Deep Reinforcement Learning for Resource Constrained Multiclass Scheduling with Random Service Rates

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

Abstract

The problem of resource constrained scheduling in a dynamic and heterogeneous setting is considered here. In our setup, the available limited resources (bandwidth) are allocated in order to serve randomly arriving service demands, which belong to different classes in terms of payload data requirements, 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. Various approaches for scheduling and resource allocation can be used, ranging from simple greedy heuristics and constrained optimization to combinatorics. Those methods are usually adjusted to meet the particularities of the specific setup and application. We adapt them to our setting but they are suboptimal and further improvements are attainable. To this purpose, we resort to Reinforcement learning and propose a Deep Distributional Deterministic Policy Gradient algorithm combined with Deep Sets to tackle the problem. We also present a novel way to use a Dueling Network, which results in performance improvement. We test the algorithm using both synthetic and real data and show consistent gains against state-of-the-art conventional methods from combinatorics, optimization, and scheduling metrics.

Index Terms

Deep Reinforcement Learning, Deep sets, Scheduling, Time-varying environment.

A. Avranas and M. Kountouris are with the Communication Systems department, EURECOM, F-06904 Sophia-Antipolis, France. Emails: apostolos.avranas@eurecom.fr, marios.kountouris@eurecom.fr

P. Ciblat is with LTCI, Telecom Paris, Institut Polytechnique de Paris, F-91120 Palaiseau, France. Email: philippe.ciblat@telecom-paris.fr

I. INTRODUCTION

Scheduling is a long-standing problem that arises in various fields under different settings. In computing, for instance, computational processes have to be arranged in an efficient way for the server to handle them; in management, certain jobs are assigned to each person for completion; in logistics, packages have to be carefully matched to each truck, to name a few. Scheduling usually comes along with a resource allocation problem. A limited amount of resources, e.g., central processing units (CPUs), human resources, trucks, is available and must be distributed efficiently among the scheduled tasks to optimize the system performance.

In this paper, we consider the following problem of scheduling and resource allocation. A base station (BS) serves data traffic of mobile users, which have different application-aware Quality of Service (QoS) requirements. Some applications may require the transmission of a large amount of data without any strict deadline, whereas some other time-sensitive or mission-critical applications impose that a small amount of data has to be reliably received within a stringent latency constraint. The increased heterogeneity in users' traffic and the diverse QoS requirements complicate substantially the provisioning of high fidelity, personalized service of guaranteed QoS. The objective of this work is to design a system architecture and an algorithm, which take as inputs the specific constraints of the traffic/service class each user belongs to and outputs which users to serve when and which resources to allocate in order to maximize the number of satisfied users.

The problem considered here is hard to solve due to at least two major technical challenges. First, with the exception of few special cases, there is no simple closed-form expression for the problem and *a fortiori* for its solution. Second, optimization algorithms that solve the problem have to be scalable with the number of active users. Existing approaches rely on combinatorial methods or suboptimal solutions, which seem to work satisfactorily in specific scenarios, failing though to perform well in more general cases with large number of users. This motivates the quest for alternative solutions. In this paper, we propose to tackle this problem by resorting to Deep Reinforcement Learning (DRL).

The scheduling problem is a well known problem arising in various scientific domains. The proliferation of new services and applications makes the problem of efficient scheduling more intricate. Researchers are resorting to new methods, such as deep reinforcement learning, which have shown very promising results in problems obeying strict game rules (Atari, Chess, Go

[5]–[7]) or physical laws (robotics and physics tasks [8], [9]). In Cloud Service provision for example [10], they use a DRL approach to schedule the incoming tasks to servers according to their heterogeneous CPU and memory requirements. DRL approaches have shown gains in wireless communication systems. For example in [11] they perform scheduling on a cellular network using DRL. Also ideas using DRL in a distributed way to perform dynamic resource allocation have appeared in [4], [12]–[14]. In [15], in a setup similar to ours and the one of Nokia's challenge [16], Deep Deterministic Policy Gradient (DDPG) is used to allocate the bandwidth to incoming data traffic but contrary to our setup they focus only on the full Channel State Information (CSI) case, without multiple traffic classes and serving only few users (typically less than 15). Initial attempts to solve the problem of scheduling on traffic of users with heterogeneous performance requirements have been seen in [17] but again only with full CSI and a limited number of users. To solve this hard problem, especially because of the high level of stochasticity, one can resort to distributional Reinforcement Learning (RL) researched in [18] and followed by [19], [20] in order to have richer representations of the environment thus obtaining better solutions. Also techniques like noisy network for better explorations [21] or architectures such as dueling networks [22] have greatly improved the stability of the trained models. Finally, recent work [3] managed to simplify and improve neural network models when permutation invariance properties apply. We combine those ideas with a deep deterministic policy gradient method [23] to reach a very efficient scheduling algorithm.

In the context of using DRL for tackling the wireless communication problem of heterogeneous scheduling and dynamic resource allocation our main contributions can be summarized as follows:

- We build a neural network with two crucial architectural choices facilitating a stable training even in the case of a traffic with a large number of users. First, we leverage the theory of *Deep Sets* [23] to exploit the permutation equivariance property of the problem and drastically reduce the number of necessary parameters. Second, we introduce a *user normalization* trick capturing the attribute of the problem that the available bandwidth resources are limited. We show that without those crucial steps the performance plummet.
- We further improve the performance with tricks like distributional DRL [20] and reward scaling as implemented in [24]. Finally, we get some extra gains by adapting the idea of dueling network [22] used in Deep Q-Networks (DQN), to apply it in Distributional RL by modifying the output to represent the distribution of the return of the agent's action.
- We demonstrate our DRL proposal can easily be implemented with minor changes in both

extremes of wireless communication scenarios, i.e. full CSI and no CSI.

- To compare our DRL solution, we design strong baselines:
  - In the *full CSI* case, the scheduling problem is solved in a myopically optimal way by reformulating it as a knapsack one. The DRL scheduler manages to outperform by reaching the same performance but requiring 13% less power and bandwidth. We also devise a baseline operating as an oracle by knowing the future traffic characteristics. With Integer Linear Programming optimally finds the resource allocation strategy and constitutes an upper bound. We experimentally show that the DRL scheduler operates close to it.
  - In the *no CSI* case, the baseline has actually access to the statistical properties of the problem and uses them to cast the scheduling problem as an optimization one. The Frank-Wolfe (FW) algorithm is employed, which guarantees that the solution is a local optimum. The DRL scheduler significantly outperforms FW, leading to our hypothesis that the more complicated the communication system is with unknown variables affecting it, the bigger possible gains may be yielded by a DRL approach since it is in principle model-free.

The paper is organized as follows: in Section II, we introduce the system model including the channel and traffic model. In Section III, we present the optimization problem we aim to solve. Section IV is devoted to the main contribution of the paper which corresponds to design a new DRL scheduler working for our setting. In Section V, baseline algorithms are described. In Section VI, we provide numerical results with synthetic data and real data. In Section VII, concluding remarks are drawn.

II. System Model

A. Network and channel model

We consider the downlink of a communication system, in which a BS sends data to multiple users 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 serving 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 extremely high, so that BS-user distances remain constant during the time interval users remain active.

Orthogonal frequency bands are assigned to concurrently served users, hence there is no interference among them. Users experience frequency flat fading channel, i.e., the channel gain of a user remains constant during a time slot and throughout all available frequency bands assigned to. 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} is 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 Markovian model

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

where $h_{u,t_0} \sim \mathcal{CN}(0,1)$ ($\mathcal{CN}(0,v)$ is a circular complex normal distribution with zero mean and variance v), and $N \sim \mathcal{CN}(0, 1-\rho^2),\ t > t_0$. The parameter $\rho = J_0(2\pi f_d T_{slot}) \in [0,1]$ [25] 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 mobility of the users), and T_{slot} the 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 block fading).

Regarding channel knowledge and the Channel State Information (CSI), we consider the following two extreme cases: (i) *full-CSI*, in which h_{u,t_c} and the users' locations (and so d_u) are perfectly known at each 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 instantaneous realization and long-term 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 sort of CSI. For that, we consider *statistical CSI*, where the scheduler knows the statistics of the channels and the locations of the users. This is in contrast to our proposed DRL algorithm, which could operate under complete absence of CSI, since the statistics are effectively learned through the training phase.

B. Traffic model

We consider a generic yet tractable traffic model, in which users with diverse stringent data and latency requirements enter and exit the system in a continuous way. A user entering the system belongs to a certain service class $c \in \mathcal{C}$ with probability p_c, where $\mathcal{C}$ denotes the set of classes. Each user 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.
- Maximum Latency L_c: the maximum number of time slots within which the user has to be satisfied, that is 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 for Service-Level Agreement (SLA)) may request for 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 appear only after for some previous user the maximum time possible to remain in the system has left. For example, if a user appears 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. We chose this approach because the more common one of allowing to randomly generate a new arrival immediately after a previous user is satisfied (in the example at time $t = 2$) would lead to a conceptual problem. The faster a scheduler serves the users the more arrivals occur, meaning that the scheduler's performance affects the traffic. On the other hand, with the approach we follow, the arrival process and its statistics remain uninfluenced by the scheduling decisions and the available resources. Therefore, at every time slot, a set of users U_t ($|U_t| \leq K$) is observed that includes all of those who still wait to receive their requested data while remaining within their latency constraint. Finally, to guarantee the randomness of the 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 in a time slot it may happen that no user appears.

C. Service Rate

The service rate is measured using Shannon's 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 $\mathcal{R}_{u,t} = \log_2(1 + g_{u,t}\mathrm{P}_{u,t}) = \log_2(1 + \kappa_u|h_{u,t}|^2)$ (bit/s), with $\mathrm{P}_{u,t}$ denoting the transmit energy per symbol

(channel use), $w_{u,t}$ the assigned bandwidth, and $\kappa_u = \frac{C_{pl}}{\sigma_N^2}\mathrm{d}_u^{-n_{pl}}$. Consider at time t_u serving a user at distance d_u from the BS, belonging to class $c \in \mathcal{C}$ with resources $(w_{u,t}, \mathrm{P}_{u,t})$. The probability of failed transmission (incorrect decoding) is given by

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

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

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

where $\Gamma(s,x) = \int_x^\infty t^{s-1}\,\mathrm{e}^{-t}\,\mathrm{d}t$ is the upper incomplete gamma function. For exposition convenience, we overload notation by allowing u in D_u, L_u, α_u to either denote a class u or a user u belonging to a class with those characteristics.

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 that belong to different classes with heterogeneous requirements. Each class defines the requirements and the expected Quality of Service (QoS) guarantees for users of that class. 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 limited resources amongst the selected user requests. The performance metric is to maximize the long-term "importance"-based weighted sum of successfully satisfied requests. A request is considered to be satisfied whenever it has received the requested data within the maximum tolerable latency specified by its service class.

The scheduling problem can be formulated as a Markov Decision Process (MDP) [1] $(\mathcal{S}, \mathcal{A}, R, P, \gamma)$, 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

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

representing the discounted sum of rewards when a trajectory of states is taken following π. An agent ideally aims to find the optimal policy $\pi^\star$ maximizing the mean reward $\mathbb{E}_\pi[Z_t^\pi]$.

At each time step t, a set of users $u \in U_t$ are pending, where each user belongs to a class $c_u \in \mathcal{C}$ described by (D_c, L_c, α_c). After L_c time steps a new user might arrive with probability p_c belonging to class c. Throughout its "lifespan" $t \in [t_0, t_0+L_c-1]$, $\mathcal{R}_{u,t}$ indicates the amount of resources $w_{u,t}$ are needed. If at any time t $w_{u,t} > D_u/\mathcal{R}_{u,t}$ then it u is satisfied. Evidently the resources are limited, $\sum_{u\in U_t} w_{u,t} \leq W, \quad \forall t$, so no more than W resources can in total 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: $r_t = \sum_{u\in U_t} \alpha_u \mathbb{1}\{w_{u,t}\mathcal{R}_{u,t} > D_u\}$

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 $\mathbb{1}\{\cdot\}$ denote the indicator function. Note that knowing the class c_u that u belongs means knowing its requirement (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 in consequence we will need a permutation *invariant function* to evaluate and train the policy.

In this work, we only perform 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 $\sum_{u\in U_t} w_{u,t} \leq W, \quad \forall t$ 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"

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

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 through eq. (1). 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 and we end up in Partially Observable MDP (POMDP) [26]. A way to reduce a POMDP to an MDP is by substituting the states with the "belief" of the states

[27]. Another way is to use the complete history $\{o_0, a_0, o_1, a_1, \cdots, a_{t-1}, o_{t-1}\}$, with $o_t \subset s_t$ the agent's observation. Notice that only the most recent part is relevant as users that have already left the system do not affect in any way how the channels of the current users will 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 $\boldsymbol{w}_{u,t} = (w_{u,t_0},\ w_{u,t_0+1}, ..., w_{u,t})$ 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 scheduler

In this section, we propose a novel DRL architecture as a means to solve the aforestated multiclass scheduling and resource allocation problem. Despite the highly challenging dynamics (stochastic wireless channel and heterogeneous traffic), we show that DRL can provide gains although it is impossible to accurately predict the number of users, their service demands, and their channel/link characteristics even after few steps.

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. The most common approach is Deep Q-learning Network (DQN); however, it is infeasible to employ it 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 (DDPG) method [23], which trains a policy $\pi_\theta : \mathcal{S} \rightarrow \mathcal{A}$ 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 with the help of eq. (4) can be given by the function

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

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 [28]:

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

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 will be 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.

1) Deep Sets: The policy we aim for falls in a category of permutation equivariant functions as discussed in Section III (permuting the users should only result in permuting likewise the resource allocation). In [3] necessary and sufficient conditions are shown for permutation equivariance in neural networks; their proposed structure called Deep Sets is adopted here. At first, the characteristics (or features as commonly termed in the machine learning field) $F_i \in \mathbb{R}^{N_u}, i \in \{1, \cdots K\}$ 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 network. Then the outputs of ϕ_{user} which corresponds to the new characteristics per user are aggregated with the permutation equivariant $f_\sigma : \mathbb{R}^{K \times H} \to \mathbb{R}^{K \times H'}$ of H (resp. H') input (resp. output) characteristics:

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

where $\mathbf{1} = [1, \cdots, 1] \in \mathbb{R}^K$ and $\sigma(\cdot)$ an element wise non-linear function. 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 non-linearity $\sigma(\cdot)$. In addition to preserving the desirable permutation equivariance property, this structure also brings a significant parameter reduction. 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 would lead to a much bigger network prone to overfitting.

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 the full CSI the exact necessary bandwidth per user is known. Therefore, we

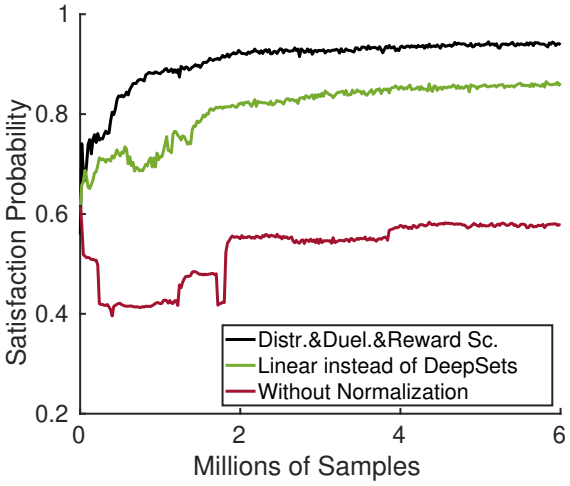


Figure 1: We conducted five experiments (with different seeds) for no-CSI using the traffic model of Table Ia, a maximum number of users $K = 75$, $\rho = 0$ and resources (total bandwidth) $W = 5$ MHz. We depict here the average probability a user to be satisfied over those five experiments to carry an ablation study on the importance of the deep sets and user normalization step.

actually only need a binary decision per user (serve or not), but this would ruin the differentiability of the policy, which is a mandatory property for DDPG to work. For that, we interpret the output $\boldsymbol{y}$ 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. According to it, 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 its 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, $\boldsymbol{y} \rightarrow \frac{\boldsymbol{y}}{||\boldsymbol{y}||_1}$ (with $||\cdot||_1$ being the ℓ_1 norm)[1]. This discrepancy in the output process is the only minor difference in the considered model between full-CSI and no-CSI.

[1]Instead of dividing by the ℓ_1 norm we also considered the $\mathrm{softmax}(\boldsymbol{y})$. Since it also provides positive output adding up to one it seemed a good choice. But it led to poor performance because no matter how much the number of users were increased, the policy insisted on evaluating advantageous to serve only a very small number of them. This makes sense since the softmax function is a smooth approximation of argmax and therefore focused on finding the one most advantageous user to be served.

3) User normalization: Before the final non-linearity of $\mathrm{softplus}(x) = \log(1 + e^x)$, as seen in Figure 5, there is the crucial "user normalization" step $\boldsymbol{x} \rightarrow \frac{\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}]}{||\boldsymbol{x}||_2}, \boldsymbol{x} \in \mathbb{R}^K$ (with $||\cdot||_2$ being the ℓ_2 norm). Consider first the full-CSI case. Without that step, the value network would perceive that the higher the "value" per resources 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 unceasing increase of every user's "value". What matters is not the actual "value" of a user but how big it 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 $\boldsymbol{y} \rightarrow \frac{\boldsymbol{y}}{||\boldsymbol{y}||_1}$ 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 1 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 degrades substantially.*

4) Exploration: A final note regards the exploration. Since the action a_t has to satisfy specific properties, such as positivity 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 [21], which introduce noise to the weights of a layer, resulting to changed decisions for the policy network. The original approach considers the variance of the added noise to be learnable; instead we keep it constant since it has provided better results. With a probability $P_{explore}$ we add noise to the parameters of ϕ_{users}, resulting to altered output features per user and therefore the policy gives 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 $\sigma_{explore}$ 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 equation (8). This is intractable to compute so 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 Deep Determinstic Policy Gradient (DDPG) is considered which uses the Bellman operator

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

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

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

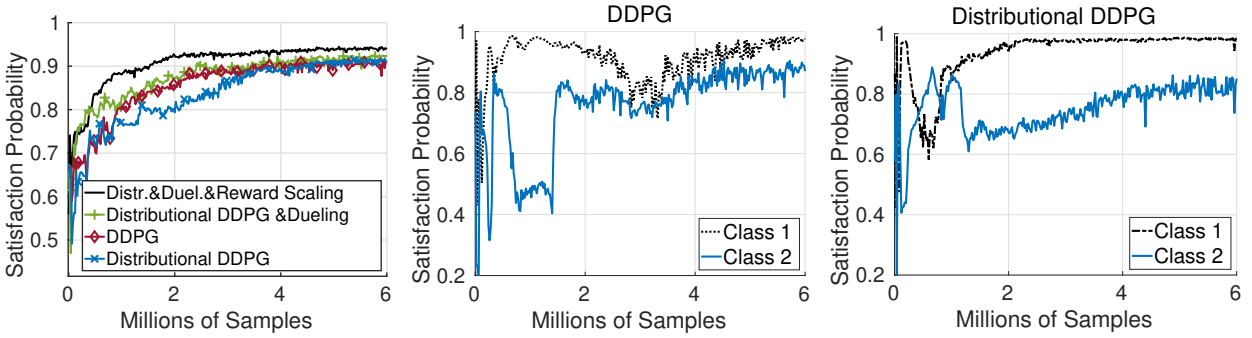
where $(\pi_{\theta'}, Q_{\psi'})$ corresponds to two separate networks called target policy and target value networks, respectively, used for stabilizing the learning. At each iteration they are gradually updated as the weighted sum between the current policy/value networks and the current target policy/value 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 [29]. An analogy might be helpful here to reason its application. 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 enables for instance to decide whether to distribute its 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 distributions with a finite number of parameters, so the value neural network $\mathcal{Z}^{\pi_\theta}_\psi : \mathcal{S}\times\mathcal{A} \to \mathbb{R}^{N_Q}$ is designed to approximate the actual Z^{π_θ} with a discrete representation. Among many variations [19], [30], we choose the representation to be a uniform (discrete) probability distribution supported at $\{(\mathcal{Z}^{\pi_\theta}_\psi)_i, i \in \{1, \cdots, N_Q\}\}$ where $(\mathcal{Z}^{\pi_\theta}_\psi)_i$ is the i-th element of the output. More rigorously, the distribution that the value neural network represents is $\frac{1}{N_Q}\sum_{i=1}^{N_Q}\delta_{(\mathcal{Z}^{\pi_\theta}_\psi)_i}$ where δ_x is a Dirac delta function at x [20]. Minimizing the 1-Wasserstein distance between this (approximated) distribution and the actual one of Z^{π_θ}, can be done by minimizing the quantile regression loss

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

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



(a) (b) (c)

Figure 2: Comparison between distributional and standard (non-distributional) RL approach. We conducted five experiments with different seeds as in Figure 1 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 RL in dealing with heterogeneous traffic.

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 teacher-student analogy. In Figure 2 we provide numerical support for distributional DDPG. Comparing 2b and 2c, we show the benefits of using distributional DDPG. This 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 see in Figure 2a a faster convergence of DDPG than the distributional one even though - when converged - the distributional DDPG has a slightly better performance. By introducing a small trick explained later (see the "dueling" paragraph), the distributional approach is enhanced, outperforming the 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 [22] as shown in Figure 5. 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 3. We set a random variable Z with a known cumulative distribution

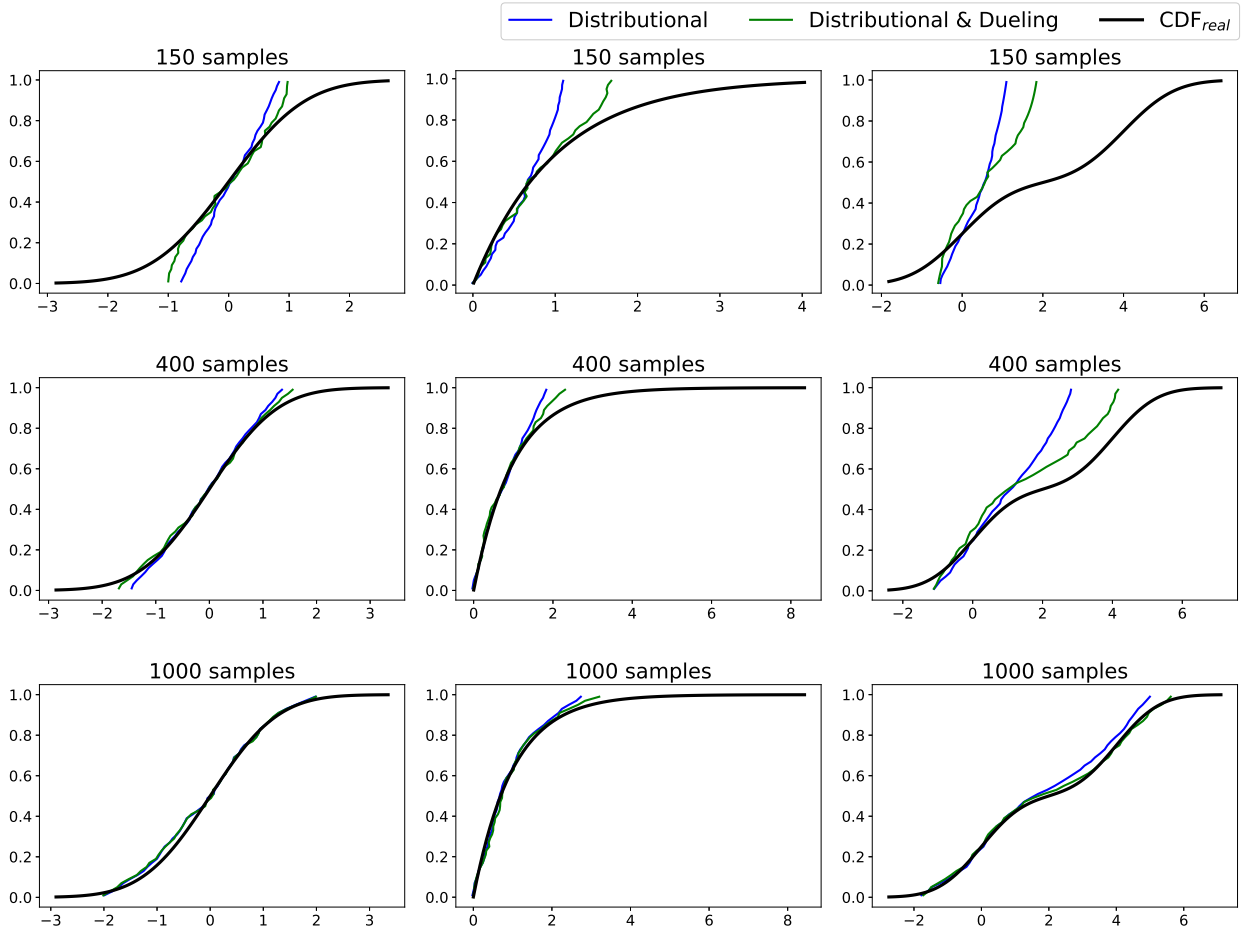


Figure 3: Estimation of a cumulative distribution function with and without the dueling trick.

function (cdf) CDF_{real} from which we draw samples. The goal is to test the distributional and the combination of distributional plus dueling approach on how fast using samples from Z they can 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 Eq. (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 (Distributional & Dueling), we use the dueling architecture 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 Eq. (14). In Figure 3, each column corresponds to a different cdf CDF_{real}:

- the first column corresponds to a normal distribution $\mathcal{N}(0, 1)$,
- the second one to a Gamma distribution $\Gamma(1, 1)$, and
- the last one 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 $N_Q = 50$. The optimization algorithm is Adam with learning rate 0.01. In all cases using dueling method leads to faster estimation of the true cdf.

4) Scaling rewards: A closer look on the range of the possible rewards reveals that they have a huge 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 returns since it is the (discounted) sum of many of those rewards. Therefore, approximating the returns which take a big 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 easily approximatable range. Given a path that a fixed agent have taken, one can compute the returns per time slot across that path. What scaling rewards does is to push 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 $\dfrac{r_t - z_t^{mean}}{\sqrt{z_t^{squares} - (z_t^{mean})^2}}$. The DRL algorithm is fed with those rewards whose discounted sum over time is the return that the policy network is trained to predict. We fix $m_{scale} = 10^{-4}$. In Figure 2a it is shown that reward normalization clearly provides additional boost in the performance.

In Figure 4 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

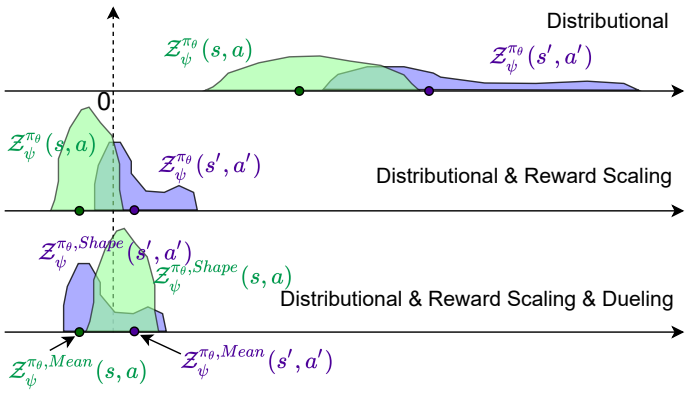


Figure 4: Effect of adding the dueling architecture in the Value Network and/or reward scaling to distributional RL.

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 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 should be designed, as discussed, in a way preserving 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 [3].

We sum up in Figure 5 the different steps of our algorithm.

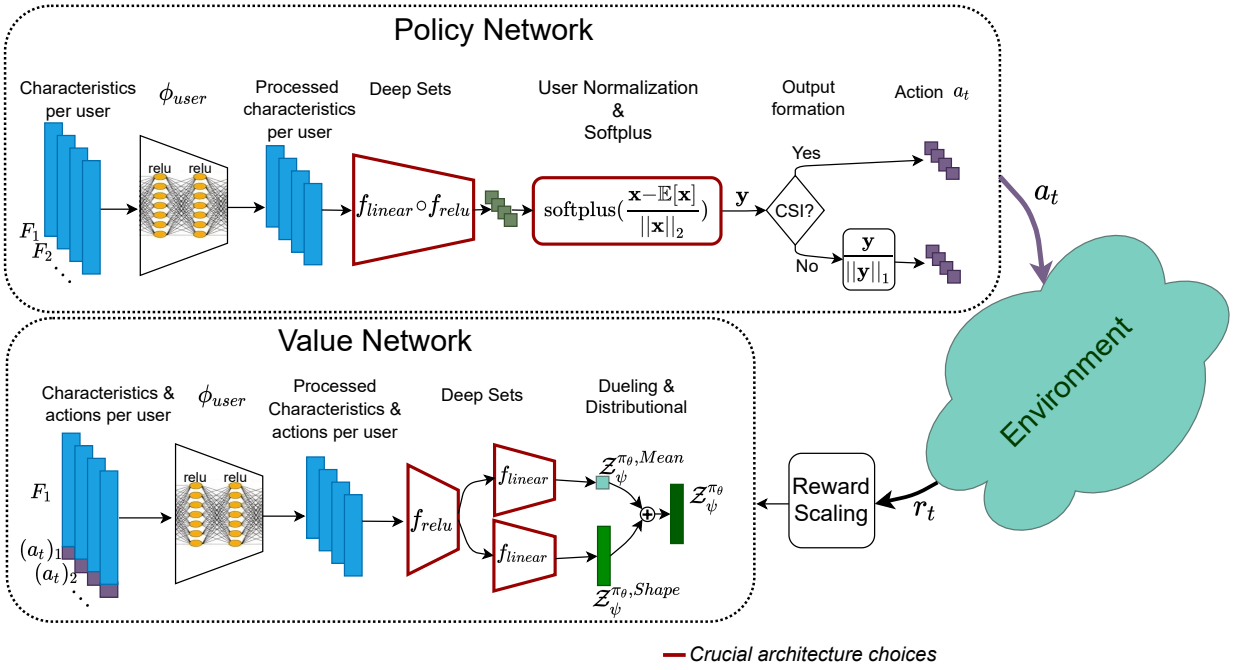


Figure 5: Network Architecture

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. They are used for performance comparison.

A. *Full-CSI case*

We remind that under full CSI assumption, the BS knows the channel realization and the location (distance) at the current time. More precisely, let us consider user u_0 who appeared at time t_0 and the current time is t_c ($t_c \geq t_0$), for which 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 following threshold

$$w_{u_0,t}^{th} = \frac{D_{u_0}}{\mathcal{R}_{u_0,t}}.$$

We first consider algorithms working with immediate horizon ($T = 1$), in which we only consider the current time t_c ignoring the effects on future. In that case, it is possible that the scheduler prefers serving two users who just appeared than one user with bad channel requiring more resources but is on the verge of its latency constraint. This optimization problem can be

easily rewritten. The variables to optimize are x_{u,t_c}, which equal 1 if user u is served at time t_c or 0 otherwise. The cost in terms of bandwidth used is $w^{th}_{u,t_c}x_{u,t_c}$, because we assume full CSI 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 follows

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

This problem is 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. In this work, we have used Google's OR-TOOLS library solving it.

A second baseline we compare with is the so-called *exponential rule* [31] which corresponds to a generalization of proportional fair scheduler taking into account the latency constraints of each user. 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 the user u remains unsatisfied and $l_{u,t}$ be the number of time slots it 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

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

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 take explicitly 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 that they will have in that time interval are known in advance, i.e., when the algorithm is executed at time t_c. This constitutes this baseline an *oracle* since it

knows for the users their future channel realizations and can choose the best moment to serve them. Thus, this method provides an *upper bound* on the performance. Specifically, if by $U_{t_c}^T$ we denote 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 $w_{u,t}^{th}$ that corresponds to the required bandwidth in order to satisfy u at time $t \in [t_c, t_c+T-1]$. The optimization problem then formulates as

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

This problem is an *Integer Linear Programming (ILP)*. To solve it, we use IBM CPLEX Optimization software, which employs the Branch and Cut algorithm [32]. 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 the set of users is non-constant.

B. Statistical CSI case

We remind that under statistical CSI, the BS knows all statistical properties of the system (channel, location, and traffic). In this section, we build baseline for comparing the proposed DRL scheduler in the *no-CSI* case as mentioned in Remark 1.

Let us first concentrate 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 is still 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 $h_{u_0,t}$:

$$\Phi(\boldsymbol{w}_{u_0,t}; \mathrm{d}_{u_0}) = \begin{cases} \mathbb{P}(A_{u_0,t} = 1 | \boldsymbol{w}_{u_0,t-1}, \mathrm{d}_{u_0}, A_{u_0,t-1}{=}1), & t > t_0 \\ \mathbb{P}(A_{u_0,t} = 1 | \mathrm{d}_{u_0}), & t = t_c = t_0. \end{cases} \tag{15}$$

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:

$$\mathfrak{g}_{u_0}^{[t_c,t]} := \mathfrak{g}(w_{u_0,t_c}, ..., w_{u_0,t}; \mathrm{d}_{u_0}) = \begin{cases} 0, \text{if } t_c > t_0 \text{ and } A_{u_0,t_c-1} = 0 \\ \alpha_{u_0}\big(1 - \prod_{j=t_c}^{t} \Phi(\boldsymbol{w}_{u_0,j}; \mathrm{d}_{u_0})\big), \text{ else.} \end{cases} \tag{16}$$

We consider now the average contribution on the gain (5) for subsequent users after user u_0. The next user (if it exists) 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 (15) and (16) over their possible locations in order to obtain their contribution on the gain function (5). So for $i \geq 1$ if $\boldsymbol{w}_{u_i,t} = (w_{u_i,t_i},\ w_{u_i,t_i+1}, ..., w_{u_i,t})$, we have

$$\mathfrak{g}_{u_i}^{[t_i,t]} = \mathfrak{g}(w_{u_i,t_i}, ..., w_{u_i,t}) = \alpha_{u_i}\big(1 - \prod_{i=t_c}^{t} \Phi(\boldsymbol{w}_{u_i,i})\big) \tag{17}$$

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

$$\Phi(\boldsymbol{w}_{u_i,t}) = \begin{cases} \mathbb{P}(A_{u_i,t} = 1 | \boldsymbol{w}_{u_i,t-1}, A_{u_i,t-1}{=}1), & t > t_i \\ \mathbb{P}(A_{u_i,t} = 1), & t = t_i. \end{cases} \tag{18}$$

Closed-form expressions for Eqs. (15) and (18) are provided in Appendix A.

To easily, in terms of notation, incorporate that in some time slot a new user may not be generated, we create the "null" class of users. It contains users serving as dummies. They appear with probability p_{null}, stay in the system for one slot ($L_{null} = 1$) and have zero contribution $\mathfrak{g}_u^{[t_i,t_{i+1}]} = 0$ where $t_{i+1}{=}t_i{+}L_{null}$. Hence, the average value of 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

$$\mathcal{G}(w_{u_0,t_c}, ..., w_{u_0,t_1-1}, w_{u_1,t_1}, ...) = \mathfrak{g}_{u_0}^{[t_c,t_1-1]}(.; d_{u_0}) + \sum_{c_1 \in \mathcal{C} \cup \text{null}} \Big(p_{c_1} \cdot \mathfrak{g}_{u_1}^{[t_1,t_2-1]} + \sum_{c_2 \in \mathcal{C} \cup \text{null}} \Big(p_{c_2} \cdot \mathfrak{g}_{u_2}^{[t_2,t_3-1]} + \sum_{c_3 \in \mathcal{C} \cup \text{null}} (...) \Big) \Big). \tag{19}$$

From (19), we observe a tree structure[2] 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

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

number of possible classes ($|\mathcal{C}|$) are raised 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

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

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

$$G(\mathbf{W}_{t_c}) = \sum_{k=1}^{K} \mathcal{G}(w_{k,t_c}, w_{k,t_c+1}, \cdots, w_{k,t_c+T-1}). \tag{20}$$

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

$$\max_{\mathbf{W}_{t_c} \in \mathbb{R}_{\geq 0}^{K \times T}} \quad G(\mathbf{W}_{t_c}) \tag{21}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} w_{k,t} \leq W, \quad \forall t \in \{t_c, \ldots, t_c+T-1\}. \tag{22}$$

It can easily be shown that the objective function $G(\cdot)$ is non-concave with multiple local optimums. The constraints given by equation (22) describe a compact and convex domain set, which allows applying the Frank-Wolfe algorithm (FW) [33] which guarantees reaching to a local optimum. The FW method is *sublinear* but computing the objective function (20) and its partial derivatives grows *exponentially* with T which leads to slow and cumbersome method in practice. Therefore, in each time slot t_c we use FW to get a local optimum solution $\mathbf{W}^{\star}_{t_c}$ from which we retrieve the first column $[w^{\star}_{1,t_c}, \cdots, w^{\star}_{K,t_c}]^{\intercal}$ corresponding to the bandwidth allocation that will be applied at the current time step t_c.

VI. Numerical Results

A. Synthetic Data

According to Long-Term Evolution (LTE) standard [34], the pathloss versus the distance d is equal to $120.9 + 37.6 \log_{10} \mathrm{d}$ (in dB), which corresponds to the constant loss component $C_{pl} =$

$10^{-12.09}$ and the pathloss exponent $n_{pl} = 3.76$. The noise spectral density is $\sigma_N^2 = -149$dBm/Hz. We consider that the distance between users and the server ranges from 0.05 km to 1 km. We keep the power per unit bandwidth equal to $1\mu W/Hz$.

For the proposed DRL scheduler, we update the target policy and value network 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 quite low (around 1800) which is mainly due to the application of Deep Set. Increasing this further unavoidably results in overfitting due to the high stochasticity of the environment. Moreover, keeping the number of parameters low makes our solution fast and cost-effective (both in terms of energy and hardware).

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 with a stringent latency constraint (of just two time slots) and one delay-tolerant class requesting a large amount of data. Every class has the same importance as can be seen from the Imp. column. In the second scenario, the 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 6, we plot the satisfaction ratio per class priority (it means all the 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-types. The Figures 6a,6c,6e have been plotted for $W = 2MHz$, $W = 5MHz$ and $W = 2MHz$ respectively whereas the 6b, 6d, 6f are all done with $\rho = 0$. Figures 6a, 6b, 6e, and 6f are done with $K = 100$ users. Figures 6c and 6d are done with $K = 60$ users.

Recall that the FW algorithm reaches to a suboptimal optima and different initializations will lead to different optima. For that, in each time slot we repeat the algorithm N_{init} times with different initialization each time and we select the best suboptimal point. This method could result in considerable performance improvement, as N_{init} increases but due to computational complexity reasons, we stopped at $N_{init} = 20$. Moreover, as the number of users K increases, unfortunately so does the number of local optima and that of solutions with poor performance making it tougher for the FW to find a good one without a big increase of N_{init}. This is a main reasons why our Deep Scheduler substantially outperforms the Frank-Wolfe algorithm even at moderate $K = 60$. The Deep Scheduler has no trouble keeping the good performance even if K is further increased.

Our proposed DRL scheduler strongly outperforms the knapsack approach. For instance, based on Figure 6b, 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. Also seeing the curves of ILP we confirm that the Deep scheduler is not far from an optimal policy since ILP which uses an oracle constitutes an upper bound on the performance and this bound is not much higher than Deep Scheduler. In Figures 6e, 6e there is a priority class of users which as expected always enjoys a higher satisfaction probability. It is interesting that the Deep Scheduler serves slightly worse the priority class than what the knapsack does. But since the priority counts for the $\frac{0.1}{0.55} \approx 18\%$ of the users and the rest 82% are much better served when Deep Scheduler is used, it is obvious that Deep Scheduler exhibits an overall better than knapsack performance.

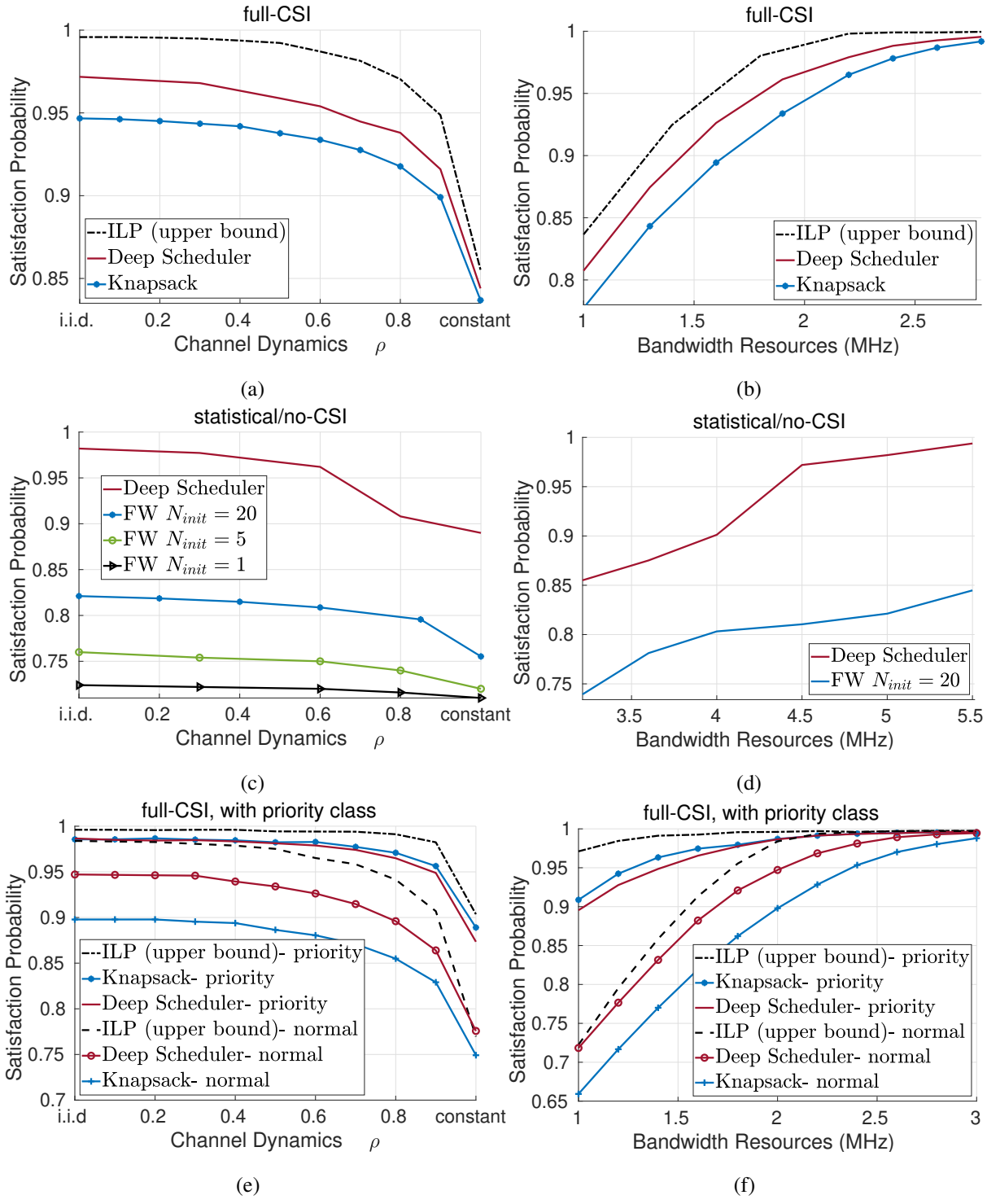


Figure 6: Satisfaction rate of the proposed DRL scheduler qnd the baselines versus ρ (left column) and W (right column). The first and last rows correspond to the case of Table Ia and the row in the middle to Table Ib. Figures 6a, 6b, 6e and 6f refer to full-CSI case while the other to statistical CSI/no-CSI case.

B. Real Data

To assess the applicability of our algorithm in a more realistic setup, we perform experiments on real data using publicly available traces based on real measurements over LTE 4G networks in a city of Belgium [35], [36]. 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 the time scale of one second 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 sec. The value of κ, which mainly depends on the user location, can safely be assumed constant within one second. As the measurements' bandwidth W is not reported, we assume to be 15MHz, giving 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 Eq. (1) to obtain ρ. This is estimated using the trajectory of GPS coordinates given from the traces. A user arriving to the system belongs to a class according to Table II, with its type of transportation chosen randomly; we then sample M_i and its location from the traces accordingly. Knowing in the previous and afterwards seconds 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, i.e. continuously; 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 increase N_{bl} with the size of the resource block constant to 200kHz, again confirming the gains from using our RL based approach. For the Exponential rule, the value of δ_u is set to $\delta_u = \delta = 10^{-2}$ in [31]. Nevertheless, since this value does not provide the best results for every N_{bl} (resource blocks), we tuned this parameter for each N_{bl} in order to offer the highest performance.

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

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

Our proposed algorithm outperforms as seen from table III the baseline ones in the context of full CSI but with real data, both in terms of data rate and satisfaction probability. Nevertheless the

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%

distance to the upper-bound is still significant but we remind that this upper-bound is optimistic since the channel is known in advance.

VII. Conclusion

The problem of scheduling and resource allocation of a time-varying set of users with heterogeneous traffic and diverse QoS requirements has been investigated 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 synthetic and real data showed that the proposed deep scheduler can achieve significant gains compared to state-of-the-art conventional combinatorial optimization methods in both full- and no-CSI cases.

Appendix A
Closed-from expressions for Eqs. (15) and (18)

a) i.i.d. fading ($\rho = 0$): This is the simplest case since there are no time dependencies on the fading, so Eqs. (15) and (18) become using the Eqs. (2) and (3)

$$\Phi(\boldsymbol{w}_{u_0,t}; \mathrm{d}_{u_0}) = P_{u_0}^{fail}(w_{u_0,t}, P; \mathrm{d}_{u_0}), \text{ and } \Phi(\boldsymbol{w}_{u_i,t}) = P_{u_i}^{fail}(w_{u_i,t}, P), \quad i \geq 1.$$

We recall that users u_i for $i \geq 1$ come after user u_0, therefore their locations are unknown and we need to average over them[3].

[3]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

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

b) Static channel ($\rho = 1$): The channel remains the same for each retransmission of the user. 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

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

Therefore, we obtain

$$\Phi(\boldsymbol{w}_{u_0,t}; \mathrm{d}_{u_0}) = \begin{cases} \dfrac{P_{u_0}^{fail}(\max\{\boldsymbol{w}_{u_0,t}\}, P; \mathrm{d}_{u_0})}{P_{u_0}^{fail}(\max\{\boldsymbol{w}_{u_0,t-1}\}, P; \mathrm{d}_{u_0})}, & \text{if } t > t_0 \\ P_{u_0}^{fail}(w_{u_0,t}, P; \mathrm{d}_{u_0}), & \text{if } \quad t = t_0. \end{cases} \tag{23}$$

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 (23), 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. More precisely, if the 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 [37, eq: 37], we have:

$$\begin{aligned}\Phi(\boldsymbol{w}_{u_0,t_0+1}; \mathrm{d}_{u_0}) &= \int_0^{x_{u_0,0}} \int_0^{x_{u_0,1}} \mathbb{P}(|h_{u_0,t_0+1}|{=}x\,|y)\mathbb{P}(|h_{u_0,t_0}|{=}y) dx dy \\ &= 1 - \frac{e^{-x_1^2} Q_1(\frac{x_{u_0,0}}{\sigma_R}, \frac{\rho x_{u_0,1}}{\sigma_R}) - e^{-x_{u_0,0}^2} Q_1(\frac{\rho x_{u_0,0}}{\sigma_R}, \frac{x_{u_0,1}}{\sigma_R})}{2(1 - e^{-x_{u_0,0}^2})}\end{aligned} \tag{24}$$

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):

$$\Phi(\boldsymbol{w}_{u,t_i+1}) = \int_{\mathrm{d}_{min}}^{\mathrm{d}_{max}} \Phi(\boldsymbol{w}_{u_i,t_i+1}; \mathrm{d}_{u_i}) f_{\mathrm{d}}(\mathrm{d}) d\mathrm{d} \tag{25}$$

where $\Phi(\boldsymbol{w}_{u_i,t_i+1}; \mathrm{d}_{u_i})$ is given by (24) by replacing u_0 with u_i. Equation (25) is *intractable* even considering only the first two adjacent retransmissions. This is exacerbated when one considers more retransmissions. 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 either $\rho = 0$ or $\rho = 1$ and keep the best result.

REFERENCES

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

[2] 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 Selected Areas in Communications*, vol. 39, no. 1, pp. 101–115, 2020.

[3] 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

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

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[6] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815*, 2017.

[7] 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.

[8] J. Kober, J. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, pp. 1238–1274, 09 2013.

[9] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

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

[11] 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*, 2018, pp. 766–774.

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

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

[14] 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 Transactions on Wireless Communications*, vol. 18, no. 11, pp. 5141–5152, 2019.

[15] 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 Selected Areas in Communications*, 2021.

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

[17] J. Li and X. Zhang, "Deep reinforcement learning-based joint scheduling of embb and urllc in 5g networks," *IEEE Wireless Communications Letters*, vol. 9, no. 9, pp. 1543–1546, 2020.

[18] 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.

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

[26] 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

[27] 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.

[28] 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.

[29] 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 *International Conference on Learning Representations, ICLR*, Vancouver, Canada, 5 2018.

[30] 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.

[31] 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.

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

[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] (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/

[35] 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 Communications Letters*, vol. 20, no. 11, pp. 2177–2180, 2016.

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

[37] A. H. Nuttall, "Some integrals involving the q_m function," *IEEE Transactions on Information Theory*, vol. 21, no. 1, pp. 95–96, 4 1975.