Deep Reinforcement Learning for Resource Constrained Multiclass Scheduling in Wireless Networks

Apostolos Avranas, Philippe Ciblat, *Senior Member, IEEE,*
and Marios Kountouris, *Fellow, IEEE*.

***Abstract*—The problem of multiclass scheduling in a dynamic wireless setting is considered here, where the available limited bandwidth resources are allocated to handle random service demand arrivals, belonging to different classes in terms of payload data request, delay tolerance, and importance/priority. In addition to heterogeneous traffic, another major challenge stems from random service rates due to time-varying wireless communication channels. Existing scheduling and resource allocation approaches, ranging from simple greedy heuristics and constrained optimization to combinatorics, are tailored to specific network or application configuration and are usually suboptimal. On this account, we resort to deep reinforcement learning (DRL) and propose a distributional Deep Deterministic Policy Gradient (DDPG) algorithm combined with Deep Sets to tackle the aforementioned problem. Furthermore, we present a novel way to use a Dueling Network, which leads to further performance improvement. Our proposed algorithm is tested on both synthetic and real data, showing consistent gains against baseline methods from combinatorics and optimization, and state-of-the-art scheduling metrics. Our method can, for instance, achieve with 13% less power and bandwidth resources the same user satisfaction rate as a myopic algorithm using knapsack optimization.**

***Index Terms*—Deep reinforcement learning, deep sets, QoS traffic scheduling, multiclass services, dynamic resource allocation.**

I. Introduction

Scheduling and resource allocation are two relevant and widely studied problems with a plethora of practical applications in various fields, ranging from computing systems and production planning to project management and logistics. Optimal resource allocation, together with the associated scheduling task, is one of the main challenges and requirements for the design of communication networks. How efficiently the available resources, such as subbands, time slots, beams, and transmit power, are managed and which users are scheduled for service have a direct impact on the communication system performance.

In this paper, we investigate the problem of scheduling and resource allocation in wireless networks. A base station (BS) sends data traffic to mobile users, which have different application-dependent Quality of Service (QoS) requirements. We consider applications that require the delivery of large amounts of data without any strict deadline, e.g., enhanced mobile broadband (eMBB) service category, as well as time-sensitive or mission-critical ones involving low payload packets that have to be reliably received within a stringent latency constraint, i.e., Ultra-Reliable and Low Latency Communications (URLLC) service category. The increased heterogeneity in users' traffic and the diverse service requirements substantially complicate the provisioning of high fidelity, personalized service with QoS guarantees. Our aim is to design a generic architecture and efficient algorithms, which take as inputs the specific constraints of the traffic/service class where each user belongs to and as output the set of users to serve, as well as the allocated resources and time slots, with the objective to maximize the number of satisfied users.

The problem considered is hard to solve due to several major technical challenges. First, with the exception of very few special cases, there is no simple closed-form expression for the problem and *a fortiori* for its analytical solution. Second, optimization algorithms that solve the problem have to be computationally efficient and implementable in large-scale wireless networks. Applying optimal methods from combinatorial optimization, such as branch and bound algorithm [1], results in solutions exhibiting prohibitively high computational complexity and being hard or impossible to meaningfully scale with the number of active users. Other existing approaches relying heuristics, approximations, or relaxations, provide suboptimal solutions, which seem to work satisfactorily in specific scenarios but fail to perform close to optimal in general cases and in the regime of large number of users. Moreover, the proliferation of new use cases makes the problem of efficient and scalable scheduling and resource allocation more intricate. This will be exacerbated with the advent of the emerging mobile systems (Beyond 5G/6G), which could involve high-dimensional optimization domains, various application scenarios, as well as heterogeneous, often conflicting, QoS requirements. This motivates the quest for alternative methods.

In this work, we resort to Deep Reinforcement Learning (DRL) for efficient and scalable scheduling and resource allocation algorithmic solutions. DRL has recently attracted considerable attention for it ability to provide very promising results in complex problems obeying strict game rules (e.g., Atari, Chess, Go [2]–[4]) or physical laws (robotics and physics-related tasks [5], [6]). In cloud service provision, DRL has been used to schedule incoming tasks to servers according to

A. Avranas is with Amadeus SAS, F-06902 Sophia Antipolis, France. Email: apostolos.avranas@amadeus.com. P. Ciblat is with LTCI, Telecom Paris, Institut Polytechnique de Paris, F-91120 Palaiseau, France. Email: philippe.ciblat@telecom-paris.fr. M. Kountouris is with the Communication Systems department, EURECOM, F-06904 Sophia-Antipolis, France. Email: marios.kountouris@eurecom.fr

their heterogeneous CPU and memory requirements [7]. DRL approaches have recently shown interesting gains in wireless communication systems [8]–[11], and has also been applied to wireless scheduling and resource allocation problems [12]–[19]. Nevertheless, none of previous works has studied the problem we consider here, that of multiclass scheduling with heterogeneous QoS constraints. Furthermore, in contrast to most prior work and in order to harness the high level of stochasticity, we consider distributional DRL [20]–[22] to obtain richer representations of the environment, which in turn yields better solutions. Our setup, which mainly focuses on time-sensitive traffic, involves two important elements that make our problem challenging and original: (i) *strict latency constraints*: each user demands to be satisfied before a specific deadline. This is a relevant scenario in 5G and beyond systems (e.g., in URLLC scenarios and mission-critical services.); and (ii) *heterogeneity*: each user has diverse QoS requirements, i.e., different strict latency constraints and data requirements. To the best of our knowledge, such setup has not be addressed (in particular considering deep learning techniques) in the sense that the proposed algorithm is by design adapted to this strict latency constraint and QoS heterogeneity.

The novelty of this paper is twofold. First, we revisit and study the problem of multiclass scheduling with heterogeneous QoS constraints and dynamic resource allocation in wireless systems using DRL. Second, we propose a DRL architecture for solving wireless networking problems with stochasticity, which judiciously and non-trivially combines several advanced DRL techniques proposed in the machine learning community. In particular, we leverage (i) noisy networks technique [23] for better explorations; (ii) dueling network architectures [24] for improved stability of the trained models; and (iii) deep sets [25] for simplifying and improving neural network models when permutation invariance properties apply. We combine these three ingredients with a deep deterministic policy gradient method [26] to propose a highly efficient general architecture and scheduling/resource allocation algorithm. In a setup similar to ours and Nokia's challenge [27], deep deterministic policy gradient is used to allocate the bandwidth to incoming data traffic in [28]. Nevertheless, unlike our work, [28] considers only full channel state information (CSI), a single traffic class, and only few users (typically less than 15). In [29] Graph Neural Networks, a similar technique to Deep Sets, are used to increase the number of users but they do not consider traffic of users. Initial attempts to solve the problem of scheduling traffic for users with heterogeneous performance requirements can be found in [30], considering though only full CSI and a limited number of users.

A. Contributions

The main contributions of this work can be summarized as follows:

- We propose and develop a DRL scheme having two important architectural components that facilitate a stable training even in the case of high traffic from a very large number of users. First, we leverage *Deep Sets* [26] as a means to exploit the permutation equivariance property of the problem and to drastically reduce the number of necessary parameters. Second, we introduce a *user normalization* trick capturing the resource-constrained attribute of our problem, namely that the available bandwidth resources are limited. We show that without those crucial steps, the performance drops significantly.
- We further improve the system performance using distributional DRL [22] and reward scaling as implemented in [31]. Finally, we harvest additional gains by adapting the idea of dueling networks [24] used in Deep Q-Networks (DQN) to distributional RL by modifying the output to represent the distribution of the return of the agent's action.
- We demonstrate that our proposed DRL architecture can easily be implemented with minor changes in both extreme cases of channel knowledge, namely full CSI and no CSI.
- We compare the performance of our DRL solution with strong baselines and state-of-the-art scheduling metrics, namely the exponential rule [32]:
  - In the *full CSI* case, the DRL scheduler reaches the same performance but with 13% less power and bandwidth requirements as compared to a optimal myopic algorithm solving a reformulation of our problem as a knapsack one. Furthermore, the proposed DRL scheduler operates close to an upper bound, devised as an oracle knowing all future traffic characteristics and finding the optimal resource allocation policy via Integer Linear Programming (ILP).
  - In the *no CSI* case, our model-free DRL scheme significantly outperforms a model-based baseline, which employs the Frank-Wolfe (FW) algorithm [33] guaranteeing a local optimum solution and is favored by knowing the statistics of the problem.

The paper is organized as follows: in Section II, we introduce the system model including the channel and traffic model. In Section III, we formulate the optimization problem and Section IV is devoted to the main contribution of the paper, that of the design a new DRL scheduler for heterogeneous multiclass traffic. In Section V, baseline algorithms, for performance comparison, are presented. In Section VI, we provide experimental results with both synthetic and real data, and Section VII concludes the paper.[1]

II. System Model

A. Network and channel model

We consider the downlink of a communication system, in which a BS serves multiple users by sending data over a wireless random time-varying channel. Users are uniformly distributed within two concentric rings of radii d_{min} and $\mathrm{d}_{max} > \mathrm{d}_{min}$. Therefore, the distance of a user u from the BS is a random variable with probability density function (PDF) $f_{\mathrm{d}}(\mathrm{d}_u) = \frac{2\mathrm{d}_u}{\mathrm{d}_{max}^2 - \mathrm{d}_{min}^2}, \mathrm{d}_u \in [\mathrm{d}_{min}, \mathrm{d}_{max}]$. We assume that mobility is not very high, so that BS-user distances remain constant during the time interval users are active.

Orthogonal frequency bands are assigned to simultaneously served users, hence there is no interference among them. Users

[1]Code is available at https://github.com/avranasa/DRL_Scheduling_Communications.

experience frequency flat fading, i.e., the channel gain of a user remains constant during a time slot and throughout all assigned frequency bands. Let a user u that has entered the system at time t_0. Its channel gain at time t is given by $g_{u,t} = \frac{C_{pl}|h_{u,t}|^2}{\sigma_N^2}\mathrm{d}_u^{-n_{pl}}$, where n_{pl} denotes the pathloss exponent, C_{pl} is a constant accounting for constant losses, and σ_N^2 is the noise power spectrum density. The small-scale fading $h_{u,t}$ evolves over time according to the following Gauss-Markov model

$$h_{u,t} = \rho h_{u,t-1} + N \tag{1}$$

where $h_{u,t_0} \sim \mathcal{CN}(0,1)$ (circular complex normal distribution with zero mean and unit variance), and $N \sim \mathcal{CN}(0, 1-\rho^2),\ t > t_0$. The parameter $\rho = J_0(2\pi f_d T_s) \in [0,1]$ [34] determines the time correlation of the channel, with $J_0(\cdot)$ denoting the zeroth-order Bessel function of the first kind, f_d the maximum Doppler frequency (determined by the user mobility), and T_s the time slot duration. If $\rho = 0$ (high mobility), a user experiences an independent realization of the fading distribution at each time slot (i.i.d. block fading). If $\rho = 1$ (no mobility), channel attenuation is constant throughout the user's lifespan (no small-scale fading fading).

We consider the following two cases for the channel state information (CSI): (i) *full-CSI*, in which h_{u,t_c} and the users' locations (and so d_u) are perfectly known at the BS for time t_c, thus enabling accurate estimation of the exact resources each user requires; (ii) *no-CSI*, in which the scheduler is completely channel-agnostic, both in terms of instantaneous fading realization and long-term channel statistics. In case of unsuccessful and/or erroneous data reception, a simple retransmission protocol (Type-I Hybrid Automatic Repeat Request (HARQ)) is employed. A packet is discarded whenever the user fails to correctly decode it (no buffering at the receiver side) and the BS will attempt to send it again in some subsequent slot.

Remark 1: For a non-trivial implementation of the Frank-Wolfe (FW) algorithm, which serves as a baseline for comparison in the the no-CSI case, we need to consider some kind of CSI. For that, we consider the case of *statistical CSI*, where the scheduler knows the statistics of the users' channels and locations. Our proposed DRL algorithm will always operate under full absence of CSI, but all statistics can effectively be learned through the training phase.

B. Traffic model

We consider a generic yet tractable traffic model, in which users with diverse data and latency requirements arrive and depart from the system. There is a set of service classes $\mathcal{C}$ to which a user entering the system belongs to with probability p_c. Each user in class $c \in \mathcal{C}$ is characterized by the tuple (D_c, L_c, α_c) as follows:

- Data size D_c: the number of information bits requested by a user belonging to class c, which have been encompassed into a packet of size D_c.
- Maximum Latency L_c: the maximum number of time slots within which the user has to be satisfied, i.e., to successfully receive its data packets of size D_c.
- Importance α_c: an index allowing the scheduler to prioritize certain service classes, e.g., users with privileged contracts (e.g., high-value Service-Level Agreement (SLA)) may demand better service and higher reliability.

We assume that a maximum number of users K can coexist per time slot and that a new user may arrive only after the departure of a user that exceeded the maximum time allowed to remain in the system. That way, the scheduling decisions do not influence the arrival process, although the packet arrival process may be determined by the scheduler in many cases. For example, if a user arrives at time $t_0 = 1$, belonging to class $c \in \mathcal{C}$ with $L_c = 4$, then even if it successfully receives its requested packet of size D_c at $t = 1$, a new arrival may randomly be generated only at time $t = t_0 + L_c = 5$ and afterwards. The rationale behind adopting this model is as follows. If a new arrival is generated right after a previous user is satisfied (in the example at time $t = 2$), then the traffic load is affected by the scheduler performance. The faster the scheduler serves the users, the more arrivals occurs. In contrast, in our model, the arrival process and its statistics remain uninfluenced by the scheduling decisions and the available resources. Therefore, at every time slot, the set of users U_t ($|U_t| \leq K$) contains all users waiting to receive their requested data while remaining within their latency constraint. Finally, to ensure random inter-arrival times, we assert that the probability $p_{null} = 1 - \sum_{c\in\mathcal{C}} p_c$ is positive , i.e., $p_{null} > 0$, leaving a probability that no user appears in a time slot.

C. Service Rate

The service rate is measured using Shannon rate expression assuming capacity-achieving codes. The achievable service rate of user u at time t is equal to $w_{u,t}\mathcal{R}_{u,t}$, where $\mathcal{R}_{u,t} = \log_2(1 + g_{u,t}\mathrm{P}_{u,t}) = \log_2(1 + \kappa_u|h_{u,t}|^2)$ (bps/Hz), with $\mathrm{P}_{u,t}$ denoting the transmit energy per symbol (channel use), $w_{u,t}$ the assigned bandwidth (in Hz), and $\kappa_u = \frac{C_{pl}}{\sigma_N^2}\mathrm{d}_u^{-n_{pl}}$. Let a user at distance d_u from the BS, belonging to class $c \in \mathcal{C}$, is served at time t_u with resources $(w_{u,t}, \mathrm{P}_{u,t})$. The probability of unsuccessful transmission is given by

$$\begin{aligned} P_u^{\text{fail}}(w_{u,t}, \mathrm{P}_{u,t}; \mathrm{d}_u) &= \mathbb{P}(w_{u,t}\mathcal{R}_{u,t} < \overline{D}_u|\mathrm{d}_u) \\ &= \mathbb{P}(|h_{u,t}|^2 < \zeta_{u,t}\mathrm{d}_u^{n_{pl}}) = 1 - e^{-\zeta_{u,t}\mathrm{d}_u^{n_{pl}}} \end{aligned} \tag{2}$$

where $\zeta_{u,t}=\frac{\sigma_N^2(2^{\overline{D}_u/w_{u,t}}-1)}{C_{pl}\mathrm{P}_{u,t}}$. If d_u is not known to the scheduler, we have

$$\begin{aligned} P_u^{\text{fail}}(w_{u,t}, \mathrm{P}_{u,t}) &= \mathbb{P}(w_{u,t}\mathcal{R}_{u,t} < \overline{D}_u) \\ &= \int_{\mathrm{d}_{min}}^{\mathrm{d}_{max}} P_u^{\text{fail}}(w_{u,t}, \mathrm{P}_{u,t}; \mathrm{d}) f_{\mathrm{d}}(\mathrm{d}) d\mathrm{d} \\ &= 1 - \frac{\Gamma(\frac{2}{n_{pl}}, \zeta_{u,t}\mathrm{d}_{min}^{n_{pl}}) - \Gamma(\frac{2}{n_{pl}}, \zeta_{u,t}\mathrm{d}_{max}^{n_{pl}})}{n_{pl}\zeta_{u,t}^{2/n_{pl}}(\mathrm{d}_{max}^2 - \mathrm{d}_{min}^2)/2} \end{aligned} \tag{3}$$

where $\Gamma(s,x) = \int_x^\infty t^{s-1}\,\mathrm{e}^{-t}\,\mathrm{d}t$ is the upper incomplete gamma function and $\overline{D}_u = D_u/T_s$ is the "normalized data/packet size". For exposition convenience, we overload notation by allowing u in $\overline{D}_u, D_u, L_u, \alpha_u$ to denote either a class u or a user u belonging to a class with those characteristics.

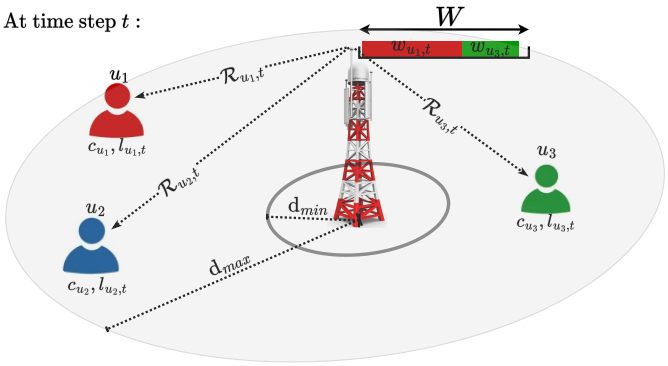


Figure 1: An instance of our system model at time step t. There are three active users $u_i, i \in \{1, 2, 3\}$. BS has available bandwidth W, which is allocated to any subset of active users. At time t, the allocated bandwidth to user u_i is denoted by $w_{u_i,t}$. Each user has its own service rate $\mathcal{R}_{u_i,t}$ that depends on its channel quality $h_{u_i,t}$ (fading coefficient) and distance d_{u_i} from the BS. User u_i requests a packet of data size D_{u_i} which depends on the class c_{u_i} it belongs to. At time t, user u_i demands to be served before $l_{u_i,t}$ time slots pass and to be satisfied at time t, bandwidth $w_{u,t} > \overline{D}_u/\mathcal{R}_{u,t}$ needs to be allocated to it. The objective of the BS's scheduler is to maximize the number of satisfied users throughout the time horizon.

III. Problem Statement

We consider the problem of heterogeneous scheduling and resource allocation, which involves a BS handling a set of randomly arriving service requests belonging to different classes with heterogeneous requirements. Each class defines the requirements and the expected Quality of Service (QoS) guarantees for its users. Observing this time-varying set of heterogeneous requests, the objective of the scheduler at each time slot is two-fold: (i) carefully select which subset of user requests to satisfy, and (ii) allocate the finite resources amongst the selected user requests. The performance metric to maximize is the long-term importance-based weighted sum of successfully satisfied requests. A request is considered to be satisfied whenever the user has received the requested data within the maximum tolerable latency specified by its service class.

The scheduling problem at hand can be formulated as a Markov Decision Process (MDP) [35] $(\mathcal{S}, \mathcal{A}, R, P, \gamma)$, where $\mathcal{S}$ is the state space of the environment and $\mathcal{A}$ is the action space, i.e., the set of all feasible allocations in our case. After action $a_t \in \mathcal{A}$ at state $s_t \in \mathcal{S}$, a reward $r_t \sim R(\cdot|s_t, a_t)$ is obtained and the next state follows the probability $s_{t+1} \sim P(\cdot|s_t, a_t)$. The discount factor is $\gamma \in [0, 1)$. Under a fixed policy $\pi : \mathcal{S} \to \mathcal{A}$ determining the action at each time step, the *return* is defined as the random variable

$$Z_t^\pi = \sum_{i=0}^{\infty} \gamma^i r_{t+i} \tag{4}$$

which represents the discounted sum of rewards when a trajectory of states is taken following π. Ideally, the aim is to find the optimal policy $\pi^\star$ that maximizes the mean reward $\mathbb{E}_\pi[Z_t^\pi]$.

At each time step t, a set of users $u \in U_t$ is waiting for service, where each user therein belongs to a class $c_u \in \mathcal{C}$ described by (D_c, L_c, α_c). A user u appearing at time t_0 has a "lifespan" $t \in [t_0, t_0 + L_{c_u} - 1]$ and at time $t = t_0 + L_{c_u}$ a new user belonging to class c might arrive with probability p_c. The amount of resources given to user u at time t is $w_{u,t}$. If at any time t, $w_{u,t} > \overline{D}_u/\mathcal{R}_{u,t}$ then user u is satisfied. Since resources are limited (finite), $\sum_{u \in U_t} w_{u,t} \leq W, \quad \forall t$, no more than W resources in total can be spent per time slot. Summing up,

- State: $s_t = \{\forall u \in U_t : c_u, \mathcal{R}_{u,t}, l_{u,t}\}$
- Action: $a_t = \{\forall u \in U_t : w_{u,t}\}$
- Reward: $r_t = \sum_{u \in U_t} \alpha_u \mathbb{1}\{w_{u,t}\mathcal{R}_{u,t} > \overline{D}_u\}$

where $l_{u,t} \leq L_u$ is the remaining number of time slots within which user u (i.e. $u \in U_t$) expects to successfully receive its packet and $\mathbb{1}\{\cdot\}$ denotes the indicator function. Note that knowing the class c_u to which user u belongs, implies knowing the requirements (D_u, L_u, α_u). An inherent attribute of this MDP is the *permutation equivariance* of an optimal policy, meaning that if we permute the indexing of the users, then permuting likewise the allocation of the resources retains the performance of the policy. For that, in our DRL approach, we only consider permutation equivariant policies, and as a consequence we need a permutation *invariant function* to evaluate and train the policy.

In this work, we focus on bandwidth allocation, assuming a fixed amount of energy spent per channel use and no power adaptation, i.e., $\mathrm{P}_{u,t} = \mathrm{P}, \forall u, t$. Specifically, for total bandwidth W, the scheduler aims at finding the $(w_{u_1,t}, w_{u_2,t}, ...) \in \mathbb{R}_{\geq 0}^{|U_t|}$ with $u_1, u_2, ... \in U_t$ and $\sum_{u \in U_t} w_{u,t} \leq W, \forall t$, so as to maximize the accumulated reward for every satisfied user over a finite time horizon. The expected reward is described by the following objective "gain-function"

$$G = \sum_t \sum_{u \in U_t} \alpha_u \mathbb{1}\{w_{u,t}\mathcal{R}_{u,t} > \overline{D}_u\}. \tag{5}$$

We stress out that a user u remains on the set U_t for a time interval less or equal to the maximum acceptable latency L_u. If not satisfied within that interval, then it does not contribute positively to the objective G.

Note that $\mathcal{R}_{u,t}$ satisfies the Markov property since $h_{u,t}$ follows a Markov model. Under full CSI, the agent (here the BS) fully observes the state s_t, while in the no-CSI case, $h_{u,t}$ is unknown resulting in a Partially Observable MDP (POMDP) [36]. One way to transform a POMDP into a MDP is by substituting the states with the "belief" of the states [37]. Another way is to use the complete history $\{o_0, a_0, o_1, a_1, \cdots, a_{t-1}, o_{t-1}\}$, with $o_t \subset s_t$ being the agent's observation. Notice that only the most recent part is relevant as users that have already left the system do not affect the way the channels of the current users evolve or the generation of future users or in general the current and future system dynamics. Therefore, we can safely consider the scheduling and allocation history of only the current users. Specifically, if $\boldsymbol{w}_{u,t} = (w_{u,t_0},$

$w_{u,t_0+1}, ..., w_{u,t})$ is the scheduling history of user u then the input of the agent is $\{\forall u \in U_t : D_u, L_u, a_u, \kappa_u, l_{u,t}, \boldsymbol{w}_{u,t}\}$.

IV. Proposed Deep Reinforcement Learning Architecture

In this section, we propose a novel DRL approach for solving the aforementioned multiclass scheduling and resource allocation problem. Our method is built upon the model-free Deep Deterministic Policy Gradient algorithm [26], which uses two neural networks. The first one is called policy network and takes as input the state of the traffic and outputs an action which in our case is the bandwidth allocation. The second network evaluates how good are the actions of the policy network and trains it accordingly. Despite the highly challenging dynamics and stochasticity (wireless channel and heterogeneous traffic) of the problem, we show that DRL can provide performance gains although it is impossible to accurately predict the number of users, their service demands, and their channel/link characteristics even after few steps. Two crucial components of our proposed network architecture are: (i) DeepSets [25], which allows to handle a big number of users without requiring to also increase the number of parameters of the networks; and (ii) user normalization operation that stabilizes the training by instilling in the architecture the notion of limited resources that should be shared among many users.

A. Policy Network

Our objective is to build a scheduler that can handle a large number of users K, even in the order of hundreds. Moreover, we require that our method works in both full CSI and no CSI cases with minor - if any - modifications. A widely used approach is Deep Q-learning Network (DQN). However, it is not feasible to employ DQN in our case since it needs a Neural Network (NN) architecture with a number of outputs equal to the number of possible actions and the action space is extremely large (in statistical CSI it is even infinitely large). For that, we resort to a Deep Deterministic Policy Gradient method [26], which trains a policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$ modeled as a NN with parameters θ.

If at time t on state s_t the action a_t is taken followed by the policy π, then the return using (4) is given by

$$Z^\pi(s_t, a_t) = r_t + \gamma Z^\pi_{t+1}, \text{with } r_t \sim R(\cdot|s_t, a_t). \tag{6}$$

Note that if even at t the action a_t comes from policy π, then $Z^\pi(s_t, a_t = \pi(s_t)) = Z^\pi_t$. Let the expected return be

$$Q^\pi(s_t, a_t) = \mathbb{E}[Z^\pi(s_t, a_t)]. \tag{7}$$

Then, the objective of the agent is to maximize

$$J(\theta) = \mathbb{E}_{s_{t_0} \sim \mathrm{p}_{t_0}}[Q^{\pi_\theta}(s_{t_0}, \pi_\theta(s_{t_0}))], \tag{8}$$

with p_{t_0} being the probability of the initial state s_{t_0} at time t_0. The gradient can be written [38]

$$\nabla_\theta J(\theta) = \mathbb{E}_{s_{t_0} \sim \mathrm{p}_{t_0}, s \sim \rho^{\pi_\theta}_{s_{t_0}}}[\nabla_\theta \pi_\theta(s) \nabla_a Q^{\pi_\theta}(s, a)|_{a=\pi_\theta(s)}], \tag{9}$$

with $\rho^{\pi_\theta}_{s_{t_0}}$ being the discounted state (improper) distribution defined as $\rho^{\pi_\theta}_{s_{t_0}}(s) = \sum_{i=0}^{\infty} \gamma^i \mathbb{P}(s_{t+i} = s|s_{t_0}, \pi_\theta)$. In practice $\rho^{\pi_\theta}_{s_{t_0}}$ is approximated by the (proper) distribution $\varrho^{\pi_\theta}_{s_{t_0}}(s) := \sum_{i=0}^{\infty} \mathbb{P}(s_{t+i} = s|s_{t_0}, \pi_\theta)$. To compute the gradient, the function $Q^{\pi_\theta}(s, a)$ is needed, which is approximated by another NN $Q_\psi(s, a)$, named *value network*, described in the next subsection.

We now explain the architecture of the model π_θ. At first, the characteristics (or *features* as commonly termed in the machine learning literature) $F_i \in \mathbb{R}^{N_u}, i \in \{1, \cdots K\}$ of each user are processed individually by the same function $\phi_{user} : \mathbb{R}^{N_u} \to \mathbb{R}^{H_u}$ modeled as a two layer fully connected neural network. The computational complexity for evaluating ϕ_{user} for all K users is $\mathcal{O}(KN_uH_u + KH_u^2)$ since we choose H_u to be also the dimension of the hidden layer. Then, the outputs of ϕ_{user} that corresponds to the new features per user are combined with the permutation equivariant layer called "Deep Sets" [25].

1) Deep Sets: As discussed in Section III, we aim for a policy that is a permutation equivariant function (i.e., permuting the users should only result in permuting likewise the resource allocation). In [25], necessary and sufficient conditions are shown for permutation equivariance in neural networks; the proposed layer called Deep Sets therein is adopted here. Deep Sets are defined as a function $f_\sigma : \mathbb{R}^{K \times H} \to \mathbb{R}^{K \times H'}$:

$$f_\sigma(x) = \sigma\left(x\Lambda + \frac{1}{K}\mathbf{1}\mathbf{1}^\intercal x\Gamma\right), \qquad \Lambda, \Gamma \in \mathbb{R}^{H \times H'} \tag{10}$$

where $\mathbf{1} = [1, \cdots, 1] \in \mathbb{R}^K$, $\sigma(\cdot)$ is an element-wise nonlinear function and the matrices Λ, Γ contain the trainable parameters. This function receives for K users their feature vectors of dimension H and transforms them into feature vectors of dimension H'. The equivariance holds because swapping the i with the j row of the input matrix x (i.e., the features of user i with ones of user j) results in swapping likewise the i with the j row of the output matrix $y = f_\sigma(x)$. We stack two of those, one $f_{\mathrm{relu}} : \mathbb{R}^{K \times H_u} \to \mathbb{R}^{K \times H'_u}$ with $\sigma(\cdot)$ being the $\mathrm{relu}(x) = \max(0, x)$ and a second $f_{\mathrm{linear}} : \mathbb{R}^{K \times H'_u} \to \mathbb{R}^{K \times 1}$ without any nonlinearity $\sigma(\cdot)$. In addition to preserving the desirable permutation equivariance property, this structure also brings a significant parameter reduction, because the number of parameters of Deep Sets contained in Λ, Γ do not depend on the number of users K. Therefore, any increase in K does not necessitate additional parameters, which could lead to a much bigger network, prone to overfitting. The computational complexity for evaluating this deep set layer with an efficient implementation is $\mathcal{O}(KHH')$. The ϕ_{user} and the two deep sets contain all the training parameters and require most of the computations. Therefore, the complexity of the total policy network is defined by these components and is $\mathcal{O}(KN_uH_u + KH_u^2 + 2KHH')$.

2) Output: The activation function for the last layer of the policy network is a smooth approximation of $\mathrm{relu}(x)$, namely $\mathrm{softplus}(x) = \log(1 + e^x)$ restricting the output $\boldsymbol{y} \in \mathbb{R}^K$ to be positive. After that, depending on the existence of CSI, there are two ways of performing the allocation. For full CSI, the bandwidth required per user is accurately known. Therefore, we only need a binary decision per user (to serve or not), which will ruin though the differentiability of the policy, a mandatory property for DDPG to work. For that, we interpret the output $\boldsymbol{y}$

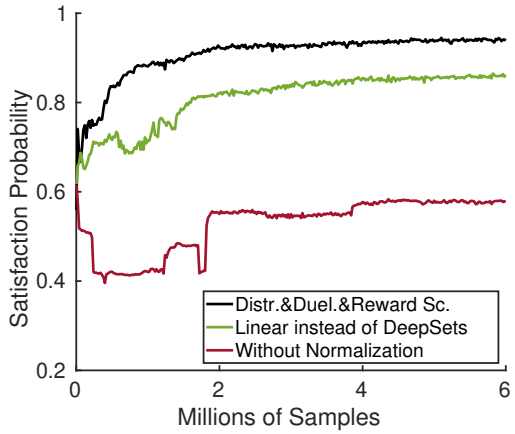


Figure 2: We conducted five experiments (with different seeds) for no CSI using the traffic model of Table Ia, a maximum number of users $K = 75$, $\rho = 0$, and resources (total bandwidth) $W = 5$ MHz. We depict here the average probability a user to be satisfied over those five experiments to carry an ablation study on the importance of the deep sets and user normalization step.

as a continuous relaxation of the binary problem. Specifically, $\boldsymbol{y}$ is the assignment to each user of a "value" per resources which after being multiplied by the number of resources the user requires, a user ranking is obtained. Then, the scheduler satisfies as many of the most "valuable" (highest rank) users as possible subject to available resources. Therefore, in full CSI, $\boldsymbol{y}$ semantically denotes how advantageous the policy believes is to allocate resource to each user. On the contrary, in the no CSI case, the action is not binary but continuous since the scheduler has to decide on the portion of the available resources each user takes. To ensure that $\boldsymbol{y}$ has the valid form of portions (i.e., positive and adding up to one) we just divide by the sum, $\boldsymbol{y} \to \frac{\boldsymbol{y}}{||\boldsymbol{y}||_1}$ (with $||\cdot||_1$ being the ℓ_1 norm)[2]. This discrepancy in the output process is the only minor difference in the considered model between full CSI and no CSI.

3) User normalization: Before the final nonlinearity of $\text{softplus}(x) = \log(1 + e^x)$, as seen in Figure 6, there is the crucial "user normalization" step $\boldsymbol{x} \to \frac{\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}]}{||\boldsymbol{x}||_2}, \boldsymbol{x} \in \mathbb{R}^K$ (with $||\cdot||_2$ denoting the ℓ_2 norm). Consider first the full CSI case. Without that step, the value network would perceive that the higher the "value" per resource assigned to a user, the more probable is for that user to get resources (and thus to be satisfied and receive reward). Unfortunately, this leads to a pointless interminable increase of every user's "value". What matters here is not the actual "value" of a user but how large this is relative to the rest of the users. To bring the notion of limited total resources, the "user normalization" subtracts from the value of each user the mean of all the users value. Hence, whenever the algorithm pushes the value of a single user to increase, the values of the rest decrease. In the no CSI case, there is an additional benefit. Since in the following step there is the operation $\boldsymbol{y} \to \frac{\boldsymbol{y}}{||\boldsymbol{y}||_1}$ so as the output to signify portions (of the total bandwidth), performing previously the normalization step (dividing by $||\boldsymbol{x}||_2$) helps keeping the denominator $||\boldsymbol{y}||_1$ stable.

In Figure 2 we show the significance of choosing the right architecture. It is clearly observed that *if either all Deeps Sets (in both policy and value network) are substituted by the most common choice of linear blocks or the user normalization step is removed, the performance degrade substantially.*

4) Exploration: Since the action a_t has to satisfy specific properties, such as positiveness and summing up to one for the no CSI case, the common approach of adding noise on the actions becomes rather cumbersome. An easy way out is through noisy networks [23], which introduce noise to the weights of a layer, resulting to change decisions for the policy network. The original approach considers the variance of the added noise to be learnable. Here, we instead keep it constant as it provided better results. With probability $P_{explore}$ we add noise to the parameters of ϕ_{users}, resulting to alter output features per user and therefore the policy outputs a different allocation. Specifically, if $\theta_{\phi_{users}}$ are the parameters of ϕ_{users}, then they are distorted as $\theta_{\phi_{users}}(1 + \sigma_{explore}\epsilon)$ with ϵ being normally distributed with zero mean and unit standard deviation and $\sigma_{explore}$ being a constant.

B. Value Network

As mentioned previously, $Q^{\pi_\theta}(s, a)$ is used for computing the gradient of the objective function described in (8). Since this is intractable to compute, a neural network, named value network, is used to approximate it. We compare three ways of employing the value network.

1) DDPG: At first, the common approach of DDPG is considered, which uses the Bellman operator

$$\mathcal{T}^{\pi} Q(s, a) = \mathbb{E}_{r \sim R(s,a), s' \sim P(s,a)}[r + \gamma Q(s', \pi(s))] \quad (11)$$

to minimize the temporal difference error, i.e., the difference between before and after applying the Bellman operator. This leads to the minimization of the loss

$$\mathcal{L}_2(\psi) = \mathbb{E}_{s_{t_0} \sim \text{p}_{t_0}, s \sim \rho^{\pi_\theta}_{s_{t_0}}}[(Q_\psi(s, a) - \mathcal{T}^{\pi_{\theta'}} Q_{\psi'}(s, a))^2] \quad (12)$$

where $(\pi_{\theta'}, Q_{\psi'})$ corresponds to two separate networks called target policy and target value neural networks, respectively, used for stabilizing the learning. At each iteration, they are gradually updated as the weighted sum between the current policy/value networks and the current target policy/value network, i.e., $\theta' \leftarrow (1 - m_{target})\theta' + m_{target}\theta$ and $\psi' \leftarrow (1 - m_{target})\psi' + m_{target}\psi$.

2) Distributional DDPG: Another way is to approximate the distribution instead of only approximating the expected value of the return, as in [39]. The following analogy is helpful here to motivate its interest. Instead of having a scheduler and its users, consider a teacher and its students. Even though the objective of the teacher is to increase the average "knowledge" of its students, using the distribution of the capacity/knowledge of the students allows for instance to decide whether to distribute

[2]Instead of dividing by the ℓ_1 norm, we also considered the $\text{softmax}(\boldsymbol{y})$, which seemed a good choice as it also provides positive outputs adding up to one. However, this approach leads to poor performance because no matter how much the number of users is increased, the policy insists on evaluating as advantageous to serve only a very small number of users. This actually makes sense since the softmax function is a smooth approximation of argmax, hence focusing on finding the one most advantageous user to be served.

his/her attention uniformly among students or to focus mostly on a fraction of them needing further support.

Algorithmically, it is impossible to represent the full space of probability distribution with a finite number of parameters, so the value neural network $\mathcal{Z}_\psi^{\pi_\theta} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^{N_Q}$ is designed to approximate the actual Z^{π_θ} with a discrete representation. Among many variations [21], [40], we choose the representation to be a uniform (discrete) probability distribution supported at $\{(\mathcal{Z}_\psi^{\pi_\theta})_i, i \in \{1, \cdots, N_Q\}\}$ where $(\mathcal{Z}_\psi^{\pi_\theta})_i$ is the i-th element of the output. More rigorously, the distribution that the value neural network represents is $\frac{1}{N_Q}\sum_{i=1}^{N_Q} \delta_{(\mathcal{Z}_\psi^{\pi_\theta})_i}$, where δ_x is a Dirac delta function at x [22]. Minimizing the 1-Wasserstein distance between this (approximated) distribution and the actual one of Z^{π_θ} can be achieved by minimizing the quantile regression loss

$$\mathcal{L}_1(\psi)=\sum_{i=1}^{N_Q} \mathbb{E}_{s_{t_0}\sim \mathrm{p}_{t_0}, s\sim\rho_{s_{t_0}}^{\pi_\theta}, z\sim\mathcal{T}^{\pi_{\theta'}}\mathcal{Z}_{\psi'}^{\pi_{\theta'}}(s,a)}[f_i(z-(\mathcal{Z}_\psi^{\pi_\theta})_i))] \tag{13}$$

where $\mathcal{T}^\pi Z^\pi(s,a) \overset{D}{=} R(s,a)+\gamma Z^\pi(s',\pi(s)), s'\sim P(s,a)$ is the distributional Bellman operator, $\mathcal{Z}_{\psi'}^{\pi_{\theta'}}$ is the target policy network (defined as before) and $f_i(x)=x(\frac{2i-1}{2N_Q}-\mathbb{1}_{\{x<0\}})$.

Notice that even though we approximate the distribution of $Z^{\pi_\theta}(s,a)$, what is actually needed for improving the policy is only its expected value, approximated as $Q^{\pi_\theta}(s,a) \approx \frac{1}{N_Q}\sum_{i=1}^{N_Q}(\mathcal{Z}_\psi^{\pi_\theta})_i$. Therefore it is natural to wonder if it indeed helps using $\mathcal{Z}_\psi^{\pi_\theta}$ instead of directly approximating the needed expected value (confirming the intuition in the teacher-student analogy). In Figure 3 we provide numerical support for distributional DDPG. Comparing Figures 3b and 3c, we show the benefits of using distributional DDPG. The distributional DDPG approach detects faster the existence of two different service classes with heterogeneous requirements, thus gradually improving the satisfaction rate for both of them. On the other hand, trying only to learn the expected value leads to a training where the performance for one class is improved at the expense of the other. Nonetheless, when aggregating the rewards coming from both classes, we observe in Figure 3a faster convergence of DDPG than the distributional DDPG even though - when converged - the latter exhibits slightly better performance. Introducing a trick (explained later in the "dueling" paragraph), the distributional DDPG approach can be enhanced and outperforms DDPG.

3) Distributional DDPG & Dueling: To facilitate the approximation of the distribution $Z^{\pi_\theta}(s_t,a_t)$, we propose to split it into two parts: one that estimates the mean $\mathcal{Z}_\psi^{\pi_\theta,Mean}$ and one that estimates the shape of the distribution $\mathcal{Z}_\psi^{\pi_\theta,Shape}$. For that, we use a *dueling* architecture [24] (shown in Figure 6). Adapting equation (9) of [24] for the distribution $Z^{\pi_\theta}(s_t,a_t)$, the output becomes $(\mathcal{Z}_\psi^{\pi_\theta})_i=\mathcal{Z}_\psi^{\pi_\theta,Mean}+(\mathcal{Z}_\psi^{\pi_\theta,Shape})_i-\frac{1}{N_Q}\sum_{i=1}^{N_Q}(\mathcal{Z}_\psi^{\pi_\theta,Shape})_i$, $\forall i \in \{1,\cdots,N_q\}$; this effectively pushes $\mathcal{Z}_\psi^{\pi_\theta,Mean}$ *to approximate* Q^{π_θ} used for training the policy. To ensure the decomposition of the distribution into shape and mean, we add a loss term $\mathcal{L}_{shape} = (\frac{1}{N_Q}\sum_i(\mathcal{Z}_\psi^{\pi_\theta,Shape})_i)^2$, centering $\mathcal{Z}_\psi^{\pi_\theta,Shape}$ around zero. The total loss function is

$$\mathcal{L}_{1+duel}(\psi) = \mathcal{L}_1(\psi) + \mathcal{L}_{shape}(\psi). \tag{14}$$

To better understand the role and the performance of using the dueling architecture to approximate the (return) distribution, we have implemented a simple experiment, whose results are shown in Figure 4. We set a random variable Z with known cumulative distribution function (cdf) CDF_{real} from which we draw samples. The objective is to test the distributional and the combination of distributional plus dueling approach on how fast using samples from Z they correctly estimate the CDF_{real}. For the first approach (termed Distributional) we use N_Q parameters $\boldsymbol{\varphi} \in \mathbb{R}^{N_Q}$ and aim to approximate the quantiles of CDF_{real} through minimizing the quantile regression loss (as in (13)): $\mathcal{L}_1(\boldsymbol{\varphi})= \sum_{i=1}^{N_Q}\mathbb{E}_{z\sim Z}[f_i(z-(\boldsymbol{\varphi})_i)]$.

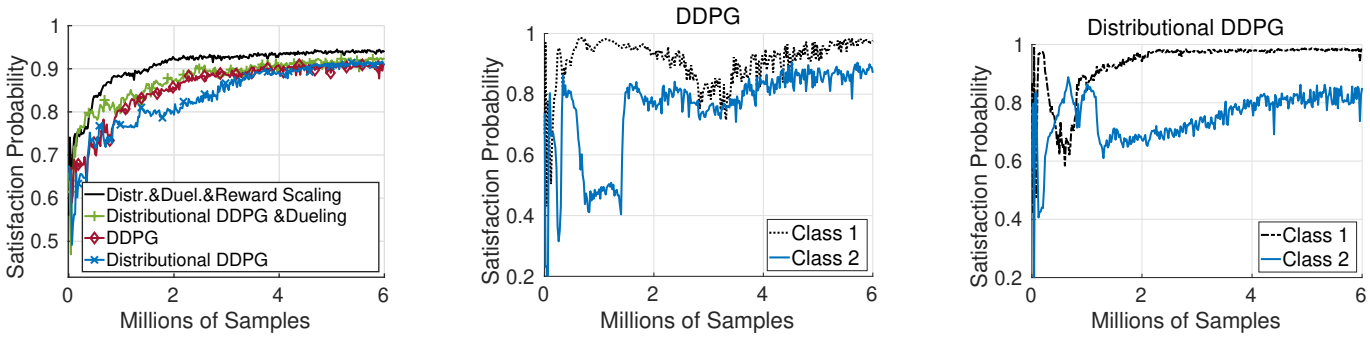
On the other hand, we use the dueling architecture (termed Distributional & Dueling) with parameters $\boldsymbol{\varphi}_{shape} \in \mathbb{R}^{N_Q}$ and $\varphi_{mean} \in \mathbb{R}$. We want with $\boldsymbol{\varphi}_{duel} := [\boldsymbol{\varphi}_{shape}, \varphi_{mean}]$ to approximate the quantiles of CDF_{real} by minimizing the loss $\mathcal{L}_{1+duel}(\boldsymbol{\varphi}_{duel})$ as defined in (14). In Figure 4, each column corresponds to a different random variable Z_i. Specifically:

- the first column corresponds to a normal distribution $Z_1 \sim \mathcal{N}(0,1)$,
- the second one to a Gamma distribution $Z_2 \sim \Gamma(1,1)$, and
- the last one Z_3 to an equiprobable mixture of two normal distributions $\mathcal{N}(0,1)$ and $\mathcal{N}(4,1)$.

Each row corresponds to a different number of samples used to estimate CDF_{real}. We depict the estimated cdf when using or not the dueling trick and compared them to the true one. We use $N_Q = 50$ and the optimization algorithm is Adam with learning rate 0.01. We can see that using dueling leads to faster estimation of the true cdf in all cases.

4) Scaling rewards: A closer look on the range of the possible rewards reveals that they have a very large range of possible values, starting from 0 (no user satisfied) to K (maximum number of users satisfied assuming all classes have equal importance $\alpha_c = 1$). Therefore both its mean and variance may take big values. This is accentuated for the return, since it is the (discounted) sum of many of those rewards. Therefore, approximating the returns which take a large range of values is demanding. Standard technique to facilitate the approximation is "scaling" the rewards. The rewards are normalized in a way that the returns take values on a more easy to approximate range. Given a path that a fixed agent have taken, the returns per time slot across that path are computed. Scaling the rewards pushes the mean of those returns to zero and the variance to one.

Specifically, the implementation of scaling the rewards involves first estimating the discounted sum of rewards $z_t \leftarrow \gamma z_{t-1} + r_t$, then the running statistic of its mean $z_t^{mean} \leftarrow m_{scale} z_{t-1}^{mean} + (1-m_{scale}) z_t$ and of its mean of squares $z_t^{squares} \leftarrow m_{scale} z_{t-1}^{squares} + (1-m_{scale}) z_t^2$. Finally the scaled reward equals to $\frac{r_t - z_t^{mean}}{\sqrt{z_t^{squares} - (z_t^{mean})^2}}$. The DRL algorithm is fed with those rewards whose discounted sum over time is the return that the policy network is trained to



(a) (b) (c)

Figure 3: Comparison between distributional and standard (non-distributional) DDPG RL. We conducted five experiments with different seeds as in Figure 2 with the same traffic model. In the first figure, we depict the average over those five experiments; in the other figures, we consider one specific experiment in an attempt to show the inherent ability of distributional DDPG in dealing with heterogeneous traffic.

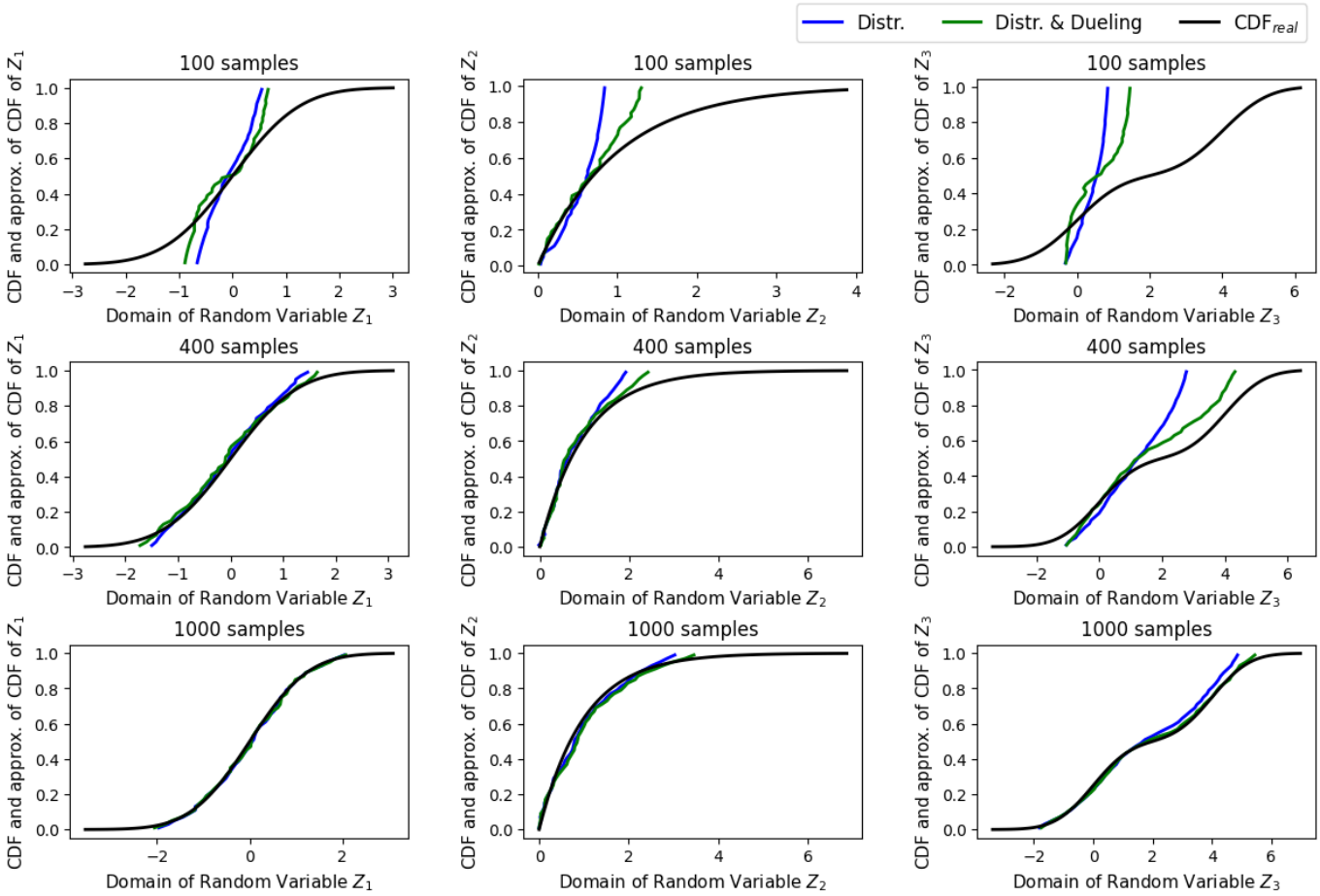


Figure 4: Estimation of a cumulative distribution function with and without the dueling trick.

predict. We fix $m_{scale} = 10^{-4}$. In Figure 3a it is shown that reward normalization clearly provides additional boost in the performance.

In Figure 5, we visualize what the value network tries to approximate. In the first row, by considering only distributional DDPG, from state s and action a the distribution of the returns $\mathcal{Z}_{\psi}^{\pi_\theta}(s, a)$ is approximated. From a different state s' and action a', there will be other possible random paths that the agent with policy π_θ may take and the value network will try to approximate the distribution $\mathcal{Z}_{\psi}^{\pi_\theta}(s', a')$. The black dots depict the average of the two distributions, which are in fact the values that the value network of a simple DDPG would like to approximate and the policy network to maximize. In the second row, the use of reward scaling shifts the distributions around zero and also shrink them. In the last row, the dueling trick is added so the value network has two outputs. One

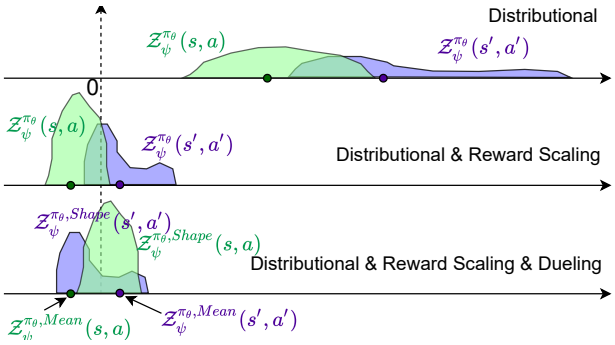


Figure 5: Effect of adding the dueling architecture in the Value Network and/or reward scaling to distributional DDPG RL.

branch of the dueling architecture approximates the value $\mathcal{Z}_\psi^{\pi_\theta,Mean}(s,a) = \mathbb{E}[\mathcal{Z}_\psi^{\pi_\theta}(s,a)]$, while the other the centered distribution $\mathcal{Z}_\psi^{\pi_\theta,Shape}(s,a) = \mathcal{Z}_\psi^{\pi_\theta}(s,a) - \mathcal{Z}_\psi^{\pi_\theta,Mean}(s,a)$.

5) Deep Sets: A final remark concerns the architecture, which, as discussed before, should be designed so as to preserve the permutation invariance. If we associate every user's characteristics with the resources given by the agent, i.e., the action corresponding to it, then permuting the users and accordingly the respective resource allocation should not influence the assessment of the success of the agent. To build such an architecture, we adopt the same architecture as in our Policy Network, capitalizing on ideas from DeepSets [25].

The different steps of our algorithm are shown in Figure 6.

V. Baseline algorithms

In this section, we present baseline scheduling algorithms, which are built upon conventional optimization techniques but are adapted to our specific problem. These algorithms are used for performance comparison in order to show the gains of our proposed DRL architecture.

A. Full CSI case

At time t_c ($t_c \geq t_0$), for user u_0, which arrived at time t_0, both channel h_{u_0,t_c} and location d_{u_0} are known. User u_0 is not satisfied at time t if and only if the allocated bandwidth $w_{u_0,t}$ is smaller than the threshold $w_{u_0,t}^{th} = \dfrac{\overline{D}_{u_0}}{\mathcal{R}_{u_0,t}}$. We first consider algorithms working with immediate horizon ($T = 1$), where only the current time t_c is considered ignoring the effects on future slots. In that case, it is possible that the scheduler prefers serving two users that just arrived in the system rather than a user with bad channel requiring more resources but being on the verge of its latency constraint expiration. The optimization problem can be easily rewritten as follows. The variables to optimize are $\{x_{u,t_c}\}_u$. The variable x_{u,t_c} is equal to 1 if user u is served at time t_c or 0 otherwise. The cost in terms of bandwidth used is $w_{u,t_c}^{th} x_{u,t_c}$, since full CSI is assumed and the scheduler allocates exactly the minimum bandwidth required to successfully send the data to user u. Then, the contribution in the reward function is $\alpha_u x_{u,t_c}$. As a result, the optimization problem can be written as

$$\begin{aligned} \max_{x_{u,t_c}} \quad & \textstyle\sum_{u\in U_{t_c}} \alpha_u x_{u,t_c} \\ \text{s.t.} \quad & \textstyle\sum_{u\in U_{t_c}} w_{u,t_c}^{th} x_{u,t_c} \leq W \\ & x_{u,t_c} \in \{0,1\}, \quad \forall u \in U_{t_c}. \end{aligned}$$

This problem boils down to a *knapsack* problem, which aims to maximize the total value by choosing a proper subset from a set of objects. Every object has its value but also its weight, thus preventing one from picking all objects since the total weight of the chosen subset should not exceed the knapsack capacity level. It is a well known $\mathcal{NP}$-complete problem with numerous efficient algorithms solving it. This method is a strong baseline as in every time slot it finds the optimal allocation that maximizes the immediate rewards. In this work, we use Google's OR-TOOLS library for solving it.

A second baseline we compare with is the so-called *exponential rule* [32], which corresponds to a generalization of proportional fair scheduler taking into account the queue state and the latency constraint of each user, and is also a state-of-the-art mixed traffic packet scheduling scheme. At each time slot t, users are ordered according to their index values and we start serving the ones with the highest rank until resources are finished. Let $v_{u,t}$ be the number of the time slots user u remains unsatisfied and $l_{u,t}$ be the number of time slots the user is eager to wait (therefore $L_u = v_{u,t} + l_{u,t}$). Denote $\overline{\mathcal{R}}_{u,t} = \dfrac{1}{v_{u,t}+1} \displaystyle\sum_{\tau=t-v_{u,t}}^{t} \mathcal{R}_{u,\tau}$ the estimated mean past rate. This value is known by the server at time t by keeping track of the history of channel gains. Then the index J_u for user u is given by

$$J_u = \gamma_{u,t} \mathcal{R}_{u,t} e^{\dfrac{a_{u,t} v_{u,t} - \overline{a_t v_t}}{1 + \sqrt{\overline{a_t v_t}}}} \tag{15}$$

with $\overline{a_t v_t} = \frac{1}{|U_t|} \sum_u a_{u,t} v_{u,t}$, $\gamma_{u,t} = a_{u,t}/\overline{\mathcal{R}}_{u,t}$ and $a_{u,t} = -\log(\delta_u)/l_{u,t}$ with δ_u being the delay violation probability.

Lastly, we focus on algorithms that explicitly take into account the effects of an action on the future of finite horizon ($T > 1$). For sake of simplicity, we assume that for the time interval $t \in [t_c, t_c + T - 1]$ for all the current users and also for the ones that will appear within that interval, the channel realizations during this time interval are known beforehand (i.e., when the algorithm is executed at time t_c). Therefore, this baseline becomes an *oracle* since it knows the future channel realizations of users and can choose the best moment to serve them. Evidently, this method provides an *upper bound* on the performance. Specifically, if $U_{t_c}^T$ denotes the set of all current users plus the ones that will arrive in the time interval $[t_c, t_c + T - 1]$, then for every user $u \in U_{t_c}^T$ this baseline knows $w_{u,t}^{th}$ which corresponds to the required bandwidth in order to satisfy u at time $t \in [t_c, t_c + T - 1]$. The optimization problem

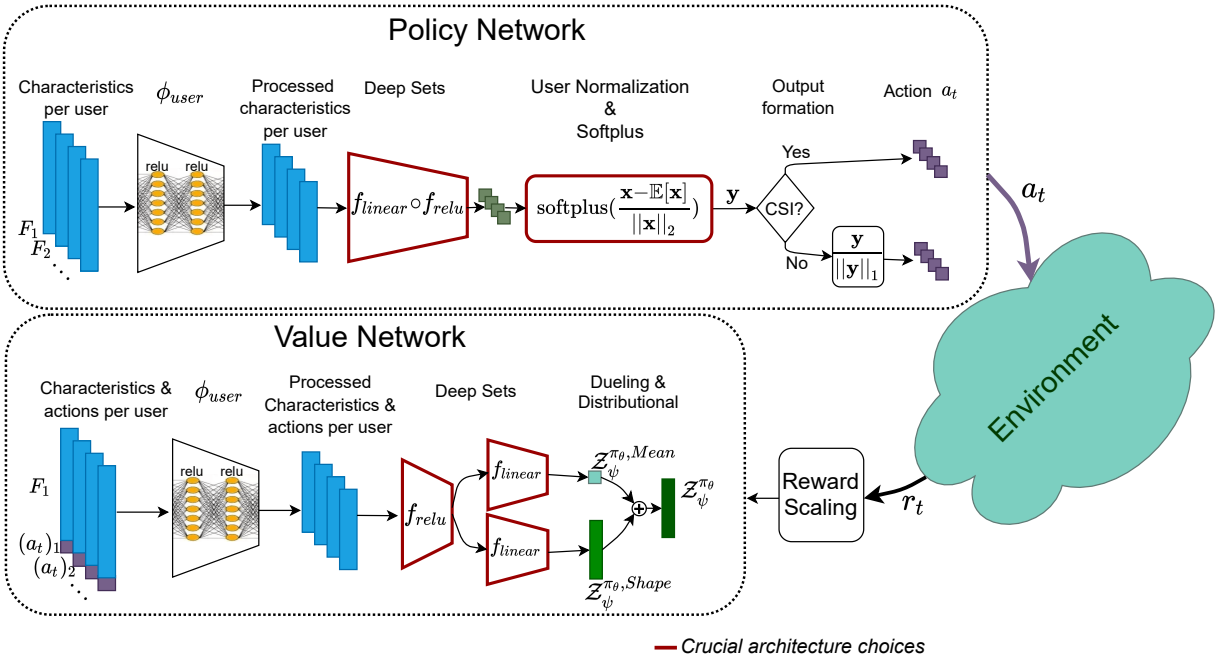


Figure 6: The proposed RL network architecture

is then cast as

$$\begin{aligned} \max_{x_{u,t}} \quad & \textstyle\sum_{u\in U_{t_c}^T} \alpha_u \sum_{t=t_c}^{t_c+T-1} x_{u,t} \\ \text{s.t.} \quad & \textstyle\sum_{U_t} w_{u,t}^{th} x_{u,t} \le W, \quad \forall t \in [t_c, t_c{+}T{-}1] \\ & \textstyle\sum_{t=t_c}^{t_c+T-1} x_{u,t} \le 1, \quad \forall u \in U_{t_c}^T \\ & x_{u,t} \in \{0,1\}, \quad \forall t \in [t_c, t_c{+}T{-}1] \text{ and } \forall u \in U_{t_c}^T. \end{aligned}$$

This problem is an ILP and we use IBM CPLEX Optimization software, which employs the Branch and Cut algorithm [1], to solve it. Notice that the above problem cannot be mapped into a knapsack one (as for $T = 1$), or even a multiple knapsack problem because the weight of each user is time-varying due to channel variability and non-constant user set.

B. Statistical CSI case

Under statistical CSI, the BS knows the statistics of the system (channel, location, and traffic). In this section, we build a baseline to compare with the proposed DRL scheduler in the *no-CSI* case as mentioned in Remark 1.

Let us first focus on the case of a single user u_0 arriving at time t_0. The current time is $t_c \in [t_0, t_0 + L_{u_0} - 1]$. We denote by $\boldsymbol{w}_{u_0,t} = (w_{u_0,t_0}, w_{u_0,t_0+1}, ..., w_{u_0,t})$ the assigned bandwidth from time t_0 (beginning of transmission for user u_0). Additionally, let $A_{u_0,t}$ be a binary random variable, where if $A_{u_0,t} = 1$, then u_0 is still unsatisfied at the end of time slot t (*after* receiving $\boldsymbol{w}_{u_0,t}$ resources) and $A_{u_0,t} = 0$ otherwise. Given that at the beginning of time t, user u_0 still remains unsatisfied and that we know $w_{u_0,t}$ is scheduled at time t, we define $\Phi(\boldsymbol{w}_{u_0,t}; d_{u_0})$ to be the probability that $w_{u_0,t}$ is not sufficient to satisfy the user's request for known location d_{u_0} and unknown channel realization $h_{u_0,t}$, i.e.,

$$\Phi(\boldsymbol{w}_{u_0,t}; \mathrm{d}_{u_0}) = \begin{cases} \mathbb{P}(A_{u_0,t} = 1 | \boldsymbol{w}_{u_0,t-1}, \mathrm{d}_{u_0}, A_{u_0,t-1}{=}1), & t > t_0 \\ \mathbb{P}(A_{u_0,t} = 1 | \mathrm{d}_{u_0}), \quad t = t_c = t_0. \end{cases} \tag{16}$$

The average contribution of user u_0 to the gain (5) on the time interval $[t_c, t]$ is given by the following equation, derived by applying the chain rule on conditional probability:

$$\mathfrak{g}_{u_0}^{[t_c,t]} := \mathfrak{g}(w_{u_0,t_c}, ..., w_{u_0,t}; \mathrm{d}_{u_0}) = \begin{cases} 0, \text{if } t_c > t_0 \text{ and } A_{u_0,t_c-1} = 0 \\ \alpha_{u_0}\big(1 - \prod_{j=t_c}^{t} \Phi(\boldsymbol{w}_{u_0,j}; \mathrm{d}_{u_0})\big), \text{ else.} \end{cases} \tag{17}$$

We consider now the average contribution on the gain (5) for subsequent users after user u_0. The next user (if any) appears at time $t_1 = t_0 + L_{u_0}$, the second next at time $t_2 = t_1 + L_{u_1}$, and so on. In other words, we consider the users, denoted $u_1, u_2, \ldots$, which appear at time $t_1 = t_0 + L_{u_0}, t_2 = t_1 + L_{u_1}, \ldots$, respectively. Users belong to classes $c_1, c_2, \ldots$, with probabilities $p_{c_1}, p_{c_2}, \ldots$, respectively. Since the locations of those future users are unknown, we need to average (16) and (17) over their possible locations in order to obtain their contribution on the gain function (5). So for $i \ge 1$ if $\boldsymbol{w}_{u_i,t} = (w_{u_i,t_i}, w_{u_i,t_i+1}, ..., w_{u_i,t})$, we have

$$\mathfrak{g}_{u_i}^{[t_i,t]} = \mathfrak{g}(w_{u_i,t_i}, ..., w_{u_i,t}) = \alpha_{u_i}\big(1 - \prod_{i=t_c}^{t} \Phi(\boldsymbol{w}_{u_i,i})\big) \tag{18}$$

where the contribution looking at time t with $t < t_i + L_{u_i}$ starts at time t_i for user u_i and where

$$\Phi(\boldsymbol{w}_{u_i,t}) = \begin{cases} \mathbb{P}(A_{u_i,t} = 1 | \boldsymbol{w}_{u_i,t-1}, A_{u_i,t-1}{=}1), & t > t_i \\ \mathbb{P}(A_{u_i,t} = 1), \quad t = t_i. \end{cases} \tag{19}$$

Closed-form expressions for Eqs. (16) and (19) are provided in Appendix A.

For notational convenience, to include the case where no new user is generated in a time slot, we introduce the "null" class of users, which contains users serving as dummies. They appear with probability p_{null}, are active for one slot ($L_{null} = 1$) and have zero contribution $\mathfrak{g}_u^{[t_i,t_{i+1}]} = 0$ with $t_{i+1}{=}t_i{+}L_{null}$. Hence, the average value of the gain function for the sequence of users $u_0, u_1, ...$ (so when there is one user at most per time slot, i.e., $K = 1$) starting at the current time t_c is

$$\mathcal{G}(w_{u_0,t_c}, ..., w_{u_0,t_1-1}, w_{u_1,t_1}, ...) = \mathfrak{g}_{u_0}^{[t_c,t_1-1]}(.; d_{u_0}) + \sum_{c_1 \in \mathcal{C} \cup \text{null}} \left(p_{c_1} \cdot \mathfrak{g}_{u_1}^{[t_1,t_2-1]} + \sum_{c_2 \in \mathcal{C} \cup \text{null}} \left(p_{c_2} \cdot \mathfrak{g}_{u_2}^{[t_2,t_3-1]} + ... \right) \right). \tag{20}$$

From (20), we observe a tree structure[3] that when a user vanishes, there is a summation over all possible classes the new user may belong to. Therefore, a number of branches equal to the number of possible classes (equal to $|\mathcal{C}|$) are created whenever a new future user is taken into account. To harness this scalability issue, we prune the tree by considering only T future time slots and work with finite horizon $[t_c, t_c + T - 1]$.

The general case with multiple users served simultaneously ($K > 1$) can easily be considered by just computing K "parallel trees". With a slight abuse of notation, we consider that the first subscript of the variables w refers now to the index of the tree (and implicitly to a specific user). As a consequence, the variables for the scheduled bandwidth resources over an horizon of length T can be put into the following matrix form

$$\mathbf{W}_{t_c} = \begin{bmatrix} w_{1,t_c} & w_{1,t_c+1} & \cdots & w_{1,t_c+T-1} \\ w_{2,t_c} & w_{2,t_c+1} & \cdots & w_{2,t_c+T-1} \\ \vdots & \vdots & \ddots & \vdots \\ w_{K,t_c} & w_{K,t_c+1} & \cdots & w_{K,t_c+T-1} \end{bmatrix} \tag{21}$$

and the average gain for these resources takes the following form:

$$G(\mathbf{W}_{t_c}) = \sum_{k=1}^{K} \mathcal{G}(w_{k,t_c}, w_{k,t_c+1}, \cdots, w_{k,t_c+T-1}). \tag{22}$$

Finally, we arrive at our optimization problem at current time t_c:

$$\max_{\mathbf{W}_{t_c} \in \mathbb{R}_{\geq 0}^{K \times T}} \quad G(\mathbf{W}_{t_c}) \tag{23}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} w_{k,t} \leq W, \quad \forall t \in \{t_c, \ldots, t_c{+}T{-}1\}. \tag{24}$$

It can easily be shown that the objective function $G(\cdot)$ is non-concave with multiple local optima. The constraints given by (24) describe a compact and convex domain set, which allows to apply the Frank-Wolfe algorithm (FW) [33] that guarantees reaching to a local optimum. FW algorithm is a type of Successive Convex Approximation algorithm that replaces the convex approximation by a linearization (i.e., first-order Taylor approximation) of the objective function. The convergence of the FW method is *sublinear*; however, the computation of the objective function (22) and its partial derivatives grows *exponentially* with T, thus leading to slow and cumbersome method in practice. In each time slot t_c, we use FW to get a local optimum solution $\mathbf{W}_{t_c}^{\star}$ from which we retrieve the first column $[w_{1,t_c}^{\star}, \cdots, w_{K,t_c}^{\star}]^{\intercal}$ corresponding to the bandwidth allocation that will be applied at the current time step t_c.

[3]This can be exploited for computing it recursively.

VI. Experimental Results

We consider the distance-dependent pathloss model $120.9 + 37.6 \log_{10} \mathrm{d}$ (in dB) [41], which corresponds to a constant loss component $C_{pl} = 10^{-12.09}$ and pathloss exponent $n_{pl} = 3.76$. The noise spectral density is $\sigma_N^2 = -149\text{dBm/Hz}$. We consider that the distance between the base station and users ranges from 0.05 km to 1 km. The power per unit bandwidth is kept equal to $1\mu\text{W/Hz}$. In all experiments, for both synthetic and real data, the timeslot is set to $T_s = 1$ ms. For measuring the reported satisfaction probabilities, the scheduler first interacts with the user traffic for a large enough time horizon until the estimated probabilities vary less than 0.1%.

For the proposed DRL scheduler, we update the target policy and value networks with momentum $m_{target} = 0.005$. We use replay buffer of capacity 5000 samples. The batch size is set to 64 and the learning rate is set to 0.001. The discount factor is $\gamma = 0.95$. We use $N_Q = 50$ quantiles to describe the distribution. The ϕ_{user} consists of two fully connected layers each with 10 neurons. We have $P_{explore} = 0.2$ and $\sigma_{explore} = 0.3$. The number of input and output dimensions in both f_{relu} and f_{linear} is 10 (i.e., $H = H' = 10$). We remark that the number of parameters is kept significantly low (around 1800), mainly due to the use of Deep Set. It is essential to keep the number of parameters low because the high stochasticity of the environment makes the model prone to overfitting. Moreover, keeping the number of parameters low makes our solution fast and cost-effective (both in terms of energy and hardware).

A. Synthetic Data

We consider two scenarios for the traffic as described in Table I.

Table I: Classes description for two scenarios

(a) Users of equal importance

	Data per user (Kbytes)	Latency (in time slots)	Imp.	Prob.
Class 1	8	2	1	0.3
Class 2	64	10	1	0.2

(b) Prioritized and normal users

	Data per user (Kbytes)	Latency (in time slots)	Imp.	Prob.
Class 1	8	2	1	0.15
Class 1+	8	2	2	0.05
Class 2	64	10	1	0.3
Class 2+	64	10	2	0.05

The first scenario consists of two classes, one with users requesting a small amount of data but within a stringent latency

constraint (of just two time slots) and one other class requesting a large amount of data with less stringent latency constraint. All classes have the same importance as seen from the Imp. column. In the second scenario, classes do not have the same importance. Note that the Prob. column describes the probability p_c with which a user of that class appears in the system at a given time slot (they do not sum up to one signifying that it is possible that no user appears during some time slots).

In Figure 7, we plot the satisfaction ratio per class priority (i.e., all users having the same priority take part to the computation of the same ratio and so depicted in the same curve) versus the channel correlation (ρ in left column) and versus the total bandwidth (W in right column) for both scenarios and different CSI knowledge. Figures 7a,7c,7e are plotted for $W = 2$ MHz, $W = 5$ MHz and $W = 2$ MHz, respectively, whereas Figures 7b, 7d, 7f are all for $\rho = 0$. Figures 7a, 7b, 7e, and 7f are done with $K = 100$ users. Figures 7c and 7d are done with $K = 60$ users.

Recall that the FW algorithm reaches to a suboptimal point and different initializations lead to different local optima. For that, at each time slot, we repeat the FW algorithm N_{init} times with a different initialization at each time and we select the best suboptimal point. This method could lead to considerable performance improvement for N_{init} increasing; however, due to computational complexity, we stop at $N_{init} = 20$. Moreover, as the number of users K increases, so does the number of local optima and that of solutions with poor performance making it tougher for the FW to find a good optimal point without significantly increasing N_{init}. This is the main reason why our DRL Scheduler substantially outperforms Frank-Wolfe algorithm even at moderate values of users ($K = 60$). Note that our DRL Scheduler continues exhibiting very good performance even if K is further increased.

The proposed DRL scheduler outperforms the knapsack algorithm. For instance, at a level of 95% of satisfaction probability, we may save about 13% of bandwidth, which is followed by a 13% power saving as the power per Hz is kept constant (see Figure 7b). We also observe that our scheduler is close to the optimal policy, since ILP which uses an oracle constitutes an upper bound on the performance. In Figures 7e and 7e, there is a priority class of users, which always enjoys a higher satisfaction probability. Interestingly, the proposed DRL scheduler serves slightly worse the priority class than what the knapsack does. Nevertheless, since the priority counts for the $\frac{0.1}{0.55} \approx 18\%$ of the users and the rest 82% is much better served using our Deep Scheduler, the latter exhibits overall better performance than the knapsack.

B. Real Data

To assess the applicability of our algorithm in a realistic setup, we perform experiments on real data using publicly available traces based on real measurements over Long Term Evolution (LTE) 4G networks in a Belgium city [42], [43]. Six different types of transportation (foot, bicycle, bus, tram, train, car) are used. The throughput and the GPS location of a mobile device continuously demanding data are recorded every second. Since this recording timescale of 1 s in the real dataset is much larger than the small-scale fading timescale represented here by the random variable h, the measurement that is provided corresponds to $M_i = \mathbb{E}_h[W \log_2(1 + \kappa|h|^2)]$ for every i-th second. The value of κ, which mainly depends on the user location, is assumed constant within $1s$. As the measurements bandwidth W is not reported in the dataset, we assume it to be 15MHz, resulting in a mean signal-to-noise ratio SNR $\approx$ 6dB in an LTE compliant system. This allows us to retrieve κ from measurement M_i. To compute the channel time variation h, the user speed is required in (1) so as to obtain ρ. This is estimated using the trajectory of GPS coordinates given from the traces. A user entering the system belongs to a class according to Table II, with its type of transportation chosen randomly; we then sample M_i and her location from the traces accordingly. Knowing in the previous and afterwards time slots the locations, we can compute the average speed and so the ρ. Finally, so far we assumed that the bandwidth can be split as small as desired (continuum); however, in practice, the bandwidth is split into N_{bl} resource blocks and each user is assigned an integer multiple of those. In Table III, we consider different possible values of N_{bl} and keep the size of each resource block constant to 200kHz. Again we confirm the performance gains from using a DRL based approach. For the exponential rule, the value of δ_u is set to $\delta_u = \delta = 10^{-2}$ [32]. Nevertheless, since this value does not provide the best results for every N_{bl} (resource blocks), we tune this parameter for each N_{bl} in order to provide the highest possible performance.

Table II: Equal Classes description (Data rate per user in Kbps, Latency in ms)

	Data (Kbits)	Latency (ms)	Imp.	Prob.
Class 1	1	5	1	0.2
Class 2	5	25	1	0.3

In Table III, we see that the proposed DRL algorithm outperforms baseline algorithms with full CSI and using real data, both in terms of data rate and satisfaction probability. The gap from the upper bound is rather significant, but this is expected as the upper bound is optimistic assuming that the channel is known in advance.

Finally, we would like to remark that the gain of the proposed DRL method is smaller in the case of real data. The main reason is that in the real measurements used in our setup, the channel remains quite constant over time, which translates to $\rho \approx 1$. For instance, in Fig. 7a, which refers to the synthetic data experiments, we observe that the closer the value of ρ is to 1 the smaller are the gains. Therefore, it is reasonable to have lower gains in the real measurements experiments, as the observed ρ is closer to 1. Intuitively, the smaller the ρ, the more diversity the channels of the users presents. This allows the scheduler to take more 'clever' decisions. On the other hand, if $\rho = 1$, it does not actually matter when the scheduler chooses to serve an active user, as their channels remain identical (or quasi identical) in future time slots, and hence the required bandwidth the scheduler needs to spend to satisfy a user. In an nutshell, *our DRL scheduler provides higher gains when ρ is small and also when the available bandwidth resources are not too large.* Since we keep the number of users constant in

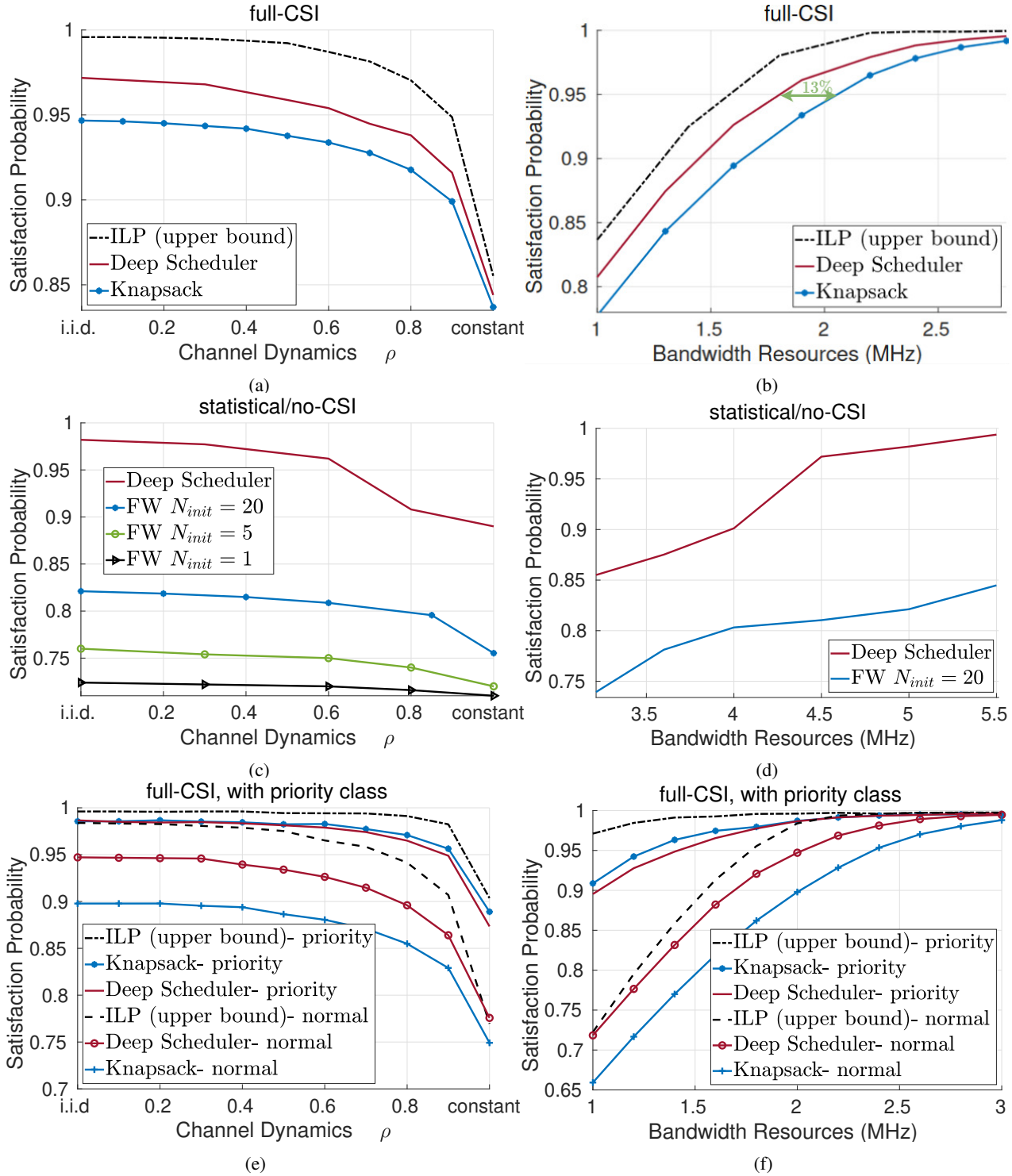


Figure 7: Satisfaction rate of the proposed DRL scheduler and the baseline algorithms versus ρ (left column) and W (right column). The first and last rows correspond to the case of Table Ia and the row in the middle to Table Ib. Figures 7a, 7b, 7e and 7f refer to the full CSI case while the other to the statistical CSI/no CSI case.

Table III: Sum Data Rate (in Mbps) / Probability of Satisfaction

N_{bl} (W)	6 (3MHz)	15 (5MHz)	25 (10MHz)	50 (15MHz)	75 (20MHz)
Knapsack	6.4 / 48.6%	10.2 / 64.3%	15.2 / 84.3%	17.0 / 91.2%	17.6 / 93.4%
Exp. Rule	5.9 / 44.0%	8.8 / 57.3%	14.0 / 80.4%	17.2 / 92.7%	18.1 / 95.8%
Proposed DRL Scheduler	**6.7 / 51.6%**	**10.6 / 67.6%**	**15.5 / 86.9%**	**17.2 / 93.0%**	**18.3 / 96.2%**
ILP (Upper Bound)	9.0 / 62.9%	14.6 / 81.8%	18.6 / 98.5%	18.9 / 98.7%	19.0 / 98.9%

our setup, if the bandwidth resources are too large, then the gains are expected to be low. The reason is that given abundant resources, it is easy to devise a reasonable scheduler that succeeds in satisfying all users. On the contrary, with limited resources the scheduler needs to smartly allocate the resources in order to maximize the satisfied users and this is when our Deep Scheduler outperforms. Finally, as we previously noted, if $\rho \approx 1$, there is not enough fluctuations (diversity) in the channel evolution from one slot to another, and hence not enough margin for smart scheduling decisions to efficiently use the available resources.

VII. Conclusion

The problem of scheduling and resource allocation of a time-varying set of users with heterogeneous traffic and QoS requirements was studied here. We leveraged deep reinforcement learning and proposed a deep deterministic policy gradient algorithm, which builds upon distributional reinforcement learning and deep sets. Our experiments on both synthetic and real data showed that the proposed scheduler can achieve significant performance gains as compared to baseline conventional combinatorial optimization methods and state-of-the-art packet scheduling metrics in both full and no CSI scenarios.

Appendix A
Closed-form expressions for (16) and (19)

a) i.i.d. fading ($\rho = 0$): This is the simplest case since there are no time dependencies on the fading, hence using (2) and (3), (16) and (19) become

$$\Phi(\boldsymbol{w}_{u_0,t}; \mathrm{d}_{u_0}) = P_{u_0}^{fail}(w_{u_0,t}, P; \mathrm{d}_{u_0}) \text{ and} \quad (25)$$

$$\Phi(\boldsymbol{w}_{u_i,t}) = P_{u_i}^{fail}(w_{u_i,t}, P), \quad i \geq 1. \quad (26)$$

We recall that users u_i for $i \geq 1$ arrive after user u_0, therefore their locations are unknown and we need to average over them[4].

b) Static channel ($\rho = 1$): The channel remains the same for each retransmission. For user u_0, the channel is time invariant ($g_{u_0} = g_{u_0,t}\ \forall t \in [t_0, t_0 + L_{u_0} - 1]$) but unknown. Only the user location is known. At time $t > t_0$, we have

$$\begin{aligned}\Phi(\boldsymbol{w}_{u_0,t}; \mathrm{d}_{u_0}) &= \mathbb{P}(w_{u_0,t} \log(1+g_{u_0}\mathrm{P}) < \overline{D}_{u_0} | \\ &\qquad w_{u_0,t'} \log(1+g_{u_0}\mathrm{P}) < \overline{D}_{u_0} \forall t' \in [t_0, t-1], d_{u_0}) \\ &= \frac{\mathbb{P}(w_{u_0,t''} \log(1+g_{u_0}\mathrm{P}) < \overline{D}_{u_0}\ \forall t'' \in [t_0, t] \,|\, d_{u_0})}{\mathbb{P}(w_{u_0,t'} \log(1+g_{u_0}\mathrm{P}) < \overline{D}_{u_0}\ \forall t' \in [t_0, t-1] \,|\, d_{u_0})}. \quad (27)\end{aligned}$$

Therefore, we obtain

$$\Phi(\boldsymbol{w}_{u_0,t}; \mathrm{d}_{u_0}) = \begin{cases} \dfrac{P_{u_0}^{fail}(\max\{\boldsymbol{w}_{u_0,t}\}, P; \mathrm{d}_{u_0})}{P_{u_0}^{fail}(\max\{\boldsymbol{w}_{u_0,t-1}\}, P; \mathrm{d}_{u_0})}, \text{if } t > t_0 \\ P_{u_0}^{fail}(w_{u_0,t}, P; \mathrm{d}_{u_0}), \quad \text{if } t = t_0. \end{cases} \quad (28)$$

For subsequent (future) users (u_i with $i \geq 1$), the expressions remain the same with the only difference that the locations of those users are also not known. Hence, in (28), we just need to omit d_u similarly to the i.i.d. case.

c) General Markovian channel ($\rho \in (0,1)$): This is the most complicated case due to the correlation between channel realizations. At time t, the distribution of $h_{u,t}$ given the past (which is not known in practice) is Rician distributed. Specifically, if user u is active at $t-1$ and t, we have $\mathbb{P}(|h_{u,t}|{=}x \,\big|\, |h_{u,t-1}|) = \mathrm{Rice}(x; v_R = \rho|h_{u_0,t-1}|, \sigma_R^2 = \frac{1-\rho^2}{2})$, where v_R and σ_R^2 is the distance and the spread parameters respectively of the Rice distribution. Let us focus on user u_0 at time $t = t_0 + 1$. According to [44, eq.(37)], we have

$$\begin{aligned}&\Phi(\boldsymbol{w}_{u_0,t_0+1}; \mathrm{d}_{u_0}) = \\ &\int_0^{x_{u_0,0}} \int_0^{x_{u_0,1}} \mathbb{P}(|h_{u_0,t_0+1}|{=}x\,|y) \mathbb{P}(|h_{u_0,t_0}|{=}y) dx dy \\ &= 1 - \frac{e^{-x_1^2} Q_1(\frac{x_{u_0,0}}{\sigma_R}, \frac{\rho x_{u_0,1}}{\sigma_R}) - e^{-x_{u_0,0}^2} Q_1(\frac{\rho x_{u_0,0}}{\sigma_R}, \frac{x_{u_0,1}}{\sigma_R})}{2(1 - e^{-x_{u_0,0}^2})} \end{aligned} \quad (29)$$

with $x_{u_i,j} = \sqrt{\zeta_{u_i,t_i+j}} d^{-\frac{n_{pl}}{2}}, i \in \{0,1\}$ and Q_M be the Marcum Q-function.

For future users ($u_i, i \geq 1$), we have at time $t = t_i + 1$ (we remind that user u_i starts its transmission at time t_i):

$$\Phi(\boldsymbol{w}_{u,t_i+1}) = \int_{\mathrm{d}_{min}}^{\mathrm{d}_{max}} \Phi(\boldsymbol{w}_{u_i,t_i+1}; \mathrm{d}_{u_i}) f_{\mathrm{d}}(\mathrm{d}) d\mathrm{d} \quad (30)$$

where $\Phi(\boldsymbol{w}_{u_i,t_i+1}; \mathrm{d}_{u_i})$ is given by (29) by replacing u_0 with u_i. Equation (30) is *intractable* even considering only the first two adjacent retransmissions. This is exacerbated when one considers additional transmissions. Therefore, the baseline algorithm is only designed for $\rho = 0$ or $\rho = 1$, even if it is also tested in the general case $\rho \in (0,1)$. Specifically, for any ρ, we apply the baseline algorithm designed for both $\rho = 0$ and $\rho = 1$ and keep the best result.

[4]It might not be easy to find the derivative of (3), which is required for first-order approximation in the Franck-Wolfe algorithm. This is done as follows

$$\begin{aligned}\frac{dP_u^{fail}}{dw} &= \int_{\mathrm{d}_{min}}^{\mathrm{d}_{max}} \frac{d\mathbb{P}(|h|^2 < \zeta_{u,t}\mathrm{d}^{n_{pl}})}{d\zeta_{u,t}} f_{\mathrm{d}}(\mathrm{d}) d\mathrm{d} \frac{d\zeta_{u,t}}{dw} \\ &= \frac{\Gamma(\frac{2+n_{pl}}{n_{pl}}, \zeta_{u,t}\mathrm{d}_{min}^{n_{pl}}) - \Gamma(\frac{2+n_{pl}}{n_{pl}}, \zeta_{u,t}\mathrm{d}_{max}^{n_{pl}})}{n_{pl}\zeta_{u,t}^{(2+n_{pl})/n_{pl}}(\mathrm{d}_{max}^2 - \mathrm{d}_{min}^2)/2} \frac{d\zeta_{u,t}}{dw}.\end{aligned}$$

REFERENCES

[1] J. E. Mitchell, "Branch-and-cut algorithms for combinatorial optimization problems," *Handbook of applied optimization*, vol. 1, pp. 65–77, 2002.

[2] V. t. Mnih, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] D. t. Silver, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *preprint, arXiv:1712.01815*, 2017.

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[5] J. Kober, J. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, pp. 1238–1274, 2013.

[6] 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," *preprint, arXiv: 1509.02971*, 2015.

[7] M. Cheng, J. Li, and S. Nazarian, "Drl-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers," in *23rd Asia and S. Pacific Design Automation Conference (ASP-DAC)*, 2018, pp. 129–134.

[8] O. Naparstek and K. Cohen, "Deep multi-user reinforcement learning for distributed dynamic spectrum access," *IEEE Trans. on Wireless Communications*, vol. 18, no. 1, pp. 310–323, 2018.

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

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

[11] B. Mao, F. Tang, Y. Kawamoto, and N. Kato, "AI models for green communications towards 6g," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 210–247, 2022.

[12] S. Chinchali, P. Hu, T. Chu, M. Sharma, M. Bansal, R. Misra, M. Pavone, and S. Katti, "Cellular network traffic scheduling with deep reinforcement learning." in *AAAI Conf. on Artificial Intelligence*, 2018, pp. 766–774.

[13] N. Zhao, Y.-C. Liang, D. Niyato, Y. Pei, M. Wu, and Y. Jiang, "Deep reinforcement learning for user association and resource allocation in heterogeneous cellular networks," *IEEE Trans. on Wireless Communications*, vol. 18, no. 11, pp. 5141–5152, 2019.

[14] J. V. Saraiva, J. Braga, Iran M., V. F. Monteiro, F. R. M. Lima, T. F. Maciel, J. Freitas, Walter C., and F. R. P. Cavalcanti, "Deep Reinforcement Learning for QoS-Constrained Resource Allocation in Multiservice Networks," *arXiv*, Mar. 2020.

[15] F. S. Mohammadi and A. Kwasinski, "Deep reinforcement learning approach to QoE-driven resource allocation for spectrum underlay in cognitive radio networks," in *IEEE Inter. Conf. on Commun. (ICC) Workshops*, Kansas City, MO, USA, May 2018.

[16] H. Yang, J. Zhao, K.-Y. Lam, S. Garg, Q. Wu, and Z. Xiong, "Deep reinforcement learning based resource allocation for heterogeneous networks," in *17th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2021, pp. 253–258.

[17] W. Lee and R. Schober, "Deep learning-based resource allocation for device-to-device communication," *CoRR*, vol. abs/2011.12757, 2020.

[18] E.-M. Bansbach, V. Eliachevitch, and L. Schmalen, "Deep reinforcement learning for wireless resource allocation using buffer state information," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[19] V. H. L. Lopes, C. V. Nahum, R. M. Dreifuerst, P. Batista, A. Klautau, K. V. Cardoso, and R. W. Heath, "Deep reinforcement learning-based scheduling for multiband massive MIMO," *IEEE Access*, vol. 10, pp. 125 509–125 525, 2022.

[20] S. C. Jaquette, "Markov decision processes with a new optimality criterion: Discrete time," *The Annals of Statistics*, vol. 1, no. 3, pp. 496–505, 1973.

[21] W. Dabney, G. Ostrovski, D. Silver, and R. Munos, "Implicit quantile networks for distributional reinforcement learning," *preprint, arXiv:1806.06923*, 2018.

[22] W. Dabney, M. Rowland, M. G. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," in *Thirty-Second AAAI Conference on Artificial Intelligence*, New Orleans, USA, 2 2018.

[23] M. Fortunato, M. G. Azar, B. Piot, J. Menick, M. Hessel, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis *et al.*, "Noisy networks for exploration," in *International Conference on Learning Representations*, 2018.

[24] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," in *International Conference on Machine Learning, ICML*, New York, USA, 6 2016.

[25] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 3391–3401. [Online]. Available: http://papers.nips.cc/paper/6931-deep-sets.pdf

[26] 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," in *International Conference on Learning Representations, ICLR*, San Juan, Puerto Rico, 5 2016.

[27] A. Valcarce, "Wireless suite," https://github.com/nokia/wireless-suite, 2020.

[28] Z. Gu, C. She, W. Hardjawana, S. Lumb, D. McKechnie, T. Essery, and B. Vucetic, "Knowledge-assisted deep reinforcement learning in 5G scheduler design: From theoretical framework to implementation," *IEEE Journal on Sel. Areas in Commun. (JSAC)*, vol. 39, no. 7, pp. 2014–2028, 2021.

[29] Y. Shen, Y. Shi, J. Zhang, and K. B. Letaief, "Graph neural networks for scalable radio resource management: Architecture design and theoretical analysis," *IEEE Journal on Sel. Areas in Commun. (JSAC)*, vol. 39, no. 1, pp. 101–115, 2020.

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

[31] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry, "Implementation matters in deep RL: A case study on PPO and TRPO," in *International conference on learning representations*, 2019.

[32] S. Shakkottai and A. L. Stolyar, "Scheduling algorithms for a mixture of real-time and non-real-time data in HDR," in *Teletraffic Science and Engineering*. Elsevier, 2001, vol. 4, pp. 793–804.

[33] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Research Logistics Quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800030109

[34] C. C. Tan and N. C. Beaulieu, "On first-order Markov modeling for the Rayleigh fading channel," *IEEE Trans. on Communications*, vol. 48, no. 12, pp. 2032–2040, 2000.

[35] R. Bellman, "A Markovian decision process," *Journal of mathematics and mechanics*, pp. 679–684, 1957.

[36] K. J. Åström, "Optimal control of Markov processes with incomplete state information," *Journal of Mathematical Analysis and Applications*, vol. 10, pp. 174–205, 1965. [Online]. Available: https://lup.lub.lu.se/search/ws/files/5323668/8867085.pdf

[37] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[38] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," *Proceedings of the 31st International Conference on Machine Learning, PMLR*, vol. 32, no. 1, pp. 387–395, 6 2014.

[39] G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, A. Muldal, N. Heess, and T. Lillicrap, "Distributed distributional deterministic policy gradients," in *Intern. Conf. on Learning Repres. (ICLR)*, Vancouver, Canada, 2018.

[40] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *International Conference on Machine Learning, ICML*, Syndey, Australia, 8 2017.

[41] (2018, 6) Technical Report: 3GPP TR 36.913 v15.0.0: Requirements for further advancements for E-UTRA (LTE-Advanced). [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/36_series/36.913/

[42] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck, "HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks," *IEEE Comm. Letters*, vol. 20, no. 11, pp. 2177–2180, 2016.

[43] H. Mao, "Pensieve," https://github.com/hongzimao/pensieve, 2017.

[44] A. H. Nuttall, "Some integrals involving the Q-function," *IEEE Trans. on Inform. Theory*, vol. 21, no. 1, pp. 95–96, 4 1975.

Apostolos Avranas received the Diploma (Hons.) in electrical and computer engineering from the Aristotle University of Thessaloniki, Greece, in 2015.In 2020 he received his Ph.D. degree from with Télécom ParisTech funded by Huawei Paris Research Center. From 2020 to 2022 he held the position of a postdoctoral researcher at EURECOM, Sophia-Antipolis, France. Currently, he is working with Amadeus, Sophia Antipolis, France as a Senior Machine Learning Engineer.

Philippe Ciblat was born in Paris, France, in 1973. He received the Engineering degree from Ecole Nationale Supérieure des Télécommunications (ENST, now called Telecom Paris) and the M.Sc. degree (DEA, in french) in automatic control and signal processing from Université Paris-Saclay, Orsay, France, both in 1996, and the Ph.D. degree from Université Gustave Eiffel, Marne-la-Vallée, France, in 2000. He eventually received the HDR degree from Université Gustave Eiffel, Marne-la-Vallée, France, in 2007. In 2001, he was a Postdoctoral Researcher with the Université de Louvain, Belgium. At the end of 2001, he joined Telecom Paris, as an Associate Professor. Since 2011, he has been (full) Professor in the same institute. From 2009 to 2020, he was the head of Digital Communications Team. He served as Associate Editor for the IEEE Communications Letters from 2004 to 2007. From 2008 to 2012, he served as Associate Editor and then Senior Area Editor for the IEEE Transactions on Signal Processing. From 2014 to 2019, he was member of IEEE Technical Committee "Signal Processing for Communications and Networking". From 2018 to 2021, he served as Associate Editor for the IEEE Transactions on Signal and Information Processing over Networks. His research areas include statistical signal processing, signal processing for digital communications, resource allocation, distributed optimization, signal over graphs, and machine learning.

Marios Kountouris (S'04–M'08–SM'15–F'23) received the diploma degree in electrical and computer engineering from the National Technical University of Athens (NTUA), Greece in 2002 and the M.S. and Ph.D. degrees in electrical engineering from Télécom Paris, France in 2004 and 2008, respectively. He is currently a Professor at the Communication Systems department, EURECOM, Sophia-Antipolis, France. Prior to his current appointment, he has held positions at CentraleSupélec, France, the University of Texas at Austin, USA, Huawei Paris Research Center, France, and Yonsei University, South Korea. He is the recipient of a Consolidator Grant of the European Research Council (ERC) in 2020 on goal-oriented semantic communication. He has served as Editor for the IEEE Transactions on Wireless Communications, the IEEE Transactions on Signal Processing, and the IEEE Wireless Communication Letters. He has received several awards and distinctions, including the 2022 Blondel Medal, the 2020 IEEE ComSoc Young Author Best Paper Award, the 2016 IEEE ComSoc CTTC Early Achievement Award, the 2013 IEEE ComSoc Outstanding Young Researcher Award for the EMEA Region, the 2012 IEEE SPS Signal Processing Magazine Award, the IEEE SPAWC 2013 Best Paper Award and the IEEE Globecom 2009 Communication Theory Best Paper Award. He is an IEEE Fellow, an AAIA Fellow, and a chartered Professional Engineer of the Technical Chamber of Greece.