Meta-Scheduling for the Wireless Downlink through Learning with Bandit Feedback

Jianhan Song
ECE Department
The University of Texas at Austin
jianhansong@utexas.edu

Gustavo de Veciana
ECE Department
The University of Texas at Austin
deveciana@utexas.edu

Sanjay Shakkottai
ECE Department
The University of Texas at Austin
sanjay.shakkottai@utexas.edu

***Abstract*—In this paper, we study learning-assisted multi-user scheduling for the wireless downlink. There have been many scheduling algorithms developed that optimize for a plethora of performance metrics; however a systematic approach across diverse performance metrics and deployment scenarios is still lacking. We address this by developing a meta-scheduler – given a diverse collection of schedulers, we develop a learning-based overlay algorithm (meta-scheduler) that selects that "best" scheduler from amongst these for each deployment scenario. More formally, we develop a multi-armed bandit (MAB) framework for meta-scheduling that assigns and adapts a score for each scheduler to maximize reward (e.g., mean delay, timely throughput etc.). The meta-scheduler is based on a variant of the Upper Confidence Bound algorithm (UCB), but adapted to interrupt the queuing dynamics at the base-station so as to filter out schedulers that might render the system unstable. We show that the algorithm has a poly-logarithmic regret in the expected reward with respect to a genie that chooses the optimal scheduler for each scenario. Finally through simulation, we show that the meta-scheduler learns the choice of the scheduler to best adapt to the deployment scenario (e.g. load conditions, performance metrics).**

***Index Terms*—wireless scheduling, multi-armed bandit, UCB**

I. Introduction

Multi-user scheduling for wireless downlink systems is a particularly challenging task for two key reasons. First, mobile users and services may have diverse performance goals/requirements that should ideally be optimized over a wide variety traffic loads/mixes and heterogeneous user service rates that can vary by over an order of magnitude. Second, because mobile users' see time-varying service rates, it is desirable to incorporate some form of opportunistic scheduling, favoring scheduling users when their service rates are high. To address these challenges wireless schedulers use a combination of the current channel conditions (e.g., obtained through channel quality feedback from mobile users) and current queue backlogs to dynamically assign users to channel resources so as to meet the desired various performance objectives including, e.g., throughput optimality (stability), mean packet/flow delay, delay tails, timely throughput, video quality of experience etc.

Although a substantial number of scheduling algorithms have been proposed, solutions that are able to systematically address the above mentioned challenges are still lacking. Indeed, an algorithm best suited for a given scenario may depend on a variety of factors including traffic load/mix and users' channels, or more generally on the usage patterns associated with the time of day. Moreover in some cases the desired performance metrics for a subset of users may not be easily pre-specified, e.g., measures of video quality, whence it is not clear what type of scheduler to deploy. Furthermore, even if one has access schedulers which are fine tuned to particular scenarios (e.g., learned through a reinforcement learning algorithm), we typically have no performance guarantees over the wide range of settings typical of wireless systems. Whence it is unclear that it is safe to deploy such scheduling policies.

In this paper, we propose a *meta-scheduler* – an online learning (bandit) algorithm which for a given operational scenario dynamically selects the best scheduler from a set of predefined policies (e.g., MaxWeight, Exp rule, Log rule, Priority rule, and Round-Robin). The scheduler in turn, determines user-to-channel assignments. In our approach, scheduling policies are viewed as bandit arms, and the meta-scheduler dynamically chooses the scheduler (aka plays an arm) based on the mobile users' feedback.

In adapting the bandit framework to our queueing setting, we need to address two challenges: *(i)* Arbitrarily switching among schedulers over time can lead to queue instability, even if each of the schedulers is stable. Indeed, one can show that switching between two Max-weight schedulers with different weights can lead to unstable queues. *(ii)* If one or more of the possible schedulers is unstable for a given scenario (e.g. a round-robin scheduler in a high-load wireless setting), then a poor choice may lead to long term instability.

Our approach uses the fact that stable queueing systems typically exhibit cyclical sample-paths associated with busy periods for the overall system. Under appropriate assumptions, the queue dynamics in a busy period are conditionally (given the scheduling policy) independent. Our meta-scheduler thus determines which scheduler (arm to play) only at the beginning of cycles and the chosen scheduler is maintained for the duration of the cycle ensuring independent reward samples across cycles). Further to ensure that cycles do not have infinite durations, the meta-scheduler interrupts[1] cycles that have exceedingly long durations. These decisions have to be properly designed such that cycles due to unstable schedulers

[1]A cycle is interrupted by forcibly making all queues to be zero, e.g., by dropping packets in the buffers.

ISBN 978-3-903176-29-4

(which have unbounded cycle lengths) are played infrequently, and when played, get interrupted (truncated) as soon as possible. Further "good" cycles associated with stable schedulers should not get interrupted. As we will see, designing a sound interruption mechanism in conjunction with online learning through bandit feedback is crucial designing a meta-scheduler which achieves a low regret with respect to a genie algorithm (baseline that always plays the best/highest-reward scheduler for a particular scenario).

A. Contribution

Our main contributions are the following:

- *Meta-scheduler:* We develop a meta-scheduler algorithm based on (UCB + Interruptions). At the beginning of each queuing cycle, the meta-scheduler determines a scheduler to be used for that cycle using a variant of the Upper Confidence Bound (UCB) Algorithm. This consists of (i) determining a score for each scheduler (empirical reward + confidence bonus) that is multiplied by a indicator that estimates if each scheduler is stable (meaning the cycle times are finite), and choosing the scheduler with the highest score; and (ii) determining an interrupt threshold for the cycle, at which time all packets in the queues are dropped if the cycle has not ended before then.
- *Theoretical guarantees:* For the meta-scheduler, we show that the regret (expected cumulative difference in reward) with respect to a genie algorithm that chooses the optimal (highest expected reward) scheduler scales as $O(\log n)$, where n is the number of cycles[2] and correspondingly $O(\log^2 \tau)$ where τ is the time-slot index. Further, the expected number of packets dropped due to interruptions also scales as $O(\log^2 \tau)$.

B. Related Work

Wireless Scheduling. The design of multi-user wireless schedulers has received substantial attention, see e.g., [22] and references therein. For infinitely backlogged user queues researchers have devised various classes of opportunistic schedulers that optimize the sum user utility (fairness criteria) of their long-term throughputs or so called timely throughput, see e.g., [12], [13], [16], [23], [28]. For settings where user queues are subject to stochastic arrivals e.g., packet streams, initial work focused on characterizing *throughput-optimal* schedulers which ensure queue stability if indeed stability can be achieved without prior knowledge of the traffic load and service capacity. These include, for example the MaxWeight rule [3], [26], Exp rule [20] and Log rule [18], which in addition to throughput optimality achieve different user-level performance objectives. Meanwhile, non-throughput-optimal policies can in certain load scenarios provide better performance, e.g., max-rate, proportionally fair, Round-Robin and the priority-based rules. Although there is substantial work in this area, the question of how to realize the best performance tradeoffs among heterogeneous users with diverse performance goals remains open and challenging.

Not surprisingly recently, *reinforcement learning* (RL) approaches have been proposed to address complex scheduling problems, including job scheduling for data centers [17] and wireless scheduling in various settings [6], [9], [19], [30]. RL algorithms provide a general approach to determine good schedulers for specific scenarios and possibly, but substantially more challenging, ones that are good for a range scenarios in terms of the user traffic, service capacity and or performance objectives. Despite showing great potential in several applications, RL based schedulers typically lack rigorous performance guarantees, and thus it is unclear they are safe to deploy.

Multi-armed Bandits. Multi-armed Bandits (MAB) problems have been studied for many decades, with applications to clinical trials, recommendation systems and online advertising; see [7] and [15] for a comprehensive discussion on the state-of-art. In our model, each time we choose a new arm, the corresponding (random) cycle time can be interpreted as a cost. Such problems where each action costs non-unit amount of resources is referred to as *budgeted bandits*. Unlike classical MAB settings, the regret is not parameterized by a time horizon; instead the regret parameterization (and thus, the best arm) involves both the reward and cost variables, which significantly increases the complexity of the problem. This line of work was started by [5] and has been followed in many directions by [1], [27], [29].

A recent study on budgeted bandits in [8] introduces the idea of MAB with interruptions. At each time, a server works on a single task that has a heavy-tailed service completion time. A task can be interrupted if it is taking too long (but with loss in reward). The authors in [8] develop a variant of the Upper Confidence Bound (UCB) algorithm [4] that selects over (a finite set of) tasks as well as a finite set of task interrupt thresholds to discard ongoing tasks, i.e. arms are (task, interrupt-threshold) pairs. Their motivation is to interrupt a task that takes too long so as to start a new one to collect more rewards, and thereby benefit the total reward. Our model is inspired by their work, but significantly differs in the way that we deal with interruptions. In contrast to [8], our goal is to eventually avoid any interruptions, thus, we do not treat interruptions as arms of a bandit. Instead, we dynamically increase the threshold for each task (aka scheduling policy) to ensure we quickly filter out unstable policies for which the cycle times are infinite, while leaving stable policies (eventually) uninterrupted. Algorithmically, our approach modifies UCB with a *multiplicative censoring* that penalizes interruptions from occurring too often, which ensures that unstable arms (with infinite expected cycle completion times) are aggressively eliminated.

Finally, bandit algorithms have also been applied to wireless resource allocation problems more broadly. These include studies in cognitive radio probing [10], spectrum access [2], decentralized wireless computing [14], [24] and most recently, cellular scheduling [25].

Throughout this paper, we use characters in bold font to

[2]The regret scaling is slightly weaker under weaker assumptions on the cycle tail distributions, please see [21] for details.

denote vectors and normal font to denote scalars. Random variables are indicated by capital letters unless stated otherwise.

II. Model Settings

In this section, we consider a multi-arm bandit model for the wireless scheduling problem. The goal is to formulate a *meta-scheduler* that can explore different scheduling policies and learn in an online manner which among the candidate policies is the best, given a certain performance metric. Before introducing the meta-scheduler in detail, we first describe the traffic model and then describe the system from a perspective of regenerative processes. We will see it is natural to allow the meta-scheduler to switch policies only when the system "regenerates". Formal definitions of a *meta-policy* (policy of a meta-scheduler) and its regret are given at the end of this section.

A. Traffic and Service Model

We consider a packet-based queuing system with a set of u different users, denoted by $\mathcal{U}$, and a single server (base station). The system operates in discrete time slots. For simplicity, suppose all packets have the same size. At any time t, define the random vector $\boldsymbol{Q}[t] = (Q_1[t], \cdots, Q_u[t]) \in \mathbb{Z}_+^u$, where $Q_i[t]$ denotes the number of packets of the i-th user at the beginning of time slot t.

The random packet arrivals at time t are denoted by $\boldsymbol{A}[t] = (A_1[t], \cdots, A_u[t])$ where $A_i[t]$ has a integer-valued distribution bounded by $\bar{a}$ for any user $i \in \mathcal{U}$. We assume $(\boldsymbol{A}[t])_{t\geq 1}$ are *i.i.d.* across time and denote its expectation by $\boldsymbol{\lambda}$. The wireless channels' service rates at time t are modeled by a random vector $\boldsymbol{S}[t] = (S_1[t], \cdots, S_u[t])$ where $S_i[t]$ denotes the service rate available to the i-th user at t. $(\boldsymbol{S}[t])_{t\geq 1}$ are *i.i.d.* over time and also independent of the queue lengths and arrival process. A scheduling *policy* will decide which user to serve at each time slot based on the queue and channel state.

Let $\mathcal{C}$ denote the long-term capacity of the system (see [22]). This means for any arrival rate that lies in $\mathcal{C}^o$ (the interior of $\mathcal{C}$), there exists at least one policy that stabilizes the system (the average queue lengths are finite). We require $\boldsymbol{\lambda} \in \mathcal{C}^o$. We say a policy is *stable* (with respect to $\boldsymbol{\lambda}$) if it stabilizes the system.

Now suppose there is a finite set of scheduling policies (or *arms* in the bandit context), denoted by $\mathcal{A}$. For a fixed $\boldsymbol{\lambda} \in \mathcal{C}^o$, $\mathcal{A}$ consists of both stable and unstable policies, denoted by $\mathcal{A}^s(\boldsymbol{\lambda})$ and $\mathcal{A}^u(\boldsymbol{\lambda})$. Assume that $\mathcal{A}^s(\boldsymbol{\lambda}) \neq \phi$.

B. Regenerative Dynamics

Suppose the arrival always occurs right after the beginning of a slot while the transmission happen right before the end of a slot. We say the system *returns idle* when the sum of users' queue lengths is down to 0 from some positive value at the end of a time slot. A *cycle* is defined as the interval of time slots between two consecutive points in time when the system returns idle.[3] Further, without loss of generality, we assume the system starts empty at the beginning of the first slot. We can describe the system's dynamics based on such cycles as follows. The notation and definitions in this section follows [8], with appropriate modifications to reflect our setting.

Each arm k is associated with a stochastic process $((C^{(k)}(n), \boldsymbol{U}^{(k)}(n)))_{n\geq 1}$ where n denotes the index of cycles. If arm k is implemented after n-th time the system returns idle, the system observes a random cycle length $C^{(k)}(n)$ (before it returns idle again), and receives a sequence of **non-negative** rewards $\boldsymbol{U}^{(k)}(n) = (U^{(k)}(n,i) : i = 1, 2, \cdots, C^{(k)}(n))$ for each time slot in the cycle. Note that $C^{(k)}(n)$ for $n \geq 1$ are *i.i.d.* and $C^{(k)}(n) \geq 1$ $a.s.$.

We consider a reward scheme where the generated rewards are *i.i.d.* over cycles and grow no faster than linearly with corresponding time, which is formally stated in the next assumption.

Assumption 1. *The cycle reward sequence $\boldsymbol{U}^{(k)}(n)$ is independent and identically-distributed over n, and satisfies that*

$$0 \leq \sum_{i=1}^{l} U^{(k)}(n,i) \leq \bar{r}l, \quad \forall n \geq 1, 1 \leq l \leq C^{(k)}(n) \tag{1}$$

for some $\bar{r} > 0$.

This assumption holds, for instance, if each packet is associated a bounded reward (e.g., in $[0, 1]$) upon transmission (such as a function of the packet's delay), which is independent of rewards seen in other cycles, and the cumulative reward over a time period is thus bounded by the maximal number of packets transmitted within that period, i.e., $\bar{r} = \bar{a}u$. We denote the (total) cycle reward by $U^{(k)}(n) = \sum_{i=1}^{C^{(k)}(n)} U^{(k)}(n,i)$. Thus, it follows that $U^{(k)}(n)$ for $n \geq 1$ are *i.i.d.* across cycles and bounded as follows

$$0 \leq U^{(k)}(n) \leq \bar{r}C^{(k)}(n) \ a.s., \quad \forall n \geq 1. \tag{2}$$

One question regarding the process is how frequently a policy forces the system to finish a cycle, i.e., the distribution of $C^{(k)}(n)$, which is vital for the meta-scheduler discussed in the sequel. When k is a stable arm, we have $\mathbb{P}(C^{(k)}(n) < \infty) = 1$ and the system will start a new cycle infinitely often. In addition, we have the following assumption on the cycle length of a stable arm.

Assumption 2. *For a given $\boldsymbol{\lambda} \in \mathcal{C}^o$, we assume if arm $k \in \mathcal{A}^s(\boldsymbol{\lambda})$, $C^{(k)}(n)$ is a sub-exponential random variable. This implies that, there exist (possibly $\boldsymbol{\lambda}$-dependent) non-negative parameters (ν_k^2, α_k), such that for all $n \geq 1$,*

$$\mathbb{P}(|C^{(k)}(n) - \mathbb{E}[C^{(k)}(n)]| \geq \varepsilon) \leq \begin{cases} 2e^{-\varepsilon^2/(2\nu_k^2)} & 0 < \varepsilon \leq \frac{\nu_k^2}{\alpha_k}, \\ 2e^{-\varepsilon/(2\alpha_k)} & \varepsilon > \frac{\nu_k^2}{\alpha_k}. \end{cases} \tag{3}$$

[3] In technical terms, a cycle consists of an *idle period* plus a *busy period*. When the system stays empty for a whole time slot, this slot is part of the idle period rather than a new cycle.

Remark 1: This assumption implies that for stable arms $k \in \mathcal{A}^s(\boldsymbol{\lambda})$, $C^{(k)}(n)$ has a light tail on the right (the left side is bounded). One can then show that the empirical average $(1/n)\sum_{i=1}^{n} C^{(k)}(i)$ is sub-exponential with parameters $(\nu_k^2/n, \alpha_k/n)$. By the linear constraint in (2), $U^{(k)}(n)$ is also sub-exponential (with possibly larger parameters). Without loss of generality, we let the parameters (ν_k^2, α_k) suffice both $C^{(k)}(n)$ and $U^{(k)}(n)$ assuming the rewards are properly normalized.

Remark 2: When the system has bounded arrival and channel distributions, and the policies considered are Markovian (choosing service vector at time t based on $\boldsymbol{S}[t]$ and $\boldsymbol{Q}[t]$ only), this assumption holds true following an argument of [11].

If an unstable arm is applied, however, the system is transient and there is a chance that the system will never start a new cycle as $\mathbb{P}(C^{(k)}(n) = \infty) > 0$ for all $k \in \mathcal{A}^u(\boldsymbol{\lambda})$. This suggests that an additional stopping mechanism is needed when an unstable arm is explored by the meta-scheduler.

When $k \in \mathcal{A}^s(\boldsymbol{\lambda})$, observe that $\big((C^{(k)}(n), U^{(k)}(n))\big)_{n\geq 1}$ form a well-defined renewal-reward process. We next define the *renewal reward rate* of a stable policy.

$$r^{(k)} = \frac{\mathbb{E}[U^{(k)}(1)]}{\mathbb{E}[C^{(k)}(1)]} \quad \forall k \in \mathcal{A}^s(\boldsymbol{\lambda}). \tag{4}$$

By Renewal Theory, this rate captures the rate of rewards generated by a policy.

C. Meta-Scheduler, Feedback and Interruptions

A meta-scheduler makes decisions on which arms to use and when, so as to maximize the rate of rewards of the system. In this paper, we will only consider meta-schedulers that comply with the following rules:

(1) A meta-scheduler can switch to another arm when the system returns idle;

(2) A meta-scheduler can *interrupt* a cycle, i.e., discarding all packets currently in the system and forcing the system to start a new cycle, so as to prevent unstable arms from occupying the system indefinitely. Furthermore, as in [8], we only consider conditions triggering such interruptions solely based on cycle time: a cycle gets interrupted when its length exceeds a threshold pre-selected before the cycle starts.

There are several advantages in adopting such rules. First, even scheduling policies that might result in unstable queues can be added to the mix, since the interruptions ensure that cycle times remain bounded. Moreover, they simplify the design of a meta-scheduler, since the system can be fully characterized by arm-independent cycle lengths and rewards, i.e., the collection of processes $\{\big((C^{(k)}(n), \boldsymbol{U}^{(k)}(n))\big)_{n\geq 1} : k \in \mathcal{A}\}$, from the meta-scheduler's point of view regardless of how the actual queues and channels vary with time. This guarantees the independence of statistics for different arms and allows us to apply classical MAB methodologies. Furthermore, such a meta-scheduler preserves properties of regenerative processes that help analysis.

According to the rules mentioned above, a meta-scheduler can only make a *decision* when the system returns idle, which consists of two selections: the arm and the interruption threshold. Formally, we let $\pi = (\pi_n)_{n\geq 1}$ be a *meta-policy* (policy of a meta-scheduler), where $\pi_n = (A_n, L_n) \in \mathcal{A}\times(\mathbb{Z}^+\cup\{+\infty\})$. A decision $\pi_n = (k, l)$ implies that arm k is selected for n-th cycle, and the cycle will be interrupted immediately if it lasts over l time slots.

In order to model cycles under our interruption policy, we let $\hat{C}^{(k,l)}(n) = \min[C^{(k)}(n), l]$ and $\hat{\boldsymbol{U}}^{(k,l)}(n) = (U^{(k)}(n,i) : i = 1, 2, \cdots, \hat{C}^{(k,l)}(n))$. The **observed** (total) cycle reward $\hat{U}^{(k,l)}(n) = \sum_{i=1}^{\hat{C}^{(k,l)}(n)} U^{(k)}(n,i)$. Note that it still holds that $0 \leq \hat{U}^{(k,l)}(n) \leq \bar{r}\hat{C}^{(k,l)}(n)$ almost surely.

If $\pi_n = (k, l)$, we assume stochastic feedback Z_n is received for n-th cycle by the meta-scheduler as follows,

$$Z_n = (\hat{C}^{(k,l)}(n), \hat{U}^{(k,l)}(n), \mathbb{1}_{\{\hat{C}^{(k,l)}(n)<C^{(k)}(n)\}}).$$

An illustration of the meta-policy dynamics is shown in Figure 1. Note that the reward for each single time slot is not required in the feedback. This suggests that if performance is evaluated at user side, additional communication cost only occurs at the end of a cycle.

We assume π_n is solely based on the history of actions and feedback up to the decision. Thus, an *admissible meta-policy* considered in this paper is formally defined as follows. This is analogous to a similar notion in [8].

Definition 1 (Admissible Meta-Policy)**.** *We call a meta-policy $\pi = (\pi_n)_{n\geq 1}$ admissible if $\pi_n \in \mathcal{F}_n$ where $\mathcal{F}_n := \sigma(\pi_1, Z_1, \pi_2, Z_2, \cdots, \pi_{n-1}, Z_{n-1})$ is the σ-field induced by all the random decisions and feedback before n-th cycle.*

Our goal is to design a good meta-policy that satisfies the following two objectives: (1) it suffers negligible throughput loss, i.e., the number of packets discarded due to interruptions by the meta-scheduler is sub-linear in time, and (2) it has a sub-linear expected regret over a given time horizon. We will define the regret in the next section.

D. Regret

As in the traditional MAB setting, we are interested in the *regret* of a meta-policy as compared to an optimal over a given time horizon τ. The regret for the meta-policy π stems from two reasons: (i) playing suboptimal arms (schedulers), and (ii) interrupting ongoing cycles. To formally define the regret, we follow a similar approach as in [8]. First, note that the number of cycles within a time horizon τ is a random variable, which can be viewed as a counting process.

Definition 2 (Counting Process)**.** *Consider a meta-policy π that is admissible. The total time of the first n-th cycle can be written as*

$$S_n^{\pi} = \sum_{i=1}^{n} \sum_{(k,l)\in\mathcal{A}\times\mathbb{Z}^+} \mathbb{1}_{\{\pi_s=(k,l)\}}\hat{C}^{(k,l)}(i).$$

Define a counting process $(N_\pi[\tau])_{\tau\geq 1}$ as follows.

$$N_\pi[\tau] = \max\{n : S_n^{\pi} \leq \tau\}.$$

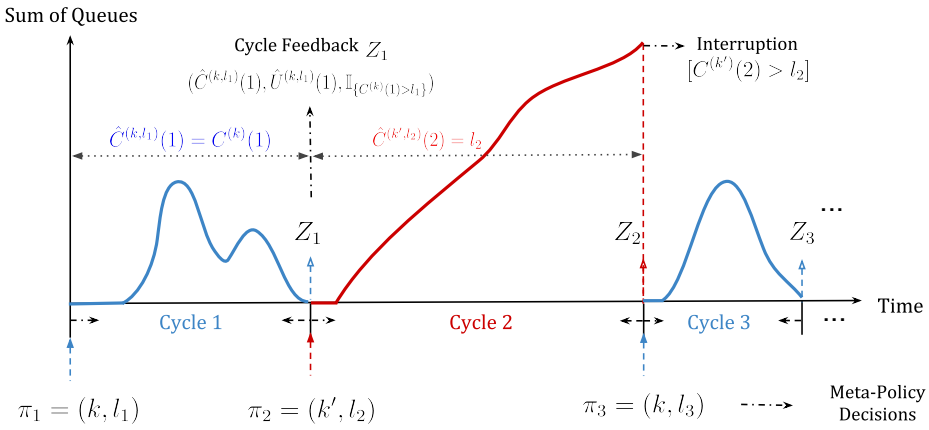

Fig. 1: Illustration of a meta-policy selecting arms from $\mathcal{A} = \{k, k'\}$. At the start of the process, the meta-scheduler makes decision $\pi_1 = (k, l_1)$, and receives feedback Z_1 after the system experiences a full cycle. Then the meta-scheduler decides $\pi_2 = (k', l_2)$, but has to interrupt the cycle as the system does not return idle before the cycle time reaches l_2. The meta-scheduler then collects feedback Z_2 and starts a new cycle with $\pi_3 = (k, l_3)$.

Note that $N_\pi[\tau]$ indicates the number of completed cycles within time horizon τ.

Definition 3 (Cumulative Reward)**.** *Given a time horizon τ, the cumulative reward for an admissible meta-policy π is a random variable given as follows. (Denote $\tilde{N} := N_\pi[\tau]$ for notation simplicity.)*

$$\begin{aligned}\mathrm{Rew}_\pi[\tau] = &\sum_{i=1}^{\tilde{N}} \sum_{(k,l)} \mathbb{1}_{\{\pi_i=(k,l)\}} \hat{U}^{(k,l)}(i) \\ &+ \sum_{(k,l)} \mathbb{1}_{\{\pi_{\tilde{N}+1}=(k,l)\}} \sum_{j=1}^{\tau - S^\pi_{\tilde{N}}} U^{(k)}(\tilde{N}+1, j).\end{aligned} \tag{5}$$

The cumulative reward is the sum of (observed) cycle rewards from the first $N_\pi[\tau]$ completed cycles and the reward from the next uncompleted cycle up to time τ.

We call a meta-policy *simple-static* if the meta-scheduler consistently selects an arm with no cycle interruption. Let $\pi^{(k)}$ be the simple-static meta-policy selecting arm k, i.e., $\pi^{(k)}_n = (k, +\infty), \forall n \geq 1$. In this paper, we define the regret with respect to the best simple-static meta-policy π^{opt} that is stable and generates the most rewards (in expectation) within a given time. By the renewal theorem, $\lim_{\tau\to\infty} \mathrm{Rew}_{\pi^{(k)}}[\tau]/\tau = r^{(k)}$ $a.s.$ for all $k \in \mathcal{A}^s(\boldsymbol{\lambda})$. This implies that $\pi^{\mathrm{opt}} = \pi^{(k^*)}$ where $k^* = \mathrm{argmax}_{k\in\mathcal{A}^s(\boldsymbol{\lambda})} r^{(k)}$. The regret is formally defined as follows.

Definition 4 (Cumulative Regret)**.** *Let π^{opt} be the optimal simple-static meta-policy, i.e.*

$$\pi^{\mathrm{opt}}_n = (k^*, \infty), \quad \forall n \geq 1 \tag{6}$$

where $k^* = \mathrm{argmax}_{k\in\mathcal{A}^s(\boldsymbol{\lambda})} r^{(k)}$. The regret of meta-policy π with respect to π^{opt} over any time horizon τ is defined as

$$\mathrm{Reg}_\pi[\tau] = \mathbb{E}[\mathrm{Rew}_{\pi^{\mathrm{opt}}}[\tau] - \mathrm{Rew}_\pi[\tau]]. \tag{7}$$

In the remaining sections, we will simply refer to k^* as the *optimal arm* (assumed to be unique). For notation simplicity, we suppress k^* as a single asterisk in the superscript when there is no ambiguity (e.g., $r^{(*)} := r^{(k^*)}$).

III. A UCB Meta-Scheduler with Interruption

To guarantee negligible throughput loss and a sub-linear regret as discussed in Section II, the meta-scheduler should wisely select the arms and interruption thresholds such that the optimal arm is being applied at most of the time, and the packet discard hardly occurs. This implies the following guidelines when designing the algorithm: 1) the number of times a suboptimal arm (either unstable or stable) gets selected should be sub-linear in time; and 2) the unstable arms' (possibly infinitely) long cycles must be stopped, while the cycles of the optimal arm should be preserved with little interruption.

Motivated by these guidelines, we propose a UCB-type meta-scheduler with a properly-designed interruption rule. Before presenting the meta-scheduler, let us first introduce several parameters to be used in our algorithm in the following assumption.

Assumption 3. *We assume the following parameters are given a priori with respect to some $\mathcal{A}_0$, a subset of arms satisfying: $\{k^*\} \subseteq \mathcal{A}_0 \subseteq \mathcal{A}^s(\boldsymbol{\lambda})$.*

(1) $\mu_{\min}$ and $\mu_{\max}$ such that $\mu_{\min} \leq \mathbb{E}[C^{(k)}(1)] \leq \mu_{\max}$ for all $k \in \mathcal{A}_0$.

(2) $r_{\max}$ such that $\mathbb{E}[U^{(k)}(1)|C^{(k)}(1) = l] \leq r_{\max}\, l$ for all $l \geq 1$ and all $k \in \mathcal{A}_0$. Note that $r_{\max}$ exists by Assumption 1, and $r_{\max} \geq r^{(*)}$.

(3) Parameters (ν^2, α) such that for all $k \in \mathcal{A}_0$, $C^{(k)}(1), U^{(k)}(1)$ are both (ν^2, α)-sub-exponential random variables as described in Assumption 2. In addition, we assume that the l-interrupted cycle reward $\hat{U}^{(k,l)}(1)$ is (ν^2, α)-sub-exponential for all $l \geq 2\mathbb{E}[C^{(k)}(1)]$.

In the algorithm, these parameters serve as hyper-parameters that need to be further tuned. To remove ambiguity, for a given set of hyper-parameters used in implementation, we will refer to $\mathcal{A}_0$ as the **largest** set of arms for which those hyper-parameters suffice the conditions. We assume $k^* \in \mathcal{A}_0$ (as the weakest notion) to achieve sub-linear packet loss and regret.

Remark 3: For technical reasons, we also require $\hat{U}^{(k,l)}(1)$ to be sub-exponential *under the same parameters*[4] (ν^2, α) as those of $U^{(k)}(1)$ when l is large enough. This does not make the assumption significantly stronger, since one can always pick the parameters large enough to satisfy this condition. The condition $l \geq 2\mathbb{E}[C^{(k)}(1)]$ is chosen for simplicity. Indeed, the condition can be replaced by $l \geq (1+\gamma)\mathbb{E}[C^{(k)}(1)]$ for any $\gamma > 0$ (the algorithm parameters will be changed accordingly).

To simplify notation and avoid ambiguity, let $C_s^{(k)}$ and $\hat{C}_s^{(k,l)}$ ($U_s^{(k)}$ and $\hat{U}_s^{(k,l)}$) be the full and observed cycle length (reward) of arm k when it is selected the s-th time (we call it s-th *sample* of k). Denote by $T_n^{(k)}$ as the number of times arm k has been chosen in the first n decisions. Thus, if $A_n = k$, $(C^{(k)}_{T_n^{(k)}}, U^{(k)}_{T_n^{(k)}}) = (C^{(k)}(n), U^{(k)}(n))$.

Not unlike the classical UCB algorithm, the meta-scheduler learns the arm statistics by keeping track of the empirical averages of cycle lengths and rewards. We formally define the *empirical rate* of arm k after s samples as $\hat{R}_s^{(k)}$. For all $s \geq 1$,

$$\hat{R}_s^{(k)} = \frac{\sum_{i=1}^{s} \hat{U}_i^{(k,F_i^{(k)})}}{\sum_{i=1}^{s} \hat{C}_i^{(k,F_i^{(k)})}}, \tag{8}$$

where $F_i^{(k)}$ denotes the threshold level for arm k's i-th sample. As a convention, the empirical rate equals 0 when $s = 0$. Let $\hat{R}^{(k)}(n) := \hat{R}^{(k)}_{T_n^{(k)}}$ be the empirical rates for the k-th arm **after** n-th cycle in the system.

Our meta-scheduler is formally presented in Algorithm 1. We will discuss the mathematical design in a more rigorous manner in the next section. Before that, let us first give some intuition as follows.

First we observe that to avoid constantly interrupting a stable arm, it is necessary (and sufficient) to apply an interruption rule where the threshold of each arm is set to slowly grow with the number of samples (note that a fixed threshold will always result in linear throughput loss). We define a threshold function as in (10), where f_s denotes the threshold for the s-th sample of **any** arm k (i.e., $F_s^{(k)} = f_s$). With this design, the expected number of interruptions imposed on the optimal arm can be bounded by a constant $\pi^2/6$ if β and κ are large enough.

[4]Note that $\hat{U}^{(k,l)}(1)$ is sub-exponential (indeed bounded). However, the fact that $U^{(k)}(1)$ is (ν^2, α)-sub-exponential does not imply the same parameters suffice $\hat{U}^{(k,l)}(1)$.

Algorithm 1 UCB Meta-Scheduler with Interruptions

Input: Set of scheduling policies $\mathcal{A}$.

Hyper-parameters: $\mu_{\min}, r_{\max}, \alpha, \nu^2, \beta, \kappa$ $\quad (\kappa > 4\alpha,$ $\beta > 2(\mu_{\max} + \nu^2/\alpha))$

$$\Delta(\epsilon, \epsilon') := \frac{\epsilon(1 + r_{\max}) + \epsilon'}{\mu_{\min} + \epsilon}. \tag{9}$$

$$f_s := \beta + \kappa \log s. \tag{10}$$

for $n = 1, \cdots, |\mathcal{A}|$ **do**

  Run every arm $k \in \mathcal{A}$ once (with interruption threshold β), then initialize $\hat{R}_1^{(k)}$

for $n = |\mathcal{A}|+1, |\mathcal{A}|+2, \ldots$ **do**

  [Before a cycle]

  $\forall k \in \mathcal{A}$, compute

$$\epsilon_n^{(k)} = \begin{cases} \sqrt{\frac{6\nu^2 \log n}{T_{n-1}^{(k)}}} & \sqrt{\frac{6\nu^2 \log n}{T_{n-1}^{(k)}}} \leq \frac{\nu^2}{\alpha}, \\ \frac{6\alpha \log n}{T_{n-1}^{(k)}} & \text{otherwise,} \end{cases} \tag{11}$$

$$\epsilon_n'^{(k)} = \frac{1}{T_{n-1}^{(k)}} \sum_{i=1}^{T_{n-1}^{(k)}} \frac{r_{\max}}{i^{\kappa/2\alpha}} e^{-\beta/4\alpha}(\beta + 2\alpha + \kappa \log i + 1), \tag{12}$$

$$B_n^{(k)} = \hat{R}^{(k)}(n-1) + \Delta(\epsilon_n^{(k)}, \epsilon_n'^{(k)}), \tag{13}$$

$$I_n^{(k)} = \begin{cases} 1 & \text{if } \sum_{i=1}^{T_{n-1}^{(k)}} \mathbb{1}_{\{C_i^{(k)} > f_i\}} < \frac{\pi^2}{6} + \sqrt{2T_{n-1}^{(k)} \log n}, \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

$$A_n = \operatorname*{argmax}_{k \in \mathcal{A}} B_n^{(k)} I_n^{(k)}. \tag{15}$$

$$L_n = f_{T_n^{(A_n)}}. \tag{16}$$

  Choose $\pi_n = (A_n, L_n)$.

  [After a cycle]

  Observe $\hat{U}^{(A_n, L_n)}(n)$ and $\hat{C}^{(A_n, L_n)}(n)$.

  Update $\hat{R}^{(A_n)}(n)$.

The meta-scheduler starts with running each policy once with an initial interruption level β and initializing the empirical rate $\hat{R}_1^{(k)}$ for any $k \in \mathcal{A}$. After this initialization phase, before each decision, the meta-scheduler will compare a "score" of each arm at decisions, which equals the sum of its empirical reward rate and an *upper confidence bound* (see (13)). The UCB term, defined in (9) and (11)(12), is used to compensate the possibly under-performing empirical rate in order to ensure adequate exploration before finding the truly optimal arm eventually. We will show that *w.h.p.*, $\hat{R}^{(*)}(n-1) + \Delta(\epsilon_n^{(*)}, \epsilon_n'^{(*)}) > r^{(*)}$ for any $n \geq |\mathcal{A}| + 1$. Meanwhile, the score of a suboptimal **stable** arm will be below $r^{(*)}$ after it is sufficiently explored.

Moreover, we design a "stability indicator" to eliminate unstable arms by utilizing interruption as a signal indicating whether an arm frequently induce long cycles. The

Bernoulli random variable $\mathbb{1}_{\{C_s^{(k)} > f_s\}}$ denotes whether the s-th sample for arm k is clipped by its threshold. The value $\mathbb{E}[\sum_{i=1}^{s} \mathbb{1}_{\{C_i^{(k)} > f_i\}}]$ is bounded by $\pi^2/6$ for the optimal arm, but grows linearly for the unstable arms (since they are frequently clipped). This motivates us to use the value of $\sum_{i=1}^{s} \mathbb{1}_{\{C_i^{(k)} > f_i\}}$ to eliminate unstable arms. The stability indicator is defined as in (14). When $\sum_{i=1}^{s} \mathbb{1}_{\{C_i^{(k)} > f_i\}}$ is larger than $\pi^2/6$ plus a concentration bound, the indicator will rule out arm k in this round.

After computing the UCB term and the stability indicator, the meta-scheduler will pick the arm with the best score $B_n^{(k)}$ and a positive value of $I_n^{(k)}$, and the threshold of that cycle is determined by the threshold function. A new cycle then starts according to this decision. When the cycle is finished, the meta-scheduler observes the (possibly clipped) cycle length and reward before updating the empirical rate for the selected arm. The updated statistics will be used to determine the next decision. We present the main result below, which shows that the meta-scheduler results in negligible packet loss and a sub-linear expected regret. Complete details are available in [21].

Theorem 1. *Provided that the hyper-parameters used in Algorithm 1 satisfy Assumption 3 when $\mathcal{A}_0 = \mathcal{A}^s(\boldsymbol{\lambda})$, the meta-policy π induced by Algorithm 1 has the following properties:*

(1) $\mathbb{E}[D_\pi[\tau]] = O(\log^2 \tau)$ where $D_\pi[\tau]$ denotes the number of packets discarded over any time horizon τ,

(2) The regret $\mathrm{Reg}_\pi[\tau] = O(\log^2 \tau)$.

Further, when the hyper-parameters suffice only for $\mathcal{A}_0 = \{k^*\}$, $\mathbb{E}[D_\pi[\tau]] = O(\log^{2+\delta} \tau)$ and $\mathrm{Reg}_\pi[\tau] = O(\log^{2+\delta} \tau)$ for any $\delta > 0$.

Acknowledgements

This work was supported by NSF grants CNS-1731658, CNS-1718089 and CNS-1910112, Army Futures Command Grant W911NF1920333, and the Wireless Networking and Communications Group Industrial Affiliates Program.

References

[1] Shipra Agrawal and Nikhil Devanur. Linear contextual bandits with knapsacks. In *Advances in Neural Information Processing Systems*, pages 3450–3458, 2016.

[2] Sahand Haji Ali Ahmad and Mingyan Liu. Multi-channel opportunistic access: A case of restless bandits with multiple plays. In *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1361–1368. IEEE, 2009.

[3] Matthew Andrews, Krishnan Kumaran, Kavita Ramanan, Alexander Stolyar, Rajiv Vijayakumar, and Phil Whiting. Scheduling in a queuing system with asynchronously varying service rates. *Probability in the Engineering and Informational Sciences*, 18(2):191–217, 2004.

[4] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, (47):235 – 256, 2002.

[5] Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 207–216. IEEE, 2013.

[6] Ravikumar Balakrishnan, Kunal Sankhe, V Srinivasa Somayazulu, Rath Vannithamby, and Jerry Sydir. Deep reinforcement learning based traffic- and channel-aware ofdma resource allocation. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2019.

[7] Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

[8] Semih Cayci, Atilla Eryilmaz, and Rayadurgam Srikant. Learning to control renewal processes with bandit feedback. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(2):43, 2019.

[9] Sandeep Chinchali, Pan Hu, Tianshu Chu, Manu Sharma, Manu Bansal, Rakesh Misra, Marco Pavone, and Sachin Katti. Cellular network traffic scheduling with deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[10] Yi Gai, Bhaskar Krishnamachari, and Rahul Jain. Learning multiuser channel allocations in cognitive radio networks: A combinatorial multi-armed bandit formulation. In *2010 IEEE Symposium on New Frontiers in Dynamic Spectrum (DySPAN)*, pages 1–9. IEEE, 2010.

[11] Bruce Hajek. Hitting-time and occupation-time bounds implied by drift analysis with applications. *Advances in Applied probability*, 14(3):502–525, 1982.

[12] I-Hong Hou. Scheduling heterogeneous real-time traffic over fading wireless channels. *IEEE/ACM Transactions on Networking*, 22(5):1631–1644, 2013.

[13] I-Hong Hou and Panqanamala Ramana Kumar. Packets with deadlines: A framework for real-time wireless networks. *Synthesis Lectures on Communication Networks*, 6(1):1–116, 2013.

[14] Yi-Hsuan Kao, Kwame Wright, Bhaskar Krishnamachari, and Fan Bai. Online learning for wireless distributed computing. *arXiv preprint arXiv:1611.02830*, 2016.

[15] Tor Lattimore and Csaba Szepesvári. Bandit algorithms. *preprint*, 2018.

[16] Xiaojun Liu, Edwin K. P. Chong, and Ness B. Shroff. Opportunistic transmission scheduling with resource-sharing constraints in wireless networks. *IEEE Journal on Selected Areas in Communications*, 19(10):2053–2064, 2001.

[17] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, pages 50–56, 2016.

[18] Bilal Sadiq, Seung Jun Baek, and Gustavo De Veciana. Delay-optimal opportunistic scheduling and approximations: The log rule. *IEEE/ACM Transactions on Networking (TON)*, 19(2):405–418, 2011.

[19] Einar Cesar Santos. A simple reinforcement learning mechanism for resource allocation in lte-a networks with markov decision process and q-learning. *arXiv preprint arXiv:1709.09312*, 2017.

[20] Sanjay Shakkottai and Alexander L Stolyar. Scheduling for multiple flows sharing a time-varying channel: The exponential rule. *Translations of the American Mathematical Society-Series 2*, 207:185–202, 2002.

[21] Jianhan Song, Gustavo de Veciana, and Sanjay Shakkottai. Meta-scheduling for the wireless downlink through learning with bandit feedback. Technical Report, UT Austin, May 2020.

[22] Rayadurgam Srikant and Lei Ying. *Communication networks: an optimization, control, and stochastic networks perspective*. Cambridge University Press, 2013.

[23] Alexander L Stolyar. On the asymptotic optimality of the gradient scheduling algorithm for multiuser throughput allocation. *Operations research*, 53(1):12–25, 2005.

[24] Yuxuan Sun, Xueying Guo, Sheng Zhou, Zhiyuan Jiang, Xin Liu, and Zhisheng Niu. Learning-based task offloading for vehicular cloud computing systems. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2018.

[25] Isfar Tariq, Rajat Sen, Gustavo de Veciana, and Sanjay Shakkottai. Online channel-state clustering and multiuser capacity learning for wireless scheduling. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 136–144. IEEE, 2019.

[26] L. Tassiulas and A. Ephremides. Dynamic server allocation to parallel queues with randomly varying connectivity. *IEEE Transactions on Information Theory*, 39:466–478, March 1993.

[27] Long Tran-Thanh, Archie Chapman, Alex Rogers, and Nicholas R Jennings. Knapsack based optimal policies for budget–limited multi–armed bandits. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[28] Pramod Viswanath, David N. C. Tse, and Rajiv Laroia. Opportunistic beamforming using dumb antennas. *IEEE transactions on information theory*, 48(6):1277–1294, 2002.

[29] Yingce Xia, Haifang Li, Tao Qin, Nenghai Yu, and Tie-Yan Liu. Thompson sampling for budgeted multi-armed bandits. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[30] Hao Ye, Geoffrey Ye Li, and Biing-Hwang Fred Juang. Deep reinforcement learning based resource allocation for v2v communications. *IEEE Transactions on Vehicular Technology*, 68(4):3163–3173, 2019.