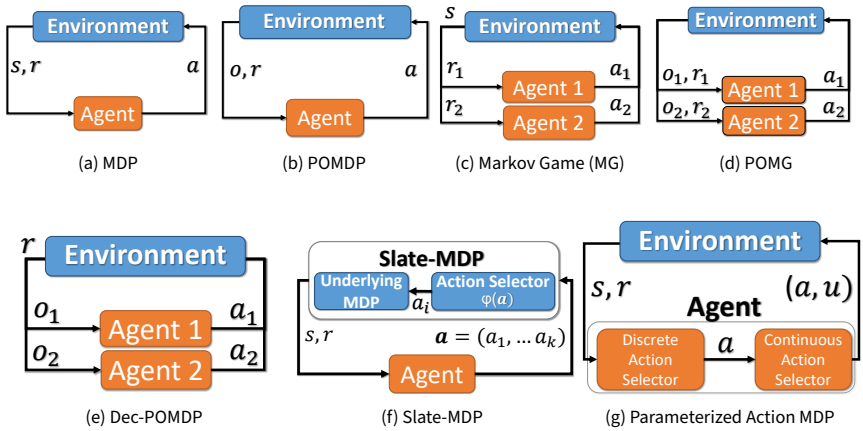


Fig. 2: An overview of the problem formulations discussed in this survey. Within these formulations we have the following variables: states s, rewards r, actions a, and observations o. For the Parameterized Action MDP we differentiate between discrete actions a and continuous actions u.

matrix $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S}' \rightarrow [0, 1]$, where $\mathcal{P}_a(s, s')$ is the probability of state s transitioning into state s' using action a. MDPs can have terminal (absorbing) states at which the episode ends.

3.2 Partially Observable Markov Decision Process

Numerous environments lack the full observability property of MDPs [151]. Here, a Partially Observable MDP (POMDP) extends an MDP $\mathcal{M}$ by adding $\langle \Omega, \mathcal{O} \rangle$, where: Ω is a finite set of observations; and $\mathcal{O}$ is an observation function defined as $\mathcal{O} : \mathcal{S} \times \mathcal{A} \times \Omega \rightarrow [0, 1]$, where $\mathcal{O}(o|s, a)$ is a distribution over observations o that may occur in state s after taking action a.

3.3 Markov Games

Many environments, including the ones that are the focus of this survey, feature more than one learning agent. Here, game theory offers a solution via Markov games (also known as stochastic games [181]). A Markov game is defined as a tuple $(n, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, that has a finite state space $\mathcal{S}$; for each state $s \in \mathcal{S}$ a joint action space $(\mathcal{A}_1 \times ... \times \mathcal{A}_n)$, with $\mathcal{A}_p$ being the number of actions available to player p; a state transition function $\mathcal{P} : \mathcal{S}_t \times \mathcal{A}_1 \times ... \times \mathcal{A}_n \times \mathcal{S}_{t+1} \rightarrow [0, 1]$, returning the probability of transitioning from a state s_t to s_{t+1} given an action profile $a_1 \times ... \times a_n$; and for each player p a reward function: $\mathcal{R}_p : \mathcal{S}_t \times \mathcal{A}_1 \times ... \times \mathcal{A}_n \times \mathcal{S}_{t+1} \rightarrow \mathbb{R}$ [181]. We allow *terminal states* at which the game ends. Each state is fully-observable.

3.4 Partially Observable Markov Games

As with POMDPs, we cannot assume the full observability property required by Markov games. A Partially Observable Markov Game (POMG) is an extension of Markov Games that includes $\langle \Omega, \mathcal{O} \rangle$ a set of joint observations Ω; and an observation probability function defined as $\mathcal{O}_p : \mathcal{S} \times \mathcal{A}_1 \times ... \times \mathcal{A}_n \times \Omega \rightarrow [0, 1]$. For each player p the observation probability function $\mathcal{O}_p$ is a distribution over observations o that may occur in state s, given an action profile $a_1 \times ... \times a_n$.

3.5 Decentralized-POMDPs

A Decentralized-POMDP (Dec-POMDP) is a Partially Observable Markov Game where, at each step, all n agents receive an identical reward [151].

3.6 Slate Markov Decision Processes

Some environments such as recommender systems require a custom model. Here, Slate-MDPs provide a solution [191]. Given an underlying MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$, a Slate-MDP is a tuple $\langle \mathcal{S}, \mathcal{A}^l, \mathcal{P}', \mathcal{R}' \rangle$. Within this formulation $\mathcal{A}^l$ is a finite discrete action space $\mathcal{A}^l = \{\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_N\}$, representing the set of all possible slates to recommend given the current state s. Each slate can be formulated as $\boldsymbol{a}_i = \{a_i^1, a_i^2, ...a_i^K\}$, with K representing the size of the slate. Slate-MDPs assume an action selection function $\varphi : \mathcal{S} \times \mathcal{A}^l \rightarrow \mathcal{A}$. State transitions and rewards are, as a result, determined via functions $\mathcal{P}' : \mathcal{S} \times \mathcal{A}^l \times \mathcal{S}' \rightarrow [0, 1]$ and $\mathcal{R}' : \mathcal{S} \times \mathcal{A}^l \times \mathcal{S}' \rightarrow \mathbb{R}$ respectively. Therefore, given an underlying MDP $\mathcal{M}$, we have $\mathcal{P}'(s, \boldsymbol{a}, s') = \mathcal{P}(s, \varphi(\boldsymbol{a}), s')$ and $\mathcal{R}'(s, \boldsymbol{a}, s') = \mathcal{R}(s, \varphi(\boldsymbol{a}), s')$. Finally, there is an assumption that the most recently executed action can be derived from a state via a function $\psi : \mathcal{S} \rightarrow \mathcal{A}$. Note, there is no requirement that $\psi(s_{t+1}) \in \boldsymbol{a}_t$, therefore, the action selected can also be outside the provided slate [1].

3.7 Parameterized Action MDPs

Parameterized Action MDPs (PA-MDPs) are a generalization of MDPs where the agent must choose from a discrete set of parameterized actions [218, 130]. More formally, PA-MDPs assume a finite discrete set of actions $\mathcal{A}_d = \{a_1, a_2, ..., a_n\}$ and for each action $a \in \mathcal{A}_d$ a set of continuous parameters $u_a \subseteq \mathbb{R}^{m_a}$, where m_a represents the dimensionality of action a. Therefore, an action is a tuple (a, u) in the joint action space, $\mathcal{A} = \bigcup_{a \in \mathcal{A}_d} \{(a, u) | \, \mathcal{U}_a\}$.

3.8 Types of Action Spaces

From the above definitions we see that environments have different requirements with regard to their action spaces [91]. This survey will discuss approaches for:

Discrete actions $a \in \{0, 1, ...N\}$, with $N \in \mathcal{N}$ available actions in a given state.

MultiDiscrete action vectors $\boldsymbol{a}$. Each a_i is a discrete action with N possibilities.

[1] There are environments that treat $\psi(s_{t+1}) \notin \boldsymbol{a}_t$ as a failure property, upon which an episode terminates [191].

Continuous actions, where an action $a \in \mathbb{R}$ is a real number, or a vector of real numbered actions.

Slate actions $\boldsymbol{a} = \{a_1, a_2, ..., a_n\}$ from which one can be selected.

Parameterized Actions, mixed discrete-continuous, e.g., a tuple (a, u) where a is a discrete action, and u is a continuous action.

3.9 Reinforcement Learning

The goal of an RL algorithm is to learn a policy π that maps states to a probability distribution over the actions $\pi : \mathcal{S} \rightarrow P(\mathcal{A})$, so as to maximize the expected return $\mathbb{E}_\pi[\sum_{t=0}^{H-1} \gamma^t r_t]$. Here, H is the length of the horizon, and γ is a discount factor $\gamma \in [\emptyset, 1]$ weighting the value of future rewards. Many of the approaches discussed in this survey use the Q-learning algorithm introduced by Watkins [216, 215] as their foundation. Using a dynamic programming approach, the algorithm learns action-value estimates (Q-values) independent of the agent's current policy. Q-values are estimates of the discounted sum of future rewards (the return) that can be obtained at time t through selecting an action $a \in \mathcal{A}$ in a state s_t, providing the optimal policy is selected in each state that follows. Q-learning is an *off-policy* temporal-difference (TD) learning algorithm.

In environments with a low-dimensional state space Q-values can be maintained using a Q-table. Upon choosing an action a in state s according to a policy π, the Q-table is updated by bootstrapping the immediate reward r received in state s' plus the discounted expected future reward from the next state, using a discount factor $\gamma \in (\emptyset, 1]$ and scalar α to control the learning rate: $Q_{k+1}(s, a) \leftarrow Q_k(s, a) + \alpha\big(r + \gamma \max_{s \in \mathcal{S}} [Q_k(s', a) - Q_k(s, a)]\big)$. Many sequential decision problems have a high-dimensional state space. Here, Q-values can be approximated using a function approximator, for instance using a neural network. A recap of popular DRL approaches is provided in Appendix A.

3.10 Joint Policies

Our domain of interest can feature multiple agents. For each agent p, the strategy π_p represents a mapping from the state space to a probability distribution over actions: $\pi_p : \mathcal{S}_p \rightarrow \Delta(\mathcal{A}_p)$. Transitions within Markov games are determined by a joint policy. The notation $\boldsymbol{\pi}$ refers to a joint policy of all agents. Joint policies excluding agent p are defined as $\boldsymbol{\pi}_{-p}$. The notation $\langle \pi_p, \boldsymbol{\pi}_{-p} \rangle$ refers to a joint policy with agent p following π_p while the other agents follow $\boldsymbol{\pi}_{-p}$.

4 Environments

A number of benchmarking environments have emerged in recent years that grant learning agents access to an abstracted version of the ACO problem, including the CybORG environment [188] and YAWNING TITAN (YT) [11]. In ACO-gyms, the Red agent is tasked with moving through the network graph, compromising nodes in order to progress. The end goal in most cases is to reach and impact a high-value target node. In all ACO-gyms, the task of the Blue agent is conceptually identical

– to identify and counter the Red agent's intrusion and advances using the available actions. These actions differ significantly per ACO-gym, with actions including scanning/monitoring for Red activity; isolating, making safe, and reducing vulnerability of nodes; and deploying decoy nodes.

Since the specific observations and actions are ACO-gym dependant, it can be surmised that these are not particular to the ACO problem. Any other environment which shares core features in its internal dynamics, and structure of observation and action spaces, should provide a suitable platform for developing methodologies to tackle ACO challenges before the ACO-gyms themselves are able to fully represent the problem. We have identified a total of 14 desirable criteria to assess the suitability of environments from other domains for developing and assessing techniques for tackling challenges in the ACO domain (see Table 1).

Code: The availability of code can significantly speed-up research efforts. We only list codeless environments that offer unique properties, or require minimal effort to implement.

Adversarial: ACO is typically an adversarial game between a Blue and a Red agent.

General Sum & Team Games: Networks can span multiple geographic locations. Physical restraints, such as data transmission capacity and latency, can therefore necessitate a multi-agent reinforcement learning (MARL) approach.

Stochastic: ACO features stochastic factors, e.g., user activity, equipment failures and the likelihood of a Red attack succeeding. To meet this criteria stochastic state transitions are required. Random starting positions alone are insufficient.

Partially Observable: In all ACO-gyms, Blue must perform a scan/monitor action to observe Red activities. Future ACO-gyms are likely to further reduce observability, requiring more sophisticated Blue policies to detect Red agent actions.

Graph-Based: With ACO taking place on networks, ACO-gyms contain underlying dynamics unique to graph-based domains. Graph-based observation spaces are not necessarily required to fulfil this criteria. However, an environment being checked implies the presence of a graph structure.

Multi-Discrete: Given the high-dimensionality challenge present in both the observation and action spaces of ACO-gyms – which increase in size with the number of nodes – the presence of multiple discrete dimensions in the observation and action spaces is highly desirable.

Extensible Dimension: A property of graph-based dynamics is a dimension that can be expanded to geometrically increase the size of the composite space, e.g., the number of nodes in a network. Environments that meet this criteria allow methodologies to be tested for scalability to larger observation and action spaces.

Continuous Dimension: As ACO-gyms scale up, a subset of action dimensions might need to be treated as continuous, e.g., networks featuring a large number of IP addresses. Additionally, continuous attributes may be present in the observation spaces, representing node vulnerability for example [11]. Therefore, environments with *mixed* input and output types are desirable.

Traditionally Intractable: By identifying environments where the observation or action space is intractable, we indicate one of two properties that pose a significant, if not impossible, challenge to traditional RL methodologies:

1. The multi-discrete space causes an exponential scaling of the size of the space when flattened, such that it quickly becomes computationally intractable.
2. The observation or action space changes in size as a function of the environment; standard DRL methods require a consistent shape and size.

Environments that meet the above criteria pose a challenge to traditional DRL methodologies, and therefore necessitate novel algorithms or formulations. However, even current ACO-gyms do not meet all these criteria (see Table 1). Below we provide a critical evaluation of current ACO-gyms.

Environment Properties																														
Categories		ACO			NBD						GD2D								DCC			RS			MG		
General	Name	CybORG [187]	Yawning Titan [11]	NASim [179]	Intelligent Communication Jamming [223]	Multicast Scheduling Problem [115]	Active Network Management [78, 79]	SUMO [125]	Iroko [172]	Categorised N-Rooms [232]	Selective Predator-Prey [184, 42]	Multi-Step Plan Environment [46]	Platform Jumping [218]	MicroRTS [84]	Pommerman [168]	DinerDash [35]	Half Field Offense [75]	MuJoCo [197, 52]	Petting Zoo [53]	Nocturne [210]	RecoGym [170]	ReSim [88]	RL4RS [213]	Zork [232]	Dou Dizhu [234]	SMAC [173, 48]
	Code	✓	✓	✓	X	X	✓	✓	✓	X	✓	X	X	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Environment Properties	General Sum / Team Games	↗	↗	X	✓	X	X	✓	X	X	✓	X	X	✓	✓	X	✓	X	✓	✓	X	X	X	X	✓	✓
	Adversarial	↗	↗	X	✓	X	X	X	X	X	✓	X	X	✓	✓	X	✓	X	X	↗	X	X	X	X	✓	✓
	Stochastic	✓	✓	✓	✓	X	↗	↗	✓	✓	✓	X	X	✓	↗	X	✓	X	✓	X	✓	✓	X	✓	X	✓
	Partially Observable	✓	✓	✓	X	X	↗	X	X	X	X	✓	X	✓	✓	X	X	X	✓	✓	X	✓	X	X	✓	✓
	Graph-Based	✓	✓	✓	X	X	✓	✓	✓	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Observation Properties	Multi-Discrete	✓	✓	✓	✓	✓	X	X	X	✓	X	X	X	✓	✓	✓	X	X	✓	X	✓	✓	✓	X	✓	✓
	Extensible Dim.	↗	✓	✓	✓	✓	↗	✓	✓	✓	✓	X	X	✓	✓	↗	X	↗	✓	✓	X	✓	X	X	X	✓
	Continuous Dim.	X	X	✓	X	✓	✓	✓	✓	X	✓	X	✓	X	✓	X	✓	✓	✓	✓	X	✓	✓	✓	X	✓
	Traditionally Intractible	↗	✓	✓	↗	X	X	X	X	↗	✓	X	X	X	X	X	X	X	↗	↗	X	✓	X	✓	X	✓
Action Properties	Multi-Discrete	✓	✓	✓	✓	✓	X	✓	X	✓	✓	✓	X	✓	X	✓	X	↗	↗	✓	X	✓	X	X	✓	✓
	Extensible Dim.	↗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	X	✓	X	↗	X	↗	✓	X	X	✓	X	X	X	✓
	Continuous Dim.	X	X	X	X	X	✓	X	✓	X	X	X	✓	X	X	X	✓	X	✓	↗	X	X	X	X	X	X
	Traditionally Intractible	↗	✓	✓	✓	✓	X	X	X	↗	✓	✓	✓	↗	X	X	X	X	↗	X	X	X	X	✓	✓	✓

Table 1: An overview of the environments that we evaluated for this survey and their properties, with ✓ and X indicating if the environment fulfills a criteria. The third symbol, ↗ indicates that the environment does not fulfil the criteria currently (due to enabling software features not being present, whether by design or due to incomplete implementations) but that the required features could be implemented with a modest amount of development effort, without the need for major extensions of environment scope. Along with ACO itself, other categories include: Gridworld Domains & 2D Games (GD2D), Discretised Continuous Control Games (DCC), Network-Based Domains (NBD), Recommender Systems (RS), and, Miscellaneous Games (MG). The latter is for unique environments which do not justify an additional category.

CybORG: The Cyber Operations Research Gym (CybORG) [187] is a framework for implementing a range of ACO scenarios. It provides a common interface at two levels of fidelity in the form of a finite state machine and an emulator. The latter provides for each node: an operating system; hardware architecture; users, and; passwords [188]. CybORG allows for both the training of Blue and Red agents. Therefore, in principle, it provides an ACO environment for adversarial learning.

However, in practice further modifications are required in order to support the loading of a DRL opponent. This is due to environment wrappers (for reducing the size of the observation space and mapping actions) only being available to the agent that is being trained. Currently, the opponent within the environment instance receives raw unprocessed dictionary observations. Furthermore, while CybORG does allow for customizable configuration files, in practice we find that specifying a custom network is non-trivial.

Yawning Titan: The YT environment intentionally abstracts away the additional information used by CybORG's emulator, facilitating the rapid integration and evaluation of new methods [11]. It is built to train ACO agents to defend arbitrary network topologies that can be specified using highly customisable configuration files. Each machine in the network has parameters that affect the extent to which they can be impacted by Blue and Red agent behaviour, including vulnerability scores on how easy it is for a node to be compromised. However, while YT is an easy to configure lightweight ACO environment, it currently lacks the ability to train Red agents, and does not support adversarial learning.

Network Attack Simulator (NASim): This ACO environment is a simulated computer network complete with vulnerabilities, scans and exploits designed to be used as a testing environment for AI agents and planning techniques applied to network penetration testing [179]. NASim meets a number of our criteria, and supports scenarios that involve a large number of machines. However, Nguyen et al. [146] find that the probability of Red agent attacks succeeding in NASim are far greater than in reality. As a result the authors add additional rules to increase the task difficulty. NASim also does not support adversarial learning.

While ACO-gyms currently do not meet all the criteria that we have outlined above, it is likely that they will in the future. As the number of observation and action space dimensions grow with ACO-gyms maturing, so too will their scale; and since, in reality, the number of nodes on a computer network does not remain constant, neither shall the observation and action space sizes. Below we discuss a number of environments from several domains which could be used to develop and evaluate approaches designed to address a subset of the challenges identified. An overview of the environments that we evaluated is provided in Table 1. Below we focus on four environments that meet a large number of ACO requirements.

MicroRTS [84] is a simple Real-Time Strategy (RTS) game. It is designed to enable the training of RL agents on an environment which is similar to PySC2 [211], the RL environment adaptation of the popular RTS game StarCraft II which possesses huge complexity. Therefore, MicroRTS is a lightweight version of PySC2, without the extensive (and expensive) computational requirements. The game features a gridworld-like observation space, where for a map of size $h \times w$, the observation space is of size $h \times w \times f$, where f is a number of discrete features which may exist in any square. This observation space can be set to be partially observable. The action space is a large multi-discrete space, where each worker is commanded via a total of seven discrete actions. The number of workers will change during the game, making the core RTS action space intractable. In order to handle this, the authors of MicroRTS implemented action decomposition as part

of the environment, and the action space is separated into the unit action space (of 7 discrete actions), and the player action space. While the authors discuss two formulations of this player action space, the one which is documented in code is the *GridNet* approach, whereby the agent predicts an action for each cell in the grid, with only the actions on cells containing player-owned units being carried out. This leads to a total action space size of $h \times w \times 7$. The challenge of varying numbers of agents is remarkably similar to the challenge of varying numbers of nodes in ACO. The MicroRTS environment, if action decomposition were to be stripped away, would be a strong candidate for handling large and variably-sized discrete action spaces.

Nocturne [210] is a 2D driving simulator, built from the Waymo Open Dataset [190]. Agents learn to drive in a partially observable environment without having to handle direct sensor inputs, e.g., video from imaging cameras. Nocturne is a general-sum game. Each agent is required to both coordinate its own actions to reach its own goal and cooperate with others to avoid congestion. It is partially observable, with objects in the environment (both static and moving) causing occlusion of objects (from the egocentric perspective of the agent). However, it is not stochastic, with the state-transition probabilities being deterministic.

SUMO [125] is a traffic simulation tool which allows for the implementation of complex, distributed traffic light control scenarios. While neither Python-based nor intended for RL, there exists a third-party interface for fulfilling these criteria [9]. There are two configurations for this environment; single-agent, and multi-agent. For this review, we consider the multi-agent configuration, but treat it as a single-agent problem since, while the multi-agent scenario introduces greater complexity with multiple junctions needing to be controlled, it does not necessitate the use of multiple agents. The goal of an agent in SUMO is to control traffic signals at junctions in order to minimise the delay of vehicles passing through. The observation space is a small, but extensible (with regard to the number of junctions) continuous space, containing information about the amount of traffic in incoming lanes. For the action space, each set of lights is controlled by a single discrete variable, each corresponding to a configuration of lights. For example, at a two-way single intersection there are four possible configurations. However, each intersection type has a different number of configurations, meaning that the action space is not only multi-discrete, but also uneven. Further, as the intersections are connected by roads in the simulation, the environment will inevitably contain graph-like dynamics, i.e., the actions at, and traffic through, each intersection will affect the state at connected junctions. As such, the action space and dynamics of SUMO are applicable to ACO. However, the small observation space and lack of adversarial or partially observable properties limit this applicability.

RecSim [88] is a configurable framework for creating RL environments for recommender systems. In RecSim, an agent observes multiple sets of features from both a user and a document database, and makes document recommendations based on these. These sets of features are comprised of features from the user, their responses to previous recommendations, and the available documents for recommendation. These observations fulfil all desirable criteria related to observation

spaces. Not only are they stochastic (due to randomness in the user's behaviour) and partially observable (due to incompleteness in the visible user features), but they are also an extensible mix of discrete and continuous features which scale exponentially with both the number of documents and the number of features extracted from them. As such, the observation space of RecSim poses a complex challenge similar to that presented in ACO. The action space, however, is less remarkable. The agent must choose a number of documents to recommend, which is a multi-discrete space which is potentially extensible as the number of recommendations and documents increases. However, this is unlikely to be intractable unless the number of documents becomes vast. Nevertheless, this large space complements the novel and complex observation space, making this a strong candidate for experimenting with approaches for ACO challenges.

In summary, in this section, we have reviewed multiple environments and domains which hold some relevance to the challenges presented by ACO. While none of the environments reviewed meet all of our desirable criteria. Each challenge is represented in at least one domain. The most scarce of these challenges, but one which is core to ACO, is the presence of graph-based dynamics. These could only be found in network-based domains, and even there, not all environments possess them. Therefore, if the ACO solution being investigated aims to tackle graph-based dynamics, one would be limited to environments in this domain. Should graph-based dynamics not be required for development, there are several options for the other desirable criteria. StarCraft II presents a particularly complex challenge, even in cut-down versions of the game such as SMAC [173, 48], which meets a large number of our requirements. However, it is a computationally demanding environment. Micro-RTS [84], an environment specifically designed to provide a less computationally demanding version of the problems presented by StarCraft II, is an attractive alternative. While the observation space is a simplified version of StarCraft II's, the action space holds much of the same complexity, with variable numbers of agents posing an issue of intractability.

5 Coping with Vast High-Dimensional Inputs

Due to an ever growing means through which large amounts of data can be harvested, the curse of dimensionality is a prevalent problem for ML and data mining tasks. This has implications for DRL. Learning efficiency is reduced due to unnecessary features contributing noise [233], and the state space can grow exponentially with the size of the state representation [28]. Here, maintaining a sufficient sample spread over the state-action space becomes challenging [41].

Dimensionality reduction techniques are a natural choice for dealing with unnecessary/noisy features. Benefits include: the elimination of irrelevant data and redundant features, while preserving the variance of the dataset; improved data quality; reduced processing time and memory requirements; improved accuracy; shorter training and inference times (i.e., reduced computing costs), and; improved performance [233]. Two popular approaches are *feature selection* and *feature extraction*. Feature selection aims to find the optimal sub-set of relevant

features for training a model, which is an Non-deterministic Polynomial (NP)-hard problem [134]. In contrast, feature extraction involves creating linear combinations of the features, while preserving the original relative distances in the latent structures. The dimensionality is decreased without losing much of the initial information [233]. However, the resulting encodings are uninterpretable for humans.

DRL uses Deep Neural Networks (DNNs) to directly extract features from high-dimensional data [142, 10]. Nevertheless, feature selection can provide a valuable pre-processing step for training DNNs. For example, anomaly detection DNNs for cyber security applications, tasked with differentiating benign from malicious packets, were found to perform better when trained on data where feature selection had been applied [39].

Below we will provide an overview of DNNs used in DRL for feature extraction. Then we shall discuss state of the art DRL approaches for further eliminating unnecessary information through learning *abstract* states and discuss advanced exploration strategies towards enabling the sufficient visitation of all *relevant* states to obtain accurate utility estimates [28, 162]. Finally, we shall take a closer look at approaches towards mitigating catastrophic forgetting, DNN's tendency to unlearn previous knowledge [154, 41]. A summary of the approaches discussed in this section is provided in Table B1 in Appendix B.

5.1 Function Approximators

Many of the successes in RL over the past decade rely on the ability of DNNs to identify intricate structures and extract compact features from complex high-dimensional samples [67, 94, 111]. These approaches work well for numerous domains that confront learners with the curse of dimensionality, such as the *Arcade Learning Environment* (ALE) [20], where DNNs are used to encode image observations. However, many of these domains are fully observable and contain state spaces with a dimensionality that is manageable for current DNNs. In contrast, this survey is focused on environments where the architectures used by standard DRL approaches cannot scale, necessitating innovative solutions. Here, considerations are required with respect to efficiently encoding an *overwhelmingly* large observation space to a low dimensional representation, while limiting concessions regarding performance. First, we will discuss two popular feature extraction techniques used by DRL: Convolutional Neural Networks (CNNs) and Graph Neural Networks (GNNs).

Convolutional Neural Networks: CNNs can be optimized to extract features from high dimensional arrays and tensors via multiple stacked linear convolution and pooling layers, banks of filters which are convolved with an input to produce an output map [111, 67]. The first layer may learn to extract edges, which can then be combined into corners and contours by the subsequent layers. These features can be combined to form the object parts that enable a classification, for instance through adding fully connected layers that precede the output layer [111, 67].

CNNs have a large learning capacity and can be trained to implement complex functions that are sensitive towards minute details within inputs [111, 104, 222]. DNNs can be trained end-to-end using stochastic gradient descent via the

back-propagation procedure, providing that the network consists of smooth functions [67]. CNNs take advantage of assumptions regarding the location of pixel dependencies within images. This allows CNNs to reduce the number of weighted connections compared to fully-connected DNNs [104]. However, while CNNs are still considered a state-of-the-art (SOTA) approach for processing data in the form of arrays and tensors, there are formulations where CNNs are not directly applicable, for instance: graph-based representations.

Graph Neural Networks: Graph-based representations are a popular choice for many domains, such as traffic forecasting, drug discovery, and ACO [144]. Conventional DNNs are not applicable to graphs, due to the graph's uneven structure, irregular size of unordered nodes, and dynamic neighbourhood compositions [99]. Here, GNNs have emerged as a powerful tool [90]. GNNs have successfully been applied to graphs that systematically model relationships between node entities, and represent simplified versions of complex problems. Tasks include node classification, link prediction, community detection and graph classification [228]. GNNs model both graph structure and node attributes via a message passing scheme, propagating relevant feature information of nodes to their neighbours until a stable equilibrium is found [99]. GNNs are primarily applicable to environments where graphs can be used to capture relationships between entities [144], e.g., enabling an effective factorisations of value functions for Multi-Agent Reinforcement Learning (MARL) in the StarCraft Multi-Agent Challenge (SMAC) [102], or for modelling relationships between agents and objects in multi-task DRL [83].

Munikoti et al. [144] recently conducted a survey on the opportunities and challenges for graph-based DRL, defining an idealised GNN for DRL as: i.) *dynamic*; ii.) *scalable*; iii.) providing *generalizability*, and; iv.) applicable to *multiagent* systems. Dynamic refers to DRL's need for GNN approaches that can cope with time varying network configurations and parameters, e.g., the number of hosts varying over time. While GNN architectures have been proposed for dynamic graphs (e.g., spatial-temporal GNNs [149]), tasks including node classification, link prediction, community detection, and graph classification could benefit from further improvements [144]. With regard to generalizability, there is a danger that a DRL agent can overfit on the graph structure(s) seen during training, and is unable to generalize across different graphs [144].

Three popular GNNs currently receiving attention from the DRL community are [144]:

1.) *Graph Convolutional Networks (GCNs):* Using a semi-supervised learning approach, GCNs were the first GNNs to apply convolutional operations similar to those used by CNNs to learn from graph-structured data [99]. The model can learn encodings for local graph structures and the features of nodes. GCNs scale linearly in the number of graph edges. However, the entire graph adjacency matrix is required to learn these representations. Therefore, GCNs cannot generalize over graphs of different sizes [144]. This has implications for DRL, since it restricts the usage of GCNs to tasks with a static network configuration.

2.) *GraphSAGE* learns the topological node structure and the distribution of node features within a confined neighbourhood, computing a node's local role in the

graph along with global position [73]. Using an inductive learning approach, GraphSAGE samples node features in the local neighbourhood of each node and learns a functional mapping that aggregates the information received by each node. It is scalable to graphs of different sizes, and can be applied to different sub-graphs, thereby not requiring all the nodes to be present during training.

3.) *Graph Attention Networks (GAT)* use masked self-attentional layers, allowing for an implicit specification of different weights for nodes in the neighbourhood, without the need for computationally expensive matrix operations or assuming knowledge of the graph structure upfront [206]. The approach selectively aggregates node contributions while suppressing minor structural details.

The type of GNN selected for DRL depends on the properties of the environment. For large graphs GNN approaches are required that can be applied to sub-graphs, such as GraphSAGE [73]. Position-aware GNNs [228] should be used when the position of a node provides critical information [144]. Meanwhile, for dynamic graphs an appropriate approach would be to fuse GNNs with a Recurrent Neural Network (RNN) to capture a graph's evolution over time, allowing the network to establish spatio-temporal dependencies [144]. Indeed, even for non-graph-based environment representations we must consider the impact of partial observations o, or a trajectory of observations $\tau = \{o_{t-n}, o_{t-n+1}...o_t\}$. Here RNN components are also utilized to retain relevant information [107, 114, 82]. This allows the learner to encode and keep track of relevant objects, e.g., the location of other agents observed during previous time-steps [107, 93].

5.2 State Abstraction

The aim of state abstraction is to obtain a compressed model of an environment that retains all the useful information – enabling the efficient training and deployment of a DRL agent over an abstract formulation. Solving the abstract MDP is equivalent to solving the underlying MDP [6, 5, 27, 28]. State abstraction groups together semantically similar states, abstracting the state space to a representation with lower dimensions [230]. A motivating example for the importance of state abstraction is the ALE game Pong [20], where success only requires access to the positions and velocities of the two paddles and the ball [62].

Abel et al. [5] identify various types of abstraction discussed in the literature that can involve states and also actions, including: state abstraction [4], temporal abstraction [165], state-action abstractions [6], and hierarchical RL approaches [105, 44]. In this section we shall focus our attention on state abstraction. Formally, given a high-dimensional state space $\mathcal{S}$, the goal of state abstraction is to implement a mapping $\phi : \mathcal{S} \rightarrow \mathcal{S}_\phi$ from each state $s \in \mathcal{S}$ to an abstract state $s_\phi \in \mathcal{S}_\phi$, where $|\mathcal{S}_\phi| \ll |\mathcal{S}|$ [1], and ϕ is an encoder [1]. This expands the set of RL problem definitions defined in Section 3, introducing the notion of an Abstract MDP, as illustrated in Figure 3.

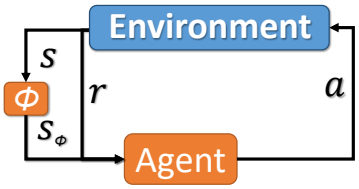


Fig. 3: Depiction of an Abstract MDP, that includes a mapping $\phi : \mathcal{S} \rightarrow \mathcal{S}_\phi$ from the full state s to an abstract state s_ϕ.

In practice low dimensional representation are often obtained using Variational AutoEncoder (VAE) based architectures [98]. For example, Giannakopoulos et al. [63] apply neural discrete representation learning, mapping high-dimensional raw video observations from an RL agent's interactions with the environment to a low dimensional discrete latent representation, using a Vector Quantized AutoEncoder (VQ-AE) trained to reconstruct the raw video data. The benefits of the approach are demonstrated within a 3D navigation task in a maze environment constructed in Minecraft.

The work from Giannakopoulos et al. [63] and others [194, 27, 28] demonstrates the ability of state abstraction to reduce noise from raw high-dimensional inputs. However, discarding too much information can result in the encoder failing to preserve essential features. Therefore, encoders must find a balance between appropriate degree of compression and adequate representational power [3]. Using *apprenticeship learning*, where the availability of an expert demonstrator providing a policy π_E is assumed, Abel et al. [5] seek to understand the role of information-theoretic compression in state abstraction for sequential decision making. The authors draw parallels between state-abstraction for RL and compression as understood in information theory. The work focuses on evaluating the extent to which an agent can perform on par with a demonstrator, while using as little (encoded) information as possible. Studying this property resulted in a novel objective function with which a VAE [98] can be optimized, enabling a convergent algorithm for computing latent embeddings with a trade-off between compression and value.

Gelada et al. [62] and Zhang et al. [235] observe that encoder-decoder approaches are typically task agnostic – encodings represent all dynamic elements that they observe, even those which are not relevant. An idealised encoder, meanwhile, would learn a robust representation that maps two observations to the same point in the latent space while ignoring irrelevant objects that are of no consequence to our learning agent(s). Both works rely on the concept of bisimulation to avoid training a decoder. The intuition behind bisimulation is as follows.

Definition 5.1 (Bisimulation.)**.** *Given an MDP $\mathcal{M}$, an equivalence relation B between states is a bisimulation relation if, for all states $s_i, s_j \in \mathcal{S}$ that are equivalent under B (denoted $s_i \equiv_B s_j$) the following conditions hold:*

$$\mathcal{R}(s_i, a) = \mathcal{R}(s_j, a), \forall a \in \mathcal{A}, \tag{1}$$

$$\mathcal{P}(G|s_i, a) = \mathcal{P}(G|s_j, a), \forall a \in \mathcal{A}, \forall G \in \mathcal{S}_B, \quad (2)$$

where $\mathcal{S}_B$ is the partition of $\mathcal{S}$ under the relation B (the set of all groups G of equivalent states), and $\mathcal{P}(G|s, a) = \sum_{s' \in G} \mathcal{P}(s'|s, a)$.

Zhang et al. [235] propose *deep bisimulation for control* (DBC) that learns directly on a bisimulation distance metric. This allows the learning of invariant representations that can be used effectively for downstream control policies, and are invariant with respect to task-irrelevant details. The encoders are trained in a manner such that distances in latent space equal bisimulation distances in the actual state space. The authors evaluated their approach on visual Multi-Joint dynamics with Contact (MuJoCo) tasks where control policies must be learnt from natural videos with moving distractors in the background. Exactly partitioning states with bisimulation is generally not feasible when dealing with a continuous state space, therefore a pseudometric space $(\mathcal{S}, d)$ is utilized, where distance function $d : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}_{\geq 0}$ measures the similarity between two states. DBC significantly outperforms SAC and DeepMDP [61] on CARLA, an open-source simulator for autonomous driving research [2]. While DBC was applied to image data, in principle it could also be applied to observations from ACO domains.

5.3 Exploration

Successful state abstraction has numerous applications, including the scaling of principled exploration strategies to DRL [194] and potential-based reward shaping [28, 27]. However, learning an abstract state space that meets all the desired criteria remains a long-standing problem. RL agents often gradually unlock new abilities, that in turn result in new areas of the environment being visited. Many initial encodings may be learned before an agent has sufficiently explored the state space [136]. In addition, exploration is intractable for domains suffering from the curse of dimensionality. Principled exploration strategies are required that enable the sufficient visitation of abstracted states [28, 136, 221]. To address this, Misra et al. [136] introduce HOMER, a state abstraction approach that accounts for the fact that the learning of a compact representation for states requires comprehensive information from the environment - something that cannot be achieved via random exploration alone.

HOMER is designed to learn a reward-free state abstraction termed *kinematic inseparability*, aggregating observations that share the same forward and backward dynamics. The approach iteratively explores the environment by training policies to visit each kinematically inseparable abstract state. Policies are constructed using contextual bandits and a synthetic reward function that incentifies agents to reach an abstract state. In addition, HOMER interleaves learning the state abstraction and the policies for reaching the new abstract states in an inductive manner, meaning policies reach new states, which are abstracted, and then new policies are learned, iteratively, until a *policy cover* has been obtained. This iterative learning approach is depicted in Figure 4. Once HOMER is trained a near-optimal policy can be found for any reward function. HOMER outperforms PPO and

[2] https://carla.org/

other baselines on an environment named the *diabolical combination lock*, a class of rich observation MDPs where the wrong choice leads to states from which an optimal return is impossible.

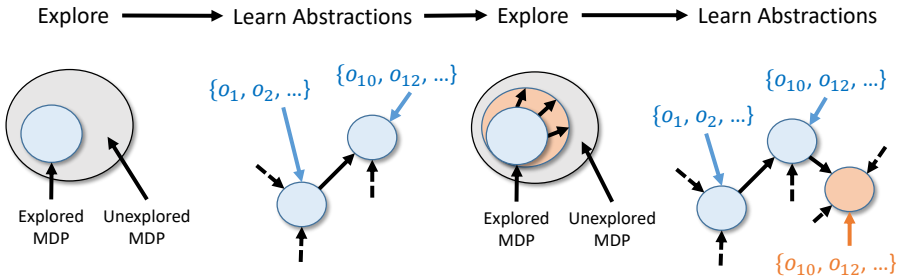


Fig. 4: HOMER (Adapted from [136]).

There are numerous *count-based* exploration approaches with strong convergence guarantees for tabular RL when applied to small discrete Markov decision processes [194]. Ladosz et al. [106] define three desirable criteria for exploration methods, including: i.) determining the degree of exploration based on the agent's learning; ii.) encouraging actions that are likely to result in new outcomes, and; iii.) rewarding the agent for exploring environments with sparse rewards. A popular approach towards encouraging exploration is via intrinsic rewards, where the reward signal consists of extrinsic and intrinsic components. When combined with state-abstraction, these tried and tested methods can be applied to environments suffering from the curse of dimensionality.

Tang et al. [194] introduce a count-based exploration method through static hashing, using SimHash. The hash codes are obtained via a trained AutoEncoder, and provide a means through which to keep track of the number of times semantically similar observation-action pairs have been encountered. A count-based reward encourages the visitation of less frequently explored semantically similar observation-action pairs. Bellemare et al. [19] proposed using Pseudo-Counts, counting salient events derived from the log-probability improvement according to a *sequential density model* over the state space. In the limit this converges to the empirical count. Martin et al. [129] focus on counts within the feature representation space rather than for the raw inputs. Other approaches for computing intrinsic rewards are based on prediction errors [161, 186, 174, 26, 22, 106] and memory based methods [59, 13], using models trained to distinguish states from one another, where easy to distinguish states are considered novel [106].

Goal based exploration represents another class of methods, which Ladosz et al. [106] further divide into: meta-controllers, where a controller with a high-level overview of the environment provides goals for a worker agent [58, 38, 208, 81, 105]; sub-goals, finding a sub-goal for agents to reach, e.g., bottlenecks in the environment [127, 128, 51], and; goals in the region of highest uncertainty, where exploring uncertain states with respect to the rewards are the sub-goals [103].

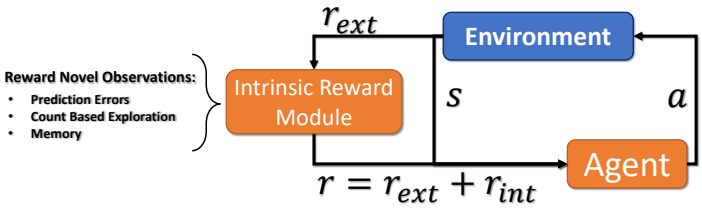


Fig. 5: Approaches rewarding agents for visiting novel states (Adapted from [106]).

Despite the above advances, learning over the entirety of the environment is neither feasible nor desirable when dealing with increased complexity. Here principled methods are required that can determine which parts of the state space are most relevant [162]. However, this requirement in itself leads to the dilemma of how one can determine with minimal effort that an area of the state space is irrelevant, which is an open research question.

5.4 Knowledge Retention

As with many online learning tasks, DRL agents are prone to *catastrophic forgetting*: the unlearning of previously acquired knowledge [12, 175]. In order to be sample efficient, DRL approaches often resort to experience replay memories, which store experience transition tuples that are sampled during training [57]. Samples are either stored long term, as in off-policy approaches such as DQN and DDPG, or short term, e.g., samples gathered using multiple workers for PPO. For the former, paying attention to the experience replay memory composition can mitigate catastrophic forgetting [41]. However, in practice, a large number of transitions are discarded, since there is a memory cost associated with storage.

An additional challenge is that the stationarity of the environment, and one's opponent(s), are a strong assumption. Samples stored inside a replay buffer can become deprecated, confronting learners with the same challenges seen in data streaming [80]. Here DRL agents require continual learning, the ability to continuously learn and build on previously acquired knowledge [220].

One approach to solve this problem is to utilize a *dual memory* where a freshly initialized DRL agent, a short-term agent (network), is trained on a new task, upon which knowledge is transferred to a DQN designed to retain long-term knowledge from previous tasks. A generative network is used to generate short sequences from previous tasks for the DQN to train on, in order to prevent catastrophic forgetting as the new task is learned [12]. However, this approach relies on a stationary environment, as an additional mechanism would be required to determine the relevance of past knowledge, given drift in the state transition probabilities.

Elastic Weight Consolidation (EWC) is another popular approach for mitigating catastrophic forgetting for DNNs [100, 86, 85]. EWC has been applied to DRL using an additional loss term using the Fisher information matrix for the difference between the old and new parameters, and a hyperparameter λ which can be used to specify how important older weights are [169, 147, 95, 220].

6 Approaches for combinatorial action spaces

Due to an explosion in the number of state-action pairs, traditional DRL approaches do not scale to high-dimensional combinatorial action spaces. Scalable methods will need to meet the following criteria.

Generalizability: For our target domains a sufficient visitation of all state-action pairs to obtain accurate value estimates is intractable. Formulations are required that allow for generalization over the action space [46].

Time-Varying Actions (TVA) can result in a policy being trained on a subset of actions $\mathcal{A}' \subset \mathcal{A}$. This has implications when the agent is later asked to choose from a larger set of actions [117] or applying actions to previously unseen objects [33, 50, 117], e.g., a new host on a network for ACO.

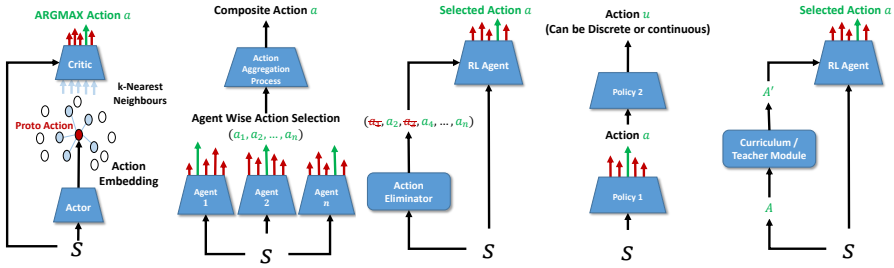
Computational Complexity: An efficient formulation is to use a DNN with $|\mathcal{A}|$ output nodes, requiring a single forward pass to compute an output for each action. However, this approach will not generalize well. Alternatively, for a value function with *a single output*, one could input observation-action pairs and estimate the utility of an arbitrary number of actions: $Q : \mathcal{O} \times \mathcal{A} \rightarrow \mathbb{R}$. However, this approach is intractable due to the computational cost growing linearly with $|\mathcal{A}|$. Instead, methods with sub-linear complexity are required.

The criteria listed above provide the axes along which the suitability of approaches for our target domains can be measured. Through reviewing the literature on high-dimensional action spaces we were able to identify five categories that conveniently cluster the approaches: i.) *proto action* based approaches; ii.) *action decomposition*; iii.) *action elimination*; iv.) *hierarchical* approaches, and; v.) *curriculum learning*. Figure 6 provides an illustrative example and short description for each category. In Table C2 in Appendix C we provide an overview of the literature and a short contributions summary.

6.1 Proto Action Approaches

Dulac-Arnold et al. [46] proposed the first DRL approach to address the high dimensional action space problem, the Wolpertinger architecture, which embeds discrete actions into a continuous space $\mathbb{R}^n$. A continuous control policy $f_\pi : \mathcal{O} \rightarrow \mathbb{R}^n$, is trained to output a *proto* action. Given that a proto action $\hat{a}$ is unlikely to be a valid action ($\hat{a} \notin \mathcal{A}$), k-nearest-neighbours (k-NN) is used to map the proto action to the k closest valid actions: $g_k(\hat{a}) = \text{argmin}^k_{a \in \mathcal{A}} |a - \hat{a}|_2$. To avoid picking outlier actions, and to refine the action selection, the selected actions are passed to a critic Q, which then selects the argmax (see Algorithm 1).

The Wolpertinger architecture meets many of the above requirements. While the time-complexity scales linearly with the number of actions k, the authors show both theoretically and in practice that there is a point at which increasing k delivers a marginal performance increase at best. Using 5-10% of the maximal number of actions was found to be sufficient, allowing agents to generalize over the set of actions with sub-linear complexity. Here, the action embedding space does require a logical ordering of the actions along each axis. Currently prior information about



(a) Proto Actions (b) Action Decomp. (c) Action Elimination (d) Hierarchical (e) Curriculum

Fig. 6: Categories of DRL approaches for high-dimensional action spaces. Figure 6a: Proto actions leverage prior domain knowledge and embed actions into a continuous space, before applying k-NN to pick the closest discrete actions, which are passed to a critic. Figure 6b: Action Decomposition reformulates the single agent problem as a Dec-POMDP with a composite action space. Figure 6c: Action elimination approaches use a module that determines which actions are redundant for a given observation. Figure 6d: Hierarchical, the action selected by a policy influences the action selected by a sub-policy. Figure 6e: Curriculum learning approaches for gradually increasing the number of available actions.

Algorithm 1 Wolpertinger Policy [46]

1: Receive an observation o from the environment.
2: $\hat{a} = f_\pi(o)$ ▷ Obtain proto-action from the actor.
3: $\mathcal{A}_k = g_k(\hat{a})$ ▷ Get k nearest neighbours.
4: $a = \text{argmax}_{a_j \in \mathcal{A}_k} Q(o, a_j)$ ▷ Action refinement step.
5: Apply a to environment and receive r, o'.

the action space is leveraged to construct the embedding space. However, Dulac-Arnold et al. [46] note that learning action representations during training could also provide a solution. The approach has also been criticised for instability during training due to the k-NN component preventing the gradients from propagating back to boost the training of the actor network [199].

Wolpertinger has been applied to: caching on edge devices to reduce data traffic in next generation wireless networks [238], voltage control for shunt compensations to enhance voltage stability [32], maintenance and rehabilitation optimization for multi-lane highway asphalt pavement [226], recommender systems (Slate-MDPs) [191, 236, 237], and, penetration testing for ACO [146]. For the latter, Nguyen et al. [146] evaluate the ability of Wolpertinger to learn a policy that launches attacks on vulnerable services on a network. Wolpertinger is applied using an embedding space that consists of three levels (illustrated in Figure 7): i.) the action characteristics (*scan subnet*, *scan host* and *exploit services*); ii.) the subnet to target, and; iii.) services that are vulnerable towards attacks. The second dimension focuses on the destination of the action with respect to the subnet,

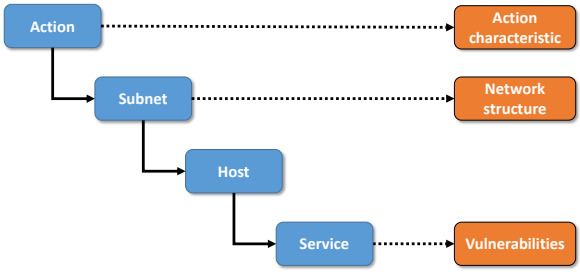


Fig. 7: Wolpertinger attack embedding used by Nguyen et al. [146] on NASim [179].

e.g., selecting "scan subnet" along axis 1 and selecting the subnet on axis 2. *Node2Vec* is used for expressing the network structure, and the authors also train a network to produce similar embeddings for correlated service vulnerabilities. Wolpertinger was shown to outperform DQN on the Network Attack Simulator environment [179].

6.2 Action Decomposition Approaches

A popular approach towards scaling DRL to large combinatorial action spaces is to apply multi-agent deep reinforcement learning (MADRL). The action space is decomposed into actions provided by multiple agents, e.g., having each agent control an action dimension [195], or via an algebraic formulation for combining the actions [199]. However, learning an optimal policy requires the underlying agents to converge upon an *optimal joint-policy*. Therefore, approaches must be viewed through the lens of MADRL within a Dec-POMDP, using equilibrium concepts from multi-agent learning.

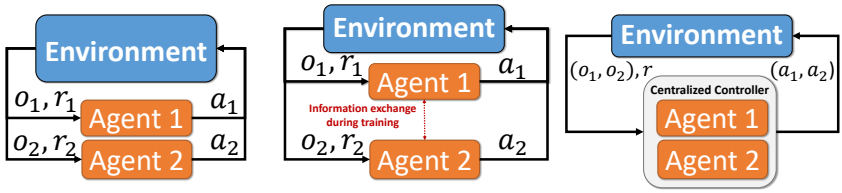
Formally, given an expected gain, $\mathcal{G}_i(\boldsymbol{\pi}) = \mathbb{E}_{\boldsymbol{\pi}}\{\sum_{k=0}^{\infty} \gamma^k r_{i,t+k+1} | x_t = x\}$, the underlying policies must find a Pareto optimal solution, i.e., a joint policy $\hat{\boldsymbol{\pi}}$ from which no agent i can deviate without making at least one other agent worse off [132]:

Definition 6.1 (Pareto Optimality)**.** *A joint-strategy $\boldsymbol{\pi}$ is Pareto-dominated by $\hat{\boldsymbol{\pi}}$* if and only if *(*iff*):*

$$\forall i, \forall s \in \mathcal{S}, \mathcal{G}_{i,\hat{\boldsymbol{\pi}}}(s) \geq \mathcal{G}_{i,\boldsymbol{\pi}}(s) \; and \; \exists j, \exists s \in \mathcal{S}, \mathcal{G}_{j,\hat{\boldsymbol{\pi}}}(s) > \mathcal{G}_{j,\boldsymbol{\pi}}(s). \tag{3}$$

A joint policy $\hat{\boldsymbol{\pi}}^*$ is Pareto optimal if it is not Pareto-dominated by any other $\boldsymbol{\pi}$.

There are three categories of training schemes for cooperative MA(D)RL (illustrated in Figure 8): independent learners (ILs), who treat each other as part of the environment; the centralized controller approach, which does not scale with the number of agents; and centralized training for decentralized execution (CTDE).



(a) Independent Learners (b) Centralized Training for Decentralized Execution (c) Centralized Controller

Fig. 8: An overview of multi-agent reinforcement learning training schemes.

Even within stateless two player matrix games with a small number of actions per agents, ILs fail to consistently converge upon Pareto optimal solutions [150, 37, 132, 92, 131, 29, 159, 160]. However, ILs are frequently used as a baseline for action decomposition approaches [195]. Therefore, to better understand the challenges that confront action decomposition approaches we shall first briefly consider the multi-agent learning pathologies that learners must overcome to converge upon a Pareto optimal joint-policy from the perspective of ILs [3]:

Miscoordination occurs when there are two or more incompatible Pareto-optimal equilibria [37, 92, 132]. One agent choosing an action from an incompatible equilibria is sufficient to lower the gain. Formally: two equilibria $\boldsymbol{\pi}$ and $\hat{\boldsymbol{\pi}}$ are incompatible *iff* the gain received for pairing at least one agent using a policy π with other agents using a policy $\hat{\pi}$ results in a lower gain compared to when all agents are using π: $\exists i, \boldsymbol{\pi}_i \neq \hat{\boldsymbol{\pi}}_i, \mathcal{G}_{i,\langle \hat{\boldsymbol{\pi}}_i, \boldsymbol{\pi}_{-i} \rangle} < \mathcal{G}_{i,\boldsymbol{\pi}}$.

Relative Overgeneralization: ILs are prone to being drawn to sub-optimal but wide peaks in the reward space, as there is a greater likelihood of achieving collaboration there [158]. Within these areas a sub-optimal policy yields a higher payoff on average when each selected action is paired with an arbitrary action chosen by the other agent [158, 219, 155].

Stochasticity of Rewards and Transitions: Rewards and transitions can be stochastic, which has implications for approaches that use optimistic learning to overcome the relative overgeneralization pathology [155, 157, 156].

The Alter-Exploration Problem: In MA(D)RL increasing the number of agents also increases *global exploration*, the probability of at least one of n agents exploring: $1 - (1 - \epsilon)^n$. Here, each agent explores according to a probability ϵ [132].

The Moving Target Problem is a result of agents updating their policies in parallel [23, 192, 202, 201]. This pathology is amplified when using experience replay memories $\mathcal{D}_i$, due to transitions becoming deprecated [57, 153, 156].

Deception: Deception occurs when utility values are calculated using rewards backed up from follow-on states from which pathologies such as miscoordination and relative overgeneralization can also be back-propagated [217]. States with

[3] For a detailed recap please read [217, 110, 92, 155].

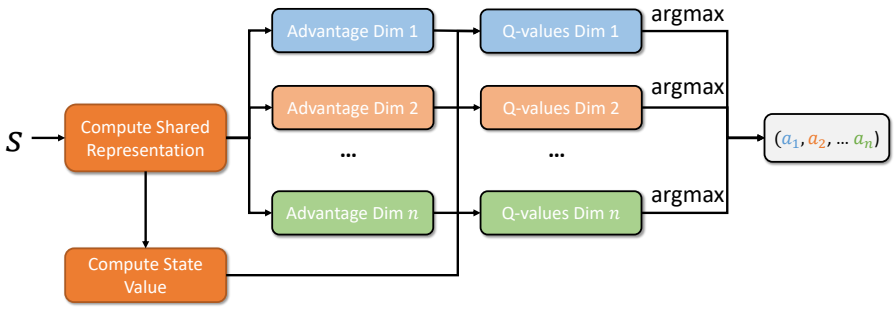


Fig. 9: An illustration Branching Dueling Q-Networks (adapted from [195]).

high local rewards can also represent a problem, drawing ILs away from optimal state-transition trajectories [217].

Non-trivial approaches are required in order to consistently converge upon a Pareto optimal solution [217, 110, 92, 155]. We shall now consider the different types of action decomposition approaches that can be found in the literature.

6.2.1 Branching Dueling Q-Network

Designed for environments where the action-space can be split into smaller action-spaces Branching Dueling Q-Network (BDQ) [195] is a branching version of Dueling DDQN [214] [4]. Each branch of the network is responsible for proposing a discrete action for an actuated joint. The approach features a *shared decision module*, allowing the agents to learn a common latent representation that is subsequently fed into each of the n DNN branches, and can therefore be considered a CTDE approach. A conceptional illustration of the approach can be found in Figure 9.

BDQ has a linear increase of network outputs with regard to number of degrees of freedom, thereby allowing a level of independence for each individual action dimension. It does not suffer from the combinatorial growth of standard vanilla discrete action algorithms. However, the approach is designed for discretized continuous control domains. Therefore, BDQ's scalability to the MultiDiscrete action spaces from ACO requires further investigation.

While BDQ achieves sub-linear complexity, the formulation is vulnerable towards the MA(D)RL pathologies outlined above. To evaluate the benefit of the shared decision module, BDQ is evaluated against Dueling-DQN, DDPG, and *independent Dueling DDQNs* (IDQ). The only mentioned distinction between BDQ and IDQ is that the first two layers were not shared among IDQ agents [195]. BDQ uses a modified Prioritized Experience Replay memory [176], where transitions are prioritized based on the *aggregated distributed TD error*. In essence, a prioritized version of Concurrent Experience Replay Trajectories (CERTS) are being utilized,

[4] Dueling DDQNs consist of two separate estimators for the value and state-dependent action advantage function.

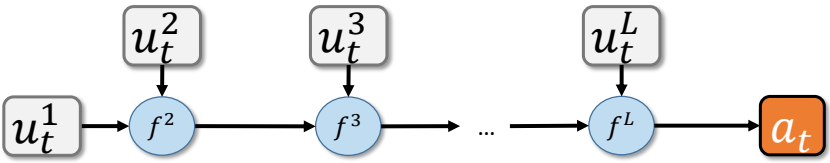


Fig. 10: Action space composition. Action u_t at time step t is algebraically constructed using actions a_t^i obtained from action subsets $\mathcal{A}^i$ (adapted from [199]).

a method from the MADRL literature that has previously been shown to facilitate coordination [153].

BDQ has been evaluated on numerous discretized MuJoCo domains [195]. The evaluation focused on two axes: granularity and degrees of freedom. BDQ's benefits over Dueling-DQNs become noticeable as the number of degrees of freedom are increased. In addition, BDQ was able to solve granular, high degree of freedom domains for which Dueling-DDQNs was not applicable. On the majority of the domains DDPG still outperformed BDQ, with the exception of Humanoid-v1. Unfortunately a comparison of BDQ against the Wolpertinger architecture was not provided. For ACO we note that BDQ will probably not scale well if one of the branches is very large, e.g., has lot of nodes.

6.2.2 Cascading Reinforcement Learning Agents

For *cascading reinforcement learning agents* (CRLA) the action space $\mathcal{A}$ is decomposed into smaller sets of actions $\mathcal{U}^1, \mathcal{U}^2, ..., \mathcal{U}^L$ [199]. For each subset of actions $\mathcal{A}^i$, the size of the dimensionality is significantly smaller than that of $\mathcal{A}$, i.e., $|\mathcal{U}^i| \ll |\mathcal{A}|$ $(\forall i \in [1, L])$. In this formulation a primitive action a_t at time step t is given by a function over actions u_t^i obtained from each respective subset $\mathcal{U}^i$: $u_t = f(u_t^1, u_t^2, ..., u_t^L)$. The action components u_t^i are chained together to algebraically build an integer identifier. This provides a formulation through which larger identifiers can be obtained using the smaller integer values provided by each action subset. A CTDE approach is used to facilitate the training of n agents, where the joint action space $\boldsymbol{\mathcal{A}}$ is comprised of $\mathcal{U}^1, \mathcal{U}^2, ..., \mathcal{U}^n$, with $\mathcal{U}^i$ representing the action space of an agent i. The concept is illustrated in Figure 10.

CRLA yields a solution that allows for large combinatorial action spaces to be decomposed into a branching tree structure. Each node in the tree represents a decision by an agent regarding which child node to select next. Each node has its own identifier, and all nodes in the tree have the same branching factor, with a heuristic being used to determine the number of tree levels: $|T| = \log_b(|\mathcal{A}|)$. Instead of having an agent for each node, the authors propose to have a linear algebraic function that shifts the action component identifier values into their appropriate range: $u_{out}^{i+1} = f(u_{out}^i) = u_{out}^i \times \beta^{i+1} + u^{i+1}$, with β^{i+1} being the number of nodes at level $i+1$. More generally, given two actions u^i and u^{i+1} obtained from agents at levels i and $i+1$, where for both actions we have identifiers in the

range $[0, ..., |\mathcal{U}^i| - 1]$, and $[0, ..., |\mathcal{U}^{i+1}| - 1]$ respectively, then the action component identifier at level $i + 1$ is $u^i \times |\mathcal{U}^{i+1}| + u^{i+1}$. The executive primitive action meanwhile will be computed via: $a = u^{L-1} \times |\mathcal{U}^L| + u^L$. An illustration of this tree structure is provided in Figure 11.

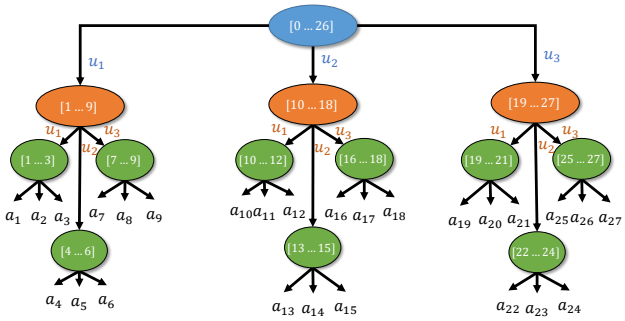


Fig. 11: An illustration of the action selection process used by CRLA (adapted from [199]). The leaf nodes represent the primitive actions from $\mathcal{A}$, while each internal node contains the action range for its children. Through using an algebraic formulation that makes use of an offset, only three agents with an action space $|\mathcal{U}^i| = 3$ are needed to capture $\mathcal{A}$.

Cooperation among agents is facilitated via QMIX [166], which uses a non-linear combination of the value estimates to compute the joint-action-value during training. The weights of the mixing network are produced using a hypernetwork [71], conditioned on the state of the environment. In CRLA agents share a replay buffer. Therefore, as with BDQ, a synchronised sampling equivalent to CERTS [153] is being used. The authors [199] recommend limiting the size of the action sets to 10 – 15 actions, and to choose an L that allows the approach to reconstruct the intended action set $\mathcal{U}$. CRLA-QMIX was evaluated on two environments against a version of CRLA using independent learners, and a single DDQN:

- A toy-maze scenario with a discretized action space, representing the directions in which the agent can move. The agent received a small negative reward for each step, and positive one upon completing the maze. An action size of 4096 was selected, with $n = 12$ actuators.
- A partially observable CybORG capture the flag scenario from Red's perspective. Upon finding a flag a large positive reward was received. A smaller reward was obtained for successfully hacking a host.

CRLA significantly outperformed DDQN on both the maze task and CybORG scenarios with more than 50 hosts. CRLA-QMIX had less variance and better stability than CRLA with ILs. However, in an evaluation scenario with 60 hosts the converged policies show similar rewards and steps per episode, potentially explained

by the fact that CRLA-ILs also makes use of CERTs. The authors note that hyperparameter tuning for the QMIX hypernetwork was time consuming.

6.2.3 Discrete Sequential Prediction of Continuous Actions

Metz et al. [135] propose a Sequential DQN (SDQN) for discretized continuous control. The original (*upper*) MDP with N (actuators) times D (dimensions) actions is transformed into a *lower* MDP with $1 \times D$ actions. The lower MDP consists of the compositional action component, where N actions are selected sequentially. Unlike BDQ actuators take turns selecting actions, and can observe the actions that have been selected by others (see Figure 12). Also, the action composition was obtained using a single DNN that learns to generalize across actuators. An LSTM [82] was used to keep track of the selected actions. For stability, SDQN learns Q-values for both the upper and lower MDPs at the same time, performing a Bellman backup from the lower to the upper MDP for transitions where the Q-value should be equal. In addition, a zero discount is used for all steps except where the state of the upper MDP changes. Another requirement is the pre-specified ordering of actions. The authors hypothesise that this may negatively impact training on problems with a large number of actuators.

The authors evaluate SDQN against DDPG on a number of continuous control tasks from the OpenAI gym [24], including Hopper ($N = 3$), Swimmer ($N = 2$), Half-Cheetah ($N = 6$), Walker2d ($N = 6$), and Humanoid($N = 17$). SDQN outperformed DDPG on all domains except Walker2d. With respect to granularity SDQN required $D \geq 4$. The authors also evaluated 8 different action orderings at 3 points during training on Half Cheetah. All orderings achieved a similar performance.

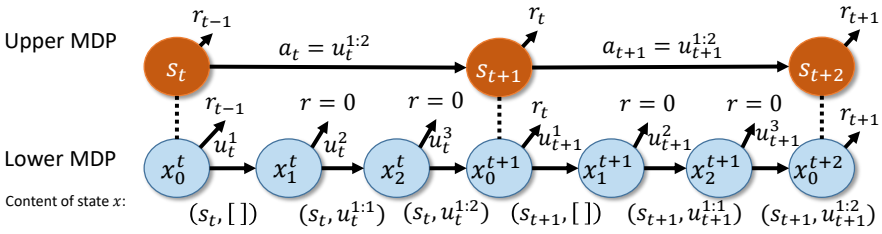


Fig. 12: Sequential DQN Architecture (adapted from [135]).

6.2.4 Time-Varying Composite Action Spaces

Li et al. [117] propose the *Structured Cooperation Reinforcement Learning (SCORE)* algorithm that accounts for dynamic time-varying action spaces, i.e., environments where a sub-set of actions become temporarily invalid [117]. SCORE can be applied to heterogeneous action spaces that contain continuous and discrete values. A series of DNNs model the composite action space. A centralized critic and decentralized actor (CCDA) approach [116] facilitates cooperation among the agents. In addition a Hierarchical Variational Autoencoder (HVAE) [47] maps the

sub-action spaces of each agent to a common latent space. This is then fed to the critic, allowing the critic to model correlations between sub-actions, enabling the explicit modelling of dependencies between the agents' action spaces. A graph attention network (GAT) [206] is used as the critic, in order to handle the varying numbers of agents (nodes). The HVAE and GAT are critical for SCORE to cope with varying numbers of heterogeneous actors. As a result, SCORE is a two stage framework, that must first learn an action space representation, before learning a robust and transferable policy. For the first phase a sufficient number of trajectories must be gathered for each sub-action space. The authors use a random policy to generate these transitions. Once the common latent action representation is acquired the training can switch to focusing on obtaining robust policies.

SCORE is evaluated on a proof-of-concept task – a Spider environment based on the MuJoCo Ant environment – and a Precision Agriculture Task, where the benefits of a mixed discrete-continuous action space comes into play. SCORE outperforms numerous baselines on both environments, including MADDPG [126], PPO [178], SAC [72], H-PPO [49], QMIX [166] and MAAC [89]. However, the code for the environments and SCORE are not made publicly available.

6.2.5 Action Decomposition Approaches for Slate-MDPs

Action decomposition approaches have also been applied to Slate-MDPs. Two noteworthy efforts are *Cascading Q-Networks* and *Slate Decomposition*.

Cascading Q-Networks (CDQNs): Chen et al. [36] introduce a model-based RL approach for the recommender problem that utilizes Generative Adversarial Networks (GANs)[66] to imitate the user's behaviour dynamics and reward function. The motivation for using GANs is to address the issue that a user's interests can evolve over time, and the fact that the recommender system can have a significant impact on this evolution process. In contrast, most other works in this area use a manually designed reward function. A CDQN is used to address the large action space, through which a combinatorial recommendation policy is obtained. CDQNs consist of k related Q-functions, where actions are passed on in a cascading fashion.

CDQNs were evaluated on six real-world recommendation datasets – *MovieLens*, *LastFM*, *Yelp*, *Taobao*, *YooChoose*, and *Ant Financial* – against a range of non-RL recommender approaches, including IKNN, S-RNN, SCKNNC, XGBOOST, DFM, W&D-LR, W&D-CCF, and a Vanilla DQN. On the majority of these datasets, the generative adversarial model is a better fit to user behaviour with respect to held-out likelihood and click prediction. With respect to the resulting model policies, better cumulative and long-term rewards were obtained. The approach took less time to adjust compared to approaches that did not make use of the GANs synthesized user. However, we caution that applying model-based RL approaches to complex asymmetrical adversarial games, such as ACO, requires further considerations.

Slate Decomposition: Slate decomposition, or *SlateQ*, is an approach where the Q-value estimate for a slate $\boldsymbol{a}$ can be decomposed into the item-wise Q-values of its constituent items u_i [87]. Having a decomposition approach that can learn $\bar{Q}(s, a_i)$ for an item i mitigates the generalization and exploration challenges listed

above. However, the ability to successfully factor the Q-value of a slate $\boldsymbol{a}$ relies on two assumptions: *Single Choice* (SC) and *Reward/Transition Dependence on Selection* (RTDS). The authors show theoretically that given the standard assumptions with respect to learning and exploration [193], as well as SC and RTDS, SlateQ will converge to the true slate Q-function $Q_\pi(x, \boldsymbol{a})$. SlateQ is evaluated in a simulation, while validity and scalability were tested in live experiments on YouTube.

6.3 Action Elimination Approaches

Learning with a large combinatorial action space is often challenging due to a large number of actions being either redundant or irrelevant within a given state [231]. RL agents lack the ability to determine a sub-set of relevant actions. However, there have been efforts towards the state dependent elimination of actions. Zahavy et al. [231] combine a DQN with an action-elimination network (AEN), which is trained via an elimination signal e, resulting in AE-DQN (Figure 13). After executing an action a_t, the environment will return a binary action elimination signal in $e(s_t, a_t)$ in addition to the new state and reward signal. The elimination signal e is determined using domain-specific knowledge. A linear contextual bandit model is applied to the outputs of the AEN, that is tasked with eliminating irrelevant actions with a high probability, balancing out exploration/exploitation. Concurrent learning introduces the challenge that the learning process of both the DQN and AEN affect the state-action distribution of the other. However, the authors provide theoretical guarantees on the convergence of the approach using linear contextual bandits. While the AE-DQN was designed for text based games the authors note that the approach is applicable to any environment where an elimination signal can be obtained via a rule-based system.

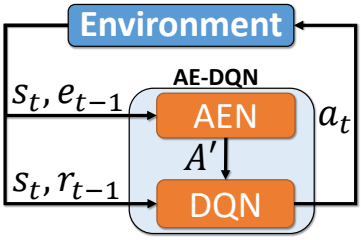


Fig. 13: Action Elimination DQN (adapted from [231]). The agent selects an action a_t, and observes a reward r_{t-1}, the next observation o_t and an elimination signal e_{t-1}. The agent uses this information to learn two function approximation deep networks: a DQN and an AEN. The AEN provides an admissible actions set $\mathcal{A}' \subseteq \mathcal{A}$ to the DQN, from which the DQN can pick the next action a_{t+1}.

AE-DQN is evaluated on both Zork and a K-Room Gridworld environment. For the K-rooms Gridworld environment a significant gain for the use of action elimination was observed as the number of categories K was increased. Similar benefits were observed in the Zork environment, e.g., AE-DQN using 215 actions

being able to match the performance of a DQN trained with a reduced action space of 35 actions, while significantly outperforming a DQN with 215 actions. However, questions remain regarding the extent to which the approach is applicable to *very* high dimensional action spaces, where additional considerations may be required as to how the AEN can generalize over actions.

Action elimination has also been applied to Slate-MDPs. Chen et al. [34] adapted the REINFORCE algorithm into a top-k neural candidate generator for large action spaces. The approach relies on data obtained through previous recommendation policies (behaviour policies β), which are utilized as a means to correct data biases via an importance sampling weight while training a new policy. Importance sampling is used due to the model being trained without access to a real-time environment. Instead the policy is trained on logged feedback of actions chosen by a historical mixture of policies, which will have a different distribution compared to the one that is being updated. A recurrent neural network is used to keep track of the evolving user interest.

With respect to sampling actions, instead of choosing the k items that have the highest probability, the authors use a stochastic policy via Boltzmann exploration. However, computing the probabilities for all N actions is computationally inefficient. Instead the authors chose the top M items, select their logits, and then apply the softmax over this smaller set M to normalize the probabilities and sample from this smaller distribution. The authors note that when $M \ll K$, one can still retrieve a reasonably sized probability mass, while limiting the risk of bad recommendations. Exploration and exploitation are balanced through returning the top K' most probable items (with $K' < k$), and sample $K - K'$ items from the remaining $M - K'$ items. The approach is evaluated in a production RNN candidate generation model in use at YouTube, and experiments are performed to validate the various design decisions.

6.4 Hierarchical Reinforcement Learning

The RL literature features a number of hierarchical formulations. Feudal RL features Q-learning with a managerial hierarchy, where "managers" learn to set tasks for "sub-managers" until agents taking atomic actions at the lowest levels are reached [40, 208]. There have also been factored hierarchical approaches that decompose the value function of an MDP into smaller constituent MDPs [43, 69].

Wei et al. [218] propose Parameterized Actions Trust Region Policy Optimization (TRPO) [177] and Parameterized Actions SVG(0), hierarchical RL approaches designed for Parameterized Action MDPs. The approaches consist of two policies implemented by neural networks. The first network is used by the discrete action policy $\pi_\theta(a|s)$ to obtain an action a in state s. The second network is for the parameter policy. It takes both the state and discrete action as inputs, and returns a continuous parameter (or a *set* of continuous parameters) $\pi_\vartheta(u|s,a)$. Therefore, the joint action probability for (a,u) given a state s is conditioned on both policies: $\pi(a,u|s) = \pi_\theta(a|s)\pi_\vartheta(u|s,a)$. This formulation has the advantage that since the action a is known before generating the parameters, there is no need to determine which action tuple (a,u) has the highest Q-value for a state s. In order to optimize

the above policies, methods are required that can back-propagate all the way back through the discrete action policy. Here the authors introduce a modified version of TRPO [177] that accounts for the above policy formulation, and a parameterized action stochastic value gradient approach that uses the Gumbel-Softmax trick for drawing an action u and back-propagating through $[\theta, \vartheta]$. The approach is depicted in Figure 14. With respect to evaluation, PATRPO outperforms PASVG(0) and PADDPG within a Platform Jumping environment. PATRPO also outperforms PADDPG within the Half Field Offense soccer [16] environment with no goal keeper [74].

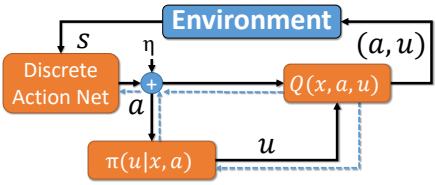


Fig. 14: PASVG(0) (adapted from [218]).

6.5 Curriculum Learning

Curriculum learning (CL) approaches attempt to accelerate the learning process through initially subjecting the RL agent to a simplified version of the problem, and subsequently gradually increasing the task complexity, e.g., via a progression function [17, 18]. This approach has also been applied to RL for combinatorial action spaces. Farquhar et al. [54] introduce a CL approach using a growing action space (GAS). For an MDP with unrestricted action space $\mathcal{A}$ the authors define a set of N action spaces $\mathcal{A}_l, l \in \{0, ..., N-1\}$. Each action space is a subset of the next level l: $\mathcal{A}_0 \subset \mathcal{A}_1 \subset ... \subset \mathcal{A}_{N-1} \subset \mathcal{A}$. A policy restricted to an action space $\mathcal{A}_l$ is denoted as $\pi_l(a, s)$. The optimal policy for this restricted policy class is $\pi_l^*(u, x)$, and the corresponding action-value and value functions are: $Q_l^*(s, a)$ and $V_l^*(s) = \max_a Q_l^*(s, a)$. Domain knowledge is used to define a hierarchy of actions. For every action $a \in \mathcal{A}_l$ where $l > 0$ there is a parent action $\texttt{parent}_l(a)$ in the space of $\mathcal{A}_{l-1}$. Given that at each level, subsets of action spaces are subsets of larger action spaces, the actions available in $\mathcal{A}_{l-1}$ are their own parents in $\mathcal{A}_l$. The authors note that in many environments Euclidean distances are a valid measure for implementing a heuristic for defining a hierarchy over actions.

An ablation study on discretized Acrobat and Mountaincart environments shows the value of efficiently using the data collected during training across levels. The authors also evaluate their approach on SMAC, using a far larger number of units compared to those usually used in MARL experiments – i.e., scenarios with 50-100 instead of 20-30. In addition, the task difficulty is increased through having randomized starting positions, and scripted opponent logic that holds its position until any agent-controlled unit is in range. Subsequently the enemy focus-fires on its closest enemy. Having to locate the enemy first increases the exploration challenge. The authors demonstrate the advantage of their approach GAS(2) (GAS with

2 levels) against various ablations of their approach and methods that directly train on the full action space.

Yu and Huang [229] take a similar approach to GAS [54] for decision making in intelligent healthcare. A Progressive Action Space (PAS) approach allows the learner to master easier tasks first, via generalized actions, before gradually increasing the granularity, e.g., modifying the precise volume of a drug to be given to a patient. This work focuses on an offline RL setting, where traditional approaches lead to inaccurate state-action evaluation for those actions seldom applied by the physicians in the historical clinical dataset – a common problem for offline RL [60]. Similar to GAS [54], PAS also requires domain knowledge for defining N abstracted action spaces $\{a_1, a_2, ..., a_N\}$ and a corresponding number of curricula $\{M_1, M_2, ..., M_N\}$. Value and policy transfer approaches are used to transfer knowledge across levels.

7 Adversarial Learning

A number of approaches discussed in the previous sections have been applied to adversarial domains, including ACO (e.g., Wolpertinger [146] and CRLA [199]). However, these approaches were trained against stationary opponents. In practice, the ACO problem is non-stationary, with Red and Blue adjusting their approach over time. In adversarial learning terminology, Red and Blue will attempt to compute *approximate best responses* (ABRs) to each others' policies [152].

7.1 The Adversarial Learning Challenge

In this section we shall formally define the adversarial learning problem, desirable solution concepts, and the approaches designed to help learning agents converge upon policies that are hard to exploit. ACO environments are adversarial. However, they are not always *zero-sum games*. For instance, in CAGE Challenge 2 Blue receives a penalty for restoring a compromised host due to the action having consequences for any of the Green agents currently working on this facility. Here, Red does not receive a corresponding reward $r_{Red} = -r_{Blue}$. Therefore, ACO lacks a key property from two player zero-sum games, where the gain of one player is equal to the loss of the other player. However, in this section we shall treat the ACO problem as a quasi zero-sum game, with the assumption that the consequences of Red winning outweigh other factors, shall assume: $\mathcal{G}_1(\pi, \mu) \approx -\mathcal{G}_2(\pi, \mu)$. An equilibrium concept commonly used in this class of games to define solutions is the Nash equilibrium [145]:

Definition 7.1 (Nash Equilibrium)**.** *A joint policy $\boldsymbol{\pi}^*$ is a Nash equilibrium* iff *no player i can improve their gain through unilaterally deviating from $\boldsymbol{\pi}^*$:*

$$\forall i, \forall \pi_i \in \Delta(\mathcal{S}, \mathcal{A}_i), \forall s \in \mathcal{S}, \mathcal{G}_i(\langle \pi_i^*, \boldsymbol{\pi}^*_{-i} \rangle) \geq \mathcal{G}_i(\langle \pi_i, \boldsymbol{\pi}^*_{-i} \rangle). \tag{4}$$

Our focus is on finite two-player (quasi) zero-sum games, where an equilibrium is referred to as a saddle point, representing the value of the game $v*$. Given two policies π_1, π_2, the equilibria of a finite zero-sum game is:

Theorem 1 (Minmax Theorem)**.** *In a finite zero-sum game:*

$$max_{\pi_1} min_{\pi_2} \mathcal{G}_i(\langle \pi_1, \pi_2 \rangle) = min_{\pi_2} max_{\pi_1} \mathcal{G}_i(\langle \pi_1, \pi_2 \rangle) = v^*. \tag{5}$$

Above is one of the fundamental theorems of game theory, which states that every finite, zero-sum, two-player game has optimal mixed strategies [203]. We shall discuss the implications of the above equations for ACO from the perspective of Blue. Given a joint-policy $\langle \pi^*_{Blue}, \pi^*_{Red} \rangle$ where π^*_{Blue} and π^*_{Red} represent optimal mixed strategies, then by definition Red will be unable to learn a new best response π_{Red} that improves on π^*_{Red}. Obtaining π^*_{Blue} guarantees that Blue will perform well, even against a worst case opponent [163]. This means that, even if the value of the game v^* is in Red's favour, assuming that Blue has found π^*_{Blue}, then Blue has found a policy that limits the extent to which that Red can *exploit* Blue.

One of the long-term objectives of MARL is to limit the exploitability of agents deployed in competitive environments [108, 152, 76]. While a number of methods for limiting exploitability exist that are underpinned by theoretical guarantees, in practice finding the value of the game is challenging even for simple games. This is due to DRL using function approximators that are unable to compute *exact* best responses to an opponent's policy. The best a DRL agent can achieve is an ABR. In addition, for complex games finding a Nash equilibrium is intractable. Here the concept of an approximate Nash equilibrium (ϵ-NE) is helpful [152, 108]:

Definition 7.2 (ϵ-Nash Equilibrium)**.** *The joint-policy $\boldsymbol{\pi}^*$ is an ϵ-NE* iff*:*

$$\forall i, \forall \pi_i \in \Delta(\mathcal{S}, \mathcal{A}_i), \forall s \in \mathcal{S}, \mathcal{G}_i(\langle \pi^*_i, \boldsymbol{\pi}^*_{-i} \rangle) \geq \mathcal{G}_i(\langle \pi_i, \boldsymbol{\pi}^*_{-i} \rangle) - \epsilon. \tag{6}$$

7.2 Approaches Towards Limiting Exploitability

The above raises the question: how can we *efficiently* limit the exploitability of ACO agents? A key insight here from the adversarial learning literature is that finding π^*_{Blue} will require learning to best respond to principled Red agents, ideally through computing a best response against π^*_{Red}. However, similarly Red will need to face strong Blue agents to find π^*_{Red}. This raises the need for an iterative adversarial learning process where Blue and Red learn (A)BRs to each others' latest policies. However, naïve independent learning approaches fail to generalize well due to a tendency to overfit on their opponents, also known as joint-policy correlation [108]. Therefore, a more principled approach is required.

Gleave et al. [65] show that an adversary can learn simple attack policies that reliably win against a static opponent implemented with function approximators. Often random and uncoordinated behavior is sufficient to trigger sub-optimal actions [5]. By adversarial attacks the study refers to attacks on the opponents

[5] The authors provide videos of the attack behaviours: https://adversarialpolicies.github.io/

observation space. This is quasi equivalent to adding perturbations to images for causing a misclassification in supervised learning, but doing so via taking actions within the environment. A worrying finding is that DNNs are more vulnerable towards high-dimensional adversarial samples [64, 96, 180]. The same applies for DRL. Empirical evaluations on MuJoCo show that the greater the dimensionality of the area of the observation space that Red can impact, the more vulnerable the victim is towards attacks [65].

The work by Gleave et al. [65] shows that agents trained via simplistic training schemes – e.g., self-play – are very far from an ϵ bounded Nash equilibrium. This raises the need for training schemes designed to limit the exploitability of agents. Perolat et al. [163] identify three categories of approaches for reducing the exploitability of agents: regret minimization, regret policy gradient methods, and best response techniques.

The first category includes approaches that scale counterfactual regret minimization (CFR) using deep learning. Deep-CFR [25] trains a regret DNN via an experience replay buffer containing counterfactual values. A limitation of this approach is the sampling method, in that it does not scale to games with a large branching factor [163]. There are model-free regret-based approaches which use DNNs that scale to larger games, such as *deep regret minimization with advantage baselines and model-free learning* (DREAM) [189] and the advantage regret-matching actor-critic (ARMAC) [68]. However, these approaches rely on an importance sampling term in order to remain unbiased. The importance weights can become very large in games with a long horizon [163]. To generalize, these techniques require the generation of an average strategy, necessitating either the complete retention of all strategies from previous iterations, or an error prone approximation, e.g., a DNN trained via supervised learning [163].

The second category of methods approximates CFR via a weighted policy gradient [185, 163]. However, the approach is not guaranteed to converge to a Nash equilibrium [163]. In contrast Neural Replicator Dynamics (NeuRD) [77], an approach that approximates the Replicator Dynamics from evolutionary game theory with a policy gradient, is proven to converge to a Nash equilibrium. NeuRD has been applied to large-scale domains, as part of *DeepNash* [163]. It is used to modify the loss function for optimizing the Q-function and the policy [163]. DeepNash was recently introduced as a means of tackling the game of Stratego. However, its wider applicability is yet to be explored. In the remainder of this section we shall therefore focus on the third category of approaches: best response techniques.

7.3 Best Response Techniques

In recent years a number of principled approaches from the MARL literature – originally designed for competitive games with a low-dimensional state space – have been scaled to *deep* MARL. Population-based and game-theoretic training regimes have shown a significant amount of potential [118]. Early work in this area by Heinrich and Silver [76] scaled *Fictitious Self-Play* for domains suffering from the curse-of-dimensionality, resulting in *Neural Fictitious Self-Play* (NFSP).

This approach approximates extensive-form fictitious play by progressively training a best response against the average of all past policies using off-policy DRL. A DNN is trained using supervised learning to imitate the average of the past best responses.

Lanctot et al. [108] introduced Policy-Space Response Oracles (PSRO), a generalization of the Double-Oracle (DO) approach originally proposed by McMahan et al. [133], a theoretically sound approach for finding a minimax equilibrium. Given the amount of interest that this approach has generated within the literature [21, 163, 212, 108, 118, 152], we shall dedicate the remainder of this section to DO based approaches. First we will discuss the DO approach's theoretical underpinnings, before providing an overview of the work that has been conducted in this area in recent years. We shall conclude the section with open challenges, in particular with respect to scaling this approach to ACO.

The DO algorithm defines a two-player zero-sum normal-form game $\mathcal{N}$, where actions correspond to policies available to the players within an underlying stochastic game $\mathcal{M}$. Payoff entries within $\mathcal{N}$ are determined through computing the gain $\mathcal{G}$ for each policy pair within $\mathcal{M}$:

$$\mathcal{R}_i^{\mathcal{N}}(\langle a_1^r, a_2^c \rangle) = \mathcal{G}_i^{\mathcal{M}}(\langle \pi_1^r, \pi_2^c \rangle). \tag{7}$$

In Equation 7, r and c refer to the respective rows and columns inside the normal-form (bimatrix) game, and $\mathcal{M}$ is the game where the policies are being evaluated. The normal-form game $\mathcal{N}$ is subjected to a game-theoretic analysis, to find an optimal mixture over actions for each player. These mixtures represent a probability distribution over policies for the game $\mathcal{M}$. The DO algorithm assumes that both players have access to a *best response oracle*, returning a *best response* (BR) policy against the mixture played by the opponent. BRs are subsequently added to the list of available policies for each agent. As a result each player has an additional action that it can choose in the normal-form game $\mathcal{N}$. Therefore, $\mathcal{N}$ needs to be augmented through computing payoffs for the new row and column entries. Upon augmenting $\mathcal{N}$ another game theoretic analysis is conducted, and the steps described above are repeated. If no further BRs can be found, then the DO algorithm has converged upon a minimax equilibrium [133].

When applying the DO algorithm to MARL the oracles must compute Approximate Best Responses (ABRs) against a *mixture of policies*. There are a number of approaches for implementing a mixture of policies, e.g., sampling individual policies according to their respective mixture probabilities at the beginning of an episode [108]. Alternatively, the set of policies can be combined into a weighted-ensemble, where the outputs from each policy are weighted by their respective mixture probabilities prior to aggregation. For generality, we define a mixture of policies as follows:

Definition 7.3 (Mixture of Policies)**.** *π_i^{μ} is a mixture of policies for a mixture μ_i and a corresponding set of policies Π_i for agent i.*

Using an oracle to obtain an *exact* best response is often also intractable. In games that suffer from the curse-of-dimensionality an oracle can at best hope to find an ABR [152]. Here we can apply *Approximate Double-Oracles* (ADO), which use linear programming to compute an *approximate mixed-strategy NE* $\langle \mu_1, \mu_2 \rangle$ for $\mathcal{N}$, where μ_i represents a *mixture* over policies $\pi_{i,1..n}$ for player i. We therefore require a function $O : \Pi_i^\mu \rightarrow \Pi_i$ that computes an ABR π_i to a mixture of policies π_i^μ:

Definition 7.4 (Approximate Best Response)**.** *A policy $\pi_i \in \Pi_i$ of player i is an approximate best response against a mixture of policies π_j^μ,* iff*,*

$$\forall \pi_i' \in \Pi_i, \mathcal{G}_i(\langle \pi_i, \pi_j^\mu \rangle) \geq \mathcal{G}_i(\langle \pi_i', \pi_j^\mu \rangle). \tag{8}$$

ABRs estimate the exploitability $\mathcal{G}_E$ of the current mixtures:

$$\mathcal{G}_E \leftarrow \mathcal{G}_i(\langle O_i(\pi_j^\mu), \pi_j^\mu \rangle) + \mathcal{G}_j(\langle \pi_i^\mu, O_j(\pi_i^\mu) \rangle) \tag{9}$$

If $\mathcal{G}_E \leq 0$, then the oracle has failed to find an ABR, and a *resource bounded Nash equilibrium* (RBNE) has been found [152]. Resources in this context refers to the amount of computational power available for obtaining an ABR:

Definition 7.5 (Resource Bounded Nash Equilibrium)**.** *Two mixtures of policies $\langle \pi_1^\mu, \pi_2^\mu \rangle$ are a resource-bounded Nash equilibrium (RBNE)* iff*,*

$$\forall_i \mathcal{G}_i(\langle \pi_i^\mu, \pi_j^\mu \rangle) \geq \mathcal{G}_i(\langle O_i(\pi_j^\mu), \pi_j^\mu \rangle). \tag{10}$$

Each agents' oracle is unable to find a response that outperforms the current mixture of policies against the opponent's one. Therefore, an RBNE has been found. The ADO approach is outlined in Algorithm 2 from the perspective of Blue and Red agents for ACO. The algorithm makes use of a number of functions, including i.) InitialStrategies, for providing an initial set of strategies for each agent, which can also include rules-based and other approaches; ii.) AugmentGame, that computes missing table entries for the new resource bounded best responses, and; iii.) SolveGame, for computing mixtures once the payoff table has been augmented [152].

7.4 Towards Scaling Adversarial Learning Approaches

Versions of ADO have been used at scale, e.g., Vinyals et al. [212] achieve a grandmaster level performance in StarCraft II, beating top human players using AlphaStar, an ADO inspired approach that computes best responses using a matchmaking mechanism that samples strong opponents with an increased probability. Imitation learning was utilized to obtain an initial pool of agents. OpenAI Five made use of similar mixture of self-play method and a dynamically-updated meta-distribution over past policies, beating top human players at Dota [21, 163]. Both achievements, although very impressive, come at a high cost with respect to engineering and training time. Indeed, despite their success, questions remain

Algorithm 2 Approximate Double Oracle Algorithm

$\langle \pi_{Blue}, \pi_{Red} \rangle \leftarrow$ INITIALSTRATEGIES()
$\langle \mu_{Blue}, \mu_{Red} \rangle \leftarrow \langle \{\pi_{Blue}\}, \{\pi_{Red}\} \rangle$ ▷ set initial mixtures
while True **do**
  $\pi_{Blue} \leftarrow$ RBBR(μ_{Red}) ▷ get new res. bounded best resp.
  $\pi_{Red} \leftarrow$ RBBR(μ_{Blue})
  $\mathcal{G}_{RBBRs} \leftarrow \mathcal{G}_{Blue}(\pi_{Blue}, \mu_{Red}) + \mathcal{G}_{Red}(\mu_{Blue}, \pi_{Red})$ ▷ Exploitability.
  if $\mathcal{G}_{RBBRs} \leq \epsilon$ **then**
    break ▷ found ϵ-RBNE
  end if
  $SG \leftarrow$ AUGMENTGAME($SG, \pi_{Blue}, \pi_{Red}$)
  $\langle \mu_{Blue}, \mu_{Red} \rangle \leftarrow$ SOLVEGAME(SG)
end while

regarding the scalability of ADO to complex domains without making concessions at the cost of theoretical guarantees. ADO approaches are expensive due to:

Computing ABRs in non-trivial environments is a lengthy process [108]. Meanwhile, even for simple environments such as Kuhn poker, ADO can require over 20 ABR iterations to approach the value of the game. For the slightly more complex Leduc this number grows to over 200, with further improvements still being obtainable [108, 118]. An even larger number of iterations would likely be necessary for ADO to approach π^*_{Blue} for ACO. Computing ABRs from scratch in each iteration would therefore be wasteful, especially if the same skills are learnt from scratch in each iteration [122]. In addition, due to the ABRs being resource bounded [152], each agent can end up with a population of under-trained policies [122]. An idealized ADO reuses *relevant* knowledge acquired over past iterations.

Payoff Matrix Augmentation. The second challenge with respect to scalability is the growing normal-form game $\mathcal{N}$. Augmenting the payoff table with entries for new best responses takes exponential time with respect to the number of policies [108]. Even for simple games, obtaining a good payoff estimate for each $\mathcal{G}_i(\langle \pi^r_i, \pi^c_j \rangle)$ can require thousands of evaluation episodes. Principled strategies are required for keeping the payoff tables to a manageable size.

Large Policy Supports. There is an overhead for having to store and utilize a large number of function approximators – one for each policy in the support [124].

While the above challenges limit the scalability of ADO, there have been efforts towards remedying this approach without harming too many of the underlying principles. Liu et al. [122] propose Neural Population Learning (NeuPL) a method that deviates from the standard ADO algorithm PSRO in two specific ways: i.) all unique policies within the population are trained continuously, and; ii.) uses a single policy that contains the entire population of policies, conditioned on an opponent mixture identifier. This allows for learning to be transferred across policies without a loss of generality. The approach is capable of outperforming PSRO while maintaining a population of eight agents on running-with-scissors, which extends the rock-paper-scissors game to the spatio-temporal and partially-observed Markov game [207].

It is also worth noting that the benefits of PSRO have been explored within single agent domains, specifically within the context of robust adversarial reinforcement learning (RARL) [164]. For RARL a single agent environment is converted into a zero-sum game between the standard agent, the protagonist, and an adversarial domain agent, that can manipulate the environment, e.g., through perturbing the protagonist's actions. Yang et al. [224] recently applied a variant of PSRO to this problem, using model agnostic meta learning (MAML) [56] as the best-response oracle. The authors conducted an empirical evaluation on MuJoCo environments, finding that the proposed method outperforms state-of-the-art baselines such as standard MAML [224].

ADO has also been applied to general-sum games, which can have more than one Nash equilibrium. This gives rise to the equilibrium selection problem. To remedy this, and enable scalable policy evaluation in general, Muller et al. [143] apply the α-Rank solution concept, which does not face an equilibrium selection problem. The α-Rank solution concept establishes an ordering over policies within the support of each player. Specifically, α-Rank uses Markov-Conley Chains to identify the presence of cycles in game dynamics, and thereby rank policies. The approach attempts to compute stationary distributions by evaluating the strategy profiles of N agents through an evolutionary process of mutation and selection. Yang et al. [225] reviewed the claim of α-ranks tractability. The authors find that instead of being a polynomial time implementation, (with respect to the total number of pure strategy profiles) solving α-Rank is NP-hard. The authors introduce α^{α}-Rank, a stochastic implementation of α-Rank that does not require an exponentially-large transitions matrix for ranking policies, and can terminate early.

Feng et al. [55] replace the computation of mixtures using linear-programming with a Neural Auto-Curricula (NAC), using meta-gradient descent to automate the discovery of the learning update rules (the mixtures) without explicit human design. The NAC is optimized via interaction with the game engine, where both players aim to minimise their exploitability. Even without human design, the discovered MARL algorithms achieve competitive or even better performance with the SOTA population-based game solvers on a number of benchmarking environments, including: Games of Skill, differentiable Lotto, non-transitive Mixture Games, Iterated Matching Pennies, and Kuhn Poker. However, the approach does not solve scalability issues with respect to the increase in time for augmenting the payoff tables, and the time required for computing ABRs.

Finally, we note that DO based methods have also been utilized for reducing the number of actions for an agent to choose from, given a state s in two player games. Bakhtin et al. [15] propose an ADO based algorithm for action exploration and equilibrium approximation in games with combinatorial action spaces: Double Oracle Reinforcement learning for Action exploration (DORA). This algorithm simultaneously performs value iteration while learning a policy proposal network. An ADO step is used to explore additional actions to add to the policy proposals. The authors show that their approach achieves superhuman performance on a two-player variant of the board game Diplomacy.

8 Discussion and Open Questions

The previous sections provide a comprehensive overview of the plethora of methods for addressing high-dimensional states, large combinatorial action spaces and reducing the exploitability of DRL agents. We shall now consider how the lessons learned from this process can help us formulate an idealised learner for ACO, and distill open research questions.

The need for improved evaluation metrics: After reading Section 7, it should be evident that success achieved against a stationary opponent should be enjoyed with caution. For example, for the CAGE challenges [30, 31, 200], the evaluation process to-date has consisted of evaluating the submitted approach against rules-based Red agents with different trial lengths. This formulation is concerning. Solutions that overfit on the provided Red agents are likely to perform best. To address this shortcoming we advocate for *exploitability* being used as a standard evaluation metric, given that it is widely used by the adversarial learning community [108, 152, 76], raising the research question:

Q1: How exploitable are current approaches for autonomous cyber operations?

The need for improved evaluation environments: In Section 4, it is identified that the challenges which would be present in an idealised ACO training environment are not currently implemented in CybORG or YAWNING TITAN. While environments are available to benchmark subsets of these challenges, no environment can be used to benchmark all of them. In particular, although *RecSim* and *Micro-RTS* are available to benchmark high-dimensional observation and action spaces, respectively, no environment is available to pose these challenges alongside graph-based dynamics. Practitioners aiming to develop methodologies which rely on graph constructs specifically have several avenues for environment development, with two open-source but difficult to modify ACO environments being available, alongside frameworks such as *Gym-ANM* which are designed to be built upon. As such, the following research (and development) questions arise:

Q2: Will methodologies developed on environments such as *RecSim* and *Micro-RTS* translate well to ACO?
Q3: Where should environment developers seeking to build towards a more idealised graph-based ACO-like environment focus their efforts?

Limiting the exploitability of cyber defence agents: Handcrafting world-beating rules-based agents that find the value of the game is non-trivial, even for simple games [209]. Therefore, our thoughts turn towards adversarial learning approaches for ACO. We have seen that best response techniques have a significant amount of potential. However, while computing an (approximate) best response to measure exploitability is a reasonable expenditure, best response techniques require this in *every* iteration. As a result the literature on adversarial learning often focuses on simple games for benchmarking such as Kuhn and Leduc poker, where the value of the game is known [109, 55, 118, 108]. Meanwhile, methods that we have identified as suitable for ACO can require lengthy training times. This raises the following research questions:

Q4: Can we implement an *efficient* best response framework for cyber defence?
Q5: What is the value of the game for non-trivial cyber defence configurations?

Efficient approximate best response oracles: While RQ4 focuses on mechanisms within the approximate best response framework to reduce training time (e.g., through knowledge reuse [122]), the endeavour is underpinned by the need for methods that themselves can be trained efficiently. However, this is not a property one would associate with the methods discussed in sections 5 and 6, raising the question:

Q6: Can we implement *wall time efficient* best response oracles for ACO?

9 Conclusions

In this work we surveyed the DRL literature to formulate an idealised learner for autonomous cyber operations. This idealised learner sits at the intersection of three active areas of research, namely environments that confront learners with the curse of dimensionality, with respect to both state and action spaces, and adversarial learning. While significant efforts have been made on each of these topics individually, each still remains an active area of research.

While SOTA DNNs have allowed RL approaches to be scaled to domains that were previously considered high-dimensional [137, 138], in this survey we provide an overview of solutions for environments that push standard DRL approaches to their limits. In sections 5 – 7 we identify components for the implementation and evaluation of an idealised learning agent for ACO, and in Section 8 we discuss both theoretical and engineering challenges that need to be overcome in-order to: implement our idealised agent, and; train it at scale. We hope that this survey will raise awareness regarding issues that need to be solved in-order for DRL to be scalable to challenging cyber defence scenarios, and that our work will inspire readers to attempt to answer the research questions we have distilled from the literature.

10 Acknowledgements

Research funded by Frazer-Nash Consultancy Ltd. on behalf of the Defence Science and Technology Laboratory (Dstl) which is an executive agency of the UK Ministry of Defence providing world class expertise and delivering cutting-edge science and technology for the benefit of the nation and allies. The research supports the Autonomous Resilient Cyber Defence (ARCD) project within the Dstl Cyber Defence Enhancement programme.

11 License

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. To view a copy of this license, visit https://creativecommons.org/licenses/by-nc-nd/4.0/ or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Appendix A Deep Reinforcement Learning

A.1 Deep Q-Learning

In DRL multi-layer Deep Neural Networks (DNNs) are used to approximate value functions or, as we shall see below, even learn a parameterized policy. We shall assume a DNN with parameters θ and sufficient representational capacity to learn our required approximations. For Deep Q-learning (DQN), DNNs map a set of n-dimensional state variables to a set of m-dimensional Q-values $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ [137, 138]. Here, m represents the number of actions available to the agent. The parameters θ are optimized via stochastic gradient descent, by randomly sampling past transitions stored within an experience replay memory $\mathcal{D}$ [120, 138, 176, 141]. The DNN is trained to minimize the time dependent loss function,

$$L_k\left(\theta_k\right) = \mathbb{E}_{(s,a,s',r)\sim U(\mathcal{D})}\Big[\left(Y_k - Q\left(s,a;\ \theta_k\right)\right)^2\Big], \tag{A1}$$

where $(s, a, s', r) \sim U(\mathcal{D})$ represents mini-batches of experiences drawn uniformly at random from $\mathcal{D}$, t the current iteration, and Y_k is the target:

$$Y_k \equiv r + \gamma Q(s', \underset{a\in\mathcal{U}}{\operatorname{argmax}}\, Q(s', a;\ \theta_k);\ \theta'_k). \tag{A2}$$

Equation (A2) uses Double Deep Q-learning (DDQN) [204], where the target action is selected using weights θ, while the target value is computed using weights θ' from a target network. Using the current network rather than the target network for the target value estimation decouples action selection and evaluation for more stable updates. Weights are copied from the current to the target network after every n transitions [205] or via a soft target update [119]. DDQNs have been shown to reduce overoptimistic value estimates [205].

A.2 Proximal Policy Optimisation

While laying important foundations, DQNs tend to be outperformed by SOTA policy gradient (PG) methods, such as *Proximal Policy Optimization* (PPO) [178]. PG approaches optimize a parameterized policy π_θ directly. The objective is to find an optimal policy π^*_θ with respect to the parameters θ, which maximizes the expected return $J(\theta) = \mathbb{E}_{x_i\sim p_\pi, a_i\sim\pi}[\mathcal{R}_0]$. PPO gathers $(s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, ..., s_{t+n}, a_{t+n}, r_{t+n})$ trajectories by interacting with the environment, potentially over multiple instances of the environment using multiple actors synchronously or asynchronously [139]. The collected samples are then used to update the policy for a number of epochs, upon which the collected data is discarded. Then, a new iteration of collecting data for updating the policy begins, making the algorithm *on policy*. The policy is optimized through estimating the PG and subsequently running a stochastic gradient ascent algorithm:

$$\hat{g} = \hat{\mathbb{E}}_t\Big[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t\Big]. \tag{A3}$$

In the above equation $\hat{A}_t$ is an estimate of the advantage function at time step t. The advantage function is computed using a learned state-value function $V(s)$, resulting in an actor-critic style architecture. The expectation $\hat{\mathbb{E}}_t[...]$ represents the fact that an empirical average over a finite batch of samples is used for computing the gradient. Therefore, as mentioned above, we assume an algorithm that alternates between sampling and updating the policy parameters θ:

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t\Big[\log \pi_\theta(a_t|s_t)\hat{A}_t\Big]. \tag{A4}$$

Performing multiple optimization steps on this loss, i.e., using the same trajectory, can lead to destructively large policy updates. To address this, PPO uses a a clipped surrogate loss function that effectively constrain the size of each update, allowing multiple epochs of mini-batch updates to be performed on gathered data. Given a probability ratio

$$\rho_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, \tag{A5}$$

PPO prevents θ from moving too far away from θ_{old}:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\Big[min(\rho_t(\theta)\hat{A}_t, clip(\rho_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)\Big]. \tag{A6}$$

A.3 Deep Deterministic Policy Gradients

For continuous control problems, the policy can also be updated by taking the gradient of the expected return. A popular actor-critic method is Deep Deterministic Policy Gradients (DDPG) [119] which uses many of the same fundamental principles of DQN but applied to continuous control problems. An actor $\mu(s)$ with network parameters ϕ is used for action selection, while the critic $Q(s,a)$ with network parameters θ provides the value function estimate. The critic is updated using temporal difference learning as with DQN in Equation A2, but the target value is estimated using the target actor and critic networks:

$$Y_k = r + \gamma \hat{Q}_{\hat{\theta}}\left(s', \hat{\mu}_{\hat{\phi}}(s')\right) \tag{A7}$$

The actor network is then updated through gradient ascent using the deterministic policy gradient algorithm [183]:

$$\nabla_\phi J_k(\phi) = \mathbb{E}\left[\nabla_a Q_\theta(s,a)|_{a=\mu(s)} \nabla_\phi \mu_\phi(s)\right] \tag{A8}$$

The target actor and critic networks are then updated using soft target updates.

Appendix B High-Dimensional State Space Approaches Overview

Papers that are relevant with regard to the high-dimensional state spaces				
Work	**Summary**	**Abstr.**	**Expl.**	**CF**
Abel et al. [4]	Authors introduce two classes of abstractions: *transitive* and *PAC* state abstractions, and show that transitive PAC abstractions can be acquired efficiently, preserve near optimal-behavior, and experimentally reduce sample complexity in *simple domains*.	✓	X	X
Abel et al. [3]	Authors investigate approximate state abstractions, Present theoretical guarantees of the quality of behaviors derived from four types of approximate abstractions. Empirically demonstrate that approximate abstractions lead to reduction in task complexity and bounded loss of optimality of behavior in a variety of environments.	✓	✓	X
Abel et al. [5]	Seek to understand the role of information-theoretic compression in state abstraction for sequential decision making, resulting in a novel objective function.	✓	✓	X
Abel et al. [6]	Combine state abstractions and options to preserve the representation of near-optimal policies.	✓	✓	X
Atkinson et al. [12]	A generative network is used to generate short sequences from previous tasks for the DQN to train on, in order to prevent catastrophic forgetting as the new task is transferred.	X	X	✓
Badia et al. [13]	Construct an episodic memory-based intrinsic reward using k-nearest neighbours over recent experiences. Encourage the agent to repeatedly revisit all states in its environment.	X	✓	✓
Bellemare et al. [19]	Use Pseudo-Counts to count salient events, derived from the log-probability improvement according to a *sequential density model* over the state space.	✓	✓	X
Bougie and Ichise [22]	Introduce the concept of fast and slow curiosity that aims to incentivise long-time horizon exploration. Method decomposes the curiosity bonus into a fast reward that deals with local exploration and a slow reward that encourages global exploration.	X	✓	X
Burda et al. [26]	Introduce a method to flexibly combine intrinsic and extrinsic rewards via a *random network distillation* (RND) bonus enabling significant progress on several hard exploration Atari games.	X	✓	X
Burden and Kudenko [27]	Automate the generation of Abstract Markov Decision Processes (AMDPs) using uniform state abstractions. Explores the effectiveness and efficiency of different resolutions of state abstractions.	✓	✓	X
Burden et al. [28]	Introduce Latent Property State Abstraction, for the full automation of creating and solving an AMDP, and apply potential function for potential based reward shaping.	✓	✓	X
Gelada et al. [62]	Introduce DeepMDP, where the ℓ_2 distance represents an upper bound of the bisimulation distance, learning embeddings that ignore irrelevant objects that are of no consequence to the learning agent(s).	✓	✓	X
Colas et al. [38]	Proposes CURIOUS, an algorithm that leverages a modular Universal Value Function Approximator with hindsight learning to achieve a diversity of goals of different kinds within a unique policy and an automated curriculum learning mechanism that biases the attention of the agent towards goals maximizing the absolute learning progress. Also focuses on goals that are being forgotten.	X	✓	✓
de Bruin et al. [41]	Study the extent to which experience replay memory composition can mitigate catastrophic forgetting.	X	X	✓
Precup [165]	Introduction of the *options* framework, for prediction, control and learning at multiple timescales. [a]	✓	✓	↗
Fang et al. [51]	Introduce Adaptive Procedural Task Generation (APT-Gen), an approach to progressively generate a sequence of tasks as curricula to facilitate reinforcement learning in hard-exploration problems.	X	✓	X
Forestier et al. [58]	Introduce an intrinsically motivated goal exploration processes with automatic curriculum learning.	✓	✓	X
Fu et al. [59]	Propose a novelty detection algorithm for exploration. Classifiers are trained to discriminate each visited state against all others, where novel states are easier to distinguish.	X	✓	X
Giannakopoulos et al. [63]	Map high-dim video to a low-dim discrete latent representation using a VQ-AE.	✓	✓	X
Hester et al. [81]	Focus on learning exploration in model-based RL.	X	✓	X
Kessler et al. [95]	Show that existing continual learning methods based on single neural network predictors with shared replay buffers fail in the presence of interference. Propose a factorized policy, using shared feature extraction layers, but separate heads, each specializing on a new task to prevent interference.	X	X	✓
Kovač et al. [103]	Set goals in the region of highest uncertainty. Exploring uncertain states with regard to the rewards are the sub-goals.	✓	✓	X
Kulkarni et al. [105]	Introduce a hierarchical-DQN (h-DQN) operating at different temporal scales.	✓	✓	X
Dietterich [44]	An overview of the MAXQ value function decomposition and its support for state and action abstraction.	✓	✓	X
Machado et al. [127]	Introduce a Laplacian framework for option discovery.	✓	✓	X
Machado et al. [128]	Look at *Eigenoptions*, options obtained from representations that encode diffusive information flow in the environment. Authors extend the existing algorithms for Eigenoption discovery to settings with stochastic transitions and in which handcrafted features are not available.	✓	✓	X
Misra et al. [136]	Authors introduce HOMER, an iterative state abstraction approach that accounts for the fact that the learning of a compact representation for states requires comprehensive information from the environment.	✓	✓	X
Martin et al. [129]	Count-based exploration in feature space rather than for the raw inputs.	✓	✓	X
Pathak et al. [161]	Intrinsic rewards based method that formulates curiosity as the error in an agent's ability to predict the consequence of its own actions in a visual feature space learned by a self-supervised inverse dynamics model.	✓	✓	X
Savinov et al. [174]	Propose a new curiosity method which uses episodic memory to form the novelty bonus. Current observation is compared with the observations in memory.	✓	✓	X
Stadie et al. [186]	Evaluate sophisticated exploration strategies, including Thompson sampling and Boltzman exploration, and propose a new exploration method based on assigning exploration bonuses from a concurrently learned model of the system dynamics.	✓	✓	X
Ribeiro et al. [169]	Train DRL on two similar tasks, augmented with EWC to mitigate catastrophic forgetting.	X	X	✓
Tang et al. [194]	Use an auto-encoder and SimHash for to enable count based exploration.	✓	✓	X
Vezhnevets et al. [208]	Introduce FeUdal Networks (FuNs): a novel architecture for hierarchical RL. FuNs employs a Manager module and a Worker module. The Manager operates at a slower time scale and sets abstract goals which are conveyed to and enacted by the Worker. The Worker generates primitive actions at every tick of the environment.	✓	✓	X
Zhang et al. [235]	Propose *deep bisimulation for control* (DBC). DBC learns directly on this bisimulation distance metric. Allows the learning of invariant representations that can be used effectively for downstream control policies, and are invariant with respect to task-irrelevant details.	✓	✓	X

Table B1: An overview of approaches for addressing the curse of dimensionality with regard to states, and the extent to which works cover the topics of abstraction, *advanced* exploration strategies, and the mitigation of catastrophic forgetting.

Appendix C High-Dimensional Action Space Approaches Overview

High-Dimensional Action Space Approaches Overview						
Algorithm	**Approach**	**Summary**	**Applications**	**SLC**	**Gen.**	**TVA**
Wolpertinger Architecture [46]	Proto Actions	Passes k nearest neighbours of proto action obtained from actor to critic for evaluation.	Flexible	✓	✓	↗
DDPG+k-NN Slate-MDP Solver [191]	Proto Actions	Uses DDPG to learn Slate Policies and Guide Attention.	Slate-MDPs	✓	✓	✓
DeepPage [236]	Proto Actions	Proposes solution for page-wise recommendations based on real-time feedback.	Slate-MDPs	✓	✓	✓
LIRD [237]	Proto Actions	Authors introduce an online user-agent interacting environment simulator and a List-wise Recommendation framework LIRD.	Slate-MDPs	✓	✓	✓
CRLA [199, 198]	Decomp. & Hierarchical	Cascaded RL agents that algebraically construct an action index.	Flexible	✓	✓	X
SEQUENTIAL DQN [135]	Decomposition	Discrete Sequential Prediction of Continuous Actions	Multi-Actuator	✓	✓	X
BDQ [195]	Decomposition	Branching Dueling Q-Network for fine control of discretized continuous control domains.	Multi-Actuator	✓	✓	↗
Cascading DQN [36]	Decomposition	Model based RL approach for recommender systems that utilizes a Generative Adversarial Network (GANs) to imitate user behaviour dynamics and reward function.	Slate-MDPs	✓	✓	X
Parameter Sharing TRPO [70]	Decomposition	Cooperative Multi-Agent Control Using Deep Reinforcement Learning.	Multi-Actuator	✓	✓	X
SCORE [117]	Decomposition	Structured Cooperative Reinforcement Learning With Time-Varying Composite Action Space.	General	✓	✓	↗
Action-Elimination DQN (AE-DQN) [231]	Elimination	Combines a DRL algorithm with an *Action Elimination Network* that eliminates sub-optimal actions.	Text-based games	✓	X	X
Top-k Candidate Generator [34]	Elimination	Adapts the REINFORCE algorithm into a top-k neural candidate generator for large action spaces.	Slate-MDPs	✓	X	X
PATRPO & PASVG(0) [218]	Hierarchical	Hierarchical architecture for the tasks where the parameter policy is conditioned on the output of the discrete action policy.	PA-MDPs	✓	X	X
Growing Action Spaces [54]	Curriculum Learning	CL approach where agents gradually learn large action spaces, based on a hierarchical action space defined using domain knowledge.	Flexible	✓	X	X
Progressive Action Space [229]	Curriculum Learning	Curriculum offline RL with a progressive action space.	Flexible	✓	X	X

Table C2: A summary of approaches designed for high dimensional action spaces. The final three columns are abbreviated: Sublinear Complexity (SLC), Generalizeabilty (Gen.) and Time Varying Actions (TVA). The ↗ symbol indicates that steps have been taken towards addressing one of the above criteria. Wolpertinger for instance can handle TVA within the current embedding space, but would require retraining if an action embedding axis was added/removed.

References

[1] D. Abel. A theory of state abstraction for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9876–9877, 2019.

[2] D. Abel, A. Agarwal, F. Diaz, A. Krishnamurthy, and R. E. Schapire. Exploratory gradient boosting for reinforcement learning in complex domains. *arXiv preprint arXiv:1603.04119*, 2016.

[3] D. Abel, D. Hershkowitz, and M. Littman. Near optimal behavior via approximate state abstraction. In *International Conference on Machine Learning*, pages 2915–2923. PMLR, 2016.

[4] D. Abel, D. Arumugam, L. Lehnert, and M. Littman. State abstractions for lifelong reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 10–19. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/abel18a.html.

[5] D. Abel, D. Arumugam, K. Asadi, Y. Jinnai, M. L. Littman, and L. L. Wong. State abstraction as compression in apprenticeship learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3134–3142, 2019.

[6] D. Abel, N. Umbanhowar, K. Khetarpal, D. Arumugam, D. Precup, and M. Littman. Value preserving state-action abstractions. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1639–1650. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/abel20a.html.

[7] A. M. K. Adawadkar and N. Kulkarni. Cyber-security and reinforcement learning—a brief survey. *Engineering Applications of Artificial Intelligence*, 114: 105116, 2022.

[8] M. Albahar. Cyber attacks and terrorism: A twenty-first century conundrum. *Science and engineering ethics*, 25:993–1006, 2019.

[9] L. N. Alegre. SUMO-RL. https://github.com/LucasAlegre/sumo-rl, 2019.

[10] P. Almasan, J. Suárez-Varela, K. Rusek, P. Barlet-Ros, and A. Cabellos-Aparicio. Deep reinforcement learning meets graph neural networks: exploring a routing optimization use case. *Computer Communications*, 196:184–194, 2022.

[11] A. Andrew, S. Spillard, J. Collyer, and N. Dhir. Developing optimal causal cyber-defence agents via cyber security simulation. In *Workshop on Machine Learning for Cybersecurity (ML4Cyber)*, 07 2022.

[12] C. Atkinson, B. McCane, L. Szymanski, and A. Robins. Pseudo-rehearsal: Achieving deep reinforcement learning without catastrophic forgetting. *Neurocomputing*, 428:291–307, 2021.

[13] A. P. Badia, P. Sprechmann, A. Vitvitskyi, D. Guo, B. Piot, S. Kapturowski, O. Tieleman, M. Arjovsky, A. Pritzel, A. Bolt, et al. Never give up: Learning directed exploration strategies. *arXiv preprint arXiv:2002.06038*, 2020.

[14] C. Baillie, M. Standen, J. Schwartz, M. Docking, D. Bowman, and J. Kim. Cyborg: An autonomous cyber operations research gym. *arXiv preprint arXiv:2002.10667*, 2020.

[15] A. Bakhtin, D. Wu, A. Lerer, and N. Brown. No-press diplomacy from scratch. *Advances in Neural Information Processing Systems*, 34:18063–18074, 2021.

[16] S. Barrett, M. Hausknecht, S. Kalyanakrishnan, P. Mupparaju, S. Narvekar, A. Siddharth, and S. Subramanian. Half field offense in robocup 2d soccer. https://github.com/LARG/HFO, 2015. Last Update: 2016-11-13, Accessed: 2017-01-27.

[17] A. Bassich and D. Kudenko. Continuous curriculum learning for reinforcement learning. In *Proceedings of the 2nd Scaling-Up Reinforcement Learning (SURL) Workshop. IJCAI*, 2019.

[18] A. Bassich, F. Foglino, M. Leonetti, and D. Kudenko. Curriculum learning with a progression function. *arXiv preprint arXiv:2008.00511*, 2020.

[19] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

[20] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[21] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[22] N. Bougie and R. Ichise. Fast and slow curiosity for high-level exploration in reinforcement learning. *Applied Intelligence*, 51:1086–1107, 2021.

[23] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

[24] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym, 2016.

[25] N. Brown, A. Lerer, S. Gross, and T. Sandholm. Deep counterfactual regret minimization. In *International conference on machine learning*, pages 793–802. PMLR, 2019.

[26] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[27] J. Burden and D. Kudenko. Using uniform state abstractions for reward shaping with reinforcement learning. In *Workshop on Adaptive Learning Agents (ALA) at the Federated AI Meeting*, volume 18, 2018.

[28] J. Burden, S. K. Siahroudi, and D. Kudenko. Latent property state abstraction for reinforcement learning. In *Proceedings of the AAMAS Workshop on Adaptive Learning Agents (ALA)*, 2021.

[29] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews, 38 (2), 2008*, 2008.

[30] CAGE. CAGE Challenge 1. *IJCAI-21 1st International Workshop on Adaptive Cyber Defense. arXiv*, 2021.

[31] CAGE. TTCP CAGE Challenge 2. https://github.com/cage-challenge/cage-challenge-2, 2022.

[32] S. Cao, S. Liao, S. Wang, and X. Luo. Optimal control with deep reinforcement learning for shunt compensations to enhance voltage stability. In *2020 5th Asia Conference on Power and Electrical Engineering (ACPEE)*, pages 398–403. IEEE, 2020.

[33] Y. Chandak, G. Theocharous, C. Nota, and P. Thomas. Lifelong learning with a changing action set. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3373–3380, 2020.

[34] M. Chen, A. Beutel, P. Covington, S. Jain, F. Belletti, and E. H. Chi. Top-k off-policy correction for a reinforce recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, page 456–464, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450359405. doi: 10.1145/3289600.3290999. URL https://doi.org/10.1145/3289600.3290999.

[35] S. Chen, X. Ma, and D. Hsu. Dinerdash gym: A benchmark for policy learning in high-dimensional action space. *arXiv preprint arXiv:2007.06207*, 2020.

[36] X. Chen, S. Li, H. Li, S. Jiang, Y. Qi, and L. Song. Generative adversarial user model for reinforcement learning based recommendation system. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1052–1061. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/chen19f.html.

[37] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *In AAAI/IAAI*, 1998:746–752, 1998.

[38] C. Colas, P. Fournier, M. Chetouani, O. Sigaud, and P.-Y. Oudeyer. Curious: intrinsically motivated modular multi-goal reinforcement learning. In *International conference on machine learning*, pages 1331–1340. PMLR, 2019.

[39] A. Davis, S. Gill, R. Wong, and S. Tayeb. Feature selection for deep neural networks in cyber security applications. In *2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, pages 1–7, 2020. doi: 10.1109/IEMTRONICS51293.2020.9216403.

[40] P. Dayan and G. E. Hinton. Feudal reinforcement learning. *Advances in neural information processing systems*, 5, 1992.

[41] T. de Bruin, J. Kober, K. Tuyls, and R. Babuška. The importance of experience replay database composition in deep reinforcement learning. In *Deep Reinforcement Learning Workshop, NIPS*, 2015.

[42] DeUmbraTX. practical_rllib_tutorial. https://github.com/DeUmbraTX/practical_rllib_tutorial/blob/main/your_rllib_environment.py, 2022. Accessed: 2023-03-20.

[43] T. G. Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of artificial intelligence research*, 13:227–303, 2000.

[44] T. G. Dietterich. An overview of maxq hierarchical reinforcement learning. In *Abstraction, Reformulation, and Approximation: 4th International Symposium, SARA 2000 Horseshoe Bay, USA, July 26–29, 2000 Proceedings 4*, pages 26–44. Springer, 2000.

[45] V. Duddu. A survey of adversarial machine learning in cyber warfare. *Defence Science Journal*, 68(4):356, 2018.

[46] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

[47] H. Edwards and A. Storkey. Towards a neural statistician. In *International Conference on Learning Representations*, 2017. URL https://openreview.

net/forum?id=HJDBUF5le.

[48] B. Ellis, S. Moalla, M. Samvelyan, M. Sun, A. Mahajan, J. N. Foerster, and S. Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning, 2022. URL https://arxiv.org/abs/2212.07489.

[49] Z. Fan, R. Su, W. Zhang, and Y. Yu. Hybrid actor-critic reinforcement learning in parameterized action space. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 2279–2285, 2019.

[50] K. Fang, Y. Zhu, A. Garg, A. Kurenkov, V. Mehta, L. Fei-Fei, and S. Savarese. Learning task-oriented grasping for tool manipulation from simulated self-supervision. *The International Journal of Robotics Research*, 39(2-3):202–216, 2020.

[51] K. Fang, Y. Zhu, S. Savarese, and L. Fei-Fei. Adaptive procedural task generation for hard-exploration problems. In *International Conference on Learning Representations*, 2021.

[52] Farama Foundation. Gymnasium mujoco documentation. https://gymnasium.farama.org/environments/mujoco/, 2023. v0.27.1, Accessed: 2023-03-20.

[53] Farama Foundation. Petting zoo documentation. https://pettingzoo.farama.org/, 2023. v1.22.3, Accessed: 2023-03-20.

[54] G. Farquhar, L. Gustafson, Z. Lin, S. Whiteson, N. Usunier, and G. Synnaeve. Growing action spaces. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3040–3051. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/farquhar20a.html.

[55] X. Feng, O. Slumbers, Z. Wan, B. Liu, S. McAleer, Y. Wen, J. Wang, and Y. Yang. Neural auto-curricula in two-player zero-sum games. *Advances in Neural Information Processing Systems*, 34:3504–3517, 2021.

[56] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[57] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. Torr, P. Kohli, and S. Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pages 1146–1155, 2017.

[58] S. Forestier, R. Portelas, Y. Mollard, and P.-Y. Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv preprint arXiv:1708.02190*, 2017.

[59] J. Fu, J. Co-Reyes, and S. Levine. Ex2: Exploration with exemplar models for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

[60] S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062. PMLR, 2019.

[61] C. Gelada, S. Kumar, J. Buckman, O. Nachum, and M. G. Bellemare. Deepmdp: Learning continuous latent space models for representation learning.

In *International Conference on Machine Learning*, pages 2170–2179. PMLR, 2019.

[62] C. Gelada, S. Kumar, J. Buckman, O. Nachum, and M. G. Bellemare. DeepMDP: Learning continuous latent space models for representation learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2170–2179. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/gelada19a.html.

[63] P. Giannakopoulos, A. Pikrakis, and Y. Cotronis. Neural discrete abstraction of high-dimensional spaces: A case study in reinforcement learning. In *2020 28th European Signal Processing Conference (EUSIPCO)*, pages 1517–1521, 2021. doi: 10.23919/Eusipco47968.2020.9287851.

[64] J. Gilmer, L. Metz, F. Faghri, S. S. Schoenholz, M. Raghu, M. Wattenberg, and I. Goodfellow. Adversarial spheres. *arXiv preprint arXiv:1801.02774*, 2018.

[65] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell. Adversarial policies: Attacking deep reinforcement learning. In *International Conference on Learning Representations*, 2019.

[66] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In *Proc. of NIPS*, pages 2672–2680, 2014.

[67] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[68] A. Gruslys, M. Lanctot, R. Munos, F. Timbers, M. Schmid, J. Perolat, D. Morrill, V. Zambaldi, J.-B. Lespiau, J. Schultz, et al. The advantage regret-matching actor-critic. *arXiv preprint arXiv:2008.12234*, 2020.

[69] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient solution algorithms for factored mdps. *Journal of Artificial Intelligence Research*, 19: 399–468, 2003.

[70] J. K. Gupta, M. Egorov, and M. Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *International conference on autonomous agents and multiagent systems*, pages 66–83. Springer, 2017.

[71] D. Ha, A. Dai, and Q. V. Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[72] 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*, pages 1856–1865, 2018.

[73] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[74] M. Hausknecht and P. Stone. Deep reinforcement learning in parameterized action space. *arXiv preprint arXiv:1511.04143*, 2015.

[75] M. Hausknecht, P. Mupparaju, S. Subramanian, S. Kalyanakrishnan, and P. Stone. Half field offense: an environment for multiagent learning and ad hoc teamwork. In *Autonomous Agents and MultiAgent Systems Adaptive Learning Agents (ALA) Workshop*, 2016.

[76] J. Heinrich and D. Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

[77] D. Hennes, D. Morrill, S. Omidshafiei, R. Munos, J. Perolat, M. Lanctot, A. Gruslys, J.-B. Lespiau, P. Parmas, E. Duenez-Guzman, et al. Neural replicator dynamics. *arXiv preprint arXiv:1906.00190*, 2019.

[78] R. Henry and D. Ernst. Gym-anm: Reinforcement learning environments for active network management tasks in electricity distribution systems. *Energy and AI*, 5:100092, 2021. ISSN 2666-5468. doi: https://doi.org/10.1016/j.egyai.2021.100092.

[79] R. Henry and D. Ernst. Gym-anm: Open-source software to leverage reinforcement learning for power system management in research and education. *Software Impacts*, 9:100092, 2021. ISSN 2665-9638. doi: https://doi.org/10.1016/j.simpa.2021.100092.

[80] P. Hernandez-Leal, B. Kartal, and M. E. Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33(6):750–797, 2019.

[81] T. Hester, M. Lopes, and P. Stone. Learning exploration strategies in model-based reinforcement learning. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1069–1076, 2013.

[82] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[83] S. Hong, D. Yoon, and K.-E. Kim. Structure-aware transformer policy for inhomogeneous multi-task reinforcement learning. In *International Conference on Learning Representations*, 2022.

[84] S. Huang, S. Ontañón, C. Bamford, and L. Grela. Gym-μrts: Toward affordable full game real-time strategy games research with deep reinforcement learning. In *2021 IEEE Conference on Games (CoG), Copenhagen, Denmark, August 17-20, 2021*, pages 1–8. IEEE, 2021. doi: 10.1109/CoG52621.2021.9619076. URL https://doi.org/10.1109/CoG52621.2021.9619076.

[85] F. Huszár. On quadratic penalties in elastic weight consolidation. *arXiv preprint arXiv:1712.03847*, 2017.

[86] F. Huszár. Note on the quadratic penalties in elastic weight consolidation. *Proceedings of the National Academy of Sciences*, 115(11):E2496–E2497, 2018.

[87] E. Ie, V. Jain, J. Wang, S. Narvekar, R. Agarwal, R. Wu, H.-T. Cheng, M. Lustman, V. Gatto, P. Covington, et al. Reinforcement learning for slate-based recommender systems: A tractable decomposition and practical methodology. *arXiv preprint arXiv:1905.12767*, 2019.

[88] E. Ie, C. wei Hsu, M. Mladenov, V. Jain, S. Narvekar, J. Wang, R. Wu, and C. Boutilier. Recsim: A configurable simulation platform for recommender systems. 2019.

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

[90] W. Jiang and J. Luo. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, 207:117921, 2022. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2022.117921. URL https://www.sciencedirect.com/science/article/pii/S0957417422011654.

[91] A. Kanervisto, C. Scheller, and V. Hautamäki. Action space shaping in deep reinforcement learning. In *2020 IEEE Conference on Games (CoG)*, pages 479–486, 2020. doi: 10.1109/CoG47356.2020.9231687.

[92] S. Kapetanakis and D. Kudenko. Reinforcement learning of coordination in cooperative multi–agent systems. *AAAI/IAAI*, 2002:326–331, 2002.

[93] S. Kapturowski, G. Ostrovski, J. Quan, R. Munos, and W. Dabney. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2019.

[94] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[95] S. Kessler, J. Parker-Holder, P. Ball, S. Zohren, and S. J. Roberts. Same state, different task: Continual reinforcement learning without interference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7143–7151, 2022.

[96] M. Khoury and D. Hadfield-Menell. On the geometry of adversarial examples. *arXiv preprint arXiv:1811.00525*, 2018.

[97] I. F. Kilincer, F. Ertam, and A. Sengur. Machine learning methods for cyber security intrusion detection: Datasets and comparative study. *Computer Networks*, 188:107840, 2021.

[98] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[99] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[100] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[101] M. Kloft and P. Laskov. Online anomaly detection under adversarial impact. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 405–412. JMLR Workshop and Conference Proceedings, 2010.

[102] R. Kortvelesy and A. Prorok. Qgnn: Value function factorisation with graph neural networks. *arXiv preprint arXiv:2205.13005*, 2022.

[103] G. Kovač, A. Laversanne-Finot, and P.-Y. Oudeyer. Grimgep: Learning progress for robust goal sampling in visual deep reinforcement learning. *IEEE Transactions on Cognitive and Developmental Systems*, pages 1–1, 2022. doi: 10.1109/TCDS.2022.3216911.

[104] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information*

processing systems, pages 1097–1105, 2012.

[105] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

[106] P. Ladosz, L. Weng, M. Kim, and H. Oh. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 2022.

[107] G. Lample and D. S. Chaplot. Playing fps games with deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[108] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Pérolat, D. Silver, and T. Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4190–4203, 2017.

[109] M. Lanctot, E. Lockhart, J.-B. Lespiau, V. Zambaldi, S. Upadhyay, J. Pérolat, S. Srinivasan, F. Timbers, K. Tuyls, S. Omidshafiei, et al. Openspiel: A framework for reinforcement learning in games. *arXiv preprint arXiv:1908.09453*, 2019.

[110] M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *In Proceedings of the Seventeenth International Conference on Machine Learning*. Citeseer, 2000.

[111] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[112] C. Li and M. Qiu. *Reinforcement learning for cyber-physical systems: with cybersecurity case studies*. Chapman and Hall/CRC, 2019.

[113] J.-h. Li. Cyber security meets artificial intelligence: a survey. *Frontiers of Information Technology & Electronic Engineering*, 19(12):1462–1474, 2018.

[114] K. Li, W. Ni, and F. Dressler. Lstm-characterized deep reinforcement learning for continuous flight control and resource allocation in uav-assisted sensor network. *IEEE Internet of Things Journal*, 9(6):4179–4189, 2021.

[115] R. Li, C. Huang, H. Zhang, and S. Jiang. Multicast scheduling for multimessage over multi-channel: A permutation-based wolpertinger deep reinforcement learning method. *arXiv preprint arXiv:2205.09420*, 2022.

[116] W. Li, B. Jin, X. Wang, J. Yan, and H. Zha. F2a2: Flexible fully-decentralized approximate actor-critic for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2004.11145*, 2020.

[117] W. Li, X. Wang, B. Jin, D. Luo, and H. Zha. Structured cooperative reinforcement learning with time-varying composite action space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):8618–8634, 2022. doi: 10.1109/TPAMI.2021.3102140.

[118] Z. Li, M. Lanctot, K. R. McKee, L. Marris, I. Gemp, D. Hennes, P. Muller, K. Larson, Y. Bachrach, and M. P. Wellman. Combining tree-search, generative models, and nash bargaining concepts in game-theoretic reinforcement learning. *arXiv preprint arXiv:2302.00797*, 2023.

[119] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv*

preprint arXiv:1509.02971, 2015.

[120] L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3-4):293–321, 1992.

[121] H. Liu and B. Lang. Machine learning and deep learning methods for intrusion detection systems: A survey. *applied sciences*, 9(20):4396, 2019.

[122] S. Liu, L. Marris, D. Hennes, J. Merel, N. Heess, and T. Graepel. Neupl: Neural population learning. *arXiv preprint arXiv:2202.07415*, 2022.

[123] X. Liu, J. Ospina, and C. Konstantinou. Deep reinforcement learning for cybersecurity assessment of wind integrated power systems. *IEEE Access*, 8: 208378–208394, 2020.

[124] E. Lockhart, M. Lanctot, J. Pérolat, J.-B. Lespiau, D. Morrill, F. TImbers, and K. Tuyls. Computing approximate equilibria in sequential adversarial games by exploitability descent. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 464–470. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/66. URL https://doi.org/10.24963/ijcai.2019/66.

[125] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. URL https://elib.dlr.de/124092/.

[126] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

[127] M. C. Machado, M. G. Bellemare, and M. Bowling. A laplacian framework for option discovery in reinforcement learning. In *International Conference on Machine Learning*, pages 2295–2304. PMLR, 2017.

[128] M. C. Machado, C. Rosenbaum, X. Guo, M. Liu, G. Tesauro, and M. Campbell. Eigenoption discovery through the deep successor representation. In *International Conference on Learning Representations*, 2018.

[129] J. Martin, S. N. Sasikumar, T. Everitt, and M. Hutter. Count-based exploration in feature space for reinforcement learning. *arXiv preprint arXiv:1706.08090*, 2017.

[130] W. Masson, P. Ranchod, and G. Konidaris. Reinforcement learning with parameterized actions. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[131] L. Matignon, G. Laurent, and N. Le Fort-Piat. Hysteretic Q-Learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 64–69, 2007.

[132] L. Matignon, G. J. Laurent, and N. Le Fort-Piat. Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. *The Knowledge Engineering Review*, 27(1):1–31, 2012.

[133] H. B. McMahan, G. J. Gordon, and A. Blum. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the 20th International*

Conference on Machine Learning (ICML-03), pages 536–543, 2003.

[134] R. Meiri and J. Zahavi. Using simulated annealing to optimize the feature selection problem in marketing applications. *European journal of operational research*, 171(3):842–858, 2006.

[135] L. Metz, J. Ibarz, N. Jaitly, and J. Davidson. Discrete sequential prediction of continuous actions for deep rl. *arXiv preprint arXiv:1705.05035*, 2017.

[136] D. Misra, M. Henaff, A. Krishnamurthy, and J. Langford. Kinematic state abstraction and provably efficient rich-observation reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6961–6971. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/misra20a.html.

[137] 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.

[138] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[139] 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*, pages 1928–1937, 2016.

[140] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[141] S. S. Mousavi, M. Schukat, and E. Howley. Deep reinforcement learning: an overview. In *Proceedings of SAI Intelligent Systems Conference*, pages 426–440. Springer, 2016.

[142] S. S. Mousavi, M. Schukat, and E. Howley. Deep reinforcement learning: an overview. In *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016: Volume 2*, pages 426–440. Springer, 2018.

[143] P. Muller, S. Omidshafiei, M. Rowland, K. Tuyls, J. Pérolat, S. Liu, D. Hennes, L. Marris, M. Lanctot, E. Hughes, et al. A generalized training approach for multiagent learning. In *ICLR*, pages 1–35. ICLR, 2020.

[144] S. Munikoti, D. Agarwal, L. Das, M. Halappanavar, and B. Natarajan. Challenges and opportunities in deep reinforcement learning with graph neural networks: A comprehensive review of algorithms and applications. *arXiv preprint arXiv:2206.07922*, 2022.

[145] J. Nash. Non-cooperative games. *Annals of mathematics*, pages 286–295, 1951.

[146] H. V. Nguyen, H. N. Nguyen, and T. Uehara. Multiple level action embedding for penetration testing. In *The 4th International Conference on Future Networks and Distributed Systems (ICFNDS)*, pages 1–9, 2020.

[147] N. D. Nguyen, T. Nguyen, and S. Nahavandi. System design perspective for human-level agents using deep reinforcement learning: A survey. *IEEE*

Access, 5:27091–27102, 2017.

[148] T. T. Nguyen and V. J. Reddi. Deep reinforcement learning for cyber security. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–17, 2021. doi: 10.1109/TNNLS.2021.3121870.

[149] A. Nicolicioiu, I. Duta, and M. Leordeanu. Recurrent space-time graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[150] A. Nowé, P. Vrancx, and Y.-M. De Hauwere. Game theory and multi-agent reinforcement learning. In *Reinforcement Learning*, pages 441–470. Springer, 2012.

[151] F. A. Oliehoek and C. Amato. A concise introduction to decentralized pomdps, 2015.

[152] F. A. Oliehoek, R. Savani, J. Gallego, E. v. d. Pol, and R. Groß. Beyond local nash equilibria for adversarial networks. In *Benelux Conference on Artificial Intelligence*, pages 73–89. Springer, 2018.

[153] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. 70:2681–2690, 2017.

[154] K. Ota, T. Oiki, D. Jha, T. Mariyama, and D. Nikovski. Can increasing input dimensionality improve deep reinforcement learning? In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7424–7433. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/ota20a.html.

[155] G. Palmer. *Independent learning approaches: Overcoming multi-agent learning pathologies in team-games*. The University of Liverpool (United Kingdom), 2020.

[156] G. Palmer, K. Tuyls, D. Bloembergen, and R. Savani. Lenient Multi-Agent Deep Reinforcement Learning. In *Proceedings of Autonomous Agents and Multi-Agent Systems*, pages 443–451, 2018.

[157] G. Palmer, R. Savani, and K. Tuyls. Negative Update Intervals in Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 43–51. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[158] L. Panait, K. Sullivan, and S. Luke. Lenience towards teammates helps in cooperative multiagent learning. In *Proceedings of Autonomous Agents and Multi-Agent Systems*, 2006.

[159] L. Panait, K. Sullivan, and S. Luke. Lenient learners in cooperative multiagent systems. In *Proceedings of Autonomous Agents and Multi-Agent Systems*, pages 801–803. ACM, 2006.

[160] L. Panait, K. Tuyls, and S. Luke. Theoretical advantages of lenient learners: An evolutionary game theoretic perspective. *Journal Machine Learning Research*, 9(Mar):423–457, 2008.

[161] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

[162] J. C. Perdomo, M. Simchowitz, A. Agarwal, and P. Bartlett. Towards a dimension-free understanding of adaptive linear control. In *Proceedings of Thirty Fourth Conference on Learning Theory*, volume 134 of *Proceedings of Machine Learning Research*, pages 3681–3770. PMLR, 15–19 Aug 2021. URL https://proceedings.mlr.press/v134/perdomo21a.html.

[163] J. Perolat, B. De Vylder, D. Hennes, E. Tarassov, F. Strub, V. de Boer, P. Muller, J. T. Connor, N. Burch, T. Anthony, et al. Mastering the game of stratego with model-free multiagent reinforcement learning. *Science*, 378(6623):990–996, 2022.

[164] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pages 2817–2826. PMLR, 2017.

[165] D. Precup. *Temporal abstraction in reinforcement learning*. University of Massachusetts Amherst, 2000.

[166] T. Rashid, M. Samvelyan, C. S. Witt, G. Farquhar, J. Foerster, and S. Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4292–4301, 2018.

[167] H. T. Reda, A. Anwar, A. N. Mahmood, and Z. Tari. A taxonomy of cyber defence strategies against false data attacks in smart grid. *arXiv preprint arXiv:2103.16085*, 2021.

[168] C. Resnick, W. Eldridge, D. Ha, D. Britz, J. Foerster, J. Togelius, K. Cho, and J. Bruna. Pommerman: A multi-agent playground. *CoRR*, abs/1809.07124, 2018. URL http://arxiv.org/abs/1809.07124.

[169] J. Ribeiro, F. Melo, and J. Dias. Multi-task learning and catastrophic forgetting in continual reinforcement learning. *EPiC Series in Computing*, 65: 163–175, 2019.

[170] D. Rohde, S. Bonner, T. Dunlop, F. Vasile, and A. Karatzoglou. Recogym: A reinforcement learning environment for the problem of product recommendation in online advertising. *arXiv preprint arXiv:1808.00720*, 2018.

[171] I. Rosenberg, A. Shabtai, Y. Elovici, and L. Rokach. Adversarial machine learning attacks and defense methods in the cyber security domain. *ACM Computing Surveys (CSUR)*, 54(5):1–36, 2021.

[172] F. Ruffy, M. Przystupa, and I. Beschastnikh. Iroko: A framework to prototype reinforcement learning for data center traffic control. *arXiv preprint arXiv:1812.09975*, 2018.

[173] M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C.-M. Hung, P. H. S. Torr, J. Foerster, and S. Whiteson. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043, 2019.

[174] N. Savinov, A. Raichuk, R. Marinier, D. Vincent, M. Pollefeys, T. Lillicrap, and S. Gelly. Episodic curiosity through reachability. *arXiv preprint arXiv:1810.02274*, 2018.

[175] M. Schak and A. Gepperth. A study on catastrophic forgetting in deep lstm networks. In *Artificial Neural Networks and Machine Learning–ICANN 2019: Deep Learning: 28th International Conference on Artificial Neural Networks,*

Munich, Germany, September 17–19, 2019, Proceedings, Part II 28, pages 714–728. Springer, 2019.

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

[177] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust Region Policy Optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

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

[179] J. Schwartz and H. Kurniawati. Autonomous penetration testing using reinforcement learning. *arXiv preprint arXiv:1905.05965*, 2019.

[180] A. Shafahi, W. R. Huang, C. Studer, S. Feizi, and T. Goldstein. Are adversarial examples inevitable? *arXiv preprint arXiv:1809.02104*, 2018.

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

[182] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed, and M. Xu. A survey on machine learning techniques for cyber security in the last decade. *IEEE Access*, 8:222310–222354, 2020. doi: 10.1109/ACCESS.2020.3041951.

[183] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

[184] H. Song, H. Jang, H. H. Tran, S.-E. Yoon, K. Son, D. Yun, H. Chung, and Y. Yi. Solving continual combinatorial selection via deep reinforcement learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3467–3474, 2019.

[185] S. Srinivasan, M. Lanctot, V. Zambaldi, J. Pérolat, K. Tuyls, R. Munos, and M. Bowling. Actor-critic policy optimization in partially observable multiagent environments. *Advances in neural information processing systems*, 31, 2018.

[186] B. C. Stadie, S. Levine, and P. Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

[187] M. Standen, M. Lucas, D. Bowman, T. J. Richer, J. Kim, and D. Marriott. Cyborg: A gym for the development of autonomous cyber agents. In *IJCAI-21 1st International Workshop on Adaptive Cyber Defense.* arXiv, 2021.

[188] M. Standen, D. Bowman, S. Hoang, T. Richer, M. Lucas, R. Van Tassel, P. Vu, M. Kiely, C. KC, and N. J. Konschnik, Collyer. Cyber operations research gym. https://github.com/cage-challenge/CybORG, 2022.

[189] E. Steinberger, A. Lerer, and N. Brown. Dream: Deep regret minimization with advantage baselines and model-free learning. *arXiv preprint arXiv:2006.10410*, 2020.

[190] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov. Scalability in perception for autonomous driving: Waymo

open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[191] P. Sunehag, R. Evans, G. Dulac-Arnold, Y. Zwols, D. Visentin, and B. Coppin. Deep reinforcement learning with attention for slate markov decision processes with high-dimensional states and actions. *arXiv preprint arXiv:1512.01124*, 2015.

[192] R. S. Sutton and A. G. Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

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

[194] H. Tang, R. Houthooft, D. Foote, A. Stooke, O. X. Chen, Y. Duan, J. Schulman, F. DeTurck, and P. Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in neural information processing systems*, pages 2750–2759, 2017.

[195] A. Tavakoli, F. Pardo, and P. Kormushev. Action branching architectures for deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[196] P. Thiyagarajan. A review on cyber security mechanisms using machine and deep learning algorithms. *Handbook of research on machine and deep learning applications for cyber security*, pages 23–41, 2020.

[197] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[198] K. Tran, A. Akella, M. Standen, J. Kim, D. Bowman, T. Richer, and C.-T. Lin. Deep hierarchical reinforcement agents for automated penetration testing. *arXiv preprint arXiv:2109.06449*, 2021.

[199] K. Tran, M. Standen, J. Kim, D. Bowman, T. Richer, A. Akella, and C.-T. Lin. Cascaded reinforcement learning agents for large action spaces in autonomous penetration testing. *Applied Sciences*, 12(21):11265, 2022.

[200] TTCP CAGE Working Group. TTCP CAGE Challenge 3. https://github.com/cage-challenge/cage-challenge-3, 2022.

[201] K. Tuyls and S. Parsons. What evolutionary game theory tells us about multiagent learning. *Artificial Intelligence*, 171(7):406–416, 2007.

[202] K. Tuyls and G. Weiss. Multiagent learning: Basics, challenges, and prospects. *AI Magazine*, 33(3):41–41, 2012.

[203] J. v. Neumann. Zur theorie der gesellschaftsspiele. *Mathematische annalen*, 100(1):295–320, 1928.

[204] H. Van Hasselt. Double Q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621, 2010.

[205] H. Van Hasselt, A. Guez, and D. Silver. Deep Reinforcement Learning with Double Q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[206] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

[207] A. Vezhnevets, Y. Wu, M. Eckstein, R. Leblond, and J. Z. Leibo. Options as responses: Grounding behavioural hierarchies in multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 9733–9742. PMLR, 2020.

[208] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 3540–3549. PMLR, 2017.

[209] F. M. Vincent and D. E. Fryer. Top score in axelrod tournament. *arXiv preprint arXiv:2104.03347*, 2021.

[210] E. Vinitsky, N. Lichtlé, X. Yang, B. Amos, and J. Foerster. Nocturne: a scalable driving benchmark for bringing multi-agent learning one step closer to the real world. *arXiv preprint arXiv:2206.09889*, 2022. URL https://arxiv.org/abs/2206.09889.

[211] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

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

[213] K. Wang, Z. Zou, Y. Shang, Q. Deng, M. Zhao, R. Wu, X. Shen, T. Lyu, and C. Fan. Rl4rs: A real-world benchmark for reinforcement learning based recommender system. *ArXiv*, abs/2110.11073, 2021.

[214] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

[215] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[216] C. J. C. H. Watkins. Learning from delayed rewards. 1989.

[217] E. Wei and S. Luke. Lenient learning in independent-learner stochastic cooperative games. *Journal Machine Learning Research*, 17(84):1–42, 2016.

[218] E. Wei, D. Wicke, and S. Luke. Hierarchical approaches for reinforcement learning in parameterized action space. *arXiv preprint arXiv:1810.09656*, 2018.

[219] R. P. Wiegand. *An analysis of cooperative coevolutionary algorithms*. PhD thesis, Citeseer, 2003.

[220] M. Wołczyk, M. Zając, R. Pascanu, Ł. Kuciński, and P. Miłoś. Continual world: A robotic benchmark for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 34:28496–28510, 2021.

[221] A. Wong, T. Bäck, A. V. Kononova, and A. Plaat. Deep multiagent reinforcement learning: Challenges and directions. *Artificial Intelligence Review*, pages 1–34, 2022.

[222] Z. Wu, C. Shen, and A. Van Den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, 90:119–133, 2019.

[223] Y. Xu, C. Wang, J. Liang, K. Yue, W. Li, S. Zheng, and Z. Zhao. Deep reinforcement learning based decision making for complex jamming waveforms. *Entropy*, 24(10), 2022. ISSN 1099-4300. doi: 10.3390/e24101441. URL https://www.mdpi.com/1099-4300/24/10/1441.

[224] C. Yang, R. WANG, X. Wang, and Z. Wang. A game-theoretic perspective of generalization in reinforcement learning. In *Deep Reinforcement Learning Workshop NeurIPS 2022*.

[225] Y. Yang, R. Tutunov, P. Sakulwongtana, and H. B. Ammar. $\alpha\alpha$-rank: Practically scaling α-rank through stochastic optimisation. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1575–1583, 2020.

[226] L. Yao, Z. Leng, J. Jiang, and F. Ni. Large-scale maintenance and rehabilitation optimization for multi-lane highway asphalt pavement: A reinforcement learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 23(11):22094–22105, 2022. doi: 10.1109/TITS.2022.3161689.

[227] O. Yavanoglu and M. Aydos. A review on cyber security datasets for machine learning algorithms. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 2186–2193, 2017. doi: 10.1109/BigData.2017.8258167.

[228] J. You, R. Ying, and J. Leskovec. Position-aware graph neural networks. In *International conference on machine learning*, pages 7134–7143. PMLR, 2019.

[229] C. Yu and Q. Huang. Curriculum offline reinforcement learning with progressive action space in intelligent healthcare decision-making. *Available at SSRN 4167820*, 2022.

[230] Y. Yu. Towards sample efficient reinforcement learning. In *IJCAI*, pages 5739–5743, 2018.

[231] T. Zahavy, M. Haroush, N. Merlis, D. J. Mankowitz, and S. Mannor. Learn what not to learn: Action elimination with deep reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

[232] T. Zahavy, M. Haroush, N. Merlis, D. J. Mankowitz, and S. Mannor. Learn what not to learn: Action elimination with deep reinforcement learning, 2019.

[233] R. Zebari, A. Abdulazeez, D. Zeebaree, D. Zebari, and J. Saeed. A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. *Journal of Applied Science and Technology Trends*, 1 (2):56–70, 2020.

[234] D. Zha, J. Xie, W. Ma, S. Zhang, X. Lian, X. Hu, and J. Liu. Douzero: Mastering doudizhu with self-play deep reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12333–12344. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/zha21a.html.

[235] A. Zhang, R. McAllister, R. Calandra, Y. Gal, and S. Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

[236] X. Zhao, L. Xia, L. Zhang, Z. Ding, D. Yin, and J. Tang. Deep reinforcement learning for page-wise recommendations. In *Proceedings of the 12th ACM*

Conference on Recommender Systems, RecSys '18, page 95–103, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450359016. doi: 10.1145/3240323.3240374. URL https://doi.org/10.1145/3240323.3240374.

[237] X. Zhao, L. Zhang, Z. Ding, D. Yin, Y. Zhao, and J. Tang. Deep reinforcement learning for list-wise recommendations. *CoRR*, abs/1801.00209, 2018. URL http://arxiv.org/abs/1801.00209.

[238] C. Zhong, M. C. Gursoy, and S. Velipasalar. A deep reinforcement learning-based framework for content caching. In *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE, 2018.