Multi-objective Scheduling in Wireless Networks with Deep Reinforcement Learning

Babacar Toure*†, Dimitrios Tsilimantos*, Theodoros Giannakas*, Omid Esrafilian†, Marios Kountouris†
*Advanced Wireless Technology Lab, Paris Research Center, Huawei Technologies France
†Communication Systems Departement, EURECOM, Sophia Antipolis, France
{babacar.toure1, dimitrios.tsilimantos}@huawei.com, tedgiannakas@gmail.com,{omid.esrafilian, kountour}@eurecom.fr

***Abstract*—Radio resource scheduling in modern wireless networks faces several challenges, including high throughput demands, fast access requirements, and a staggering amount of users. AI-based schedulers have recently gained increasing interest as a solution to these problems since they can handle complex network settings by coming up with non-trivial scheduling schemes. Nevertheless, they have not yet been used to address multiple, often conflicting objectives, such as network throughput, latency, and fairness, without essentially reducing them to a single scalar objective. In this work, we develop a multi-objective reinforcement learning agent that acts as a scheduler of downlink transmissions to multiple devices. The agent is trained to accommodate differing and varying operator preferences for throughput and fairness while minimizing packet drops. Our simulation results show that, with a single agent trained only once, we outperform existing scheduling baselines on all objectives.**

***Index Terms*—deep reinforcement learning, multi-objective optimization, scheduling, wireless networks.**

I. Introduction

In recent years, mobile network operators have experienced an unprecedented increase in mobile traffic demand, as users are becoming more eager to use their mobile devices for streaming, gaming, and teleconferencing purposes, among others. The fact that there were finally more *unlimited* mobile data plans available in 2019 than capped ones [1] is a significant evidence of this phenomenon. To make matters worse, the wireless community has set very high standards for 5G+ networks, expecting considerable gains in performance over 4G, such as $10\times$ reduction in latency and a $10\times$ increase in throughput. Furthermore, the emerging generation of communication systems needs to perform well across a diverse range of services, such as enhanced mobile broadband (eMBB) and ultra-reliable and low-latency communications (URLLC). One of the main issues with managing many services is that each one could call for a different trade-off between network objectives; for instance, eMBB requires high throughput, whereas URLLC prioritizes low latency and error rate.

In this paper, we study a wireless setting in which a Base Station (BS) allocates resources to a set of devices for downlink communication. In particular, at each timeslot, the BS must determine which of its connected devices will occupy the channel to download data, based on device-related information, such as channel gain. Our main goal is to enable the BS to effectively and efficiently balance multiple objectives, such as throughput, fairness, and reliability.

A. Related Work

Conventional approaches in wireless networks typically optimize the performance of a single objective or combination of objectives. For example, Proportional Fair (PF) scheduler optimizes a mix of fairness and throughput [2], and MaxWeight (MW) targets throughput while trying to stabilize the data queues [3]. Recently, an AI-based approach that has gained popularity in wireless is Reinforcement Learning (RL) [4]. RL views the BS as an agent that tries to learn a (scheduling) policy by interacting with the wireless stochastic environment without knowledge of its statistics. Most often, RL uses function approximation, namely Deep Neural Networks (DNNs) [5], because of the large action and state spaces in networking applications. Deep RL (DRL), combining RL with DNNs, has shown remarkable results in the context of resource allocation [6]–[8], especially in dense scenarios where standard methods fail due to their myopic decision-making.

B. Key Challenge: Multi-objective RL-based Scheduling

In the intersection of RL with scheduling, some works assume a single objective [6], [7], [9], and others consider multiple rewards (objectives) but scalarize them; typically, this is done by using a linear function as follows. Let $\mathbf{r} \in \mathbb{R}^M$ denote the reward vector of M objectives and $\mathbf{w} \in \mathbb{R}^M$ express the importance on each objective; the scalarized multi-objective reward is then defined as $f_{\mathbf{w}}(\mathbf{r}) = \mathbf{w}^\top \mathbf{r}$, from which the BS learns [10]–[12]. Nonlinear scalarization has also been studied [13], as well as other network tasks like Internet congestion control [14].

Importantly, the aforementioned works optimize the performance for a single, fixed $\mathbf{w}$, hence for a single balance between objectives. This approach, although simple, is *not robust in the changes* of objective preferences $\mathbf{w}$. Consider the case of training a BS in a setting with a varying preference $\mathbf{w}$. A straightforward option is to retrain the agent (i.e., a DNN) for each $\mathbf{w}$ that arrives. This is an approach that requires a lot of computational resources. Alternatively, the BS could train multiple independent agents for each $\mathbf{w}$, but this is not a scalable approach in terms of memory. For these reasons, it would be preferable and more efficient to learn a single DNN that can *generalize* across the different $\mathbf{w}$. Such an approach allows network operators to handle various services

with potentially different trade-off requirements between the objectives, without the need for online retraining.

C. Contributions

We address the problem of tuning a single DRL agent to perform well for a wide range of preferences $\mathbf{w}$ over throughput and fairness, while also accounting for packet-drop rate. Our contributions are summarized as follows:

(C.1) We formulate the wireless downlink scheduling with three objectives as a multi-objective RL (MORL) problem. Importantly, we allow the operator's preference $\mathbf{w}$ over throughput and fairness to vary arbitrarily across the time horizon.

(C.2) We optimize a single Deep Q-Network (DQN) to learn multiple trade-off points between throughput and fairness while keeping the packet drop events to a bare minimum.

(C.3) We provide a thorough numerical evaluation of our proposed method, showing that it dominates all baselines on both throughput and fairness objectives, while having interesting gains in terms of packet drops.

The paper is structured as follows. Section II introduces the multi-objective scheduling problem. In Section III, we formulate the problem as a MORL task and describe the training algorithm. In Section IV, we present our simulation results, and finally, we draw our conclusions in Section V.

II. Problem Setup

A. System Model

We consider the downlink of a wireless network with $\mathcal{K} = \{u_1, \dots u_K\}$ devices connected to a single BS, as shown in Fig. 1. Time is slotted with a slot duration of T_s and at each slot the BS can schedule at most one device for transmission. The BS has a dedicated buffer of size B for each device, measured in packets; each packet is of size ρ bits. At slot t, the buffer k, dedicated to u_k, has x_t^k packets that wait for transmission, while A_t^k packets arrive according to Poisson distribution with mean λ_k. Packets arriving at a full buffer are dropped; we denote the number of dropped packets as d_t^k.

We adopt a block fading channel model, where the channel gain between u_k and the BS, denoted by g_t^k, remains constant during slot t and changes independently in each slot as

$$g_t^k = l^k l_{sh}^k ||h_k^t||_2^2, \tag{1}$$

where l^k is the distance-dependant path-loss following the empirical 3GPP model for Urban Macrocell propagation in Line of Sight conditions (UMa-LOS) [15], $l_{sh}^k \sim \mathcal{N}(0, \sigma)$ is the shadowing realization from a log-normal distribution and h_k^t evolves at each slot following a Gauss-Markov fast fading model [16]: $h_k^t = \alpha h_k^{t-1} + z_k^t, t > 0$, where $0 \leqslant \alpha \leqslant 1$ is the correlation factor and $z_k^t = \sqrt{\frac{1-\alpha}{2}}\left(z_{k,1}^t + i z_{k,2}^t\right)$ are i.i.d. circular complex Gaussian variables, $z_{k,j}^t \sim \mathcal{N}(0,1)$ for $j \in \{1,2\}$. Below, we define our main system variables.

Rate. Using the Shannon capacity of an additive white Gaussian noise (AWGN) channel, the estimated rate of u_k at t is

$$R_t^k = b_w \log_2\left(1 + \frac{g_t^k P}{N_0 b_w}\right) \tag{2}$$

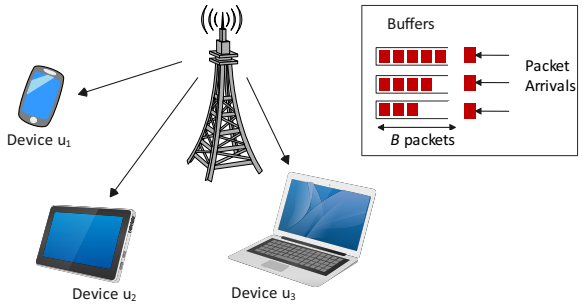


Fig. 1: Example of downlink communication

where P is the transmission power, b_w is the bandwidth and $N_0/2$ is the noise power spectral density.

Scheduled device and Throughput. At every slot, the BS allows only one device to occupy the channel and to download data from its buffer. We denote $c_t^k \in \{0,1\}$ a binary variable indicating if u_k has been scheduled at time t. Using the above definitions, u_k can potentially transmit several packets

$$\hat{P}_t^k = \left\lfloor \frac{T_s c_t^k R_t^k}{\rho} \right\rfloor, \tag{3}$$

but as the transmission is limited by the buffer content x_t^k, the real number of transmitted packets is written as

$$P_t^k = \min\{x_t^k, \hat{P}_t^k\}. \tag{4}$$

Therefore, the throughput of u_k at slot t is given by

$$H_t^k = \frac{\rho P_t^k}{T_s}. \tag{5}$$

The floor function $\lfloor \cdot \rfloor$ models the fact that only full packets are downloaded by the devices.

Buffer Evolution. We assume that the number of packets x_t^k in buffer k evolves based on the number of transmitted packets P_t^k and the arrivals A_t^k, but cannot exceed B as

$$x_{t+1}^k = \min\left(x_t^k - P_t^k + A_t^k, B\right). \tag{6}$$

Finally, at every slot t, the number of dropped packets in buffer k, which represents the excess of packets that would overflow the buffer capacity B after considering transmissions and arrivals, is written as follows

$$d_t^k = \left[x_t^k - P_t^k + A_t^k - B\right]^+, \tag{7}$$

with $[x]^+ \triangleq \max(x, 0)$.

B. Network Metrics

The operator has a scheduling policy $\boldsymbol{\pi}$, which drives the system evolution and performance. In particular, policy $\boldsymbol{\pi}$ is associated with the performance of the three long-term objectives of interest. Hence, for a time horizon T, first, throughput (in bps) is expressed as

$$H^{\boldsymbol{\pi}} = \sum_{t=1}^{T} \sum_{k=1}^{K} H_t^k. \tag{8}$$

Second, fairness, as experienced by the K devices [17], is

$$F^{\boldsymbol{\pi}} = \frac{(\sum_{k=1}^{K}\sum_{t=1}^{T} H_t^k)^2}{K\sum_{k=1}^{K}(\sum_{t=1}^{T} H_t^k)^2}. \quad (9)$$

Note that the minimum of (9) is equal to $1/K$ when only one device has occupied the channel on all T slots, whereas its maximum is equal to 1 when all devices download the same amount of packets across horizon T. Third, the number of dropped packets, normalized by the number of arrivals, is

$$D^{\boldsymbol{\pi}} = \frac{\sum_{k=1}^{K}\sum_{t=1}^{T} d_t^k}{\sum_{k=1}^{K}\sum_{t=1}^{T} A_t^k}. \quad (10)$$

Depending on the scenario, the objectives $H^{\boldsymbol{\pi}}$ and $F^{\boldsymbol{\pi}}$ can be conflicting. As an example, consider that device u_k has the highest rate in every slot, and the arrival rate is high enough to constantly keep all buffers full. In that case, a policy $\boldsymbol{\pi}$ that always schedules u_k is throughput-optimal, and results in the minimum possible fairness, $F^{\boldsymbol{\pi}} = 1/K$.

C. Problem Formulation

We formulate our optimization problem with multiple objectives, namely throughput, fairness and packet drop rate. In mathematical terms, the optimization problem becomes:

***Optimization Problem** 1:*

$$\max_{\pi}(H^{\boldsymbol{\pi}}, F^{\boldsymbol{\pi}}, -D^{\boldsymbol{\pi}}) \quad (11)$$

Since these objectives may conflict, our goal is to find multiple trade-off points as Pareto optimal solutions.

III. Multi-objective Reinforcement Learning

In this section, we formalize the scheduling problem of Section II-C with RL and present an algorithm for the training of the BS/agent, based on DQN as in [18].

A. Formulation

The scheduling problem we study is inherently "state-based", mainly due to *finite size* buffers. Scheduling a device at slot t affects the state of all buffers at $t+1$, rendering RL an attractive modeling framework. Moreover, since we are interested in considering more than one objectives without beforehand knowledge of the desired trade-off between those objectives, we resort to the Multi-Objective RL (MORL) framework. A MORL problem is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathbf{R}, \mathcal{W}, f_{\mathbf{w}})$, where $\mathcal{S}$ is the set of environment states, $\mathcal{A}$ is the set of actions and $\mathcal{P}$ is the state transition matrix. Importantly, the reward function $\mathbf{R}$ produces a vector, $\mathbf{r} \in \mathbb{R}^M$ with M the number of objectives, and not a scalar as in typical RL formulations, while $\mathcal{W}$, the preference space, is a $(M-1)$-simplex representing all possible preferences over the objectives: $\mathcal{W} = \{\mathbf{w} \in \mathbb{R}^M : \sum_i \mathrm{w}_i = 1, \mathrm{w}_i \geqslant 0, \forall i = 1, \ldots, M\}$, where w_i represents the importance weight of the i-th objective. Finally, function $f_{\mathbf{w}}$ scalarizes $\mathbf{r}$; here we are interested in linear scalarization, of the form $f_{\mathbf{w}}(\mathbf{r}) = \mathbf{w}^\top \cdot \mathbf{r}$.

State Space. For each device u_k at $t > 1$, we keep track of:

(a) Rate R_{t-1}^k as computed by (2).
(b) Average historical throughput up to time $t-1$, namely $\overline{H}_{t-1}^k = \frac{1}{t-1}\sum_{\tau=1}^{t-1} H_\tau^k$, see (5).
(c) Free buffer space, namely $B - x_t^k$, see (6).

Rates are useful to capture the throughput objective. We assume *no* access to the current channel rates, as in practice channel gain information needs to be reported by the devices for downlink [19]; and instead use the rates from the previous slot. Moreover, the historical throughput per device is mainly included to capture fairness accross devices and finally, the free buffer space is useful in order to avoid packet drops.

Each device u_k then has a local state $s_t^k = [R_t^k, \overline{H}_t^k, B - x_t^k]$ and the concatenation across all devices defines the state $\mathbf{s}_t$, which the BS observes at every t:

$$\mathbf{s}_t = \{s_t^1, \ldots, s_t^K\}. \quad (12)$$

As $\mathcal{S}$ is prohibitively large, we resort to the use of DNNs [5].

Action space. The BS action a_t indicates the device chosen to download data from the BS at slot t.

Reward. Through the instantaneous reward signal, we aim to help the agent accomplish its long-term goals. We set as a reward at slot t the vector that represents instantaneous measurements of the long-term objectives.

$$\mathbf{r}_t = \left[\sum_{k=1}^{K} H_t^k, \frac{(\sum_{k=1}^{K}\sum_{i=1}^{t} H_i^k)^2}{K\sum_{k=1}^{K}(\sum_{i=1}^{t} H_i^k)^2}, -\sum_{k=1}^{K} d_t^k\right]. \quad (13)$$

Recall that we would like to strike a balance between throughput (8) and fairness (9) while minimizing the packet drops (10). A way of capturing this is by setting a constant weight to the packet drop rewards. Thus, the preferences are in the shape $\mathbf{w}_0 = (\mathrm{w}, 1-\mathrm{w}, c)$ with $\mathrm{w} \in [0,1]$, and $c \geqslant 0$ modulates the level of constraint on the drop objectives. Since c is fixed, in the rest of the paper we only consider the 2D preference $\mathbf{w} = (\mathrm{w}, 1-\mathrm{w})$.

B. Training a Multi-Objective DQN (MOQ) Agent

In the standard scalar reward setting, starting from state s, taking action a and following policy π, the state-action value function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined as follows

$$Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{t \geqslant 0} \gamma^t r_t | s_0 = s, a_0 = a\right]. \quad (14)$$

***Definition** 1: The Bellman optimality operator $H : \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|} \to \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$ is defined as*

$$(HQ)(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P}\left[\max_{a' \in \mathcal{A}} Q(s', a')\right]. \quad (15)$$

The fixed point $Q^\star$ of the operator H is the optimal state-action function from which the optimal policy can be found.

Extending the single reward setting to the multi-objective case, the reward becomes a vector $\mathbf{r}_t \in \mathbb{R}^M$ and the state-action value function is also parameterized by $\mathbf{w}$, namely, $\mathbf{Q}^\pi : \mathcal{S} \times \mathcal{A} \times \mathcal{W} \to \mathbb{R}^M$ and is expressed as

$$\mathbf{Q}^\pi(s,a,\mathbf{w}) = \mathbb{E}_\pi\left[\sum_{t \geqslant 0} \gamma^t \mathbf{r}_t | s_0 = s, a_0 = a\right]. \quad (16)$$

Observe that for fixed s, a and two preferences $\mathbf{w} \neq \mathbf{w}'$, $\mathbf{Q}^\pi$ may result in two different M-dimensional vectors. Below, we define the Bellman operator for the multi-objective case.

***Definition** 2: The Bellman optimality operator* $\mathbf{H} : \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|\times|\mathcal{W}|} \rightarrow \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|\times|\mathcal{W}|}$ *is defined as*

$$(\mathbf{HQ})(s,a,\mathbf{w}) = \mathbf{r}(s,a) + \gamma \mathbb{E}_{s'\sim P}\Big[(\mathbf{FQ})(s',\mathbf{w})\Big]. \quad (17)$$

Similar to (15), where the max operator chooses the best Q-value from the next state s', here we need to extract M Q-values, one for every reward dimension. The main idea of [18] gives an optimistic way to resolve this by defining $\mathbf{F}$ as:

$$\mathbf{FQ}(s,\mathbf{w}) = \arg_{\mathbf{Q}} \max_{a'\in\mathcal{A},\mathbf{w}'\in\mathcal{W}} \mathbf{w}^T\mathbf{Q}(s,a',\mathbf{w}'). \quad (18)$$

The key step is to compute the scalarized utility $\mathbf{w}^T\mathbf{Q}(s,\cdot,\cdot)$ across all action-preference pairs (so in total $|\mathcal{A}|\cdot|\mathcal{W}|$ utilities), and return the $\mathbf{Q}$ vector evaluated at the pair that achieved the highest utility, namely, $\mathbf{Q}(s,a^*,\mathbf{w}^*)$. Intuitively, this generalizes the idea of being optimistic over the next action to being optimistic over the next action-preference pair. Applying repeatedly this Bellman operator provides the optimal $\mathbf{Q}^\star$ [18].

Training Procedure. We train the MOQ algorithm with double Q-Learning which yields more accurate action-value estimates [20]. The DQN is parametrized by θ and we define a loss between the $\mathbf{Q}$-estimations and target values obtained using the target network parametrized by θ^-, i.e.,

$$L^A(\theta) = \mathbb{E}_{s,a,\mathbf{w}}\Big[||\mathbf{y} - \mathbf{Q}(s,a,\mathbf{w};\theta)||^2\Big], \quad (19)$$

where $\mathbf{y} := \mathbf{r} + \gamma\mathbf{Q}(s',\tilde{a},\tilde{\mathbf{w}};\theta^-)$ with $(\tilde{a},\tilde{\mathbf{w}}) = \arg\max_{a'\in\mathcal{A},\mathbf{w}'\in\mathcal{W}} \mathbf{w}^T\mathbf{Q}(s,a',\mathbf{w}';\theta)$, using (17)-(18).

Such a loss drags the Q-value vector in the direction of the target vector. We also add an auxiliary loss that takes into account the scalarized utility as done in [18]:

$$L^B(\theta) = \mathbb{E}_{s,a,\mathbf{w}}\Big[\,\big|\mathbf{w}^T\mathbf{y} - \mathbf{w}^T\mathbf{Q}(s,a,\mathbf{w};\theta)\big|\,\Big]. \quad (20)$$

Our final loss then becomes:

$$L(\theta) = \frac{1}{2}\left(L^A(\theta) + L^B(\theta)\right). \quad (21)$$

The expectation in (19)-(20) is estimated by sampling experience tuples from a replay buffer. We describe the training steps in Alg. 1. As various preferences are sampled during training, this learning paradigm acquires optimized policies for all possible preferences, thereby eliminating the need for additional training or fine-tuning (see step 12 in Alg. 1). Note that since our scheduling problem does not have a real terminal state, we apply bootstrapping [21] to the target of the last state.

Algorithm 1: MOQ algorithm

1 **Input:** a preference sample distribution $\mathcal{D}_\mathcal{W}$, a replay buffer $\mathcal{B}$, a number of training episodes L, a number of sampled preferences $N_\mathbf{w}$.
2 Initialize replay buffer $\mathcal{B}$ and $\mathbf{Q}$ function parameters θ.
3 **for** *episode* $e = 1,\dots,L$ **do**
4 Sample $\mathbf{w}$ from $\mathcal{D}_\mathcal{W}$
5 **for** *timeslot* $t = 1,\dots,T$ **do**
6 Observe state s_t.
7 Sample ϵ-greedily an action:
$$a_t = \begin{cases}\text{random action in } \mathcal{A} & \text{w.p. } \epsilon\\ \max_{a\in\mathcal{A}} \mathbf{w}^\top\mathbf{Q}(s_t,a,\mathbf{w};\theta) & \text{w.p. } 1-\epsilon.\end{cases}$$
8 Receive vectorized reward $\mathbf{r}_t$ and observe s_{t+1}.
9 Save transition $(s_t,a_t,\mathbf{r}_t,s_{t+1})$ in buffer $\mathcal{B}$.
10 **if** *update* **then**
11 Sample N_τ transitions $(s_j,a_j,\mathbf{r}_j,s_{j+1})$
12 Sample $N_\mathbf{w}$ preferences $W = \{\mathbf{w}_i \sim \mathcal{D}_\mathcal{W}\}$.
13 Compute target $\forall 1 \leqslant i \leqslant N_\mathbf{w}, 1 \leqslant j \leqslant N_\tau$:
$$\mathbf{y}_{ij} = \mathbf{r}_j + \gamma\mathbf{Q}(s_{j+1},\tilde{a},\tilde{\mathbf{w}};\theta^-)$$
$$(\tilde{a},\tilde{\mathbf{w}}) = \arg\max_{a'\in\mathcal{A},\mathbf{w}'\in W} \mathbf{w}_i^\top\mathbf{Q}(s_{j+1},a',\mathbf{w}';\theta).$$
14 Update $\theta \leftarrow \theta - \alpha\nabla_\theta L(\theta)$ by stochastic gradient descent, using (21).
15 Update target network parameters: $\theta^- \leftarrow \tau\theta + (1-\tau)\theta^-$

IV. Performance Evaluation

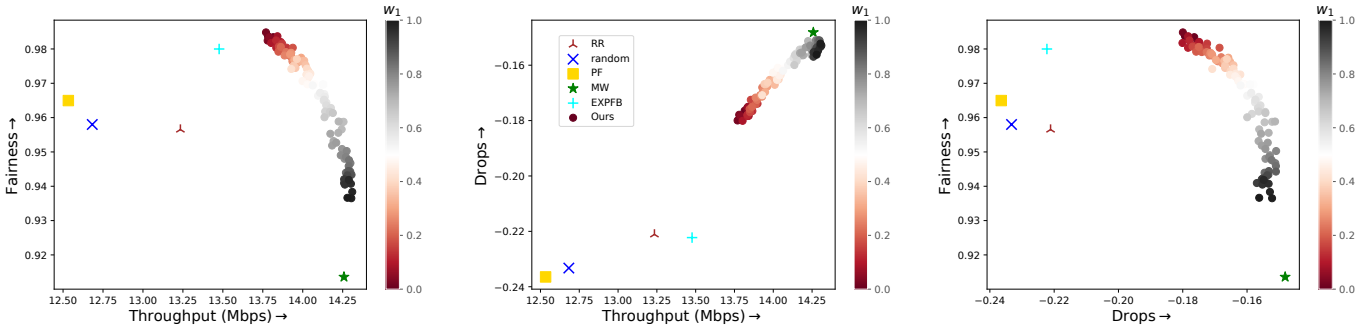
A. Simulation setup

Baselines. To evaluate the performances of our proposed DQN agent, we use the following baselines:

- Max-Weight (MW): $a_t = \arg\max_{k\in\mathcal{K}} x_t^k R_{t-1}^k$. It optimizes the throughput while stabilizing the buffers.
- Proportional Fairness (PF): $a_t = \arg\max_{k\in\mathcal{K}} \frac{R_{t-1}^k}{\overline{R}_{t-1}^k}$. It optimizes a mix of throughput and fairness.
- Exponential PF using Buffer state (EXPB) [22]: $a_t = \arg\max_{k\in\mathcal{K}} \frac{R_{t-1}^k}{\overline{R}_{t-1}^k} \exp\left(\frac{x_t^k - \bar{x}_t}{1+\sqrt{\bar{x}_t}}\right)$ s.t. $\bar{x}_t = \frac{\sum_k x_t^k}{K}$. EXPB extends PF by considering buffer lengths.
- Round-Robin: Schedules devices in a cyclic fashion.
- Random: Picks a device at random at every time slot.

The above baseline schedulers do not require training and each of them, by design, solves a fixed trade-off between our different objectives, which can be restrictive in the context of modern networks with variable service requirements. On the contrary, our MOQ agent is first trained offline, experiencing different trade-offs weights between the objectives. Then, in real-time, it is able to produce any required trade-off or adapt to a change in the operator inputted weights.

Environment and DQN model. We set up a network with a single BS and 8 devices with fixed positions. The maximum buffer size is $B = 100$ packets and each packet contains $\rho = 700$ bits. We fix the transmission time to $T_s = 1$ ms. The arrival rate λ is set to create significant network congestion, making the problem challenging. Note that our approach works for any λ. The buffers are initialized full, so that packets can get dropped from the beginning of the



(a) Throughput vs Fairness (b) Drops vs Throughput (c) Fairness vs Drops

Fig. 2: Trade-off analysis. Each marker represents a scheduler. Our algorithm (circled markers with gradient of colors from red to black) is employed with different preferences $\mathbf{w} = (\mathrm{w}_1, 1 - \mathrm{w}_1)$ for throughput weight w_1.

TABLE I: Simulation parameters

System parameters	
Packet arrivals λ	3000 packets/s
Scheduling duration T	500 slots
Cell radius	1000 m
Bandwidth b_w	1 MHz
Transmission power P	20 dBm
Noise spectral density N_0	-174 dBm/Hz
Center frequency f_c [15]	2 GHz
Shadowing std σ	7 dB
Fast-fading correlation	0.95
DNN parameters	
Discount factor γ	0.99
Soft target update τ	0.005
Preference sample size $N_{\mathbf{w}}$	4
Batch size N_τ	32
Training episodes L	3000
Learning rate α	$1e^{-4}$

scheduling duration. A random realization of the environment is saved for evaluation so that all the schedulers experience the same channel conditions and packet arrivals. For the DQN, we employ a fully connected DNN with 2 layers, each one with 256 neurons. The packet drop constraint is set to $c = 0.5$. More network and DQN parameters are presented in Table I.

State and reward normalization. Input and output scaling is widely used to enhance DNN training [23]. Following this principle, we normalized states and rewards yielding significant gains in performance and convergence speed. Practically, we collected a dataset of 75000 states and rewards by performing 50 scheduling episodes using each of the following three baselines: MW, Random and PF. Each episode corresponds to a random realization of the environment with different channel conditions due to shadowing and fast-fading. In addition, the reward dataset is diversified because the three selected baselines prioritize different objectives. From the obtained dataset, we computed the following statistics: the maximum rate R and maximum average historical throughput $\overline{H}$ observed across all devices, and (μ_m, σ_m) for $m = 1, \cdots, M$ the average and standard deviation of the rewards on each objective. Our normalized state and reward then become:

$$\hat{\mathbf{s}}_t = \left\{ \forall u_k \in \mathcal{K} : \frac{R^k_{t-1}}{R}, \frac{\overline{H}^k_{t-1}}{\overline{H}}, \frac{B - x^k_t}{B} \right\}, \tag{22}$$

$$\hat{r}_{t,m} = \frac{r_{t,m} - \mu_m}{\sigma_m} \quad \text{for } m = 1, \cdots, M. \tag{23}$$

Here $r_{t,m}$ denotes the m-th element of the vector reward $\mathbf{r}_t$.

B. Results and discussion

Pareto front.

Fig. 2 shows the achieved performance of different schedulers over the throughput, fairness, and drop ratio objectives, as defined in (8)-(10). The performance is evaluated over an episode of the evaluation environment. Our scheduler has been called over the preferences $\{(\frac{k}{100}, 1 - \frac{k}{100}), k \in [\![0, 100]\!]\}$, which are illustrated with different colors on the plots. Recall that the packet drop weight is set to 0.5. Also note that for illustration purposes, we show the drop packet ratio with negative values, so that the values in all 3 objectives increases for higher performances.

Our scheduler manages to learn a trade-off between throughput and fairness (see Fig. 2.a), while minimizing the packet drops (see Fig. 2.b and Fig. 2.c). When the input preference is $\mathbf{w} = (0, 1)$, i.e. fully on fairness, our scheduler performs $(13.78 \text{ Mbps}, 0.984, 17.9\%)$ respectively on throughput, fairness, and drop ratio, dominating EXPB and PF on all three objectives. Note that in our setting, the PF scheduler performs poorly in throughput and packet drop due to its insensitivity to buffer sizes (see its definition in IV.A). EXPB corrects that behaviour thanks to the exponential term in buffer lengths, but is still worse than our solution. On the other hand, for preferences close to $\mathbf{w} = (1, 0)$ favoring throughput, we observe that the MW scheduler minimizes the packet drops and is throughput-efficient at the cost of a lower fairness compared to other baselines. Remarkably, our scheduler performs $(14.30 \text{ Mbps}, 0.93, 15.4\%)$, slightly outperforming MW

TABLE II: Monte-Carlo evaluation (mean and std)

Algo	**w**	Throughput (Mbps)	Fairness	Drops (%)
MW	-	14.4 $\pm$0.09	0.90 $\pm$0.009	13.7 $\pm$0.6
Ours	(1,0)	14.45 $\pm$0.1	0.92 $\pm$0.01	14 $\pm$0.7
EXPB	-	13.62 $\pm$0.1	0.97 $\pm$0.003	20 $\pm$0.8
Ours	(0,1)	13.84 $\pm$0.1	0.976 $\pm$0.003	17.3 $\pm$0.8

in throughput but with a significantly higher fairness, as shown by the x-axis of Fig. 2.a.

Besides the fact that our scheduler performs well in both extremes, as shown previously, one of the main interests of our approach lies in the fact that it embeds, with a single training, optimal policies over the whole preference space. For example, we notice that as the preference gradually changes from $(1, 0)$ to $(0, 1)$, we are able to continuously capture trade-off points in the Pareto front of the studied objectives. On another note, our results show a strong correlation between the throughput and drop ratio objectives (see Fig 2 .b).

Monte Carlo evaluation. Knowing that our wireless environment has high dynamics due to the randomness of the channel conditions and packet arrivals, we set up a test to ensure that the performance of our scheduler is guaranteed across different realizations of the environment. For 100 different environments, we measure the long-term performances achieved by the EXPB scheduler, the MW scheduler, and our scheduler with $\mathbf{w} = (0, 1)$ and $\mathbf{w} = (1, 0)$. The results are reported in Table II. From this table, we see that our scheduler has a similar variance to the standard schedulers in all the settings and on all objectives. This supports the reliability of our RL solution in comparison to these well-established scheduling baselines. In terms of average performance, we confirm the observation made on the Pareto front of the evaluation environment. Our scheduler reaches a performance similar (or slightly higher) than the baselines in the objective of interest while outperforming them on the other objectives.

V. Conclusions

In this work, we proposed an all-in-one scheduler that determines optimal policies for every input preference on throughput and fairness objectives with only a single training. Our simulations demonstrate that, for all objectives, our scheduler is as reliable and as effective as existing baseline schedulers. Its key advantage is its ability to implement the optimal policy for any preference, thus eliminating the need for online re-training or maintaining multiple, possibly suboptimal models at the BS. In our future work we plan to extend this study to scenarios involving multiple interfering base stations in high-density networks. The addtional challenge would be the non-stationarity introduced by agents acting according to their own preferences.

References

[1] U. Cisco, "Cisco annual internet report (2018–2023) white paper," *Cisco: San Jose, CA, USA*, vol. 10, 2020.

[2] M. Andrews *et al.*, "Providing quality of service over a shared wireless link," *IEEE Communications magazine*, vol. 39, no. 2, pp. 150–154, 2001.

[3] ——, "Scheduling in a queuing system with asynchronously varying service rates," *Probability in the Engineering and Informational Sciences*, vol. 18, no. 2, pp. 191–217, 2004.

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

[5] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[6] Y. S. Nasir and D. Guo, "Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2239–2250, 2019.

[7] A. Destounis and D. Tsilimantos, "Distributed reinforcement learning for low-delay uplink user scheduling in multicell networks," in *GLOBECOM IEEE Global Communications Conference*. IEEE, 2022, pp. 2303–2308.

[8] A. Avranas, P. Ciblat, and M. Kountouris, "Deep reinforcement learning for resource constrained multiclass scheduling in wireless networks," *IEEE Transactions on Machine Learning in Communications and Networking*, vol. 1, pp. 225–241, 2023.

[9] F. Meng, P. Chen, L. Wu, and J. Cheng, "Power allocation in multi-user cellular networks: Deep reinforcement learning approaches," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6255–6267, 2020.

[10] S. Chinchali *et al.*, "Cellular network traffic scheduling with deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[11] J. Stigenberg, V. Saxena, S. Tayamon, and E. Ghadimi, "Qos-aware scheduling in new radio using deep reinforcement learning," in *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2021, pp. 991–997.

[12] C. Xu *et al.*, "Buffer-aware wireless scheduling based on deep reinforcement learning," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2020, pp. 1–6.

[13] N. Naderializadeh, J. J. Sydir, M. Simsek, and H. Nikopour, "Resource management in wireless networks via multi-agent deep reinforcement learning," *IEEE Transactions on Wireless Communications*, vol. 20, no. 6, pp. 3507–3523, 2021.

[14] N. Jay *et al.*, "A deep reinforcement learning perspective on internet congestion control," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3050–3059.

[15] 3GPP, "Study on channel model for frequencies from 0.5 to 100 GHz (v. 17.0.0, release 17)." TS 38.901, 2022.

[16] M. Kobayashi and G. Caire, "Joint beamforming and scheduling for a multi-antenna downlink with imperfect transmitter channel knowledge," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 7, pp. 1468–1477, 2007.

[17] R. K. Jain, D.-M. W. Chiu, W. R. Hawe *et al.*, "A quantitative measure of fairness and discrimination," *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, vol. 21, p. 1, 1984.

[18] R. Yang, X. Sun, and K. Narasimhan, "A generalized algorithm for multi-objective reinforcement learning and policy adaptation," *Advances in neural information processing systems*, vol. 32, 2019.

[19] J. Korhonen, "Scheduling of data transmissions," https://www.3gpp.org/technologies/scheduling, 2023, [Online; accessed 4-April-2024].

[20] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16. AAAI Press, 2016, p. 2094–2100.

[21] F. Pardo, A. Tavakoli, V. Levdik, and P. Kormushev, "Time limits in reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018, pp. 4045–4054.

[22] J.-H. Rhee, J. Holtzman, and D.-K. Kim, "Scheduling of real/non-real time services: adaptive exp/pf algorithm," in *The 57th IEEE Semiannual Vehicular Technology Conference, 2003. VTC 2003-Spring.*, vol. 1, 2003, pp. 462–466 vol.1.

[23] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, *Efficient BackProp*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 9–48. [Online]. Available: https://doi.org/10.1007/978-3-642-35289-8_3