2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC) | 978-1-6654-6483-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/PIMRC56721.2023.10293754

Deep Reinforcement Learning for Downlink Scheduling in 5G and Beyond Networks: A Review

Michael Seguin, Anjali Omer, Mohammad Koosha, Filippo Malandra, Nicholas Mastronarde
University at Buffalo, Department of Electrical Engineering
Email: {mpseguin, anjaliom, mkoosha, filippom, nmastron}@buffalo.edu

***Abstract*—The coexistence of a wide variety of different applications with diverse Quality of Service (QoS) and Quality of Experience (QoE) requirements calls for more sophisticated radio resource scheduling in 5G and beyond (5GB) networks compared to previous generations. To address this challenge, a growing body of research has explored deep reinforcement learning (DRL) to solve the radio resource scheduling problem. In this paper, we review representative literature on the topic of downlink scheduling for 5GB networks using DRL, with emphasis on fine-grained approaches that directly allocate resource blocks (RBs) to user equipments (UEs). We conclude by discussing four ways to improve upon this early-stage research and identify some open problems that must be solved to make DRL a viable solution to the downlink scheduling problem in 5GB networks.**

I. Introduction

With the evolution of cellular networks from 4G to 5G and beyond (5GB), end-user Quality of Service (QoS) and Quality of Experience (QoE) requirements have become more stringent and varied. In [1], 3GPP lists the driving factors for 5G as Virtual Reality (VR), industrial control, Internet of Things (IoT), and ubiquitous on-demand coverage. These key applications can be served by the three main connectivity types in 5G New Radio (NR): Enhanced Mobile Broadband (eMBB), Ultra Low Latency Communication (URLLC), and Massive Machine Type Communication (mMTC) [2].

To meet the challenging requirements introduced by these applications, performance in 5GB mobile networks needs to considerably improve with respect to older generations. In [3], 3GPP focuses on Key Performance Indicators (KPI) for each connectivity type. For example, eMBB aims for peak data rates of 20 Gbps and a downlink user plane latency of 4 ms; URLLC aims for a downlink user plane latency of 0.5 ms; and KPIs for mMTC focus on power-saving and connection density requirements. 6G networks will require at least an order of magnitude improvement in data rates, latency, connectivity density, and energy efficiency compared to 5G [4].

In parallel to these developments, Deep Reinforcement Learning (DRL) has emerged as an effective data-driven approach for solving sophisticated sequential-decision problems with high-dimensional, continuous, or infinite state and action spaces. In 2015, DeepMind used a Deep Q-Network (DQN) to play Atari 2600 games [5]. In 2016, they developed AlphaGo using a combination of supervised learning and Reinforcement Learning (RL) to play the game Go [6]. Three years later, OpenAI developed a DRL algorithm to play Dota 2, which was able to win 99.4% of 7,000 games [7]. Such breakthroughs have driven interest in DRL across many domains, including radio resource scheduling in 5GB networks.

The coexistence of a wide variety of different applications with diverse and ambitious QoS and QoE requirements calls for more sophisticated radio resource scheduling. DRL offers a promising solution to address these challenges because it can: (1) pull more parameters into the decision-making process than traditional scheduling algorithms; (2) consider more complex objective functions comprising multiple (potentially competing) objectives; (3) account for how scheduling decisions affect the immediate and expected future network performance; and (4) enable sub-ms scheduling decisions after training. Consequently, a growing body of literature has focused on the use of DRL to solve the 5G downlink scheduling problem.

Existing DRL approaches for 5G scheduling can be roughly broken into two categories: coarse- and fine-grained approaches [8]. Coarse-grained approaches use DRL to select among several traditional rule-based schedulers to make the scheduling decision in each time slot. In contrast, fine-grained approaches use DRL to directly allocate resource blocks (RBs) to User Equipments (UEs). Scheduling may be performed one RB at a time or jointly across all RBs. Both methods have shown promise for downlink scheduling in 5G cellular networks; however, **we focus on the fine-grained approach in this review** as it has greater potential to exceed the performance of existing schedulers, but poses more challenges due to the exponential number of possible scheduling actions.

The papers in this review were selected as follows. We searched Google Scholar using combinations of keywords "deep reinforcement learning," "downlink scheduling," "4G," and "5G." We identified **32** papers published since **2019** that matched the keywords AND focused on the allocation of RBs. Out of these, we retained **11** papers on **fine-grained 5G downlink scheduling using DRL.** Based on our selection criteria, the review excludes papers that focus on course-grained scheduling, that consider problems beyond RB allocation (beamforming, power control, interference coordination, network slicing, etc.), or use standard RL (e.g., Q-learning).

A. Omer's, M. Koosha's, and N. Mastronarde's work was supported by NSF award #2030157 and F. Malandra's work was supported by NSF award #2105230.

Our contributions are as follows: 1) We review 11 papers published from 2019-2022 that use DRL to solve the fine-grained downlink scheduling problem; 2) We compare these papers in terms of the selected DRL algorithms, the states, actions, and rewards used in their problem formulations, and the unique aspects of each approach; and 3) We identify four directions to improve upon this early stage research and highlight some open problems. Although the reviewed papers largely focus on DRL in 5G networks, our insights hold true for 6G networks whose higher complexity and performance requirements will demand improved DRL algorithms. The remainder of this paper is organized as follows: Section II reviews the 5G resource grid, the radio resource scheduling problem, Markov Decision Processs (MDPs), and DRL; Section III reviews the literature on DRL for downlink scheduling in 5G networks; and Section IV concludes the paper.

II. Preliminaries

A. Resource Grid

Similar to LTE, the physical layer in 5G NR is represented using a *resource grid*, as illustrated in Fig. 1. The resource grid shows the subdivision of resources in time (y-axis) and frequency (x-axis). The smallest units in the figure are the *resource elements (REs)*, each of which spans one subcarrier in the frequency domain and one Orthogonal Frequency-Division Multiplexing (OFDM) symbol in the time domain.

Unlike 4G/LTE, which only supports one subcarrier spacing ($\Delta f = 15$ kHz), 5G NR supports five subcarrier spacings: $\Delta f = 2^{\mu} \cdot 15$ kHz, where $\mu \in \{0, 1, 2, 3, 4\}$ is the so-called *numerology*. Note that numerology $\mu = 0$ corresponds to the subcarrier spacing used in 4G/LTE. In the time-domain, downlink channels are divided into *frames* of 10 ms each. Each frame consists of ten 1 ms *subframes*. Each subframe is further divided into $N_{\text{slot}} = 2^{\mu}$ *slots* with duration $T_{\text{slot}} = \frac{1}{N_{\text{slot}}}$ ms, such that higher numerologies have shorter slot durations.

Finally, the smallest scheduling unit in time and frequency is an RB, which is twelve subcarriers wide and one slot long. The total number of RBs in the resource grid depends on the total system bandwidth and the numerology.

B. Radio Resource Scheduling

As previously mentioned, the number of RBs available to UEs in both uplink and downlink directions is limited by the bandwidth. The problem of allocating RBs to UEs, performed by the gNodeB, is usually referred to as *radio resource scheduling*, and it is proven to be NP-hard [9]. In the remainder of this paper, we will focus our discussion on the downlink radio resource scheduling problem. An illustration of this problem is shown in Fig. 1, where, for example, UE1 is allocated one RB in one slot and UE2 is allocated one RB in another slot. Due to the complexity of this problem and its hard time requirements (a decision needs to be taken each ms), it is typically solved with *heuristic* or *rule-based* algorithms that are based on a comparison of per-UE per-RB metrics. Specifically, in each slot, the jth RB is allocated to

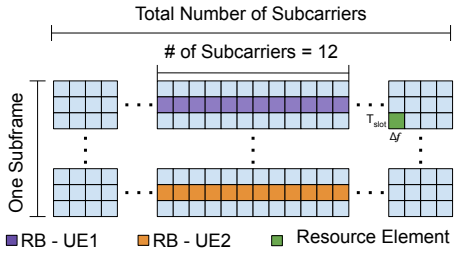

Fig. 1. 5G resource grid and radio resource scheduling

the ith UE if the metric m_{ij} is the largest across all UEs, i.e., $m_{ij} = \max_k m_{kj}$. Each mobile operator can choose the metrics and scheduling policies that work best in their network since 3GPP has intentionally not standardized them in LTE and 5G. Different radio resource scheduling algorithms use different metrics to achieve different goals. An extensive list of downlink schedulers can be found in [10].

While rule-based schedulers can achieve excellent performance in terms of throughput and fairness, DRL based schedulers can potentially optimize networks when UEs have heterogeneous QoS and QoE requirements in terms of not only throughput and fairness, but also Packet Drop Rates (PDRs), buffer delays, and application-specific utility functions.

C. Markov Decision Process

Before we discuss DRL and its application to downlink scheduling, we introduce the concept of an MDP. An MDP is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, R, P, \gamma)$, where: $\mathcal{S}$ is a set of states and $s \in \mathcal{S}$ denotes a state; $\mathcal{A}$ is a set of actions and $a \in \mathcal{A}$ denotes an action; $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function that maps states and actions to a real-valued reward (we will also refer to cost functions, denoted by $C : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$); $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition probability function that defines the probability of transitioning to state $s' \in \mathcal{S}$ after taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$; $\gamma \in [0, 1]$ is a discount factor, which determines the relative importance of immediate and future rewards.

MDPs are used to model sequential-decision problems in which the actions taken in the current state not only affect the immediate reward, but also the subsequent state [11]. The objective of an MDP is to determine an optimal *policy* $\pi : \mathcal{S} \to \mathcal{A}$ that specifies the action to take in each state and maximizes the *expected discounted reward*: i.e.,

$$\max_{\pi} V^{\pi}(s) = \max_{\pi} \mathbb{E}\left[\sum\nolimits_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s = s_0, \pi\right], \quad (1)$$

where V^{π} denotes the *value function* under policy π and the expectation is over the sequence of states. The optimal value function satisfies the following Bellman equation:

$$V^*(s) = \max_{a \in \mathcal{A}} \underbrace{\left\{R(s, a) + \gamma \sum\nolimits_{s' \in \mathcal{S}} P(s'|s, a) V^*(s')\right\}}_{Q^*(s,a)}, \quad (2)$$

where V^* denotes the *optimal value function*, which indicates how good it is to be in each state under the *optimal policy* π^*, and Q^* denotes the *optimal action-value function*, which indicates how good it is to take an arbitrary action a in state

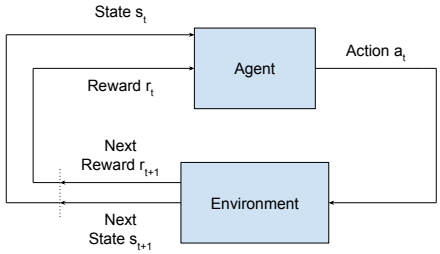

Fig. 2. The agent–environment interaction in an MDP [12]

s and then follow the optimal policy thereafter. Note that $\pi^*(s) = \max_{a\in\mathcal{A}} Q^*(s,a)$. If we consider a cost function, then we replace – in (1) and (2) – the reward function with the cost function and $\max$ with $\min$.

If the state and action spaces are discrete and finite and both the transition probability and reward functions are known, then an MDP can be solved numerically using dynamic programming (e.g., value iteration). However, problems with high-dimensional, continuous, or infinite state and action spaces, or for which the transition probability and reward functions are unknown, cannot be solved in this manner. DRL, which uses Deep Neural Networks (DNNs) to approximate the value function and/or policy, is a promising solution to such problems and has attracted significant attention as a solution to the downlink scheduling problem in 5G networks, which involves high-dimensional state spaces (e.g., comprising the buffer and channel states of all UEs) and high-dimensional action spaces (e.g., determining the allocation of RBs to UEs).

Note that MDPs and DRL differ from classical convex/integer/non-linear optimization methods since MDPs optimize long-term performance (see (1)) rather than a one-shot objective function and DRL is data-driven rather than model-based.

D. Deep Reinforcement Learning

In DRL, an agent interacts with an environment over discrete time steps as illustrated in Fig. 2. In each time step $t \in \{0,1,\ldots\}$, the agent observes its state $s_t \in \mathcal{S}$, takes an action $a_t \in \mathcal{A}$, receives a reward $r_t \in \mathbb{R}$, and transitions to a new state $s_{t+1} \in \mathcal{S}$. The environment is assumed to behave according to an MDP, such that $s_{t+1} \sim P(\cdot|s_t,a_t)$ and $E[r_t] = R(s_t,a_t)$. Each experience tuple (s_t,a_t,r_t,s_{t+1}) is then stored in an *experience replay buffer* $\mathcal{E}$. In 5G scheduling, each time step t typically corresponds to one slot.

We now briefly introduce two of the most popular DRL algorithms used for downlink scheduling: DQN [5] and Deep Deterministic Policy Gradient (DDPG) [13]. In a DQN, we approximate the action-value function using a DNN parameterized by $\boldsymbol{\theta}$ and aim to minimize the following mean squared error (MSE) loss function:

$$L(\boldsymbol{\theta}) = \mathbb{E}_{s^t\sim\rho^\beta,\ a^t\sim\beta,\ r^t}\left[\left(Q(s^t,a^t;\boldsymbol{\theta}) - y^t\right)^2\right], \quad (3)$$

where β is the behavioral policy (which determines the scheduling actions), ρ^β is the discounted state visitation probability (which is a measure of the frequency with which different states are visited), and $y^t = r^t + \gamma Q(s^{t+1},\pi(s^{t+1});\boldsymbol{\theta})$. During training, $L(\boldsymbol{\theta})$ is estimated by sampling mini-batches of experience tuples from the experience replay buffer $\mathcal{E}$.

DQNs are not well-suited for problems with large action spaces because determining the action in each time slot requires maximizing over the action-value function, i.e., $a^t = \arg\max_{a\in\mathcal{A}} Q(s^t,a;\boldsymbol{\theta})$, which is computationally impractical. DDPG, a so-called *actor-critic* method, overcomes this limitation by directly learning a parameterized policy (or *actor*) $\pi(s;\boldsymbol{\phi})$. The action-value function (or *critic*) network $\boldsymbol{\theta}$ is then updated to minimize a similar loss function to (3) and the actor network $\boldsymbol{\phi}$ is updated using a sampled policy gradient (using randomly sampled mini-batches of experience tuples), i.e.,

$$\begin{aligned}\nabla_{\boldsymbol{\phi}} J &\approx \mathbb{E}_{s^t\sim\rho^\beta}\left[\nabla_{\boldsymbol{\phi}} Q(s^t,\pi(s^t;\boldsymbol{\phi});\boldsymbol{\theta})\right] \\ &= \mathbb{E}_{s^t\sim\rho^\beta}\left[\nabla_a Q(s^t,a;\boldsymbol{\theta})|_{a=\pi(s^t;\boldsymbol{\phi})} \nabla_{\boldsymbol{\phi}}\pi(s^t;\boldsymbol{\phi})\right], \quad (4)\end{aligned}$$

which is the gradient of the policy's performance. Although DDPG is designed to solve problems with continuous actions, it can also work with a large number of discrete actions, as in the downlink scheduling problem. In our review, we will see another actor-critic algorithm called Advantage Actor-Critic (A2C), which is a synchronous version of the Asynchronous Advantage Actor-Critic (A3C) algorithm [14].

III. DRL for Downlink Scheduling In 5G

In this section, we review 11 papers published from 2019 to 2022 that are representative of the literature using DRL for fine-grained 5G downlink scheduling. A high-level summary of each paper is provided in Table I, including a brief description of the considered problem, the specific DRL algorithm that the authors use to solve it, and basic information about the underlying MDP (i.e., states, actions, and rewards).

In [15], Wang et al. consider a single-carrier downlink scheduling scenario with saturated traffic conditions, in which a single resource block group (RBG) is allocated to a UE in each scheduling period. They investigate three approaches to solve the problem: direct-learning, dual-learning, and expert learning. Direct-learning is based on a vanilla DDPG algorithm that directly learns from the environment. Dual-learning leverages two independent DDPG agents (say, agents A and B) that are trained alternately, and rewards Agent A when it does better than agent B, and vice versa. Lastly, expert-learning is similar to dual-learning, but one of the DDPG agents is replaced with the PF scheduling algorithm, i.e., the "expert." Direct- and dual-learning achieve better throughput than Proportional Fair (PF), but are less fair. Expert-learning, on the other hand, obtains nearly the same performance as PF.

In [16], Sharma et al. consider downlink scheduling for time-sensitive applications. They define the ith UE's cost as the change in its buffer state from time step t to $t+1$ and the total cost as the sum of this metric across UEs. They solve the problem using vanilla DDPG. Their simulation results show improvements in average delay and fairness compared to Max-Weight [25], PF, and Max-Channel Quality Indicator (CQI) under heterogeneous traffic conditions, achieving up to 55%

TABLE I
COMPARISON OF FINE-GRAINED DRL-BASED DOWNLINK SCHEDULING PAPERS

Ref.	Description	Algorithm	State	Action	Reward/Cost
[15]	Single-carrier DL scheduling with a single RBG and saturated traffic	DDPG-based direct, dual, and expert learning	Instantaneous and average rates for each UE	Scheduled UE on one RBG	*Direct*: weighted sum of throughput and JFI; *Dual/Expert*: competition-based
[16]	DL scheduler for time-sensitive traffic	DDPG	Buffer states and CQI for all UEs	UE scheduled on each RB (jointly for all RBs)	Sum of buffer state changes
[17]	DL scheduling with high priority UEs	DDPG	Channel power fading, sum of resource requests, inverse minimum time-to-timeout for each UE	Fraction RBs allocated to each UE	Weighted sum of sum capacity, sum timeouts, and sum packet rate
[18]	DL scheduler for time-sensitive IoT traffic	K-DDPG	HoL delays and SNRs for each UE	#RBs allocated to each UE	#packets received by UEs
[19]	Joint DL scheduler for eMBB and URLLC traffic	DDPG	SNRs of eMBB/URLLC UEs, capacity of eMBB UEs, URLLC UE traffic arrivals	Fraction RBs allocated to eMBB/URLLC UEs, fraction RBs each eMBB UE loses to URLLC UEs	UCTS: product of sum-SSLs of eMBB/URLLC UEs; SWTS: product of sum-reliability of eMBB/URLLC UEs
[20]	DL scheduling of one RB at a time. Generalizes across the number of UEs.	LSTM/Pointer Network trained with A2C	Throughput, buffer state, JFI, CQI	Scheduled UE on current RB (one RB at a time)	Total throughput and JFI
[21]	DL scheduler that generalizes across changing numbers of UEs and RBs	LSTM/Pointer Network trained with DDPG	CQIs and buffer states for each UE	Scheduled UE on each RB (jointly for all RBs)	Rewards reducing UE buffer states, penalizes scheduling inactive UEs and leaving UEs unscheduled for a long time
[8]	DL scheduler called LEASCH that schedules UEs one RB at a time	DDQN	Eligibility, data rate, and fairness	Scheduled UE on current RB (one RB at a time)	Modified Max-CQI with data rate discounted by resource sharing fairness
[22]	Delay-aware DL scheduler	DQN-based on RNN	Tx rates for each UE on each RB, UE buffer states, UE HoL delays, and UE data arrivals	Scheduled UE on each RB (jointly for all RBs)	Sum-HoL delays
[23]	DL scheduling of one RBG at a time	DQN	Achievable data rate and fairness indicator for each UE	Scheduled UE on current RBG (one RBG at a time)	Product of the scheduled UE's throughput and its PF ratio
[24]	DL scheduling of one RB at a time	A2C	Instantaneous and average rate, spare space in the buffer, HoL delay	Scheduled UE on current RB (one RB at a time)	Weighted sum of throughput, JFI, and PDR

lower delay than the best benchmark (Max-Weight) as the numbers of UEs and RBs increase.

In [17], Gracla et al. propose a DDPG-based downlink scheduler that provides higher QoS to high-priority users, such as emergency vehicles. The algorithm determines the fraction of RBs allocated to each UE and includes a post-processing step to remove inefficient actions. They compare their proposed DDPG-based scheduler against Maximum Throughput (MT), Max-Min-Fair, Delay-Sensitive, DQN-based, and Random scheduling algorithms. In their simulation results, DDPG achieves the highest sum packet rate, highest sum-reward, and fewest emergency vehicle timeouts. However, MT and Delay-Sensitive achieved slightly better capacities and fewer latency violations, respectively, than DDPG.

In [18], Gu et al. focus on downlink scheduling for time-sensitive IoT traffic with the goal of maximizing the throughput of packets that meet certain head-of-line (HoL) delay constraints. They observe that vanilla DDPG converges slowly and achieves poor performance. To overcome its limitations, they propose an improved algorithm called Knowledge-assisted Deep Deterministic Policy Gradient (K-DDPG) that uses a multi-head critic, reward shaping, and importance sampling. They conduct an ablation study that shows that the multi-head critic has the biggest impact on the average and worst-case reward among users. The multi-head critic uses a critic network (that approximates the value function) with one output representing the performance of each UE rather than a single output representing the aggregate performance across all UEs as in vanilla DDPG. This improves learning performance by directing gradient updates towards underperforming UEs. They demonstrate a 30%-50% reduction in packet losses (due to HoL delay violations) compared to Earliest Deadline First and Round-Robin (RR) schedulers.

In [19], Li and Zhang focus on joint scheduling of URLLC and eMBB traffic when URLLC UEs can be scheduled on resources that were already allocated to eMBB UEs (so-called *puncturing*). They investigate two DDPG-based solutions to this problem, namely, system-wide trade-off scheduling (SWTS) and user-centric trade-off scheduling (UCTS), which use different reward functions. SWTS uses a reward function defined based on the reliability of eMBB and URLLC UEs, which is set to 1 if a UE's rate requirement is met, and 0 otherwise. UCTS uses a reward function defined based on the service satisfaction levels (SSLs) of UEs, which denote the fraction of a UE's rate requirement that is met. They determine that SWTS does a better job guaranteeing the performance of eMBB UEs and that it achieves the best trade-off between eMBB and URLLC performance.

All papers that we have reviewed so far use feed forward neural network architectures that do not support varying input sizes. This has two important consequences. First, if the number of UEs varies over time, then the DRL agent needs to be trained to accommodate the maximum number of UEs that could be encountered, and the inputs will need to be padded

with zeros when there are fewer UEs. Second, if the number of UEs is fixed, then new DRL agents will need to be trained to handle networks with different numbers of UEs and RBs. To overcome this limitation, some papers have exploited so-called *pointer networks* [20, 21]. A pointer network can take in a variable length input sequence and select members of the input sequence as its output. In the context of DRL for 5G scheduling, this means that the states of a variable number of UEs can be provided as input to the pointer network, and the output(s) can specify which UE(s) to allocate to the RB(s).

In [20], Al-Tam et al. propose a Long Short-Term Memory (LSTM)/pointer network-based downlink scheduler that they train using A2C. Their solution is designed to schedule one RB at a time and allows the same network to be reused for scheduling each RB. This considerably reduces the network complexity compared to solutions that jointly schedule all RBs in each time step. Their simulation results show that the pointer network can be trained once and used for different numbers of UEs while achieving comparable performance to a PF scheduler and better performance than an RR scheduler in terms of both throughput and fairness.

In [21], Robinson and Kunz propose another LSTM/pointer network-based downlink scheduler, but they train it using DDPG rather than A2C. Their solution is designed to jointly allocate all RBs in each time step. In their simulations, they demonstrate that their pointer network can be trained once and used for a variable number of UEs and RBs while achieving comparable performance to PF and RR in terms of throughput, but worse performance in terms of fairness.

In [8], Al-Tam et al. propose Learn to Schedule (LEASCH), which uses a Double DQN (DDQN) to schedule UEs one RB at a time. Their simulation results show that LEASCH achieves 5–21% better throughput and 2–5% better fairness than PF and RR under different numbers of UEs, different numerologies, and different system bandwidths.

In [22], Zhang et al. propose a DQN-based delay-aware downlink scheduler that is implemented with a Recurrent Neural Network (RNN) and uses the sum-HoL delay as a cost function. They simulate their algorithm in a network with two UEs and an unspecified number of RBs. They compare their algorithm to PF, max-CQI, and RR. Their proposed algorithm achieves 33% lower delay and 59% lower queue lengths than the next best benchmark (PF), with approximately 7% lower throughput than PF and max-CQI. In [23], Walaa et al. also use a DQN-based downlink scheduler, but for the joint optimization of throughput and fairness. Their approach allocates one RB at a time and they evaluate it using the 3GPP 5G NR compliant MATLAB 5G toolbox. Their DQN-based approach outperforms PF and RR in terms of median throughput while achieving slightly lower Jain's fairness index (JFI) under two different numerologies.

In [24], Xu et al. use A2C with the goal of maximizing long-term throughput and fairness, and minimizing the PDR. Similar to [8, 20, 23], they learn a policy to schedule one RBG at a time. They propose to use the same network to make scheduling decisions for each RBG, while iteratively updating the state as RBGs are allocated to UEs. Their solution outperforms PF by 10.54%, 0.3% and 7.64% in terms of throughput, fairness, and PDR, respectively, and achieves similar performance to two genie-aided schedulers that have access to non-causal information about the channel states and packet arrivals. Additionally, they show that it is possible to train a policy on a single RBG and deploy it in a network with ten RBGs; however, this comes at a slight performance penalty compared to directly training the policy on ten RBGs.

IV. Discussion and Conclusion

We have reviewed 11 articles that use DRL for fine-grained downlink scheduling in 5GB networks based on the selection criteria in Section I. While significant steps have been made, we have identified four ways to improve upon this early-stage research and some open problems that must be solved to make DRL a viable solution to the downlink scheduling problem.

Learned policies have limited generalizability: A major hurdle in adopting DRL to solve the downlink scheduling problem is its limited flexibility in dealing with mobility, time-varying traffic arrivals, and varying numbers of UEs and RBs. Policies learned with DRL work best when they are deployed in the same environment in which they were trained and when the environment is stationary. However, if UEs are mobile and have non-stationary traffic arrivals, then their experienced channel conditions, achievable rates, and buffer state transition probabilities will vary over time. Unfortunately, if the scheduling policies learned using DRL cannot deal with these variations, then their performance will quickly degrade below that achieved by simple rule-based schedulers. How to effectively deal with non-stationary environments is an open problem that demands further investigation.

In parallel, the DRL algorithms used in many of the reviewed papers need to be retrained if the numbers of UEs or RBs change. However, among the reviewed papers, two promising approaches to overcome this challenge have been proposed. One option is to schedule one RB at a time and use the same policy to make the scheduling decision at each RB [8, 20, 23, 24]. In this way, it is possible to train a single policy that can be deployed in networks with different numbers of RBs. Second, two papers explore the use of pointer networks that can be trained once and then deployed in settings with different numbers of UEs [20] or UEs and RBs [21]. These approaches warrant deeper investigation to understand the involved tradeoffs in performance and generalizability.

Reward functions must be selected carefully: Many of the reviewed papers consider reward functions that are based on some combination of throughput and fairness. These generally achieve performance on par with rule-based schedulers such as PF. This is an important result because it shows that DRL can effectively learn good scheduling policies. Meanwhile, several of the reviewed papers consider buffer backlogs [16] or HoL delays in their reward functions [22], or enforce delay constraints through specific state variables [17, 18]. These

often show substantial gains over rule-based schedulers with respect to delay-based performance metrics.

Max-Weight scheduling is throughput optimal, while PF can flexibly trade off throughput and fairness. Since these simple myopic algorithms can achieve excellent performance in terms of throughput and fairness, it is difficult to imagine a situation where DRL would *definitively* outperform them with respect to these metrics. On the other hand, the inclusion of delay-based rewards and state variables leads to a sequential-decision problem in which the current scheduling action affects the immediate and future delays experienced by the UEs. We believe that future work should identify and focus on such sequential-decision problems because DRL can solve them better than rule-based schedulers. For example, VR streaming could benefit from DRL due to its stringent delay constraints and inter-packet dependencies that arise during video encoding. In such problems, however, it may still be beneficial to initialize the DRL algorithm to behave like PF as a good starting point (e.g., using the approach in [15]).

Problem formulations are not rigorous: Pioneering research on DRL in the Machine Learning (ML) community often focuses on developing algorithms that can solve a wide range of problems. The algorithms are typically evaluated using well-known benchmark environments, ranging from Atari games to complex control problems. Nokia's Wireless Suite [26] (used in [20]) is the only effort that we are aware of to provide benchmark environments for wireless communications. Detailed mathematical models of the environments are usually not presented and in many cases (e.g., Atari games) an MDP describing the environment cannot be rigorously formulated.

In contrast, each paper in this review considers a unique variation of the downlink scheduling problem with different states, actions, rewards, and transition probabilities (see Table I). Although these problems can all be rigorously formulated as MDPs, there is often insufficient detail to understand exactly what problem is being solved and why DRL is an appropriate solution to it. We argue that future work using DRL for downlink scheduling (and optimizing wireless networks in general) must clearly formulate the underlying MDP and, most importantly, describe how the *next state* depends on the *current state* and *action* in the problem under study. Since MDPs are used to model sequential-decision problems (see Section II-C), demonstrating these dependencies is critical to justify formulating the problem as an MDP. It is also critical to highlight the components of the *reward* and *transition probability functions* that are unknown and to quantify the sizes of the state and action spaces. These details help justify why DRL is needed to solve the problem.

Research is challenging to reproduce and compare: Reproducibility is fundamental to science, and is especially important to push the frontiers of ML-based research. In the context of DRL for 5G downlink scheduling, there are four essential components needed to reproduce results: 1) a complete problem formulation; 2) the simulation environment; 3) the DRL algorithm implementation; and 4) the hyperparameters used to generate results. Without these, it becomes incredibly challenging to replicate results and comparatively evaluate different solutions, ultimately hindering research advancements in the application of DRL in 5GB networks.

REFERENCES

[1] *3GPP TS 22.261, Service requirements for the 5G system*, V18.3.0 ed., 3GPP, 2021, http://bit.ly/3TuoD6F.

[2] S. Yost, "5G—it's not here yet, but closer than you think." EMBEDDED REVOLUTION, 2017.

[3] *3GPP TR 38.913, Study on Scenarios and Requirements for Next Generation Access Technologies*, V16.0.0 ed., 3GPP, 2020.

[4] Z. Zhang et al., "6G wireless networks: Vision, requirements, architecture, and key technologies," *IEEE Veh. Technol. Mag.*, vol. 14, no. 3, pp. 28–41, 2019.

[5] V. Mnih et al., "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[6] D. Silver et al., "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[7] C. Berner et al., "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[8] F. Al-Tam, N. Correia, and J. Rodriguez, "Learn to schedule (LEASCH): A deep reinforcement learning approach for radio resource scheduling in the 5G MAC layer," *IEEE Access*, vol. 8, pp. 108 088–108 101, 2020.

[9] H.-S. Liao, P.-Y. Chen, and W.-T. Chen, "An efficient downlink radio resource allocation with carrier aggregation in LTE-advanced networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 10, pp. 2229–2239, 2014.

[10] F. Capozzi, G. Piro, L. A. Grieco, G. Boggia, and P. Camarda, "Downlink packet scheduling in LTE cellular networks: Key design issues and a survey," *IEEE Commun. Surveys Tuts.*, vol. 15, pp. 678–700, 2012.

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

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

[13] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[14] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in *Intl. Conf. Machine Learning*, 2016, pp. 1928–1937.

[15] J. Wang, C. Xu, Y. Huangfu, R. Li, Y. Ge, and J. Wang, "Deep reinforcement learning for scheduling in cellular networks," in *11th Intl. Conf. Wireless Commun. and Signal Process. (WCSP)*, 2019, pp. 1–6.

[16] N. Sharma et al., "Deep reinforcement learning for delay-sensitive LTE downlink scheduling," in *IEEE Intl. Symp. Pers., Indoor, Mobile Radio Commun.*, 2020, pp. 1–6.

[17] S. Gracla, E. Beck, C. Bockelmann, and A. Dekorsy, "Learning resource scheduling with high priority users using deep deterministic policy gradients," in *IEEE Intl. Conf. Commun.*, 2022, pp. 4480–4485.

[18] Z. Gu et al., "Knowledge-assisted deep reinforcement learning in 5G scheduler design: From theoretical framework to implementation," *IEEE J. Sel. Areas Commun.*, 2021.

[19] J. Li and X. Zhang, "Deep reinforcement learning-based joint scheduling of eMBB and URLLC in 5G networks," *IEEE Wireless Commun. Lett.*, vol. 9, no. 9, pp. 1543–1546, 2020.

[20] F. Al-Tam, A. Mazayev, N. Correia, and J. Rodriguez, "Radio resource scheduling with deep pointer networks and reinforcement learning," in *IEEE Intl. Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2020, pp. 1–6.

[21] A. Robinson and T. Kunz, "Downlink scheduling in LTE with deep reinforcement learning, LSTMs and pointers," in *IEEE Mil. Commun. Conf.*, 2021, pp. 763–770.

[22] T. Zhang, S. Shen, S. Mao, and G.-K. Chang, "Delay-aware cellular traffic scheduling with deep reinforcement learning," in *IEEE Global Commun. Conf.*, 2020, pp. 1–6.

[23] W. AlQwider, T. F. Rahman, and V. Marojevic, "Deep Q-Network for 5G NR downlink scheduling," in *IEEE Intl. Conf. Commun. Workshops*, 2022, pp. 312–317.

[24] C. Xu et al., "Buffer-aware wireless scheduling based on deep reinforcement learning," in *IEEE Wireless Commun. and Netw. Conf.*, 2020.

[25] L. Tassiulas and A. Ephremides, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Trans. Inf. Theory*, vol. 39, no. 2, pp. 466–478, 1993.

[26] Nokia, "Wireless suite," 2021, https://github.com/nokia/wireless-suite.