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/PIMRC56721.2023.10293980

Performance Evaluation of 5G Delay-Sensitive Single-Carrier Multi-User Downlink Scheduling

Anjali Omer†, Filippo Malandra†, Jacob Chakareski‡, Nicholas Mastronarde†
†*Dept. of Electrical Engineering, University at Buffalo,* ‡*Ying Wu College of Computing, New Jersey Institute of Technology*
Email: anjaliom@buffalo.edu, filippom@buffalo.edu, jacob.chakareski@njit.edu, nmastron@buffalo.edu

***Abstract*—The coexistence of a wide variety of different applications with diverse Quality of Service (QoS) requirements calls for more sophisticated radio resource scheduling (RRS) in 5G networks compared to previous generations. To address this challenge, a growing body of research formulates the RRS problem as a Markov decision process (MDP) and aims to solve it using deep reinforcement learning (DRL). A key consideration when formulating an MDP is the choice of *reward* function, which determines the goal of the decision agent. Despite the reward function being a critical component of an MDP, there is currently no systematic study comparing how different reward functions affect network performance. To this end, we carry out a comparative study of the delay and overflow performance using several reward functions that aim to minimize packet delays. Through extensive simulations under different traffic and channel conditions, we identify a reward function that can achieve near optimal delay with up to $55-67\%$ fewer packet drops than the other investigated options, and does not require any tuning.**

I. Introduction

With the evolution of cellular networks from 4G to 5G, new RRS algorithms are needed to meet the more stringent and varied end-user QoS requirements. It is well-known that the RRS problem is NP-hard [1]. Due to its complexity, conventional optimization-based solutions to the RRS problem often cannot allocate resources to user equipments (UEs) in the required sub-ms time span. On the other hand, due to the end-user's diverse and demanding QoS requirements, simple rule-based schedulers provide sub-optimal solutions. To overcome these limitations, a growing body of research formulates the RRS problem as an MDP and solves it using DRL. A key consideration when formulating an MDP is the choice of *reward* function, which determines the goal(s) of the decision agent. In the context of RRS, reward functions can be roughly divided into two categories: 1) those that focus on maximizing network capacity, throughput, and/or fairness (e.g., [2]–[6]) and 2) those that focus on minimizing packet delays (e.g., [7]–[10]). Despite the reward function being a critical component of an MDP, there is currently no systematic study comparing how different ones affect network performance.

In this paper, we aim to bridge this gap by carrying out a comparative study of the delay and overflow performance achieved using different reward/cost functions that target minimizing queuing delays, buffer overflows, and/or packet drops. We focus on such reward/cost functions for two reasons. First, 5G networks aim to enable new delay-sensitive applications, such as industrial control and virtual/augmented/mixed reality. Second, existing rule-based schedulers can effectively maximize network capacity, throughput, and/or fairness, but do not effectively optimize packet delays in general [11].

Our contributions are as follows: 1) we rigorously formulate the multi-user downlink RRS problem as an MDP; 2) we review five reward/cost functions used in prior literature to minimize queuing delays; 3) we further parameterize the reward/cost functions to penalize buffer overflows at different levels (ranging from zero to severe penalty); and 4) through extensive simulations under different traffic and channel conditions, we identify a reward function that frequently achieves near optimal delay with up to 55-67% fewer packet drops than the other investigated options, does not require any tuning, and outperforms a benchmark rule-based scheduler.

As noted earlier, the RRS problem is NP-hard. Therefore, finding an optimal scheduling policy is unfeasible in systems with many UEs and resource blocks (RBs). Although DRL algorithms can be used to optimize scheduling policies in such scenarios, they require careful hyperparameter optimization to learn good policies and the learned policies are not guaranteed to be optimal. Since our goal is to rigorously assess the performance of RRS under different reward functions, it is crucial that we can evaluate the optimal policy in each case, to allow us to prospectively identify key performance insights and more broadly applicable conclusions. To this end, we formulate the downlink RRS problem assuming a single-carrier (i.e., one RB). Furthermore, due to the well-known *curse of dimensionality* [12], we limit our simulations to a scenario with finite packet buffers and two UEs.

The remainder of this paper is organized as follows. We review related work in Section II; present our system model in Section III; formulate the RRS problem as an MDP and describe how to solve it using value iteration in Section IV; present our results in Section V; and conclude in Section VI.

II. Related Work

In this section, we briefly highlight related work that focuses on minimizing packet delays. Broadly speaking, there are two distinct approaches to minimize packet delays when formulating an MDP-based scheduling problem. The first approach introduces state variables to keep track of the head of line (HoL) delays or individual packet delays for each UE [7], [8], [13]. For example, Gu et al. [8] formulate the

A. Omer's and N. Mastronarde's work was supported by NSF award #2030157, F. Malandra's work was supported by NSF award #2105230, and J. Chakareski's work was supported by NSF awards #2032033 and #2106150.

downlink scheduling problem to maximize the total number of packets received by UEs, while dropping packets that do not meet certain HoL delay constraints. As another example, Nokia's Wireless Suite [13], which provides an Open AI Gym compatible environment for allocating resources to UEs, includes state variables tracking the age of every packet in every UE's buffer. However, tracking individual packet delays results in an enormous state space, which limits the scalability of this approach even when solving it with DRL.

The second approach introduces state variables to track each UE's buffer state and uses the buffer state as a proxy for delay in the cost function. For example, Sharma et al. [9] use a cost function based on the sum of changes in the buffer state of every UE to minimize delay. Inspired by Sharma, Robinson et al. [10] use an integer reward system based on the change in the buffer state of UEs. Additionally, a significant body of research leveraging reinforcement learning for optimal transmission scheduling in single- and multi-user scenarios makes use of a buffer state-based approach [14]–[19]. Due to its favorable scalability compared to tracking individual packet delays, in this paper, we focus on a buffer state based approach.

III. System Model

RRS refers to the problem of allocating available radio resources (i.e., RBs) to the requesting UEs to satisfy their Quality of Service (QoS) requirements. In this paper, we consider a 5G single-carrier multi-user downlink RRS problem with a single macro base station (gNodeB) that needs to allocate a single RB to one of the N requesting UEs indexed by $i \in \{1, 2, \ldots, N\}$ in each transmission time interval (TTI), as illustrated in Fig. 1. We let Δt denote the TTI duration (s) and index TTIs by $t \in \{0, 1, \ldots\}$. In TTI t, the gNodeB observes the following two state variables for each UE: 1) the number of packets b_i^t awaiting transmission in UE i's buffer; and 2) UE i's channel state h_i^t. Based on the observed states, the gNodeB takes a scheduling action $a^t \in \{1, 2, \ldots, N\}$, where $a^t = i$ indicates that UE i is scheduled in TTI t. We now describe each component of Fig. 1 in detail.

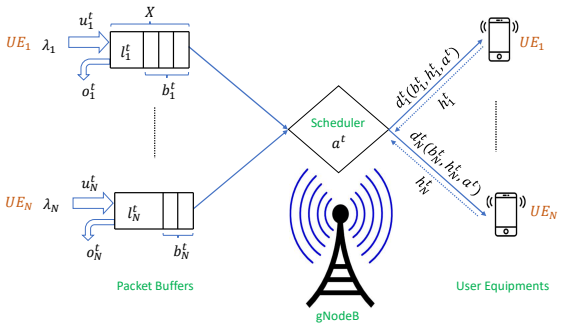


Fig. 1: 5G multi-user downlink scheduling model.

Channel Model: We let $h_i^t \in \mathcal{S}_h$ denote the channel state (signal-to-noise ratio (SNR)) of UE i at time t, where $\mathcal{S}_h$ is a discrete and finite set of channel states. As in [9], we assume that: 1) the channel states h_i^t, $t = 0, 1, \ldots$, evolve as a Markov chain with stationary transition probability function $P_i^h(h_i'|h_i)$; 2) the channel state can be estimated perfectly; and 3) the channel state is constant in each TTI. Given the scheduling action a^t and h_i^t, we define the transmission rate $c_i^t(h_i^t, a^t)$ (bits/TTI) of UE i in TTI t as:

$$c_i^t(h_i^t, a^t) = \begin{cases} B \log_2(1 + h_i^t)\triangle t, & \text{if } a^t = i \\ 0, & \text{if } a^t \neq i, \end{cases} \quad (1)$$

where B is the RB bandwidth (Hz).

Buffer Model: Packet arrivals for UE i are stored in a buffer at the gNodeB and are transmitted in first-in first-out (FIFO) order. We denote the i-th UE's buffer state in TTI t by $b_i^t \in \mathcal{S}_b = \{0, 1, ..., X\}$, where $\mathcal{S}_b$ is a finite set of possible buffer states and X denotes the maximum buffer occupancy in packets. We assume that packets have a fixed size of L bits and that packet arrivals u_i^t are independent and identically distributed in each TTI, i.e., $u_i^t \sim P_i^u(u)$, where P_i^u denotes the packet arrival distribution. We let λ_i denote UE i's average packet arrival rate in bits/s.

Given its buffer state b_i^t, its channel state h_i^t, and the scheduling action a^t, the ith UE's next buffer state b_i^{t+1} can be determined by the following Lindley recursion:

$$b_i^{t+1} = \min(b_i^t - d_i^t(b_i^t, h_i^t, a^t) + u_i^t, X), \quad (2)$$

where $d_i^t(b_i^t, h_i^t, a^t)$ denotes the number of packets that UE i transmits in TTI t. Importantly, $d_i^t(b_i^t, h_i^t, a^t)$ cannot exceed the number of buffered packets; therefore,

$$d_i^t(b_i^t, h_i^t, a^t) = \min\left(b_i^t, \lfloor c_i^t(h_i^t, a^t)/L \rfloor\right), \quad (3)$$

where $\lfloor x \rfloor$ denotes the floor of x.

Due to the finite buffer size X, packet overflows will occur when more packets arrive than can be stored in the buffer. Let o_i^t denote the number of packet overflows and l_i^t denote the number of packets that enter UE i's buffer in TTI t: i.e.,

$$o_i^t = \max(b_i^t - d_i^t(b_i^t, h_i^t, a^t) + u_i^t - X, 0) \text{ and } l_i^t = u_i^t - o_i^t. \quad (4)$$

IV. Problem Formulation

In this section, we formulate the scheduling problem under study as an MDP. We first introduce the definition of an MDP in Section IV-A. We then formulate the single-carrier multi-user downlink scheduling problem as an MDP in Section IV-B.

A. Markov Decision Process (MDP) Framework

An MDP is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, C, P, \gamma)$ or $\mathcal{M} = (\mathcal{S}, \mathcal{A}, R, P, \gamma)$, 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; $C : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ are cost and reward functions that map states and actions to real-valued costs and rewards, respectively; $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ 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 $s \in \mathcal{S}$; and $\gamma \in [0, 1]$ is a discount factor, which determines the relative importance of immediate and future costs/rewards.

MDPs model sequential-decision problems in which the action taken in the current state not only affects the immediate cost/reward, but also affects the future costs/rewards through the next state [12]. This is a fitting model of the RRS

problem because scheduling decisions affect the immediate and expected future queuing delays experienced by the UEs.

In the following formulation, we focus on cost functions. To use a reward function, we replace in the analysis the cost function with the reward function and the operator $\min$ with $\max$. The objective of an MDP is to determine an optimal *policy* $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that specifies the action to take in each state and minimizes the *expected discounted cost*:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty}(\gamma)^t C^t(s^t, \pi(s^t))|s = s^0\right], \quad (5)$$

where V^{π} denotes the *value function* under policy π; $(\gamma)^t$ denotes the discount factor to the t-th power; the expectation is taken over the sequence of states governed by the transition probabilities $P(s'|s,a)$; and s^0 is the initial state. We can rewrite $V^{\pi}(s)$ recursively by using the one-step transition probability function to represent the expected future costs:

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

The *optimal value function* is defined as follows:

$$V^*(s) = \min_{\pi \in \Pi} V^{\pi}(s),$$

where Π denotes the set of possible policies and $V^*(s)$ satisfies the following Bellman equation for all $s \in \mathcal{S}$:

$$V^*(s) = \min_{a \in \mathcal{A}} \left\{ C(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V^*(s') \right\}. \quad (6)$$

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 (6) for all $s \in \mathcal{S}$.

In this paper, we use the well-known value iteration algorithm to determine $\pi^*(s)$. The value iteration algorithm, formulated in Algorithm 1, takes the cost and transition probability functions as inputs and provides the corresponding optimal policy $\pi^*(s)$ as its output.

Algorithm 1 Value Iteration

Initialize V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}$)
repeat
  $\triangle \leftarrow 0$
  for each $s \in \mathcal{S}$ **do**:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \min_a \left\{ C(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s') \right\}$
    $\triangle \leftarrow \max(\triangle, |v - V(s)|)$
  end for
until $\triangle < \theta$ (a small positive number)
Output a deterministic policy, π^*, such that for all $s \in \mathcal{S}$:

$$\pi^*(s) = \arg\min_a \left\{ C(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s') \right\}$$

B. The Scheduling Problem as an MDP

We now map the system model in Section III to an MDP.

State: The ith UE's state in TTI t comprises its buffer and channel states, i.e., $s_i^t \triangleq (b_i^t, h_i^t) \in \mathcal{S}_i = \mathcal{S}_b \times \mathcal{S}_h$. The system state at TTI t is the combined state of all UEs, given by $s^t \triangleq (s_1^t, s_2^t, ..., s_N^t) \in \prod_{i=1}^{N} \mathcal{S}_i$, where $\prod_{i=1}^{N} \mathcal{S}_i$ denotes the Cartesian product of the UEs' state sets.

Action: The action $a^t \in \mathcal{A} = \{1, \ldots, N\}$ determines which UE is scheduled in TTI t.

Transition Probability Function: Given its packet arrival distribution P_i^u, channel state h_i, and the scheduling action a, it follows from (2) that UE i transitions from buffer state b_i to b_i' with probability:

$$P_i^b(b_i'|b_i, h_i, a) = \sum_{u_i=0}^{\infty} P_i^u(u_i) \mathbb{I}_{\{b_i' = \min(b_i - d_i(b_i,h_i,a) + u_i, X)\}}, \quad (7)$$

where $\mathbb{I}_{\{\cdot\}}$ is an indicator function that is set to 1 when $\{\cdot\}$ is true and is set to 0 otherwise. Given each UE's channel state transition probability function P_i^h, the joint state transition probability can be expressed as:

$$P(s'|s,a) = \prod_{i=1}^{N} P_i^b(b_i'|b_i, h_i, a) P_i^h(h_i'|h_i). \quad (8)$$

Cost/Reward Functions: The total expected cost $C(s,a)$ (resp., reward $R(s,a)$) of taking action a in state s is equal to the sum of the expected costs (resp., rewards) over all UEs:

$$C(s,a) = \sum_{i=1}^{N} C_i(s_i, a) \text{ and } R(s,a) = \sum_{i=1}^{N} R_i(s_i, a), \quad (9)$$

where $C_i(s_i,a)$ (resp., $R_i(s_i,a)$) denotes the expected cost (resp., reward) for UE i in state s_i given action a.

In this paper, we explore scheduling policies based on five different cost and reward functions. Each cost/reward function aims to minimize delay in a different way, but all contain an *overflow cost* term to penalize packet overflows. Given the ith UE's packet arrival distribution P_i^u and its state s_i, its expected overflow cost $O_i(s_i,a)$ when action a is taken is defined as its expected number of overflows: i.e.,

$$O_i(s_i, a) = \sum_{u_i=0}^{\infty} P_i^u(u_i) \max(b_i - d_i(b_i, h_i, a) + u_i - X, 0). \quad (10)$$

We now describe the five cost and reward functions in detail.

Cost function for Policy 1 (P1):

$$C_i(s_i, a) = b_i + \alpha O_i(s_i, a). \quad (11)$$

This cost function is defined as a weighted sum of the ith UE's buffer state (b_i) and its expected overflow cost. By Little's law, the time-average buffer state is proportional to the time-average queuing delay experienced by packets that are admitted into the buffer. The expected overflow cost is multiplied by a tunable parameter $\alpha \geq 0$, which denotes the penalty per packet overflow. The term b_i in this cost function has been used in prior work on delay-sensitive point-to-point transmission scheduling (e.g., [14], [16]) and multi-user scheduling (e.g., [15]).

Cost function for Policy 2 (P2):

$$C_i(s_i, a) = \sum_{b_i'=0}^{X} P_i^b(b_i'|b_i, h_i, a)(b_i' - b_i) + \alpha O_i(s_i, a). \quad (12)$$

In (12), $b_i' - b_i$ is equivalent to the difference between the number of packets admitted into the buffer (l_i defined in (4))

and the number of transmitted packets (d_i defined in (3)). The expected discounted cost of this term can be expressed as:

$$\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t(b_i^{t+1}-b_i^t)\right]=\mathbb{E}\left[-b^0+(1-\gamma)\sum_{t=0}^{\infty}\gamma^t b_i^{t+1}\right],$$

which is similar to the expected discounted value $\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t b_i^t\right]$ of the holding cost in (12), but with the addition of a constant ($-b^0$), a scaling factor ($1-\gamma$), and a one-step delay ($\gamma^t b_i^{t+1}$). We used this cost function in our prior work on downlink scheduling [9].

Cost function for Policy 3 (P3):

$$C_i(s_i,a)=b_i-d_i(b_i,h_i,a)+\alpha O_i(s_i,a). \tag{13}$$

Eq. (13) is similar to (12) but it does not count the transmission delay (i.e., the TTI in which packets are transmitted). This has been used in prior work on delay-sensitive point-to-point transmission scheduling (e.g., [17], [18]).

Cost function for Policy 4 (P4):

$$C_i(s_i,a)=b_i/\lambda_i+\alpha O_i(s_i,a). \tag{14}$$

The first three cost functions defined above can effectively minimize delay in single-user scenarios as in [14], [16]–[18]; however, in multi-user scenarios, the time-average sum of the buffer states across UEs is *not* proportional to the time-average sum of delays across UEs if UEs have *heterogeneous* arrival rates. In (11), we correct for this by replacing b_i in (12) with $\frac{b_i}{\lambda_i}$ since, by Little's law, its time-average is *equal* to the time-average delay. Unfortunately, to find the optimal policy in the finite buffer scenario, we face a chicken-and-egg problem: we have to replace the arrival rate λ_i with the rate of packets *admitted* into the buffer, but this quantity depends on the policy and the optimal policy depends on this quantity. Consequently, in our numerical results, we directly use $\frac{b_i}{\lambda_i}$. This cost function was used in prior work on delay-constrained MIMO transmission control [19].

Reward function for Policy 5 (P5):

$$R_i(s_i,a)=d_i(b_i,h_i,a)-\alpha O_i(s_i,a). \tag{15}$$

This reward function is defined as the difference between the number of transmitted packets and the weighted overflow cost. Our numerical results show that maximizing the time-average sum of packets transmitted over time effectively minimizes packet overflows. This follows from the fact that maximizing the system's throughput is equivalent to minimizing the number of blocked packets [11]. As described in Section II, this reward function was used in [8].

V. Simulation Results

We now carry out a comparative study of the delay and overflow performance achieved by scheduling policies that optimize the five cost/reward functions in Section IV. We also include the performance of a benchmark rule-based scheduler, namely, Max-Weight, which schedules the UE a^t that has the highest capacity-buffer product in the current TTI t, i.e.,

$$a^t=\arg\max_{i\in\mathcal{A}}\left\{B\log_2(1+h_i^t)\triangle t\times b_i^t\right\}. \tag{16}$$

We choose Max-Weight instead of the more common Proportional Fair (PF) scheduling algorithm for two reasons. First, unlike PF, Max-Weight is throughput and delay optimal under the assumptions of a) finite buffers, b) binary arrivals, and c) arrival, channel, and service processes with identical statistics at different UEs [11]. Second, in our prior work [9], Max-Weight achieved better performance in terms of delay, throughput, and fairness than PF in a similar multi-user downlink scheduling scenario to the one considered herein.

To evaluate the various scheduling policies, we implemented a single-carrier multi-user downlink scheduling simulator in Python based on the system model in Section III. Table I lists the key simulation parameters. For the reasons explained in Section I, we consider $N=2$ UEs and one RB. We let packets have a fixed size of $L=200$ bits and each UE's buffer store a maximum of $X=10$ packets. We consider a TTI duration of $\Delta t=1$ ms and RB bandwidth of $B=180$ kHz. We assume Poisson arrival processes with average arrival rates of $\lambda=100-275$ kbps. We use two sets of channel states $\mathcal{S}_h^1$ and $\mathcal{S}_h^2$ such that channel states in $\mathcal{S}_h^2$ are better than those in $\mathcal{S}_h^1$. We consider discount factor $\gamma=0.98$ and overflow penalty factors $\alpha=0-100$.

We use four network scenarios with different arrival rates and channel states. Arrival rates are either *homogeneous*, where both UEs have the same arrival rates ($\lambda_1=\lambda_2$), or *heterogeneous*, where both UEs have different arrival rates ($\lambda_1\neq\lambda_2$). Similarly, channel states are either *homogeneous*, where both UEs' channel states belong to $\mathcal{S}_h^1$, or *heterogeneous*, where UE i's channel states belong to $\mathcal{S}_h^i$ ($i=1,2$). We implement value iteration as described in Algorithm 1 to obtain optimal policies using every cost/reward function in Section IV under different arrival rates, channel states, and overflow penalty factors (α). We evaluate each scheduling policy based on its average packet delay and packet drop rate. All the results are averaged over 30 epochs of 50,000 TTIs.

TABLE I: Simulation Parameters

Parameter	Value
Number of UEs, N	2
Number of RBs	1
Packet Length, L	200 bits
Buffer Size, X	10 pkts
TTI, $\triangle t$	1 ms
RB Bandwidth, B	180 kHz
Average Arrival Rate, λ	$\{100, 125, 150, 175, 200, 225, 250, 275\}$ kbps
Channel States (SNR), $\mathcal{S}_h^1, \mathcal{S}_h^2$	{3.22, 7.37, 10.48, 13.83 }, {20.02, 22.4, 25.56, 29.72 } dB
Discount Factor, γ	0.98
Overflow Penalty Factor, α	$\{0, 20, 40, 60, 80, 100\}$
Number of Epochs	30
Number of TTIs	50000

Fig. 2 shows the packet drop percentage (%) versus average packet delay (ms) of different policies under homogeneous states with overflow penalty factor $\alpha=0$. Figs. 2a and 2b show results for homogeneous and heterogeneous arrivals, respectively. It is clear that P5 achieves better delay and overflow performance than Max-Weight under all tested arrival rates. P1-P4 perform similar to each other and incur lower average packet delays but higher packet loss rates than P5 and Max-Weight. For example, in Fig. 2a, when $\lambda_1=\lambda_2=250$ kbps, P5 shows a 6.18% decrease in delay and 9.14% decrease

in packet drops compared to Max-Weight, while P1-P4 show a $7.56-8.05\%$ decrease in delay but experience a $58.31-63\%$ increase in packet drops compared to P5. Similarly, in Fig. 2b, when $\lambda_1 = 250$ kbps and $\lambda_2 = 175$ kbps, P5 shows a 14.57% decrease in packet drops and 7.2% decrease in delay compared to Max-Weight, while P1-P4 show a modest $4.78 - 7.46\%$ decrease in delay but experience an $85.28-150.17\%$ increase in packet drops compared to P5.

Fig. 3 is set up the same as Fig. 2, but it shows results under heterogeneous channel states, where UE 2 has better channel states than UE 1. As before, P5 achieves better delay and overflow performance than Max-Weight under the tested arrival process while P1-P4 perform similar to each other and incur lower average packet delays but higher packet loss rates than P5 and Max-Weight. For example, in Fig. 3a, when $\lambda_1 = \lambda_2 = 250$ kbps, P5 shows a 19.15% decrease in delay and a 77.64% decrease in packet drops compared to Max Weight, while P1-P4 show a marginal $2.63 - 3.07\%$ decrease in delay but experience a $127.47-157.14\%$ increase in packet drops compared to P5. Similarly, in Fig. 3b, P5 shows a 76.39% decrease in packet drops and 17.71% decrease in delay compared to Max-Weight, while P1-P4 show a negligible $1.35 - 2.24\%$ decrease in delay but experience $123.61 - 201.39\%$ increase in packet drops compared to P5.

Overall, Figs. 2 and 3 show that P5's performance relative to Max-Weight and P1-P4 improves when arrival rates and/or channel conditions are heterogeneous rather than homogeneous. Max-Weight has higher delays under heterogeneous arrival rates and channel conditions because, as noted earlier, it is only delay optimal under specific homogeneous arrival and channel processes. P1-P4 have poor overflow performance because, without any overflow penalty ($\alpha = 0$), the sum delay across UEs can be minimized by allowing more packet drops.

We now examine how varying the overflow penalty factor affects the delay and overflow performance of the policies. Fig. 4 shows the packet drop percentage (%) versus average packet delay (ms) of P1-P5 and Max-Weight under homogeneous channel conditions and fixed arrival rates for overflow penalty factors $\alpha = 0 - 100$. Figs. 4a and 4b show results for homogeneous ($\lambda_1 = \lambda_2 = 250$ kbps) and heterogeneous arrivals ($\lambda_1 = 250$ Kbps and $\lambda_2 = 175$ kbps), respectively. Different overflow penalty factors are distinguished by different markers. Since Max-Weight does not include an overflow penalty, it is represented with only one data point in each subfigure (a cyan $\times$). There are two regions in each subfigure where values are difficult to distinguish. We use an inset plot to highlight the left-most crowded region in each subfigure since the variation in the right-most region is relatively small.

Under homogeneous arrivals (Fig. 4a), increasing α decreases P5's packet drop rate by $0.26-1.73\%$ while increasing its delay by $4.38 - 5.6\%$. Similarly, under heterogeneous arrivals (Fig. 4b), increasing α decreases P5's packet drop rate by $1.68 - 4.36\%$ while increasing its delay by $7.19 - 8.38\%$. These results suggest that using P5 with $\alpha > 0$ has a relatively small impact on its performance as it cannot dramatically improve its already low packet drop rate.

P1, P3, and P4 behave similarly as α increases: their packet drop rates decrease and their packet delays increase. We observe that, in contrast to P5 with $\alpha = 0$, under P1, P3, and P4 the overflow penalty factor requires some tuning to simultaneously achieve low delay and low packet overflows. P2 also behaves similar to P1, P3, and P4, but it is much more sensitive to the overflow penalty factor. This is because, under P2, the expected future buffer costs are scaled by $1 - \gamma = 1 - 0.98 = 1/50$ (see the expected cost expression after (12)) and therefore get dominated by the overflow cost. We have also analyzed the impact of the overflow penalty factor on the delay and packet drop rates under heterogeneous channel states. The results are very similar to the homogeneous channel state case, and are omitted due to space constraints.

VI. Conclusion

We carried out a systematic analysis of the delay and overflow performance of different cost/reward functions used for minimizing the delay in a single-carrier multi-user downlink RRS scenario. Towards this objective, we formulated the related multi-user downlink RRS problem as an MDP (i.e., the underlying mathematical model used to describe environments in which DRL agents learn to act). In our evaluation, we compared the performance of each policy with Max-Weight as a benchmark rule-based scheduler and identified that a reward function that aims to maximize the number of transmitted packets over time frequently achieves near optimal delay and superior overflow performance among the investigated options, and does not require any tuning. This reward function achieves up to 19.15% lower delay and up to 77.64% fewer packet drops than Max-Weight. In future work, we will extend our investigation using DRL, while considering more than two UEs, more than one RB, and larger packet buffers.

References

[1] 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.

[2] W. AlQwider, T. F. Rahman, and V. Marojevic, "Deep Q-network for 5G NR downlink scheduling," in *IEEE ICC Workshops*, 2022, pp. 312–317.

[3] M. Kim, S. Hwang, and I. Lee, "Deep reinforcement learning approach for fairness-aware scheduling in wireless networks," in *13th Int. Conf. on Inf. and Commun. Technol. Convergence (ICTC)*, 2022, pp. 1229–1232.

[4] 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.

[5] F. AL-Tam, A. Mazayev, N. Correia, and J. Rodriguez, "Radio resource scheduling with deep pointer networks and reinforcement learning," in *IEEE 25th Int. Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2020, pp. 1–6.

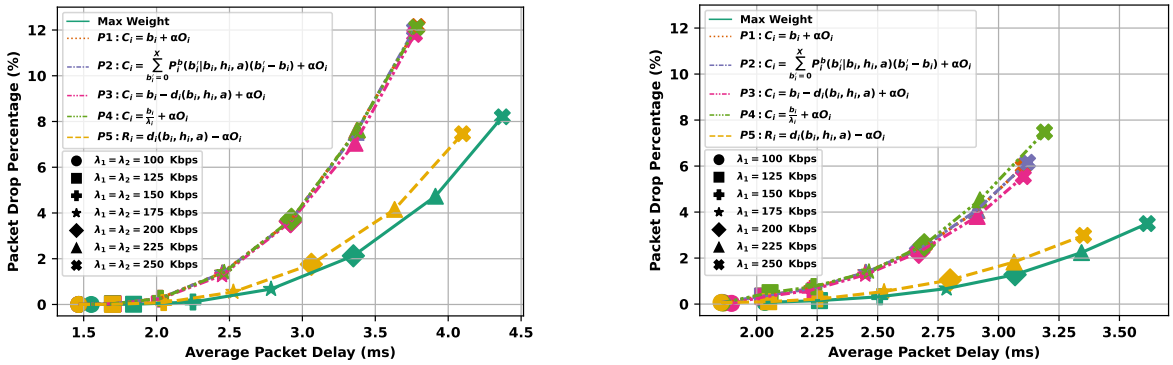
[6] J. Wang et al., "Deep reinforcement learning for scheduling in cellular networks," in *11th Int. Conf. on Wireless Commun. and Signal Process. (WCSP)*, 2019, pp. 1–6.

[7] H. Chang, R. Sree, H. Chen, J. Zhang, and L. Liu, "MADRL based scheduling for 5G and beyond," in *IEEE MILCOM*, 2022, pp. 873–878.

[8] Z. Gu and et al., "Knowledge-assisted deep reinforcement learning in 5G scheduler design: From theoretical framework to implementation," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2014–2028, 2021.

[9] N. Sharma et al., "Deep reinforcement learning for delay-sensitive LTE downlink scheduling," in *IEEE PIMRC*, 2020, pp. 1–6.

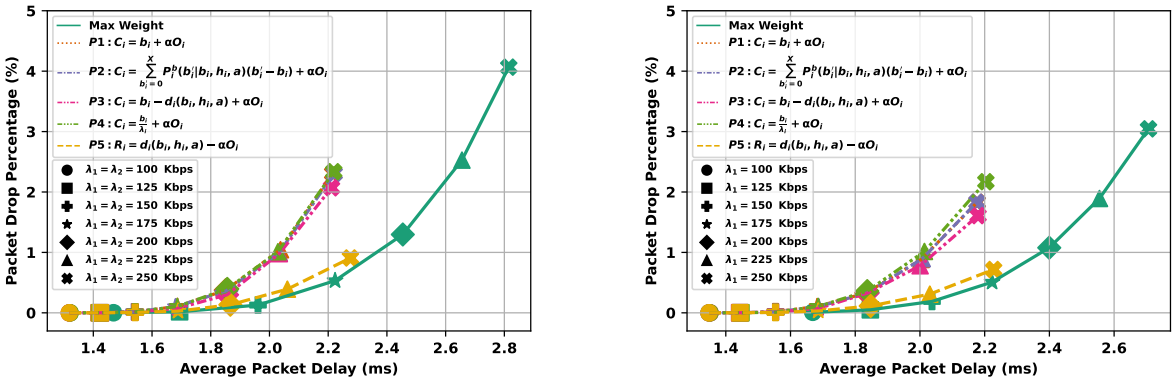
[10] A. Robinson and T. Kunz, "Downlink scheduling in LTE with deep reinforcement learning, LSTMs and pointers," in *IEEE MILCOM*, 2021, pp. 763–770.



(a) Homogeneous arrivals ($\lambda_1 = \lambda_2$)

(b) Heterogeneous arrivals ($\lambda_2 = 175$ Kbps)

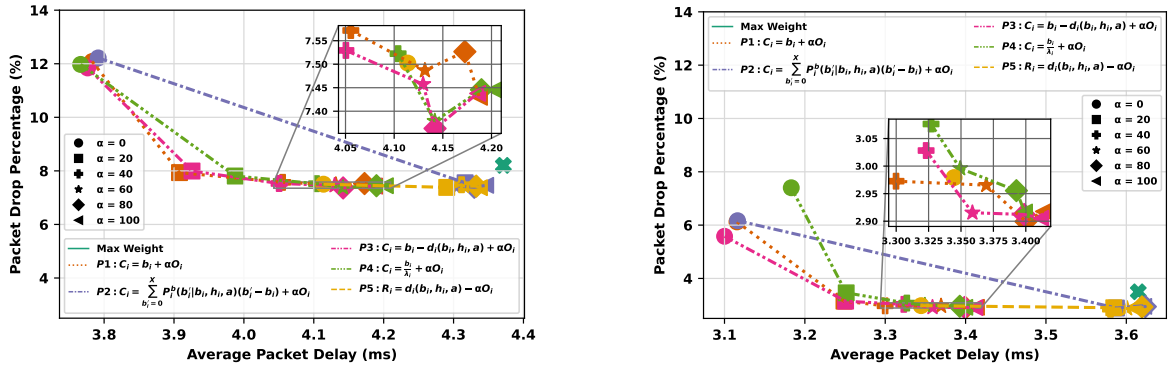
Fig. 2: Homogeneous channel states with overflow penalty factor $\alpha = 0$.



(a) Homogeneous arrivals ($\lambda_1 = \lambda_2$)

(b) Heterogeneous arrivals ($\lambda_2 = 175$ Kbps)

Fig. 3: Heterogeneous channel states with overflow penalty factor $\alpha = 0$.



(a) $\lambda_1 = \lambda_2 = 250$ Kbps

(b) $\lambda_1 = 250$ Kbps and $\lambda_2 = 175$ Kbps

Fig. 4: Homogeneous channel states with overflow penalty factor $\alpha \in \{0, 20, 40, 60, 80, 100\}$.

[11] 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.

[12] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, 2007.

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

[14] N. Salodkar et al., "An on-line learning algorithm for energy efficient delay constrained scheduling over a fading channel," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 4, pp. 732–742, 2008.

[15] N. Salodkar, A. Karandikar, and V. S. Borkar, "A stable online algorithm for energy-efficient multiuser scheduling," *IEEE Trans. Mobile Comput.*, vol. 9, no. 10, pp. 1391–1406, 2010.

[16] N. Sharma, N. Mastronarde, and J. Chakareski, "Accelerated structure-aware reinforcement learning for delay-sensitive energy harvesting wireless sensors," *IEEE Trans. Signal Process.*, vol. 68, 2020.

[17] N. Mastronarde and M. van der Schaar, "Fast reinforcement learning for energy-efficient wireless communication," *IEEE Trans. Signal Process.*, vol. 59, no. 12, pp. 6262–6266, 2011.

[18] F. Fu and M. Van der Schaar, "Structure-aware stochastic control for transmission scheduling," *IEEE Trans. Veh. Technol.*, 2012.

[19] D. V. Djonin and V. Krishnamurthy, "MIMO transmission control in fading channels – a constrained Markov decision process formulation with monotone randomized policies," *IEEE Trans. Signal Process.*, vol. 55, no. 10, Oct. 2007.