Deep Reinforcement Learning for Delay-Sensitive LTE Downlink Scheduling

Nikhilesh Sharma*, Sen Zhang*, Someshwar Rao Somayajula Venkata*, Filippo Malandra*,
Nicholas Mastronarde*, and Jacob Chakareski†

*Dept. Electrical Engineering, U. Buffalo, †Ying Wu College of Computing, New Jersey Institute of Technology

***Abstract*—We consider an LTE downlink scheduling system where a base station allocates resource blocks (RBs) to users running delay-sensitive applications. We aim to find a scheduling policy that minimizes the queuing delay experienced by the users. We formulate this problem as a Markov Decision Process (MDP) that integrates the channel quality indicator (CQI) of each user in each RB, and queue status of each user. To solve this complex problem involving high dimensional state and action spaces, we propose a Deep Reinforcement Learning based scheduling framework that utilizes the Deep Deterministic Policy Gradient (DDPG) algorithm to minimize the queuing delay experienced by the users. Our extensive experiments demonstrate that our approach outperforms state-of-the-art benchmarks in terms of average throughput, queuing delay, and fairness, achieving up to 55% lower queuing delay than the best benchmark.**

I. Introduction

The rise of Internet based multimedia applications has massively driven the evolution of cellular networks [1–3]. Many such applications have stringent quality-of-service (QoS) requirements, such as low latency and high throughput. The Long Term Evolution (LTE) specifications introduced by 3GPP [4] have served as a guideline to address these requirements. With an increasing number of such applications and tightening QoS requirements, a common characteristic in 4G/5G networks, it is imperative for any scheduling framework to make fast (sub-millisecond) and accurate scheduling decisions, and fairly allocate the available resources. Therein lies a need to study such resource allocation algorithms.

We study an LTE downlink scheduling system, illustrated in Figure 1, where a Deep Reinforcement Learning (RL) algorithm is employed at the base station (BS) to allocate resources to minimize the sum-delay experienced by the users. We assume data arrivals and channel dynamics to be unknown to the users and the scheduler. Our contributions are as follows:

- We formulate the LTE downlink scheduling problem as a Markov Decision Process (MDP).
- We utilize the Deep Deterministic Policy Gradient (DDPG) algorithm to learn an optimal scheduling policy that minimizes the users' sum-delay. After extensive offline training, the resulting policy can make near-optimal scheduling decisions in 210 – 370 μs.
- We show using extensive simulation results that the proposed Deep RL based resource allocation algorithm outperforms state-of-the-art benchmark algorithms across diverse performance metrics.

The work of N. Sharma and N. Mastronarde was supported in part by the National Science Foundation (NSF) under Award ECCS-1711335. The work of J. Chakareski was supported in part by the NSF under Awards CCF-1528030, ECCS-1711592, CNS-1836909, and CNS-1821875.

The rest of the paper is organized as follows. We review related work in Section II, present our LTE system model in Section III, formulate our delay minimization problem in Section IV, describe our proposed DDPG-based algorithm in Section V, present our numerical results in Section VI, and present the conclusions of this research in Section VII.

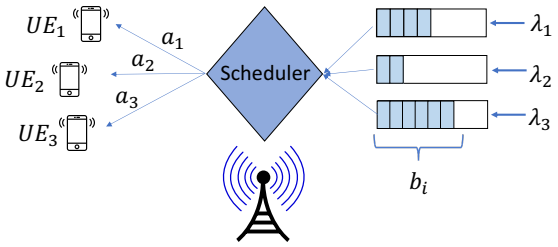


Fig. 1: System Model. λ_i, b_i, and a_i represent the arrival rate, buffer status, and allocated bits, respectively, for UE i.

II. Related Work

In practical networks, the stochastic processes governing the channel dynamics and data arrivals are unknown a priori. This drives the need for learning scheduling policies. In this regard, RL [5, 6] is a widely used tool. In traditional communications scenarios, RL is often used where the dynamics of the system are unknown, with Q-learning [7] being the most widely used RL technique. For instance, in [8], Blasco et al. propose using Q-learning to maximize the throughput of an energy harvesting transmitter. In their work, Tabrizi et al. [9], use Q-learning to develop an algorithm for making dynamic handoff decisions, which learns the optimal handoff policy from the user's past experience and performs close to an optimal oracle algorithm. Other studies use function approximation and Bayesian RL techniques to develop algorithms for minimizing the queuing delay, and maximizing the throughput, respectively [10–12].

This growing push towards RL systems has been supported by advances in deep neural networks, which are shown to be universal function approximators [13]. RL techniques taking advantage of these advances are collectively known as Deep RL. Among Deep RL techniques, the most popular is the Deep Q-Network (DQN) [14], which has been used in various wireless communications applications [15]. Challita et al. [16] use DQN for proactive resource allocation in small BSs and show that the proposed approach permits achieving a mixed-strategy Nash equilibrium. The authors in [17] use DQN with a Long Short-Term Memory (LSTM) neural network to reduce the latency in uplink scheduling in Mission Critical Devices, and show that their approach performs competitively against a conventional Q-learning based latency-minimizing approach.

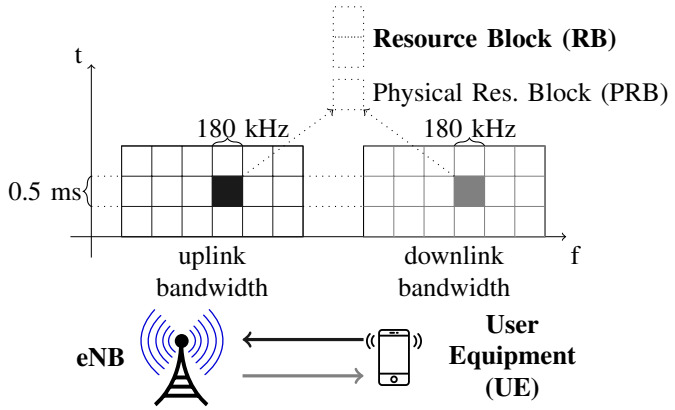


Fig. 2: LTE Resource Grid.

While DQN works well with problems involving high-dimensional observation spaces, it can only handle low-dimensional and small action spaces. This makes it infeasible for problems characterized by either high-dimensional action spaces, or by a large number of actions, such as the one we consider. Due to these shortcomings, we consider the DDPG algorithm [18] whose Actor-Critic based framework makes it suitable for problems involving continuous or large action spaces, such as the LTE downlink scheduling problem.

III. LTE System Model

An LTE downlink employs Orthogonal Frequency Division Multiplexing (OFDM) together with advanced antenna techniques and Adaptive Modulation and Coding (AMC) to achieve better spectral efficiency, enabling the User Equipments (UEs) to transmit more data at lower cost per bit. OFDM is a multi-carrier transmission scheme providing benefits of robustness against frequency selective fading. The available bandwidth is divided into multiple narrow sub-carriers and data is transmitted on these sub-carriers in parallel streams. Each sub-carrier can possibly use a different modulation scheme, ranging from BPSK to 64-QAM.

A. LTE Resource Allocation

Resource allocation in LTE is done jointly across the time/frequency domains. In the time-domain, the downlink channels are divided into *Frames* of 10 ms each. Each frame further consists of 10 *Subframes* of 1 ms each. Each subframe interval is referred to as a *Transmission Time Interval* (TTI).

In the frequency domain, the total available system bandwidth is divided into sub-channels of 180 kHz, with each sub-channel comprising 12 equally spaced sub-carriers of 15 kHz each. A time-frequency radio resource spanning over a 0.5 ms slot in the time domain and a 180 kHz sub-channel in the frequency domain is called a *Resource Block* (RB). Each RB carries OFDM symbols and the number of symbols per RB depends on the used cyclic prefix. The number of resource blocks in the available bandwidth is called the *Resource Grid*, as shown in Fig. 2. The number of RBs in a resource grid depends on the bandwidth, with RBs ranging from 6 to 100 for bandwidth values between 1.4 MHz and 20 MHz.

B. LTE Downlink Scheduling Framework

Multi-user scheduling is one of the main features in LTE systems, which works by distributing available resources among active users in order to satisfy their QoS requirements.

Since the data channel is shared among users, portions of the spectrum should be distributed every TTI (smallest unit in time domain which can be allocated to UEs) among them. Packet schedulers are deployed at the BS or *Evolved NodeB* (eNB) and work with a granularity of one TTI and one RB in the time and frequency domains, respectively.

Resource allocation for each UE is usually based on a comparison of per-RB metrics. For every decision interval, the k-th RB is allocated to the i-th user if the value of its metric $m_{i,k}$ is the largest one, i.e., if it satisfies,

$$m_{i,k} = \max_j \{m_{j,k}\}. \tag{1}$$

These metrics denote the transmission priority of each user on a specific RB. There are various parameters on which to base the transmission priority, such as:

- **Channel Quality:** Reported *Channel Quality Indicator* (CQI) values can be used to allocate resources to users experiencing better channel conditions
- **Resource Allocation History:** Information about the past achieved performance can be used to improve fairness
- **Queue status:** Status of transmission queues.

C. Considered System Model

We consider an LTE downlink scheduling model, illustrated in Fig. 1, in which a base station allocates M RBs to N requesting UEs. The UEs are assumed to be running delay-sensitive multimedia applications and the BS allocates resources every TTI. The BS monitors the buffer status of all UEs and the channel fading state in each RB, and decides what RBs should be assigned to which UEs. The scheduler then schedules the transmission of $a_i \in \mathcal{A}$ bits from the i-th UE, determined by the *Transport Block Size* (TBS) table [4].

Channel Model: We consider an Urban Macro (UMa) pathloss model as defined in [4]. The pathloss experienced by the UE at a distance d from the BS is given by,

$$PL(d) = 128.1 + 37.6 \times \log_{10}(d), \tag{2}$$

The channel state of UE i in RB k at time t, characterized by its Signal to Interference plus Noise Ratio (SINR), is denoted by $h^t_{ik} \in \mathcal{S}_h$. The combined channel state of UE i in all RBs is given by the vector $\mathbf{h}^t_i = (h^t_{i1}, \ldots, h^t_{iM}) \in \mathcal{S}^M_h$, and the combined state of all UEs in all RBs is given by the matrix $\mathbf{H}^t$. Each channel is also assumed to undergo log-normal fading. Note that we ultimately map the SINR of each channel to a 4-bit CQI, which in turn maps to a modulation and coding scheme (MCS) using LTE's CQI table.

Buffer Model: The i-th UE's data buffer status at time t is denoted by $b^t_i \in \mathcal{S}_b = \{0, 1, ...\}$ (bits). The combined buffer state of all UEs at time t is given by the vector $\mathbf{b}^t = (b^t_1, \ldots, b^t_N) \in \mathcal{S}^N_b$. We assume that new data arrivals are independently and identically distributed (i.i.d.). The average

arrival rate of the i-th UE is denoted by λ_i (bits/s), new bits arriving in time slot t are denoted by l_i^t, and the UE's buffer state in time slot $t+1$ is given by the following Lindley's recursive equation,

$$b_i^{t+1} = \max\left(b_i^t - a_i^t(\mathbf{b}^t, \mathbf{H}^t), 0\right) + l_i^t, \tag{3}$$

where $a_i^t(\mathbf{b}^t, \mathbf{H}^t)$ denotes the number of bits scheduled for transmission using the TBS table. Given its arrival distribution P_i^l and the scheduling action a_i, the probability that UE i transitions from buffer state b_i to b_i' is,

$$P_i^b(b_i'|b_i, a_i) = \mathbb{E}_l\left[\mathbb{I}_{\{b_i' = \max(b_i - a_i, 0) + l_i\}}\right], \tag{4}$$

where $\mathbb{I}_{\{\}}$ is an indicator function that is set to 1 when $\{\cdot\}$ is true, and is set to 0 otherwise. UE i's state is denoted by $s_i \triangleq (b_i, \mathbf{h}_i) \in \mathcal{S}_i = \mathcal{S}_b \times \mathcal{S}_h^M$ and the system state by $s = (s_1, \ldots, s_N) \in \Pi_{i=1}^N \mathcal{S}_i$.

IV. Problem Formulation

Let $\pi : \mathcal{S} \to \mathcal{A}$ denote a *policy* that maps states to scheduling actions. The objective of the LTE downlink scheduling problem is to determine the optimal policy π^* that minimizes the average sum of queuing delays (sum-delay) across the UEs. Since the scheduling decisions at the current time affect the present *and* future delays experienced by the UEs, we formulate the delay-sensitive LTE downlink scheduling problem as a MDP [19].

We define a *buffer cost* to penalize large queue backlogs. Formally, we define UE i's buffer cost in time slot t given its state s_i^t and the scheduling action a_i^t as the change in its buffer state from time t to $t+1$: i.e.,

$$c_i^t(s_i^t, a_i^t) = \left[\max\left(b_i^t - a_i^t, 0\right) + l_i^t\right] - b_i^t, \tag{5}$$

where the term in square brackets is equal to b_i^{t+1} from (3). Based on the arguments in [20, Section II.C], minimizing the long-term average of (5) is equivalent to minimizing the UE's delay. We define the total cost incurred in time slot t as the sum of costs incurred by the UEs: i.e.,

$$c^t(s^t, a^t) = \sum\nolimits_{i=1}^{N} c_i^t(s_i^t, a_i^t), \tag{6}$$

where $s_t = (s_1^t, \ldots, s_N^t)$ and $a_t = (a_1^t, \ldots, a_N^t)$ are the joint state and action, respectively.

We now introduce the *value function*, $V^\pi(s)$, which estimates how good (or bad) it is to be in each state, while following the policy π. We define $V^\pi(s)$ as follows,

$$V^\pi(s) = \mathbb{E}\left[\sum\nolimits_{t=0}^{\infty} (\gamma)^t c^t(s^t, \pi(s^t)) | s = s^0\right], \tag{7}$$

where $\gamma \in [0, 1)$ is a discount factor; $(\gamma)^t$ denotes the discount factor to the t-th power; and the expectation is taken over the sequence of states, governed by the controlled Markov chain with transition probabilities $P(s'|s, a) = \Pi_{i=1}^N P^b(b_i'|b_i, a_i)\Pi_{j=1}^M P^h(h_{ij}'|h_{ij})$. We can rewrite $V^\pi(s)$ recursively by taking advantage of the one-step transition probability function to represent the expected future costs:

$$V^\pi(s) = c(s, \pi(s)) + \gamma \sum\nolimits_{s' \in \mathcal{S}} P(s'|s, \pi(s)) V^\pi(s').$$

Then, the scheduling problem's objective is to determine the scheduling policy that solves the following optimization:

$$\min\nolimits_{\pi \in \Pi} V^\pi(s), \tag{8}$$

where Π denotes the set of all possible policies. The optimal solution to (8) satisfies the Bellman equation, $\forall s \in \mathcal{S}$:

$$\begin{aligned} V^*(s) &= \min\nolimits_{a \in \mathcal{A}} \left\{ c(s, a) + \gamma \sum\nolimits_{s' \in \mathcal{S}} P(s'|s, a) V^*(s') \right\}, \\ &\triangleq \min\nolimits_{a \in \mathcal{A}} Q^*(s, a), \end{aligned} \tag{9}$$

where $V^*(s)$ is the optimal *value function*, and $Q^*(s, a)$ is the optimal *action-value function* denoting the value of taking action a in state s and then following the optimal policy thereafter. The optimal policy $\pi^*(s)$, which gives the optimal action to take in each state, can be determined by taking the action that minimizes the right-hand side of (9).

If there are a finite number of states, then in principle dynamic programming techniques such as value iteration or policy iteration can be used to compute $V^*(s)$ and $\pi^*(s)$ [5]. However, in our problem, we assume that the buffer sizes are infinite and, even if we did not, the number of states and actions is enormous. For example, if there are N UEs and M RBs, and each UE has 10 possible buffer states and there are 16 possible channel states per UE per RB, then there are a total of $10^N \times 16^{N \times M}$ possible states and N^M possible scheduling actions. Hence, we resort to a deep RL to solve the problem.

V. DDPG based LTE Downlink Scheduler

The DDPG algorithm we use [18] is a deep neural network (DNN) adaptation of the off-policy Deterministic Policy Gradient (DPG) algorithm developed by Silver, et. al [21]. It is designed to solve problems involving continuous, high-dimensional, and large action spaces since it is based on an Actor-Critic framework. It has also been shown that DDPG can perform well in problems involving large but discrete action spaces [22], such as the one we consider here.

The Actor-Critic framework of DDPG comprises two parts, an actor function that chooses what actions to take, and a critic function that evaluates the actor's selections. The actor function $\mu(s|\theta^\mu)$, a DNN parameterized by θ^μ, specifies the current policy by deterministically mapping states to a specific action, while the critic function $Q(s, a|\theta^Q)$, a DNN parameterized by θ^Q, is learned using the Bellman equation and gives feedback on the selected action.

The actor network is updated by taking a gradient of the expected return J, similar to (7), with respect to the parameters θ^μ. We formally define this operation as:

$$\nabla_{\theta^\mu} J \approx \mathbb{E}_{s^t \sim \rho^\beta}\left[\nabla_{\theta^\mu} Q(s, a|\theta^Q)|_{s=s^t, a=\mu(s^t|\theta^\mu)}\right]. \tag{10}$$

The critic network, which learns the action-value function, is updated by minimizing the Mean Squared Error (MSE) loss,

$$L(\theta^Q) = \mathbb{E}_{s^t \sim \rho^\beta, a^t \sim \beta, c^t \sim E}\left[\left(Q(s^t, a^t|\theta^Q) - y^t\right)^2\right], \tag{11}$$

where β is the behavior policy, and,

$$y^t = c(s^t, a^t) + \gamma Q(s^{t+1}, \mu(s^{t+1})|\theta^Q), \tag{12}$$

denotes the target value.

Convergence of DNN approximators is not guaranteed. Still, inspired by the success of DQN [14], DDPG uses several modifications to its architecture to use DNN function approximation to learn in large state and action spaces.

Similar to DQN, to address the problem of sample correlation, DDPG uses a *replay buffer*, which is a finite sized cache used to store transition tuples (s^t, a^t, c^t, s^{t+1}). At each time step, the actor and critic are updated by sampling a mini-batch of stored transitions from the replay buffer. This allows the algorithm to learn from a set of uncorrelated transitions.

DDPG uses *target networks* to address the problem of divergence caused by using the same Q-network for updating the network and the target value y^t. This is addressed by creating copies of the actor and critic networks, and then updating the copies slowly, resulting in improved stability.

Finally, to address the problem of exploration, DDPG uses an exploration policy by adding noise sampled from a noisy process $\mathcal{N}$ to the actor policy, $\mu'(s_t) = \mu(s_t|\theta_t^\mu) + \mathcal{N}$, where $\mathcal{N}$ can be chosen to suit the environment. The complete description of the algorithm is presented in [18].

Although DDPG mostly works in continuous action spaces, it has been shown to work well with discrete action spaces with slight modifications [22]. We adopt similar practices to adapt the DDPG algorithm to the LTE scheduling problem.

State normalization: Recall that UE i's state, $s_i \triangleq (b_i, \mathbf{h}_i) \in \mathcal{S}_i$, depends on its buffer state s_i and its CQI in each RB, represented by the vector $\mathbf{h}_i$. DDPG accepts states of all the UEs as its input. All buffer and CQI values in a *minibatch* of samples are normalized to $[-1, 1]$ using Batch Normalization for improved performance [18].

Action scaling and quantization: The DDPG algorithm has been shown to work effectively in problems involving large but discrete action spaces such as the problem we consider. For M RBs, the actor network outputs a vector of size $[1 \times M]$. Each entry of this vector is a normalized value between -1 and 1. Inspired by the approach in [22], we then scale these values to the range $[1, N]$ as follows:

$$\Big\lceil \big(\frac{\mathbf{y}+1}{2} \times (N-1)\big) + 1 \Big\rceil, \tag{13}$$

where $\mathbf{y} \in [-1, 1]^M$ is the vector output of the actor network, N is the number of UEs, and $\lceil \cdot \rceil$ denotes the ceiling operator.

VI. Experimental Results

We now present our simulation results where we compare the performance of our proposed DDPG based LTE downlink scheduler against three well known scheduling algorithms: Max CQI, Proportional Fair (PF), and Max-Weight.

Max CQI [23]: The Max CQI scheduler aims to maximize the eNB's throughput by opportunistically allocating an RB to the user with the best CQI (or achievable rate) in that RB. To implement Max CQI, we use $R_j(k,t)$, the rate of UE j in RB k at time t, as the metric $m_{j,k}$ in (1). The Max CQI scheduler is found to be not fair in practice.

Proportional Fair (PF) [24]: The PF scheduler operates by scheduling a user when its instantaneous channel quality is high relative to its average channel condition over time. This approach has been found to be throughput fair in practice. The metric $m_{j,k}$ from (1), used to allocate RB k at time t is,

$$m_{j,k} = R_j(k,t)/T_j(t), \tag{14}$$

where $R_j(k,t)$ is the achievable rate, and $T_j(t)$ is the past throughput achieved by UE j, calculated as an exponential moving average of its throughput up to time t.

Max-Weight [25]: The Max-Weight scheduler considers the queue (buffer) length along with the achievable rate to allocate RBs. It is known to be throughput optimal [25]. We use a buffer weight along with the achievable rate $R_j(k,t)$ to establish the Max-Weight metric:

$$m_{j,k} = R_j(k,t) \times \Big(B_j(t) / \sum\nolimits_{j'=1}^{N} B_{j'}(t) \Big), \tag{15}$$

where $B_j(t) = b_j^t$ is the buffer state of UE j at time t. This ensures that the relative buffer states and achievable rates are considered when allocating the RB.

TABLE I: Simulation Parameters

Notation	Value	Notation	Value
num UEs	{6, 20, 30}	Cell Radius	0.375 km
Interf. BSs	6	Noise PSD	-174 dBm/Hz
numEpochs	30	Bandwidth	{1.4, 10, 20} MHz
TTI, τ	1 ms	RB Duration	0.5 ms
num PRBs	{10, 50, 100}	Log-Norm. mean	0
numSlots	50000	Arrival Rate	50 or 10–100 Kbps
Tx. Power	46 dBm	Log-Norm. Std. Dev.	8

We developed an LTE scheduling simulator in Python. In our simulations, we consider a hexagonal grid with 6 interfering BSs and a central BS that schedules resources to the UEs in its coverage under three different LTE scenarios: 6 UEs/10 RBs, 20 UEs/50 RBs, and 30 UEs/100 RBs. Each scenario considers two different settings: *homogeneous* (UEs all have 50 Kbps arrival rates) and *heterogeneous* arrivals (UEs have 10–100 Kbps arrival rates). We assume *Poisson arrivals* for both settings, thus accounting for the possibility of bursty arrival traffic. Table I lists the key simulation parameters. We implemented the DDPG algorithm as described in [18] and Section V. The actor and critic networks are *fully-connected* DNNs with 2 hidden layers, comprising 400 and 300 neurons, respectively, with ReLU activation. The algorithm is trained using samples collected by running the simulator for 200 epochs of 10,000 time slots each with a minibatch of 32 samples. Under both homogeneous and heterogeneous arrivals, the total training times for the different LTE scenarios were 6.5 hours, 34 hours and 83 hours, respectively, on a Dell 40-core "Cascade Lake" GPU node with 192 GB RAM and 32 GB video memory.

We compare the performance of our proposed DDPG based LTE downlink scheduler against the benchmarks described above, across four metrics: average throughput (*thrpt.*), average queuing delay, average throughput fairness, and average delay fairness. Fairness is calculated using Jain's Fairness Index [26], which quantifies how fairly resources are distributed among users (an index closer to 1 indicates a fairer allocation). All results are averaged over 30 epochs of 50,000 time slots each, and were also computed on the Dell GPU node.

Table II presents the performance of different algorithms under the scenarios listed above. Under homogeneous arrivals,

DDPG performs very close to Max-Weight, which was proved to be delay-optimal under homogeneous arrivals in [25]. Moreover, DDPG outperforms the benchmark algorithms in all other metrics. Under heterogeneous arrivals, the DDPG algorithm outperforms the benchmarks in all scenarios, achieving up to 15% higher throughput and 55% lower queuing delays than the best benchmark. We generally see more improvement as we scale up the number of UEs and RBs.

We plot these results for homogeneous and heterogeneous arrivals in Fig. 3 and4 respectively with error bars representing 95% confidence intervals. Max CQI predictably performs the worst since it is not fair, and thus leads to high delays and low actual throughput. We can see from Fig. 3b and 3d that the DDPG algorithm outperforms the other algorithms in terms of fair resource allocation. The general trends for heterogeneous arrivals are similar to those for homogeneous arrivals, save for two key differences. First, the average fairness values are lower, which is due to having more variation in arrival rates across users. Second, from Fig. 4c, DDPG outperforms Max-Weight even for average delay. This is most pronounced in the case of 30 UEs and 100 RBs, where the DDPG algorithm performs almost 55% better than Max-Weight.

Feasibility: Considering the constraint that the algorithm has to take an action every TTI, the proposed algorithm performs well in that regard. Even though the algorithm takes long to train for large networks, it only takes between $210\mu s$ – $370\mu s$ with an average of $270\mu s$ to schedule an action, thus, making it suitable for real-time scenarios.

TABLE II: Performance of LTE scheduling algorithms

6 UEs, 10 RBs	Homogeneous Arrivals				Heterogeneous Arrivals			
	Max CQI	PF	Max-Wt.	DDPG	Max CQI	PF	Max-Wt.	DDPG
Thrpt. (Mbps)	0.160	0.211	0.276	**0.285**	0.160	0.209	0.282	**0.289**
Thrpt. Fairness	0.612	0.987	0.994	**0.999**	0.558	0.884	0.915	**0.942**
Delay (s)	0.721	0.405	**0.162**	0.184	0.663	0.369	0.161	**0.127**
Delay Fairness	0.602	0.670	0.846	**0.894**	0.588	0.627	0.718	**0.761**
20 UEs, 50 RBs	Homogeneous Arrivals				Heterogeneous Arrivals			
	Max CQI	PF	Max-Wt.	DDPG	Max CQI	PF	Max-Wt.	DDPG
Thrpt. (Mbps)	0.298	0.847	0.948	**0.977**	0.432	0.939	1.057	**1.202**
Thrpt. Fairness	0.319	0.957	0.981	**0.992**	0.408	0.821	0.887	**0.925**
Delay (s)	1.961	0.801	**0.251**	0.286	1.881	0.726	0.153	**0.105**
Delay Fairness	0.533	0.635	0.803	**0.908**	0.515	0.609	0.759	**0.846**
30 UEs, 100 RBs	Homogeneous Arrivals				Heterogeneous Arrivals			
	Max CQI	PF	Max-Wt.	DDPG	Max CQI	PF	Max-Wt.	DDPG
Thrpt. (Mbps)	0.437	1.088	1.210	**1.293**	0.506	1.117	1.285	**1.334**
Thrpt. Fairness	0.359	0.963	0.979	**0.986**	0.427	0.837	0.879	**0.921**
Delay (s)	2.184	0.852	**0.288**	0.339	2.278	0.785	0.239	**0.109**
Delay Fairness	0.585	0.674	0.897	**0.905**	0.533	0.653	0.795	**0.863**

VII. Conclusion

We formulated the LTE downlink scheduling problem as an MDP and proposed a Deep RL algorithm to solve it. The algorithm can learn near-optimal scheduling policies that minimize the UEs' sum-delay, without requiring the scheduler to have upfront knowledge of the stochastic data arrivals and channel dynamics. We compared the proposed scheduling algorithm to well-known LTE-downlink scheduling benchmarks and we observed an improvement in performance both in terms of fairness and delay. As future work, we plan to adapt this algorithm to the more challenging problems of LTE uplink scheduling and 5G uplink/downlink scheduling.

References

[1] J. Chakareski, "UAV-IoT for next generation virtual reality," *IEEE Trans. Image Processing*, vol. 28, no. 12, pp. 5977–5990, Dec. 2019.

[2] J. Chakareski and P. Frossard, "Distributed collaboration for enhanced sender-driven video streaming," *IEEE Trans. Multimedia*, Aug. 2008.

[3] X. Corbillon, A. Devlic, G. Simon, and J. Chakareski, "Viewport-adaptive navigable 360-degree video delivery," in *Proc. Int'l Conf. Communications*. Paris, France: IEEE, May 2017.

[4] 3GPP TR 25.913, "Requirements for Evolved UTRA (E-UTRA) and Evolved UTRAN (E-UTRAN)," 2009.

[5] R. Sutton and A. Barto, *Reinforcement learning: An introduction*, 2nd ed. MIT Press Cambridge, 2018.

[6] N. Mastronarde and M. van der Schaar, "Joint physical-layer and system-level power management for delay-sensitive wireless communications," *IEEE Trans. Mobile Comput.*, vol. 12, no. 4, 2013.

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

[8] P. Blasco, D. Gunduz, and M. Dohler, "A learning theoretic approach to energy harvesting communication system optimization," *IEEE Trans. Wireless Commun.*, vol. 12, no. 4, pp. 1872–1882, 2013.

[9] H. Tabrizi, G. Farhadi, and J. Cioffi, "Dynamic handoff decision in heterogeneous wireless systems: Q-learning approach," in *Proc. Int'l Conf. on Communications (ICC)*. IEEE, 2012, pp. 3217–3222.

[10] A. Ortiz, H. Al-Shatri, X. Li, T. Weber, and A. Klein, "Reinforcement learning for energy harvesting point-to-point communications," in *Proc. IEEE Int'l Conf. Communications*, 2016.

[11] N. Sharma, N. H. Mastronarde, and J. Chakareski, "Accelerated structure-aware reinforcement learning for delay-sensitive energy harvesting wireless sensors," *IEEE Trans. on Signal Process.*, 2020.

[12] Y. Xiao, Z. Han, D. Niyato, and C. Yuen, "Bayesian reinforcement learning for energy harvesting communication systems with uncertainty," in *Proc. IEEE ICC*, 2015, pp. 5398–5403.

[13] K. Hornik, M. Stinchcombe, H. White *et al.*, "Multilayer feedforward networks are universal approximators." *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[14] 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*, vol. 518, no. 7540, pp. 529–533, 2015.

[15] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[16] U. Challita, L. Dong, and W. Saad, "Proactive resource management for LTE in unlicensed spectrum: A deep learning perspective," *IEEE Transactions on Wireless Communications*, vol. 17, no. 7.

[17] M. Elsayed and M. Erol-Kantarci, "Deep reinforcement learning for reducing latency in mission critical services," in *Proc. Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–6.

[18] 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.

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

[20] M. J. Neely, "Stochastic optimization for Markov modulated networks with application to delay constrained wireless scheduling," in *Proc. IEEE Conf. Decision and Control*, 2009, pp. 4826–4833.

[21] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," 2014.

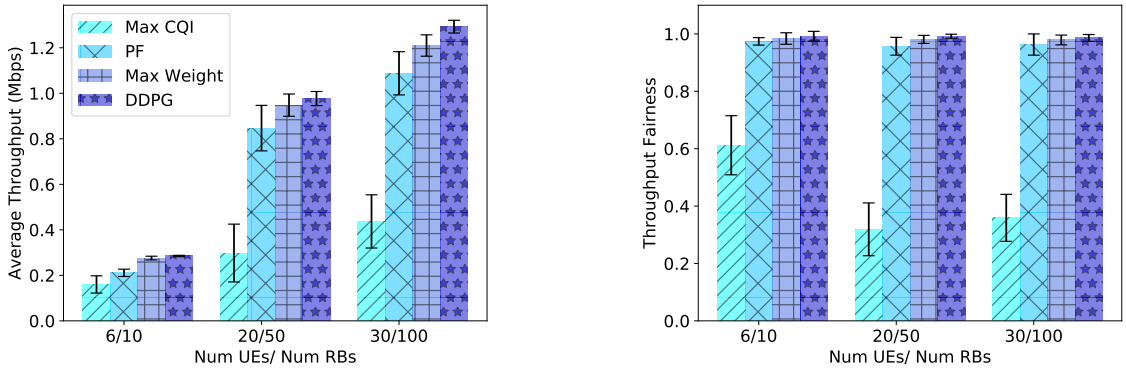
[22] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin, "Deep reinforcement learning in large discrete action spaces," *arXiv preprint arXiv:1512.07679*, 2015.

[23] 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 Comm. Surveys & Tutorials*, vol. 15, no. 2, pp. 678–700.

[24] S. Sesia, I. Toufik, and M. Baker, *LTE-the UMTS long term evolution: from theory to practice*. John Wiley & Sons, 2011.

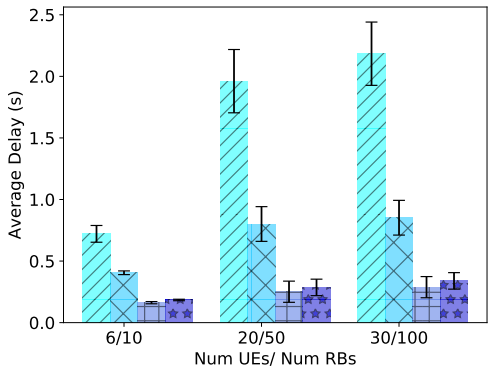
[25] L. Tassiulas and A. Ephremides, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Trans. on Information Theory*, vol. 39, no. 2, pp. 466–478, 1993.

[26] R. Jain, A. Durresi, and G. Babic, "Throughput fairness index: An explanation," in *ATM Forum contribution*, vol. 99, no. 45, 1999.

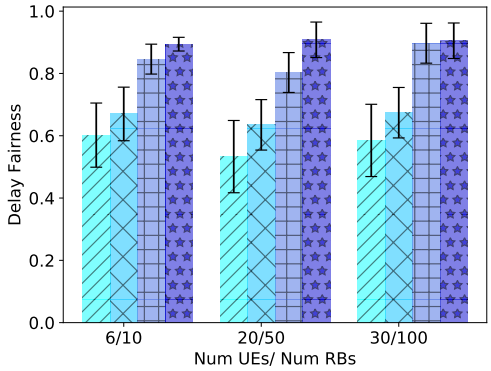


(a) Avg. Throughput

(b) Throughput Fairness

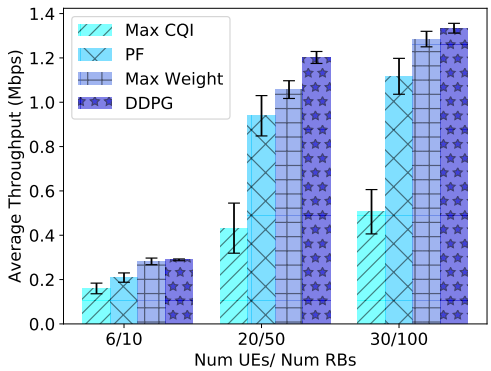


(c) Avg. Delay

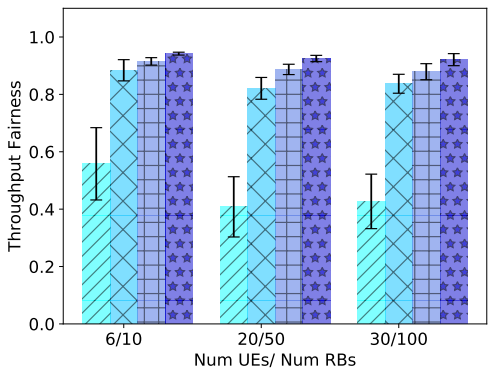


(d) Delay Fairness

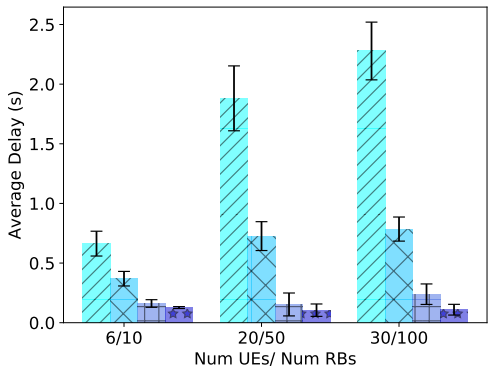
Fig. 3: Performance for Homogeneous Arrivals



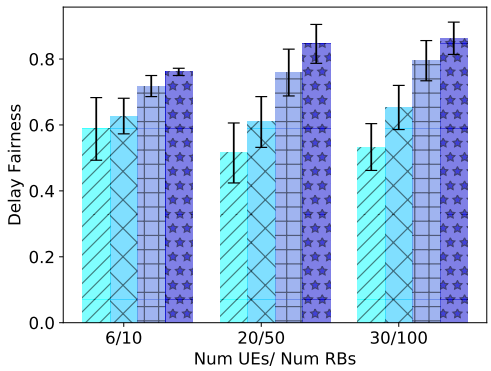
(a) Avg. Throughput



(b) Throughput Fairness



(c) Avg. Delay



(d) Delay Fairness

Fig. 4: Performance for Heterogeneous Arrivals.