The Dissertation Committee for Jianhan Song
certifies that this is the approved version of the following dissertation:

Online Learning Algorithms For Wireless Scheduling

Committee:

Gustavo de Veciana, Supervisor

Sanjay Shakkottai, Supervisor

Constantine Caramanis

Aryan Mokhtari

John Hasenbein

Online Learning Algorithms For Wireless Scheduling

by

Jianhan Song

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

November 2023

Dedicated to my Parents.

Acknowledgments

It is with great pleasure that I express my gratitude to all those who have offered their help and guidance to me throughout these past years. To anyone whose name I may have missed, please know that I am sincerely grateful for the assistance you have provided, contributing significantly to the completion of my dissertation.

I would like to thank the faculty from my undergraduate years who introduced me to the world of mathematics and engineering and fostered my passion for academic research. Prof. Hong Cheng, Prof. Chandra Nair, Prof. Sherman Chow, Prof. Jaggi Sidharth from the Chinese University of Hong Kong, and Prof. Lav Varshney from the University of Illinois at Urbana-Champaign during my exchange study, are just a few among the many great mentors. Their guidance inspired me to explore my intellectual potential and embark on a journey as a Ph.D. student.

I consider myself incredibly fortunate to have Prof. Gustavo de Veciana and Prof. Sanjay Shakkottai as my doctoral advisors. They have provided invaluable support throughout my Ph.D. journey, assisting me in every step from selecting research topics, navigating complex mathematical proofs, to refining my paper writing styles. I am particularly grateful for not only their technical guidance but also their extension of patience, kindness, and encouragement during many challenging times in both my academic and personal life in the past few years. It goes without saying that I could not have undertaken this journey without their strong support.

I would also like to express my gratitude to my committee members Prof. Constantine Caramanis, Prof. Aryan Mokhtari, and Prof. John Hasenbein, for their kind and valuable feedback on my research. My sincere thanks also go to Prof. Hao Zhu, my academic advisor in my first year, during which I completed my first research project at UT Austin under her mentorship. I also extend my appreciation to Prof. Hyeji Kim for granting me an opportunity to serve as a teaching assistant in her probability class during my last semester, a role that I feel greatly honored to fulfill while passing the torch to the young academic prospects. I would also like to thank all my instructors at UT Austin, whose courses laid a strong foundation of knowledge, greatly supporting my research efforts while also satisfying my intellectual curiosity.

I am grateful for all the help I received from my fellow WNCG members. A special acknowledgment should go to Isfar Tariq for guiding me in my initial projects. I also extend my thanks to colleagues such as Alejandro Gomez-Leos, Parikshit Hedge, Jean Abou Rahal, Jeong Yeol Kwon, Shubhankar Agarwal, Jiaxiao Zheng, Yan Han, Yi Zhang, Kartik Patel, Yuqiang Heng, Ronshee Chawla, Hongxiang Xie, Xiangxu Yu, Zaixi Shang, and many others for their support and counsel in my research, coursework, and various aspects of my Ph.D. life. Additionally, I would like to thank Gulick Melanie, Karen Little, and numerous other administrative staff members at the ECE Department and

WNCG for their assistance as I navigated through the Ph.D. experience.

During my Ph.D., I had the opportunity to intern at several prestigious tech companies. These experiences not only broadened my knowledge and boosted my research endeavors but also served as essential preparation for my future career. I am thankful to V Srinivasa Somayazulu at Intel Lab, Guohao Zhang, Sukwon Chung, Peng Wu at Meta, and Yinghao Sun at Google for providing me with important learning experience during their roles as supervisors/managers in my internships.

Finally, I would like to extend my sincere thanks to my family members. I am deeply grateful to my parents, Fengjun Yu and Shijun Song, for their unconditional love and support throughout the pursuit of my Ph.D. degree. I also extend my heartfelt gratitude to my wife, Jiaru, along with our cherished feline companions — Wilbur, Milky, Raku, and Ali for their unwavering emotional support during the highs and lows of my journey.

In conclusion, I would like to express my appreciation to every mentor, classmate, colleague, friend, acquaintance, and stranger who, at various points, intersected with my life's journey, offering warmth, depth, complexity, and uniqueness to my experience, which has shaped the individual I have become and my perception of the world. Each encounter is a great fortune. May we continue to discover joy and love as we fulfill our existence in this vast universe.

Online Learning Algorithms For Wireless Scheduling

Publication No. ____________

Jianhan Song, Ph.D.
The University of Texas at Austin, 2023

Supervisors: Gustavo de Veciana
Sanjay Shakkottai

Online learning, and more specifically, multi-armed bandit algorithms, has recently garnered significant interest across diverse fields. Within an online learning framework, agents can leverage past interactions with their environment to optimize future decisions, making it an ideal mechanism for use in applications such as recommendation systems. Driven by these advantages, we believe that the online learning approach can be effectively employed to address resource allocation and scheduling challenges in wireless systems, with the potential to enhance the adaptability and robustness of system performance. In this dissertation, we explore the applications of multi-armed bandit algorithms in various wireless settings, showcasing their efficacy through both theoretical analysis and empirical demonstrations.

We first studied the multi-user scheduling problem for the wireless downlink with instantaneous channel rate and queue information. We introduced the concept of ”meta-scheduling”, which formulates the task of selecting an optimal wireless scheduler as a bandit problem, and proposed a UCB-type bandit algorithm designed to adapt to the dynamics of a queueing system. Expanding on the meta-scheduling concept, we then studied a model of hierarchical scheduling in the context of network slicing, in which the base station learns the optimal option among infinitely-many arms. Our approach involves formulating the problem as a blackbox optimization and addressing it using an HOO-type bandit algorithm adaptive to random queueing cycles. Lastly, we transitioned into a multi-agent setting, where decisions of learning agents in close proximity are coupled with each other through interference. Within this context, we identified a low-complexity structure termed the ”weakly-coupled system”, and developed a decentralized bandit algorithm to facilitate the learning of optimal collective actions. Throughout each of these segments, we presented rigorous theoretical proofs demonstrating that the proposed algorithms exhibit the desired sub-linear regret compared to an idealized genie. Furthermore, we validated the efficacy of the algorithms through a series of experiments using simulation.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter aims to establish the context and drive behind the research underlying this dissertation, as well as to outline our primary contributions. Section 1.1 delves into the background of the problems, while Section 1.2 highlights the motivation that propels the proposed research. In Section 1.3, the dissertation statement is presented alongside a summary of the research contributions. Lastly, Section 1.4 addresses the organization of this dissertation.

1.1 Background

Efficient resource allocation is a crucial part of the design of wireless systems, which encompasses dynamic scheduling of various resources such as time slots, frequency sub-channels, power, and more. It is a particularly challenging task that involves real-time decisions that need to be optimized over a wide variety of scenarios and objectives, such as diverse performance goals/requirements, heterogeneous traffic loads and channel conditions, interference, etc. For instance, in the multi-user downlink scheduling problem, a base station may choose which user or users to serve for each time slot based on queue backlogs, time-varying service rates, and users' QoS requirements.

Previous research has emphasized developing policies tailored to specific application use cases – for example, Log-Rule [1] was introduced for multi-user scheduling to optimize packets' mean delay. Despite the numerous scheduling policies/algorithms proposed for each specific task, however, manually selecting a policy based on domain knowledge is not a systematic solution, and can lead to several drawbacks. First, the effectiveness of an algorithm for a particular scenario is contingent on several factors, such as heterogeneous traffic loads and user channel conditions, which can fluctuate across different time scales. Furthermore, with the increasing diversity of service requirements in modern wireless systems to cater to varying needs, it may not be straightforward to map the application scenario to any well-studied performance metric and apply good strategies accordingly. Therefore, it is of significant interest to develop a general approach that helps decision-making agents deploy well-behaved policies that address the system's complexity and adapt to changes in the environment.

Reinforcement learning (RL) based solutions have recently gained much attention in both academia and industry. One approach is to frame resource allocation as a Markov Decision Process (MDP) problem and apply a data-driven method to train/learn the optimal strategy for a specified reward/performance metric. However, in order to be robust to the change of environments, this method requires considerable exploration of various regimes as well as a significant computational effort, which is not suitable to be conducted online. It is crucial to properly model the performance metric and traffic environment where the scheduler will be deployed during the offline training. This further raises safety concerns due to the potential mismatch between the training and deployment environments.

1.2 Motivation of Proposed Research

Given the constraints of current approaches, we believe that multi-armed bandit (MAB) frameworks can be applied as an online learning scheme to address wireless resource allocation problems and have great potential to improve the adaptivity and robustness of the performance of a scheduling system. In a bandit framework, the agent or agents (e.g., base stations, APs, etc.) explore multiple (sometimes infinitely many) possible options, collect reward feedback regarding the performance of the corresponding decisions, and adaptively adjust further actions (regarding which option to explore) based on the feedback history.

It is worth noting that, unlike general RL methods where an MDP is formulated, there is no state transition in bandit frameworks — this may seem contradictory to the settings of typical scheduling problems, as the decisions are usually dependent on system states such as queues and channels. We tackle this issue by carefully choosing new timescales and action (aka arm) sets such that each action corresponds to a single reward. In particular, we propose the concept of "meta-scheduling", where pre-defined policies are viewed as bandit arms, and the action reward is measured by the performance of each policy over a period of time, ensuring that each policy starts from the same system state.

This approach has the following advantages. Firstly, meta-scheduling allows the agents to benefit from existing policies (which can be either manually designed or fine-tuned RL policies) without additional design complexity. Secondly, in the context of wireless systems, this framework is more computationally efficient to implement in an online manner compared to the data-driven RL approach, and can quickly adapt to environment change without additional training. Lastly, it allows a thorough mathematical analysis, which can lead to rigorous performance guarantees.

1.3 Summary of Contributions

The thesis statement for this dissertation is as follows:

Wireless resource allocation problems can be approached by multi-armed bandit algorithms to systematically address the complexity of systems and improve the robustness to changing environments.

First, we study online learning-assisted multi-user scheduling for the wireless downlink. We propose the notion of meta-scheduling — given a diverse collection of schedulers, we develop a bandit-based overlay algorithm (meta-scheduler) that learns which is the "best" for the current deployment scenario. Our meta-scheduler algorithm is based on a variant of the Upper Confidence Bound (UCB) algorithm, but adapted to interrupt the queuing dynamics at the base station so as to avoid 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. Using simulation, we show that the meta-scheduler is able to quickly learn the best choice of schedulers to best adapt to the deployment scenario (e.g. load conditions, performance metrics).

Second, we build on the meta-scheduling idea and study a model of hierarchical scheduling for use in conjunction with network slicing, in which the base station learns the optimal option among infinitely many arms in a continuously-valued parameter set. In network slicing applications, wireless users are grouped into “slices”, and a hierarchical scheduler is implemented by combining an inter-slice scheduler allocating resources amongst slices (which can be parameterized by a weight vector), and intra-slice schedulers which opportunistically allocate resources to users/services within slices. We formulate the problem of optimizing the inter-slice scheduler to maximize system utility as an online black-box optimization — The goal is to learn the best weight vector. We develop a bandit algorithm operating across queueing cycles by building on Hierarchical Optimistic Optimization (HOO) [2]. Theoretical analysis of our algorithm shows a sub-linear regret with respect to an omniscient genie. We validate our approach via simulations, showing that the algorithm quickly learns the optimal weight vectors when combined with opportunistic and/or utility-maximizing intra-slice schedulers.

Finally, we further extend our work to a multi-agent setting. We propose and evaluate a bandit framework to address resource allocation in coupled wireless systems. In particular we consider, multiple agents that choose from a set of resource allocation options towards achieving their own performance objectives/requirements, and where the performance observed at each agent is further coupled with the actions chosen by the other agents, e.g., through interference, channel leakage, etc. The challenge is to find the best collective action. Our focus is on systems that are "weakly-coupled" wherein the best arm of each agent is invariant to others' arm selection the majority of the time – this majority structure enables one to develop lightweight efficient algorithms. We develop a bandit algorithm based on the Track-and-Stop strategy [3], which shows a logarithmic regret with respect to a genie. Through simulation, we exhibit the potential use of our model and algorithm in several wireless application scenarios.

We summarize our main contributions as follows.

- Chapter 2: Meta-Scheduling for the Wireless Downlink through Learning with Bandit Feedback.

    1. We propose "meta-scheduling", a multi-armed bandit framework that aims at dynamically selecting the best scheduler amongst a candidate set.
    2. We introduce a meta-scheduler algorithm based on the Upper Confidence Bound (UCB) algorithm and cycle interruptions, in adaptive to queueing system settings.

3. We provide theoretical guarantees of the algorithm: the regret scales as $O(\log n)$ with respect to the optimal scheduler, and the expected number of packets dropped due to interruptions scales as $O(\log^2 n)$.

4. We simulate the algorithm under varying conditions such as different loads and performance metrics, showing the algorithm's adaptability in choosing different schedulers that maximize the reward.

   This work has been published in [4].

- Chapter 3: Online Learning for Hierarchical Scheduling to Support Network Slicing in Cellular Networks

  1. We consider hierarchical schedulers for network slicing parameterized by a slice-level allocation weight, and formulate the scheduler selection as a blackbox optimization problem under a bandit framework.

  2. We propose the Cycle-Based HOO with Clipping (CHOOC) algorithm, a modified Hierarchical Optimistic Optimization (HOO) algorithm adapting to queueing systems.

  3. We conduct a theoretical analysis on the cumulative regret of CHOOC, which is shown to scale in the same order as HOO.

  4. We empirically evaluate the algorithm in various wireless settings. Our results highlight the power of CHOOC to handle performance tradeoffs across slices.

     This work has been published in [5].

- Chapter 4: Online Learning for Multi-Agent Based Resource Allocation in Weakly Coupled Wireless Systems

    1. We develop a bandit framework to address the multi-agent resource allocation problem. In particular, we formally define weak-coupling, and discuss its properties and potential application usage.
    2. We propose a decentralized bandit algorithm designed for weakly coupled systems based on Track-and-Stop.
    3. We show that the algorithm has a low communication cost and is efficient with a logarithmic cumulative regret.
    4. We demonstrate two wireless systems with potential weak-coupling properties. In simulation, our algorithm successfully learns the best collective action with reasonable cost.

       This work has been published in [6].

1.4 Organization

In the subsequent chapters of this dissertation, we elaborate on our contributions. Chapter 2 covers our study on meta-scheduling for wireless downlink problems through an online learning framework. Chapter 3 discusses our work on bandit learning-based hierarchical network slicing in cellular networks. Chapter 4 elucidates our work on online learning multi-agent based resource allocation in weakly coupled wireless systems. Lastly, Chapter 5 concludes this report and outlines potential directions for future research.

Chapter 2

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

In this chapter, we study learning-assisted multi-user scheduling for the wireless downlink[1]. 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

[1]The content of this chapter is based on Song, Jianhan, et al. "Meta-Scheduling for the Wireless Downlink through Learning with Bandit Feedback." *IEEE/ACM Transactions on Networking*, 2021. The author, Jianhan Song, took on most of the responsibility for the problem formulation and the theoretical analysis, and performed all of the simulations.

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). This work is completed and has been published in [4].

2.1 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 of 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, the 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.[2] 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 that are fine-tuned to particular scenarios (e.g., learned through a reinforcement learning (RL) 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 chapter, 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, Log rule, Exp rule, Priority rule, RL schedulers, etc.). 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. The goal is to provide a learning framework that efficiently identifies the best among a pre-selected set of state-of-the-art policies for a given underlying scenario (characterized by traffic, channel states, user metrics, etc.)

In adapting the bandit framework to our queueing setting, we need to address two challenges: *(i)* Arbitrarily switching among schedulers over

[2]As an example, consider a set of users with different latency requirements. When the traffic load is low, it may be desirable to give scheduling priority to users with stricter packet deadlines without degrading other users' performance; when the load increases, however, a fairer scheduler may be preferred.

time can lead to queue instability, even if each of the schedulers is stable. Indeed, one can show that switching between two MaxWeight 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[3] cycles that have exceedingly long durations. These decisions have to be properly designed such that cycles due to unstable schedulers (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 in designing a meta-scheduler that achieves a low regret with respect to a genie algorithm (baseline that always plays the best/highest-reward scheduler for a particular

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

scenario).

Finally, it is worth noting that there are possibly two parallel methodologies to systemically address the scheduling problem for various environments/applications. The first one is to formulate scheduling as an MDP problem for any given performance metric and utilize a data-driven method to train and learn the optimal strategy for the given model, i.e., the reinforcement learning approach. Although this is a powerful method to generate *new* schedulers, the training process typically requires considerable exploration of different regimes and a large computational effort and is often conducted offline (i.e., before deployment). Therefore, it is essential to properly model the performance metric and traffic environment where the scheduler will be deployed, which further raises difficulty and safety concerns (due to the mismatch between the training and deployment environments). By contrast, in this chapter, we tackle the problem from the second perspective. We will answer the following question: Given a set of *existing* policies (which could include one or more pre-trained RL schedulers optimized for specific settings), how one can determine the best scheduler *among these candidates* without assuming prior knowledge on the current environment/performance metric. Our bandit framework learns in an online manner with lightweight computation and relatively low convergence time needed, and can in principle keep re-optimizing to changing scenarios (through re-running the meta-scheduling algorithm when the environment significantly changes). When the optimal policy is unclear, which is common in many real applications given all the uncertainties, our approach transfers the burden of choosing the best-suited *environment-specific state-of-the-art* scheduler to a learning algorithm.

2.1.1 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[4] and correspondingly $O(\log^2 \tau)$ where τ is the time-slot index. Further, the expected

[4]The regret scaling is slightly weaker under weaker assumptions on the cycle tail distributions, please see Section 2.4 for details.

number of packets dropped due to interruptions also scales as $O(\log^2 \tau)$. When packet drop is forbidden, an alternative mechanism to clear up the queueing system is introduced at a slight expense of the total regret.

- *Simulation Results:* We simulate the meta-scheduler in a variety of wireless settings. These include different rewards for performance metrics such as mean delay, delivering packets on time, and penalizing bursty service, and various schedulers including the MaxWeight, Exp, Log, max-rate and round-robin and opportunistic priority, and different load conditions. Our simulations show that as conditions vary (e.g. different loads, or different performance metrics), the meta-scheduler adapts to choose a different scheduler that maximizes the reward for each scenario.

2.1.2 Related Work

Wireless Scheduling. The design of multi-user wireless schedulers has received substantial attention, see e.g., [7] 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., [8, 9, 10, 11, 12]. 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 [13, 14], Exp rule [15] and Log rule [1], 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 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 [16] and wireless scheduling in various settings [17, 18, 19, 20]. RL algorithms provide a general approach to determining good schedulers for specific scenarios and possibly, but substantially more challenging, ones that are good for a range of scenarios in terms of user traffic, service capacity, and or performance objectives. Despite showing great potential in several applications, providing theoretical performance guarantees for RL-based schedulers remains an open question. Limited success has been achieved in some simple settings (in terms of traffic model or user metrics) – see e.g., [21, 22, 23]. However, advanced RL methods, especially those involving neural networks that have attracted the most attention in practice, typically lack rigorous performance guarantees, and thus it is unclear whether they are safe to deploy. The goal of this chapter differs from the common focus of designing practically or theoretically good RL schedulers in current RL-networking literature. Instead, our framework aims at better utilizing the knowledge of existing schedulers (including RL schedulers) to address various scheduling scenarios (in particular those with complicated performance metrics or traffic models), while ensuring queueing stability.

In the literature, some authors proposed scheduling policies that utilize online *statistical learning*, e.g., [24, 25], which involve learning system statistics online to improve the performance of certain schedulers. We note that the above methodology is different from the bandit-based online learning framework we propose in this chapter. Our framework is adaptive to statistics, but by learning the best scheduler among a predefined candidate set of policies for a specific scenario, instead of refining specific policies.

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 [26] and [27] 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 a 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 [28] and has been followed in many directions by [29, 30, 31].

A recent study on budgeted bandits in [32] 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 [32] develop a variant of the

Upper Confidence Bound (UCB) algorithm [33] 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 [32], our goal is to eventually avoid any interruptions, thus, we do not treat interruptions as the 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 [34], spectrum access [35], decentralized wireless computing[36, 37] and most recently, cellular scheduling [38].

2.1.3 Notation

Throughout this paper, we use characters in bold font to denote vectors and normal font to denote scalars. Random variables are indicated by capital letters unless stated otherwise. We adopt the following technical abbreviations:

"*w.h.p.*" for "with high probability", "*a.s.*" for "almost surely" and "*i.i.d.*" for "independently and identically distributed". Finally, we use $\mathbb{1}$ for the $\{0,1\}$ indicator function.

2.2 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.

2.2.1 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 $(\boldsymbol{A}[t])_{t\geq 0}$ 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. $(\boldsymbol{S}[t])_{t\geq 0}$ 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 [7]). 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 $\mathcal{A}^s(\boldsymbol{\lambda}) \neq \phi$.

2.2.2 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.[5] 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 [32], 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.* .

Remark 1. *To be precise, in order to make $C^{(k)}(n)$ i.i.d. over cycles, in addition to i.i.d. arrivals and channels assumed in our traffic and service model, it is necessary to assume the scheduling policy k only uses information associated with its current cycles. For example, a Markovian policy which chooses a service vector at time t only based on the current system states (e.g., $\boldsymbol{S}[t]$ and $\boldsymbol{Q}[t]$), such as MaxWeight, Log rule, etc., naturally satisfies this requirement. For a policy that keeps internal states which utilize information from the past, e.g., a proportionally fair scheduler using an exponential moving average of past throughput, we need to additionally reset internal states when a cycle begins.*

[5]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.

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* $(\boldsymbol{U}^{(k)}(n))_{n\geq 1}$ *for all* $k \in \mathcal{A}$ *are i.i.d. over* n*, and such that for* $l = 1, \ldots, C^{(k)}(n)$,

$$0 \leq \sum_{i=1}^{l} U^{(k)}(n,i) \leq \bar{r}l, \qquad \textit{almost surely} \tag{2.1}$$

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

Remark 2. *This assumption holds, for instance, if each packet is associated a bounded reward (e.g., over* $[0,1]$*) upon reception, and the cumulative reward over a time period is thus bounded by the maximal number of packets transmitted within that period, i.e.,* $\bar{r} = \bar{a}u$*. In practice, for example, the cycle reward evaluated by a latency-sensitive user can be the number of packets that arrive on time within a cycle. It should be noted that the manner in which rewards are calculated/defined is not necessarily known by the base station in our model, which allows for more flexible user-customized reward schemes.*

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:

$$0 \leq U^{(k)}(n) \leq \bar{r}C^{(k)}(n), \qquad \textit{almost surely.} \tag{2.2}$$

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 $n \geq 1$,*

$$\mathbb{P}(|C^{(k)}(n) - \mathbb{E}[C^{(k)}(n)]| \geq \varepsilon) \leq \begin{cases} 2e^{-\varepsilon^2/(2\nu_k^2)} & 0 < \varepsilon \leq \frac{\nu_k^2}{\alpha_k}, \\ 2e^{-\varepsilon/(2\alpha_k)} & \varepsilon > \frac{\nu_k^2}{\alpha_k}. \end{cases} \tag{2.3}$$

Remark 3. *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). When the system has bounded arrival and channel distributions, and the policies considered are Markovian, this assumption holds true following an argument of [39]. 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 Assumption 1, i.e., (2.2), $U^{(k)}(n)$ is also sub-exponential (with possibly larger parameters). Without loss of generality, we will assume both $C^{(k)}(n)$ and $U^{(k)}(n)$ are (ν_k^2, α_k)-sub-exponential, assuming the rewards are properly normalized.*

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.

$$r^{(k)} = \frac{\mathbb{E}[U^{(k)}(1)]}{\mathbb{E}[C^{(k)}(1)]} \quad \forall k \in \mathcal{A}^s(\boldsymbol{\lambda}). \tag{2.4}$$

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

2.2.3 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 chapter, 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 [32], 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.

Remark 4. *To allow for simpler analysis, we require that all packets to be discarded when a cycle is interrupted. As is shown later, a good meta-scheduler should be designed such that this event occurs rarely. If such packet drops are unacceptable, instead of interrupting and dropping, we can switch to a default policy that is guaranteed to be stable (e.g., MaxWeight) when an interruption* *is triggered, and a cycle can be restarted when the system returns to the idle state. Later we will show that the loss of rewards induced by this extra process grows logarithmically in time.*

There are several advantages in adopting those two 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 cycle lengths and rewards, i.e., the collection of processes $\{((C^{(k)}(n), \boldsymbol{U}^{(k)}(n)))_{n\geq 1} : k \in \mathcal{A}\}$, 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

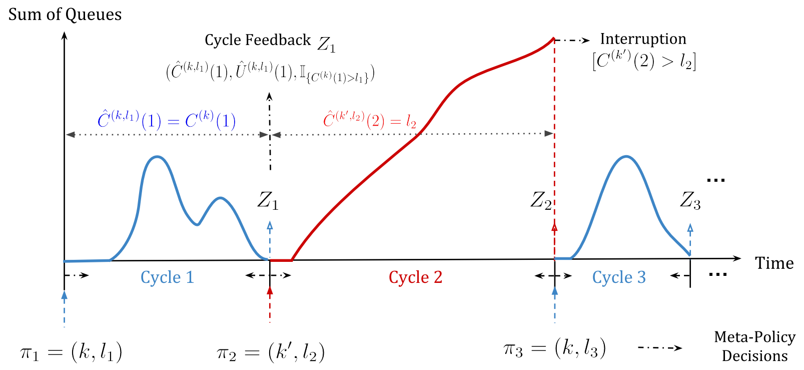


Figure 2.1: Illustration of a meta-policy selecting arms from $\mathcal{A} = \{k, k'\}$. At the start of the process, the meta-scheduler makes decision $\pi_1 = (k, l_1)$, and receives feedback Z_1 after the system experiences a full cycle. Then the meta-scheduler decides $\pi_2 = (k', l_2)$, but has 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 $\pi_3 = (k, l_3)$.

$0 \leq \hat{U}^{(k,l)}(n) \leq \bar{r}\hat{C}^{(k,l)}(n)$ 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 2.1. Note that the reward for each single time slot is not required in the feedback. This suggests that if performance is evaluated at the 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 chapter is formally defined as follows. This is analogous to a similar notion in [32].

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 regret* over a given time horizon. We will define the regret in the next section.

2.2.4 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 [32]. 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.*

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

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.)*

$$\begin{aligned}\mathrm{Rew}_\pi[\tau] =& \sum_{i=1}^{\tilde{N}} \sum_{(k,l)} \mathbb{1}_{\{\pi_i=(k,l)\}} \hat{U}^{(k,l)}(i) \\ &+ \sum_{(k,l)} \mathbb{1}_{\{\pi_{\tilde{N}+1}=(k,l)\}} \sum_{j=1}^{\tau - S^\pi_{\tilde{N}}} U^{(k)}(\tilde{N}+1, j).\end{aligned} \tag{2.5}$$

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 chapter, 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.

$$\pi_n^{\mathrm{opt}} = (k^*, \infty), \quad \forall n \geq 1 \tag{2.6}$$

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*

$$\mathrm{Reg}_\pi[\tau] = \mathbb{E}[\mathrm{Rew}_{\pi^{\mathrm{opt}}}[\tau] - \mathrm{Rew}_\pi[\tau]]. \tag{2.7}$$

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., $r^{(*)} := r^{(k^*)}$).

2.3 UCB Meta-Scheduler with Interruption

To guarantee negligible throughput loss and a sub-linear regret as discussed in Section 2.2, 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.

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 $A_n = k$,

$$(C^{(k)}_{T_n^{(k)}}, U^{(k)}_{T_n^{(k)}}) = (C^{(k)}(n), U^{(k)}(n)).$$

Similar to a 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 $s \geq 1$,

$$\hat{R}_s^{(k)} = \frac{\sum\limits_{i=1}^{s} \hat{U}_i^{(k,F_i^{(k)})}}{\sum\limits_{i=1}^{s} \hat{C}_i^{(k,F_i^{(k)})}}, \tag{2.8}$$

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.

As an overview, we present a simplified version of our meta-scheduler in Algorithm 1. The mathematical design of key variables will be discussed in a 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 otherwise a fixed threshold will always result in linear throughput loss). Hence, we define a threshold function f_s as in line 2. For *any* arm k, we will use f_s as the

Algorithm 1 UCB based Meta-Scheduler with Interruption

```
1: **Input:** Set of scheduling policies $\mathcal{A}$.
2: **Threshold Function**: $f_s := \beta + \kappa \log s$.
                                              ▷ $\beta$ and $\kappa$ to be defined
3: **Initialization**: Run every arm $k \in \mathcal{A}$ once with interruption threshold $\beta$,
   then initialize empirical rate $\hat{R}_1^{(k)}$.
4: **for** $n = |\mathcal{A}|+1, |\mathcal{A}|+2, \cdots$ **do**
5:     [*before cycle n*]
6:     **for all** $k \in \mathcal{A}$ **do**
7:         Compute *Exploration Bonus* $\Delta_n^{(k)}$.
8:         Compute *Stability Indicator* $I_n^{(k)}$.
                                              ▷ $I_n^{(k)}$: a Boolean variable
9:     **Arm decision:**
           $$A_n \in \operatorname*{argmax}_{k\in\mathcal{A}} \; I_n^{(k)} \times (\hat{R}^{(k)}(n-1) + \Delta_n^{(k)}).$$
10:    **Cycle interruption decision:**
           $$L_n = f_{T_n^{(A_n)}}.$$
11:    [*after cycle n*]
12:    Observe cycle feedback $\hat{U}^{(A_n,L_n)}(n), \hat{C}^{(A_n,L_n)}(n)$, then update $\hat{R}^{(A_n)}(n)$.
```

interruption threshold for its s-th sample. With this design, the expected number of interruptions imposed on the optimal arm can be bounded by a *constant* if β and κ are large enough.

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 the "score" (i.e., *upper confidence bound*) for each of its arms. The score is the sum of its empirical reward rate and an *exploration bonus* (line 7).

The exploration bonus is used to compensate for possibly under-performing empirical rate estimates in order to ensure adequate exploration before finding committing to optimal arm. We will show that *w.h.p.*, $\hat{R}^{(*)}(n-1)+\Delta_n^{(*)} > r^{(*)}$ 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* (line 8) to eliminate unstable arms by utilizing accumulated interruptions as a signal indicating whether an arm frequently induces long cycles. We keep track of the number of times each arm is interrupted, and $I_n^{(k)}$ is set 0 only if the total number of interruptions exceeds a limit (which is a function of n).

After computing the exploration bonus and the stability indicator, the meta-scheduler will pick the arm with the best score $B_n^{(k)} := \hat{R}^{(k)}(n-1)+\Delta_n^{(k)}$ and a positive value of $I_n^{(k)}$ (line 9), and the threshold of that cycle is determined by the threshold function (line 10). 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 are then used to determine the next decision.

2.4 Main Results and Discussion

Compared with previous works in the literature, our UCB-type algorithm tackles more challenging assumptions on cycle variables (sub-exponential instead of sub-Gaussian), and further the cycle lengths can be unbounded with positive probability, due to unstable schedulers. Thus, it requires several novel design choices such as dynamic interruption thresholds, a stability indicator, and a suitably modified exploration bonus. In this section, we will discuss these key design choices that differ from the classical UCB, followed by the result of the regret analysis.

2.4.1 Analysis of Key Design Choices

2.4.1.1 Hyper-Parameters

Before discussing the details of the meta-scheduler, let us first introduce several parameters used in our algorithm in the following assumption.

Assumption 3. *We assume parameters $\mu_{\min}$, $\mu_{\max}$, $r_{\max}$ and (ν^2, α) are given a priori such that there exists a subset of arms $\mathcal{A}_0$ satisfying*

$$\{k^*\} \subseteq \mathcal{A}_0 \subseteq \mathcal{A}^s(\boldsymbol{\lambda})$$

and for all $k \in \mathcal{A}_0$,

(1) $\mu_{\min} \leq \mathbb{E}[C^{(k)}(1)] \leq \mu_{\max}$.

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

(3) $C^{(k)}(1), U^{(k)}(1)$ 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 which satisfy these conditions with respect to those hyper-parameters. We assume $k^* \in \mathcal{A}_0$ (as the weakest notion) to achieve sub-linear regret.

Remark 5. *For technical reasons, we also require $\hat{U}^{(k,l)}(1)$ to be sub-exponential under the same parameters*[6] *(ν^2, α) as those of $U^{(k)}(1)$ when l is sufficiently large. 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), which will be explained in Section 2.4.1.4.*

2.4.1.2 Threshold Function

Recall that the interruption threshold for arm k's s-th sample is given by

$$f_s := \beta + \kappa \log s. \tag{2.9}$$

We require β and κ satisfy that

$$\beta > \mu_{\max} + \nu^2/\alpha, \quad \kappa/\alpha \geq 4.$$

Under these conditions, and by Assumption 2, it is easy to verify that the interruption probability for the best arm $\mathbb{P}(C_s^{(*)} > f_s^{(*)}) \leq 1/s^2$. This implies

[6]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 $\hat{U}^{(k,l)}(1)$.

that, under our threshold function, the expected number of interruptions of the best arm is bounded by a constant (since $\sum_{s=1}^{\infty} 1/s^2 = \pi^2/6$), and thus packet drops induced by the "wrong" interruptions do not grow faster than $O(\log n)$ over n cycles.

2.4.1.3 Stability Indicator

The stability indicator is defined as follows:

$$I_n^{(k)} = \begin{cases} 1 & \text{if} \sum_{i=1}^{T_{n-1}^{(k)}} \mathbb{1}\{C_i^{(k)} > f_i\} < \frac{\pi^2}{6} + \sqrt{2T_{n-1}^{(k)} \log n}, \\ 0 & \text{otherwise.} \end{cases} \tag{2.10}$$

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 as previously discussed, but grows linearly for the unstable arms (since they are frequently clipped).

This motivates us to distinguish unstable arms by comparing the total number of interruptions of each arm to $\pi^2/6$ plus a concentration bound. By McDiarmid's inequality, the optimal arm's indicator $I_n^{(*)}$ is equal to 1 *w.h.p.* for any $n \geq 1$. In contrast, for any unstable k, the value of $\sum_{i=1}^{s} \mathbb{1}\{C_i^{(k)} > f_i\}$ will eventually exceed the limit.

2.4.1.4 Upper Confidence Bound

As discussed in the last section, an exploration bonus term is designed to compensate the empirical reward rates of arms for arm selection. Specifically, at least for the best arm, the empirical rate plus its exploration bonus should be above the true rate *w.h.p.*, which ensures the best arm gets sufficiently explored.

The exploration bonus is formally defined as follows:

$$\Delta_n^{(k)} = \Delta(\epsilon_n^{(k)}, \epsilon_n'^{(k)}) \tag{2.11}$$

where

$$\Delta(\epsilon, \epsilon') := \frac{\epsilon(1 + r_{\max}) + \epsilon'}{\mu_{\min} + \epsilon}, \tag{2.12}$$

and

$$\epsilon_n^{(k)} = \begin{cases} \sqrt{\frac{6\nu^2 \log n}{T_{n-1}^{(k)}}} & \sqrt{\frac{6\nu^2 \log n}{T_{n-1}^{(k)}}} \le \frac{\nu^2}{\alpha}, \\ \frac{6\alpha \log n}{T_{n-1}^{(k)}} & \text{otherwise,} \end{cases} \tag{2.13}$$

$$\epsilon_n'^{(k)} = \frac{1}{T_{n-1}^{(k)}} \sum_{i=1}^{T_{n-1}^{(k)}} \frac{r_{\max}}{i^{\kappa/2\alpha}} e^{-\beta/4\alpha}(\beta + 2\alpha + \kappa \log i + 1). \tag{2.14}$$

To see why this term works, first let us suppose every arm is stable and there is no interruption, i.e., $f_s = \infty$. As in [32], we can introduce the following term,

$$\bar{\Delta}(\epsilon) := \frac{\epsilon(1 + r_{\max})}{\mu_{\min} + \epsilon} \ge r^{(*)} - \frac{\mathbb{E}[U_1^{(*)}] - \epsilon}{\mathbb{E}[C_1^{(*)}] + \epsilon}. \tag{2.15}$$

By simple observation, we have that

$$\begin{aligned} &\{\hat{R}_s^{(*)} + \bar{\Delta}(\epsilon) \le r^{(*)}\} \subset \\ &\{\frac{1}{s}\sum_{i=1}^{s} \hat{C}_i^{(*,f_i)} > \mathbb{E}[C_1^{(*)}] + \epsilon\} \bigcup \{\frac{1}{s}\sum_{i=1}^{s} \hat{U}_i^{(*,f_i)} \le \mathbb{E}[U_1^{(*)}] - \epsilon\}. \end{aligned} \tag{2.16}$$

These two events at the bottom allow us to use the concentration properties stated in Assumption 2 to bound the probability of the original event. Following a trick in [40], let $\epsilon = \epsilon_{n,s}$ where

$$\epsilon_{n,s} = \begin{cases} \sqrt{\frac{6\nu^2 \log n}{s}} & \sqrt{\frac{6\nu^2 \log n}{s}} \leq \frac{\nu^2}{\alpha}, \\ \frac{6\alpha \log n}{s} & \text{otherwise.} \end{cases} \tag{2.17}$$

We can then show that both of those two events happen with probability $1/n^3$ for all $s \leq n$. Hence, by taking a union bound on all possible $T_{n-1}^{(*)} \leq n$, we have that *w.h.p.* the exploration bonus $\bar{\Delta}(\epsilon_{n,T_{n-1}^{(*)}})$ suffice to compensate the best arm's empirical rate $\hat{R}^{(*)}(n)$ such that $\hat{R}^{(*)}(n) + \bar{\Delta}(\epsilon_{n,T_{n-1}^{(*)}}) > r^{(*)}$, which is as desired.

When we consider the threshold f_s as defined in (2.9), however, $\bar{\Delta}$ is not sufficient to compensate for $\hat{R}_s^{(*)}$ to exceed $r^{(*)}$. This is due to the bias of estimating $\mathbb{E}[U_1^{(*)}]$ by the average *truncated* reward $(1/s)\sum_{i=1}^{s} \hat{U}_i^{(*,f_i)}$, and the second bottom event in (2.16) is no longer with a negligible probability. This motivates us to adjust ϵ to account for the additional bias.

When $\beta > (1+\gamma)(\mu_{\max} + \nu^2/\alpha)$, by simple algebra, we have that the bias is bounded as follows (see the detailed derivation in Appendix A.1).

$$\begin{aligned} &\mathbb{E}[U_1^{(*)}] - \frac{1}{s}\sum_{i=1}^{s} \mathbb{E}[\hat{U}_i^{(*,f_i)}] \\ &\leq \epsilon'_{n,s} := \frac{1}{s}\sum_{i=1}^{s} \frac{r_{\max}}{i^{\kappa/2\alpha}}(\beta + 2\alpha + \kappa \log i + 1) e^{-\beta\gamma/2(1+\gamma)\alpha}. \end{aligned} \tag{2.18}$$

For simplicity, we let $\gamma = 1$ in our algorithm. Note that $\epsilon'_{n,s} \sim O(\log s/s)$ when $\kappa/\alpha \geq 4$.

Now we can define an updated exploration bonus $\Delta(\epsilon, \epsilon')$ as in (2.12). Note that

$$\Delta(\epsilon, \epsilon') := \frac{\epsilon(1 + r_{\max}) + \epsilon'}{\mu_{\min} + \epsilon} \geq r^{(*)} - \frac{(\mathbb{E}[U_1^{(*)}] - \epsilon') - \epsilon}{\mathbb{E}[C_1^{(*)}] + \epsilon}. \tag{2.19}$$

Following the same logic as in (2.16), we can conclude[7] that *w.h.p.*, $\hat{R}^{(*)}(n) + \Delta(\epsilon_{n,T_{n-1}^{(k)}}, \epsilon'_{n,T_{n-1}^{(k)}}) > r^{(*)}$.

As a sanity check, if $T_n^{(k)}$ grows faster than $O(\log n)$ for any stable arm k, then its exploration bonus will converge to 0. Thus, the UCB-compensated empirical rate of a stable suboptimal arm will eventually fall short of $r^{(*)}$ after sufficient explorations.

2.4.2 Main Results

In this part we present the main theorem, which justifies that the meta-scheduler given in Algorithm 1 satisfies our requirements regarding negligible packet loss and a sub-linear regret. Before that, let us first introduce a key lemma, stating that the number of sub-optimal decisions under our algorithm grows quasi-logarithmically.

For simplicity, let $d^{(k)} := r^{(*)} - r^{(k)}$ denote the gap of reward rates between arm k and the optimal arm. We will use symbols $\wedge$ and $\vee$ as the shorthand notations for min and max functions respectively.

Lemma 1. *A meta-scheduler implementing Algorithm 1 satisfies the following regarding* $\mathbb{E}[T_n^{(k)}]$.

[7]The proof requires the technical assumption in item (3) of Assumption 3.

(1) For all unstable arms $k \in \mathcal{A}^u(\boldsymbol{\lambda})$,

$$\mathbb{E}[T_n^{(k)}] \leq \lceil \mathsf{K}_1^{(k)} \log n \rceil + \frac{\pi^2}{6} + 1$$

where

$$\mathsf{K}_1^{(k)} = \frac{18}{\mathbb{P}(C_1^{(k)} = \infty)^2}.$$

(2) For stable suboptimal arms that lie in set $\mathcal{A}_0 \setminus \{k^*\}$, we have that

$$\mathbb{E}[T_n^{(k)}] \leq \lceil \mathsf{K}_2^{(k)} \log n \vee \mathsf{M}_1^{(k)} \vee \mathsf{M}_2^{(k)} \rceil + \pi^2 + 1$$

where

$$\mathsf{K}_2^{(k)} = \frac{24(1+r_{\max})^2 \nu^2 (\mu_{\min}+1)^2}{(d^{(k)})^2 \mu_{\min}^2} \vee \frac{12(1+r_{\max})\alpha(\mu_{\min}+1)}{d^{(k)} \mu_{\min}}$$

and $\mathsf{M}_1^{(k)}, \mathsf{M}_2^{(k)}$ are (smallest possible) constants such that

$$\begin{aligned} \mathsf{M}_1^{(k)} &\geq \frac{4}{d^{(k)}} \bar{r} \, (\mathbb{E}[C_1^{(k)}] + 6\alpha_k \log \mathsf{M}_1^{(k)}) (\frac{\pi^2}{6} + \sqrt{\mathsf{M}_1^{(k)} \log \mathsf{M}_1^{(k)}}), \\ \mathsf{M}_2^{(k)} &\geq \frac{4}{d^{(k)}} r_{\max} \cdot \frac{\pi^2}{6} e^{-\beta/4\alpha} (\beta + 2\alpha + \kappa \log \mathsf{M}_2^{(k)} + 1). \end{aligned}$$

(3) For stable suboptimal arms that lie in $\mathcal{A}^s(\boldsymbol{\lambda}) \setminus \mathcal{A}_0$, we have that for any $\delta > 0$ and $\chi > 1$,

$$\mathbb{E}[T_n^{(k)}] \leq \lceil \mathsf{K}_3^{(k)} \log n \vee \mathsf{J}_1^{(k)} \log^{1+\delta} n \vee \chi \vee \mathsf{M}_2^{(k)} \rceil + \pi^2 + 1$$

where

$$\mathsf{K}_3^{(k)} = \frac{24(1+r_{\max})^2 \nu_k^2 (\mu_{\min}+1)^2}{(d^{(k)})^2 \mu_{\min}^2} \vee \frac{12(1+r_{\max})\alpha_k(\mu_{\min}+1)}{d^{(k)} \mu_{\min}}$$

and $\mathsf{J}_1^{(k)}$ *is a constant (up to* δ *and* χ*) such that*

$$\mathsf{J}_1^{(k)} \geq \frac{4}{d^{(k)}} \bar{r} \frac{\left(\mathbb{E}[C_1^{(k)}] + 6\alpha_k \log(\mathsf{J}_1^{(k)} \log^{1+\delta}\chi)\right)\left(\frac{\pi^2}{6} + \sqrt{2\mathsf{J}_1^{(k)}\log^{2+\delta}\chi} + 1\right)}{\log^{1+\delta}\chi}.$$

Note that $\mathsf{M}_1^{(k)}, \mathsf{J}_1^{(k)}$ *are roughly* $O((1/d^{(k)})^2)$ *and* $\mathsf{M}_2^{(k)}$ *is roughly* $O(1/d^{(k)})$. *To summarize,*

$$\mathbb{E}[T_n^{(k)}] = \begin{cases} O(\log n) & \forall k \in \mathcal{A}^u(\boldsymbol{\lambda}), \\ O(\log n) & \forall k \in \mathcal{A}_0 \setminus \{k^*\}, \\ O(\log^{1+\delta} n)\ \forall \delta > 0 & \forall k \in \mathcal{A}^s(\boldsymbol{\lambda}) \setminus \mathcal{A}_0. \end{cases}$$

Proof of Lemma 1. Here we will present a proof sketch. The complete proof can be found in Appendix A.2.

The proof consists of two parts. First, we prove the case when arm k is unstable. Observe that for any $k \in \mathcal{A}^u(\boldsymbol{\lambda})$, $p_k := \mathbb{P}(C_1^{(k)} = \infty) > 0$. Thus, *w.h.p.*, the value $\sum_{i=1}^{T_{n-1}^{(k)}} \mathbb{1}_{\{C_i^{(k)} > f_i\}}$ grows no slower than $p_k T_{n-1}^{(k)}$ minus a concentration bound $\sqrt{2T_{n-1}^{(k)} \log n}$ (by Bernstein's inequality). Therefore, if the n-th arm decision $A_n = k \in \mathcal{A}^u(\boldsymbol{\lambda})$, by the stability indicator, we have that $p_k T_{n-1}^{(k)} - \sqrt{2T_{n-1}^{(k)} \log n} < \pi^2/6 + \sqrt{2T_{n-1}^{(k)} \log n}$. This implies $\mathbb{E}[T_n^{(k)}] = O(\log n)$.

Next, we study the case for any stable suboptimal arm. If $A_n = k \in \mathcal{A}^s(\boldsymbol{\lambda}) \setminus \{k^*\}$, either of the following two events will happen: (a) the stability indicator of the best arm is 0; or (b) the UCB score (empirical rate plus UCB) of arm k is larger than the best arm's score. As discussed in 2.4.1.3, the first event has a negligible chance to happen. For the second event, we already guarantee that *w.h.p.*, $\hat{R}^{(*)}(n) + \Delta(\epsilon_n^{(*)}, \epsilon_n'^{(*)}) > r^{(*)}$ by the design of the exploration bonus in 2.4.1.4. Then to show the second event cannot occur either, it suffices to show that $\hat{R}^{(k)}(n) + \Delta(\epsilon_n^{(k)}, \epsilon_n'^{(k)}) < r^{(*)}$. Observe that $\hat{R}^{(k)}(n) \approx r^{(k)} < r^{(*)}$ and that $\Delta(\epsilon_n^{(k)}, \epsilon_n'^{(k)})$ can be arbitrarily small when $T_n^{(k)} > \mathsf{C} \log n$ for a sufficiently large C. This suggests $T_n^{(k)}$ cannot grow faster than $O(\log n)$. When $k \in \mathcal{A}^s(\boldsymbol{\lambda}) \setminus \mathcal{A}_0$, an additional $1+\delta$ is needed in the exponent of $\log n$. This is caused by a technical issue[8] due to the bias of (possibly) overestimating $r^{(k)}$ by $\hat{R}^{(k)}(n)$. $\square$

Now we present the main theorem as follows.

Theorem 1. *Let $f_\tau = (\kappa + \alpha \log \tau)$ and*

$$\begin{aligned}
\mathsf{g}_1^{(k)}(\tau) &= \lceil \mathsf{K}_1^{(k)} \log \tau \rceil + 1, \\
\mathsf{g}_2^{(k)}(\tau) &= \lceil \mathsf{K}_2^{(k)} \log \tau \vee \mathsf{M}_1^{(k)} \vee \mathsf{M}_2^{(k)} \rceil + 1, \\
\mathsf{g}_3^{(k)}(\tau) &= \lceil \mathsf{K}_3^{(k)} \log \tau \vee \mathsf{J}_1^{(k)} \log^{1+\delta} \tau \vee \chi \vee \mathsf{M}_2^{(k)} \rceil + 1,
\end{aligned}$$

where $\mathsf{K}_1^{(k)}$, $\mathsf{K}_2^{(k)}, \mathsf{K}_3^{(k)}, \mathsf{J}_1^{(k)}, \mathsf{M}_1^{(k)}$ and $\mathsf{M}_2^{(k)}$ are defined as in Lemma 1. The meta-policy π induced by Algorithm 1 has the following properties.

(1) For the expected number of packets discarded over the time horizon

[8] If the hyper-parameters, in particular β and α, do not suffice the conditions in Assumption 3 for a stable suboptimal arm k, the number of interruptions of k is no longer well bounded under our threshold function. This leads to an additional bias between $r^{(k)}$ and $\hat{R}^{(k)}(n)$.

τ, denoted by $\mathbb{E}[D_\pi[\tau]]$, we have for any $\delta > 0$ and $\chi > 1$,

$$\mathbb{E}[D_\pi[\tau]] \leq \bar{a}uf_\tau \Bigg(\sum_{k\in\mathcal{A}_0} \frac{\pi^2}{6} + \sum_{k\in\mathcal{A}^u(\boldsymbol{\lambda})} \Big(((\frac{\pi^2}{6} + \sqrt{2\mathsf{g}_1^{(k)}(\tau)\log\tau} + 2) \wedge \mathsf{g}_1^{(k)}(\tau)) + \frac{\pi^2}{6} \Big) + \sum_{k\in\mathcal{A}^s(\boldsymbol{\lambda})\setminus\mathcal{A}_0} \Big(((\frac{\pi^2}{6} + \sqrt{2\mathsf{g}_3^{(k)}(\tau)\log\tau} + 2) \wedge \mathsf{g}_3^{(k)}(\tau)) + \pi^2 \Big) \Bigg).$$

(2) For the regret $\mathrm{Reg}_\pi[\tau]$, we have for any $\delta > 0$ and $\chi > 1$,

$$\mathrm{Reg}_\pi[\tau] \leq \sum_{k\in\mathcal{A}^u(\boldsymbol{\lambda})} (r^{(*)} - \tilde{r}^{(k)})(\mathsf{g}_1^{(k)}(\tau) + \frac{\pi^2}{6})f_\tau + \sum_{k\in\mathcal{A}^s(\boldsymbol{\lambda})\setminus\mathcal{A}_0} (r^{(*)} - \tilde{r}^{(k)})(\mathsf{g}_3^{(k)}(\tau) + \pi^2)f_\tau + \sum_{k\in\mathcal{A}_0} (r^{(*)} - r^{(k)})(\mathsf{g}_2^{(k)}(\tau) + \pi^2)\mu_{\max} + r^{(*)}f_\tau + r^{(*)}\frac{\mathbb{E}[(C^{(*)}(1))^2]}{\mathbb{E}[C^{(*)}(1)]} + \sum_{k\in\mathcal{A}_0} \mathsf{h}(\tau),$$

where

$$\tilde{r}^{(k)} = \inf_{l\geq\beta} \frac{\mathbb{E}[\hat{U}^{(k,l)}(1)]}{\mathbb{E}[\hat{C}^{(k,l)}(1)]}, \quad \mathsf{h}(\tau) = r_{\max}\frac{\pi^2}{6}e^{-\beta/4\alpha}(f_\tau + 2\alpha + 1).$$

This theorem implies that if $\mathcal{A}_0 = \mathcal{A}^s(\boldsymbol{\lambda})$, i.e, all the stable arms satisfy the conditions in Assumption 3 with respect to the hyper-parameters used in the algorithm, then

$$\mathbb{E}[D_\pi[\tau]] = O(\log^2(\tau)), \quad \mathrm{Reg}_\pi[\tau] = O(\log^2\tau).$$

Otherwise,

$$\mathbb{E}[D_\pi[\tau]] = O(\log^{2+\delta}(\tau)), \ \ \mathrm{Reg}_\pi[\tau] = O(\log^{2+\delta}(\tau)), \forall\delta > 0.$$

Proof of Theorem 1. The complete proof can be found in Appendix A.3.

Claim (1) of the theorem: The expected number of interruptions on arms in $\mathcal{A}_0$ is bounded by $\pi^2/6$ due to the threshold design. For any arm not in $\mathcal{A}_0$, the expected number of interruptions is bounded by the number of arm selections $\mathbb{E}[T_\tau^{(k)}]$ (note that $N_\pi[\tau] \leq \tau$), but a nicer bound can be given since some of the cycles might not be interrupted. For each interrupted cycle, the number of packets dropped is bounded by $\bar{a}uf_\tau = O(\log \tau)$, where f_τ is a (coarse) bound on the longest possible cycle before time τ. This concludes the claim.

Claim (2) of the theorem: For this part we follow a similar approach as in the budgeted bandit literature [30, 32]. We first bound the number of sub-optimal actions that have been taken in order to bound regret. Intuitively, the expected cumulative reward for the optimal meta-policy is roughly $r^{(*)}\tau$, and the regret of π is due to the reward loss during the period of suboptimal decisions, where the loss in reward rate is either $r^{(*)} - r^{(k)}$ (for $k \in \mathcal{A}_0 \setminus \{k^*\}$) or bounded by $r^{(*)} - \tilde{r}^{(k)}$ (for $k \notin \mathcal{A}_0$)[9]. We can show that

$$\begin{aligned}\mathrm{Reg}[\tau] \leq &\sum_{k \in \mathcal{A}_0} \mathbb{E}[T_\tau^{(k)}]\mu_{\max}(r^{(*)} - r^{(k)}) \\ &\quad + \sum_{k \notin \mathcal{A}_0} \mathbb{E}[T_\tau^{(k)}] f_\tau (r^{(*)} - \tilde{r}^{(k)}) + O(\log \tau).\end{aligned}$$

The value of $\mathbb{E}[T_\tau^{(k)}]$ is then bounded using Lemma 1. □

[9]When $k \in \mathcal{A}_0$, the expected reward rate of arm k's period (which may include interrupted cycles) is roughly $r^{(k)}$, since only few cycles are interrupted; otherwise, we use a weaker bound $\tilde{r}^{(k)}$ instead.

As described in Remark 4, if packet dropping is unacceptable in the system, one can instead switch to a queue-stablizing (throughput-optimal) policy like MaxWeight to clear out the queues. This process introduces a small extra cost to the total regret.

Corollary 1. *Once a cycle is interrupted, if the meta-scheduler switches to a MaxWeight (instead of dropping packets) until the system returns idle, the total regret* $\mathrm{Reg}_\pi[\tau]$ *satisfies the following:*

$$\mathrm{Reg}_\pi[\tau] = \begin{cases} O(\log^3 \tau) & \text{if } \mathcal{A}_0 = \mathcal{A}^s(\boldsymbol{\lambda}), \\ O(\log^{3+\delta} \tau), \ \forall \delta > 0 & \text{otherwise.} \end{cases}$$

Proof. By Lyapunov stability, it can be shown that the average time required for MaxWeight to clear out queues of a total length q is in the order of $O(q^2)$. Recall that when a cycle is interrupted before the horizon τ, the total queue length is bounded by $\mathsf{C} \cdot \log \tau$ for some constant C. Therefore, it takes $O(\log^2 \tau)$ time slots for each queue-clearing process to end, and the extra regret induced is thereby in the order of $O(r^{(*)} \cdot \log^2 \tau)$ since $r^{(*)}$ is the rate of the optimal policy. Then the claim is shown by utilizing Lemma 1 and combining with Theorem 1. A complete proof is presented in Appendix A.4. □

2.5 Extension: System with Constraints

In the previous sections, we introduced a UCB-type meta-scheduler that determines the best stable policy optimizing the renewal reward rate of the system. In some applications, the system might be also interested in satisfying a certain performance guarantee besides maximizing the main reward. For

Algorithm 2 UCB Meta-Scheduler with Interruption for System with Constraints

```
1: **Input:** Set of scheduling policies $\mathcal{A}$.
2: **Threshold Function**: $f_s := \beta + \kappa \log s$.
3: **Initialization**: Run every arm $k$ once with interruption threshold $\beta$, then
   initialize empirical rates $\hat{R}_1^{(k)}$ and $\hat{W}_{i,1}^{(k)}, \forall i = 1, 2, \cdots, h$.
4: **for** $n = |\mathcal{A}| + 1, |\mathcal{A}| + 2, \cdots$ **do**
5:     [*before cycle n*]
6:     **for all** $k \in \mathcal{A}$ **do**
7:         Compute *Exploration Bonus* $\Delta_n^{(k)}$.
8:         Compute *Stability Indicator* $I_n^{(k)}$.
                                   ▷ $\Delta_n^{(k)}$ and $I_n^{(k)}$ as defined in Section 2.4.
9:         Compute *Constraint Indicator* $J_n^{(k)}$.
10:    **Arm decision:**
           $$A_n \in \operatorname*{argmax}_{k \in \mathcal{A}} \; I_n^{(k)} \times J_n^{(k)} \times (\hat{R}^{(k)}(n-1) + \Delta_n^{(k)}).$$
11:    **Cycle interruption decision:** $L_n = f_{T_n^{(A_n)}}$.
12:    [*after cycle n*]
13:    Observe $\hat{C}^{(A_n,L_n)}(n), \hat{U}^{(A_n,L_n)}(n)$ and $\hat{V}_i^{(A_n,L_n)}(n)$.
14:    Update $\hat{R}^{(A_n)}(n)$ and $\hat{W}_i^{(A_n)}(n), \forall i$.
```

instance, the system may attempt to minimize mean packet delay of all traffic but also promise that certain users get sufficient service (*e.g.*, 80% of packets must arrive within 5 ms). If the guarantee can be described as a constraint on the reward rate of another renewal-reward process (other than the main reward), we can extend Algorithm 1 to locate the optimal constraint-satisfying policy with a simple modification.

First let us generalize the basic model as follows. For arm k, the n-th cycle $C^{(k)}(n)$ is associated with 1 cycle reward $U^{(k)}(n)$ and h auxiliary rewards $V_1^{(k)}(n), \cdots, V_h^{(k)}(n)$. Both the main and auxiliary rewards satisfy Assumption 1. If $\pi_n = (k, l)$, a stochastic feedback Z_n is observed for n-th cycle as follows,

$$\begin{aligned} Z_n =& (\hat{C}^{(k,l)}(n), \hat{U}^{(k,l)}(n), \\ & \hat{V}_1^{(k,l)}(n), \cdots, \hat{V}_h^{(k,l)}(n), \mathbb{1}_{\{C^{(k)}(n)>l\}}). \end{aligned}$$

As in (2.4), let $w_i^{(k)} := \mathbb{E}[V_i^{(k)}(1)]/\mathbb{E}[C^{(k)}(1)]$ be the renewal reward rate for i-th auxiliary reward of arm k. We call a scheduling policy *acceptable* if it guarantees that the reward rates for the h auxiliary rewards exceed a given threshold $\boldsymbol{\xi} = (\xi_1, \cdots, \xi_h)$, which is known a priori by the meta-scheduler. The optimal arm k^* is thus defined as

$$\begin{aligned} k^* = \quad & \underset{k}{\operatorname{argmax}} \quad r^{(k)} \\ & \text{s.t.} \quad k \in \mathcal{A}^s \cap \{k' | w_i^{(k')} \geq \xi_i, \forall i = 1, 2, \cdots, h\}. \end{aligned}$$

Inspired by the stability indicator, the constraints can also be handled by an indicator that eliminates unacceptable arms with high probability. Since auxiliary rewards still satisfy Assumption 1, we can use the same UCB bound as defined in (2.19), and an arm's *constraint indicator* is set to be false when its empirical rate of auxiliary rewards compensated by the exploration bonus is below $\boldsymbol{\xi}$. Denote $\hat{W}_i^{(k)}(n)$ as the (observed) empirical rate (see (2.8)) of i-th auxiliary reward after n cycles. Formally, the constraint indicator $J_n^{(k)}$ is as follows,

$$J_n^{(k)} = \prod_{i=1}^{h} \mathbb{1}_{\{\hat{W}_i^{(k)}(n-1) + \Delta_n^{(k)} \geq \xi_i\}}.$$

The algorithm is formally presented in Algorithm 2.

Table 2.1: A summary of policies used in our simulations. Here we omit all time indices, and Q_i, H_i, S_i denote the current queue length, head-of-line delay and available service rate of user i **at the time of decision for each packet** (remind that user scheduling is done packet by packet sequentially in a time slot). Unless specifically mentioned we will set policy parameters as below. Note that the policies are all weighted (i.e., b_i) by the inverse of mean rate of each user, which is a common practice suggested in [1].

	Policy Rules	Parameter Settings
MaxWeight	$\text{argmax}_{i\in\mathcal{U}}\, b_i S_i Q_i$	$b_i = 1/\mathbb{E}[S_i]$
Exp-Rule	$\text{argmax}_{i\in\mathcal{U}}\, b_i S_i \exp\left(\frac{a_i Q_i}{c+(u^{-1}\sum_{j\in\mathcal{U}} a_j Q_j)^{\eta}}\right)$	$b_i = 1/\mathbb{E}[S_i], a_i = 1, c = 0.3, \eta = 0.6$
Log-Rule	$\text{argmax}_{i\in\mathcal{U}}\, b_i S_i \log(c + a_i Q_i)$	$b_i = 1/\mathbb{E}[S_i], a_i = 2,\ c = 1$
MaxWeight-HOL	$\text{argmax}_{i\in\mathcal{U}}\, b_i S_i H_i$	$b_i = 1/\mathbb{E}[S_i]$
Max-Rate	$\text{argmax}_{i\in\mathcal{U}}\, b_i S_i$	$b_i = 1/\mathbb{E}[S_i]$

Table 2.2: User profile in the simulation system of Chapter 2.

User Indices	1,2	3,4	5,6	7,8	9,10	11, 12
Distance to BS (m)	50	80	110	140	170	200
Mean Rate (packets/slot)	9.16	6.54	4.84	3.62	2.71	2.04

2.6 Performance Evaluation

In this section, we evaluate the performance of our meta-scheduler algorithms. The simulation setting is based on the IMT Advanced evaluation guidelines for urban macro-cell deployments [41]. We consider a wireless network consisting of a single base station (BS) and $u = 12$ down-link users. The BS is located at the center of the cell with a radius of 250 m, and the user terminals are located in the cell. We assume the total channel bandwidth is 10 MHz. Further, the bandwidth can be divided into 200 resource units of 0.05 MHz each, which can be assigned to different users within a time slot. Scheduling decisions, which consist of the allocation of each resource unit, are made once in each time slot of duration 0.5 ms.

We assume that the size of packets in the system is fixed at 5 kb. At each slot, each users' packets arrive as *i.i.d.* binomial random variables. For simplicity we do not allow one packet to be transmitted across several time slots. The user scheduling within one slot is done in a sequential manner: one of the users is first scheduled for 1 packet based on current queue and rate values, then the updated queues and rates (of the remaining resource units) are used to determine the next user. The process is iterated until remaining resource units cannot support another packet transmission.

The Signal-to-Interference-Noise ratio (SINR) at time t is modeled as $\texttt{SNR}_i[t] = P_b g_i[t]/(\sigma^2 + I_i[t])$ where P_b is the transmit power of BS, $g_i[t]$ denotes the channel gain of user i, σ^2 and $I_i[t]$ denote the noise and the interference level respectively. The channel gain is a combination of path loss, fast fading and antenna gain, Following [41], we set $P_b = 47$ dBm, $\sigma^2 = -104$ dBm, path loss (in dB) computed as $39.1 \log_{10}(\texttt{dist}) + 13.5 + 20 \log_{10}(f_c)$ where $f_c = 2.0$ GHz and $\texttt{dist}$ denotes the user distance, and antenna gain of 17 dBi. Fast fading follows a Rayleigh distribution and is independent over users. For simplicity, we assume the interference is identical to all users and $I_i[t] = -56$ dBm. In any time slot t, the channel state (rate supported by the channel) of the user i is given by

$$S_i[t] = \mathtt{BW} \times \log_2(1 + 10^{0.1(\mathtt{SNR}_i[t] - \mathtt{L})}) \quad \text{bps}$$

where the parameter $\mathtt{L} = 3$dB describes a loss to Shannon capacity.

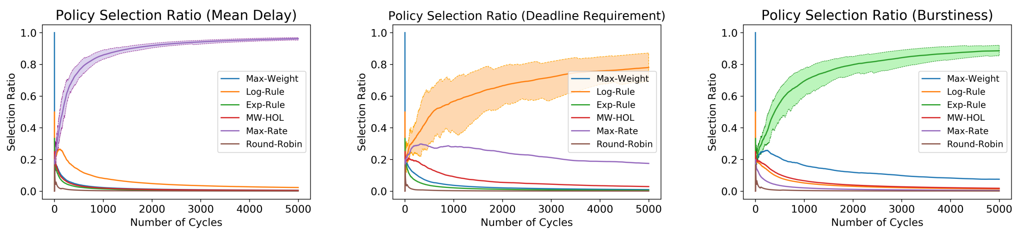
In the following simulations, we will fix the locations of 12 users. The location profile and the mean data rates are given in Table 2.2. Several classical scheduling policies we use are summarized in Table 2.1.

2.6.1 Meta-scheduler behavior and reward design

In this experiment we select various types of rewards and show that the meta-scheduler can indeed pick the optimal policy. We set *i.i.d.* random arrival $A_i[t] \sim \text{Binomial}(3, 0.12)$ for each $i \in \mathcal{U}$ described in Table 2.2. Under this arrival rate ($\lambda_i = 0.36$ packets/slot), cycle lengths induced by the policies in Table 2.1 are no more than 60 ms.

Suppose each packet of the system is associated with a reward and the cycle reward is simply defined as the sum of all packet rewards with proper normalization such that $r_{\max} = 1$ (see Assumption 3). Three types of packet rewards are considered as follows:

Type-1: Mean delay: The reward of each packet equals $(1 - \mathtt{delay} * 0.1)^+$. To optimize this type of reward is equivalent to minimize the mean delay of packets provided the delays are smaller than 10 time slots.



(a) Mean Delay (b) Deadline Requirement (c) Burstiness

Figure 2.2: Mean policy selection ratio of Algorithm 1 for 3 types of rewards defined in Section 2.6.1, after simulating 40 times in each case. The area between 10% and 90% quantile of the best arm is shaded.

Type-2: Deadline requirement: Each packet receives a reward of '1' only if its delay is less than `ddl` slots. Otherwise the packet receives '0' reward. We use $\texttt{ddl} = 8$ in this experiment.

Type-3: Burstiness: This reward favors spreading the service allocations to a user across slots rather than serving a user multiple packets in a single slot. If a user receives a single packet within one slot, this packet is associated with a reward of '1'. If two or more packets are received in the same slot, it will be considered as "bursty" and no rewards are given to any of the packets.

Besides the policies given in Table 2.1, we also consider a Round-Robin scheduler as a baseline which may not be stable even if the traffic loads are within the capacity region. In Table 2.3, we list the average cycle length and reward rates induced by each policy. We set the parameters of Algorithm 1 as $\alpha = 4, \nu^2 = 1, \kappa = 50, \beta = 200, \mu_{\min} = 20, r_{\max} = 1$ and run 40 simulations for each type of rewards. Define the selection ratio of arm k after n cycles as

$(1/n)\sum_{i=1}^{n} \mathbb{1}_{\{A_i=k\}}$. Figure 2.2 exhibits the mean selection ratios of all arms for the three types of rewards (with 10% and 90% quantile shown for the best arms). In each case, Algorithm 1 correctly determines the optimal policy. We observe the rate of convergence largely depends on the performance gap between the best and second best arms: Type-2 reward takes the longest time to separate between Log-Rule and Max-Rate since they have the least gap. As we would expect, Round-Robin scheduler gets discarded quickly in all cases.

Table 2.3: A summary of mean cycle lengths and reward rates for 3 types of rewards considered induced by each policy used in Section 2.6.1. The reward rate of the optimal arm for each reward type is in bold font.

	Mean Cycle Length (ms)	Average Reward per Packet		
		Type 1	Type 2	Type 3
MaxWeight	37	0.717	0.887	0.589
Log-Rule	22	0.805	**0.954**	0.557
Exp-Rule	57	0.627	0.796	**0.599**
MW-HOL	48	0.694	0.931	0.567
Max-Rate	20	**0.832**	0.949	0.484
Round-Robin	$+\infty$	N/A	N/A	N/A

2.6.2 Meta-scheduler behavior dependence on the load

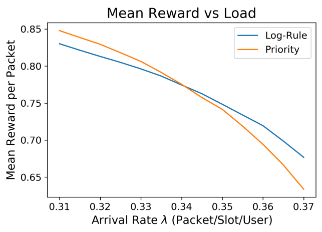
In this experiment, we show the robustness of Algorithm 1 over variations in the traffic load. We design a case where the best policy shifts from one to another when we adjust the load of the system. The goal is to verify the optimal arm is picked by our algorithm in all scenarios.

Suppose Users 6, 12 are two reward Type-2 users with `ddl` $= 2$ (as defined in the last experiment) that are quite strict with packet deadlines, while other users are Type-1 users. The cycle reward is still defined as the sum of packet rewards of all users. We consider two policies: 1) Log-Rule and 2) Priority Rule. The second policy is defined as follows: at each slot, Type-2 users are always scheduled first using a Max-Rate policy; Type-1 users are scheduled using Log-Rule only when there are no more Type-2 packets that can be transmitted. Clearly, the second policy provides better performance for Type-2 users.

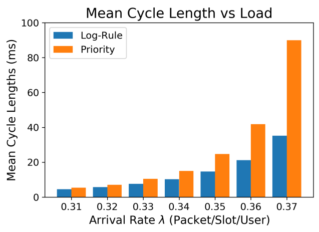
We consider a system where each user has random arrivals $A_i[t] \sim \text{Binomial}(3, \lambda_i/3)$. We increase the traffic load from $\lambda_i = 0.32$ to 0.36 for each $i \in \mathcal{U}$. Figure 2.3a shows the reward per packet under the two policies as a function of the traffic load. When the load is relatively light, the priority-based scheduler outperforms Log-Rule; however, when the load is larger than 0.34, the reward boost of Priority Rule for Type-2 users does not compensate the loss in mean delay for Type-1 users and Log-Rule prevails instead. Indeed, Priority Rule is not even stable for even higher loads (see Figure 2.3b).

Figure 2.3d to 2.3f exhibit the simulation results for $\lambda_i = 0.32, 0.34$ and 0.36. Algorithm 1 correctly locates the optimal policy in the low and high load scenario. When $\lambda_i = 0.34$, the selection ratio of two policies barely separate as the performance gap is almost 0. This is not an issue for any MAB algorithm as both arms can be viewed as the best arm in this scenario.

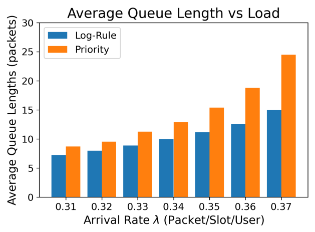
Remark 6. *Figure 2.3d-2.3f illustrate each arm's selection ratio over the*



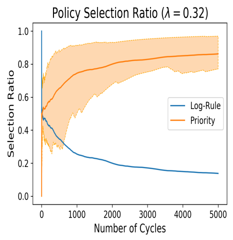
(a) Load vs Reward Rate.



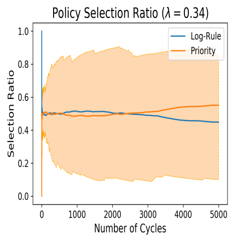
(b) Load vs Cycle Length.



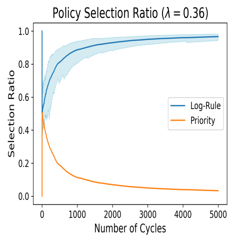
(c) Load vs Sum-Queue Length.



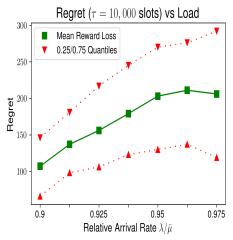
(d) $\lambda_i = 0.32$



(e) $\lambda_i = 0.34$



(f) $\lambda_i = 0.36$



(g) Regret vs Load.

Figure 2.3: Results for the experiment in Section 2.6.2. (a-c) Reward rate, mean cycle length and mean queue length induced by Log-Rule and Priority under changing traffic loads. (d-f) Mean policy selection ratio (40 simulations) of the meta-scheduler when arrival rate $\lambda_i = 0.32, 0.34, 0.36$ respectively, where the area between 10% and 90% quantile of the best arm is shaded. (g) Mean reward loss (aka regret) of meta-scheduler π for time horizon $\tau = 10\text{k}$ over varying traffic loads. Let $\bar{\mu}$ be the mean (non-opportunistic) service rate and $\bar{\mu} = 0.4$ packet/user/slot. We focus on the high load region of $\lambda = 0.36$ to 0.39 (i.e., relative arrival rate > 0.9) where the best policy is Log-Rule. As the load increases, the expanding performance gap of two policies and the growing cycle lengths have opposing effects on the regret, and thus it does not grow monotonically.

number of cycles. Indeed, the meta-scheduler's rate of convergence ***in real time scale*** *also depends on cycle lengths. In general, two factors affect the rate of convergence. First, the larger is the performance gap between the best and second best arm, the easier it is to learn. Thus, as load increases, it is indeed possible that the instance becomes easier due to the increased gap between* *the best and second best schedulers. Second, the longer is the system's cycle time, the slower the learning process is. With this effect, in general, the system with higher loads will exhibit longer cycle times. To get some insight on the load-cycle relation, consider for simplicity a standard M/M/1 queue with $\bar{\lambda}$ and $\bar{\mu}$ as the mean arrival and service rate respectively. For a stable queue, we have the load parameter $\rho := \bar{\lambda}/\bar{\mu} < 1$. From standard analysis of such queues, the mean cycle length is $\bar{\mu}/(1-\rho) + 1/\bar{\lambda}$ and the sub-exponential parameter α roughly scales as $O(1/\log \rho^{-1}) \approx O(\rho/(1-\rho))$. Recall that the regret scales linearly in these parameters, and thus, the regret has an inverse dependence in $(1-\rho)$, assuming the performance gap is fixed. The system we consider is more complex, and includes opportunism, multi-user scheduling and a non-stationary schedule, thus making it hard to analytically quantify the effect.*

In Figure 2.3g, we numerically explore how the regret varies with load and indeed see a mixed impact – as the load increases, the regret does not change monotonically due to the different effects of enlarging performance gaps and growing cycle lengths.

2.6.3 Meta-scheduler behavior with performance constraints

In this experiment, we consider the case where additional constraints are imposed on the system. Let $\lambda_i = 0.36$ packet per slot for any $i \in \mathcal{U}$. Let User 6 and User 12 be Type-2 users. Suppose we impose the following performance guarantee: 75% of packets for user 6 and 12 must arrive with a delay less than 5 slots ($\texttt{ddl} = 5$). And the target is to pick the policy that minimizes the mean delay of the other 10 users while satisfying this constraint.

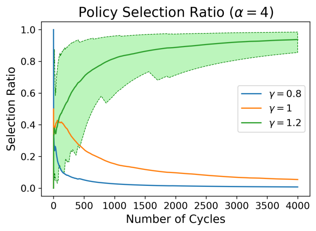
We are given 3 Log-Rule schedulers with different weight parameters b_i (See Table 2.1): $b_i = \frac{1}{\mathbb{E}[S_i]^\gamma}$ where $\gamma = 0.8, 1$ and 1.2. Here γ roughly tunes the fairness of each user, and a larger γ is good for users with low average rates. Table 2.4 summarizes the reward rates for the constraints and main objective. Only the policy with $\gamma = 1.2$ satisfies the constraints.

We run Algorithm 2 40 times using the same parameters as in the first experiment. Figure 2.4a shows the policy selection ratio over number of cycles with the constraints described above. As a comparison, we drop the constraint (by setting $\boldsymbol{\xi} = 0$) and the result is shown in Figure 2.4b. In both cases, Algorithm 2 locates the best constraint-satisfying policy.

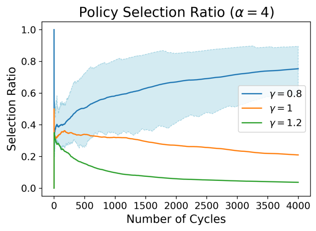
To clearly show the behavior of Algorithm 2, we manually slow the convergence of learning by increasing hyper-parameter α from 4 to 20, which corresponds to a more conservative upper confidence bound. As shown in Figure 2.4c, the third policy prevails the selection ratio after the other two policies sequentially get dropped by the constraint indicators.

2.7 Conclusion

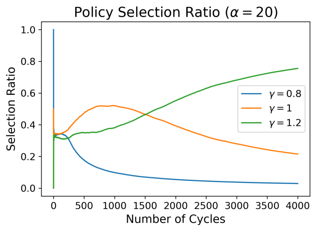
In this chapter, we move from the traditional approach of designing a good downlink wireless scheduler given a scenario and/or rewards to that of determining which amongst a set of possible (good) schedulers is the best for the given context, e.g., user loads, service capacity, and performance requirements.



(a) System with a constraint



(b) System without constraints



(c) System with a constraint (slow convergence)

Figure 2.4: Results for the experiment in Section 2.6.3 using Algorithm 2. (a) The meta-scheduler finds the best policy ($\gamma = 1.2$) subject to the performance constraint defined in Section 2.6.3. (b) The meta-scheduler finds the best policy ($\gamma = 0.8$) when the constraint does not exist. (c) Repeating (a) with $\alpha = 20$ to see clear convergence behavior of the meta-scheduler.

Our, so-called, meta-scheduler, provides a systematic approach to achieve robustness to uncertainty in the demand, environment, or users' needs. This is accomplished by leveraging a budgeted multi-armed bandit framework, which uses the queuing system's regeneration cycles as natural times to make choices amongst arms (scheduling policies), but also by introducing a cycle interruption policy that is shown to ensure that eventually only stable policies are chosen. We provide a theoretical analysis that shows two objectives are met: (1) the approach has sub-linear regret, and (2) the losses due to interruptions are negligible. Our simulations show the meta-scheduler approach is effective, and exhibits the ability to achieve robust decisions in selecting a context-dependent best scheduling policy.

Finally, there has been a renewed interest in using Reinforcement Learning (RL) algorithms for wireless resource allocation. However, designing the ideal wireless scheduler that will achieve optimal performance in all possible

Table 2.4: A summary of mean renewal reward rates for the main and auxiliary rewards induced by each policy used in Section 2.6.3. Only the last policy is acceptable when $\boldsymbol{\xi} = [0.75, 0.75]$.

	Average Reward per Packet		
	User 6	User 12	Others (Main Reward)
$\gamma = 0.8$	0.855	0.577	**0.808**
$\gamma = 1.0$	0.810	0.725	0.803
$\gamma = 1.2$	**0.772**	**0.834**	0.787

settings is likely an impossible goal, even with current RL techniques. Our meta-scheduler framework provides an approach to leverage a collection of state-of-art schedulers (possibly even RL-based) which are known to be good for specific settings, and achieve "universality" by learning which amongst these provides the best results for the given operational scenario.

Chapter 3

Online Learning for Hierarchical Scheduling to Support Network Slicing in Cellular Networks

In this chapter, we study a learning-based hierarchical scheduling framework in support of network slicing for cellular networks. [1] This addresses settings where users and/or service classes are grouped into slices, and resources are allocated hierarchically. The hierarchy is implemented by combining a slice-level scheduler which allocates resources to slices, and flow-level schedulers within slices which opportunistically allocate resources to users/services. Optimizing the slice-level scheduler to maximize system utility is typically hard due to underlying heterogeneity and uncertainty in user channels and performance requirements. We address this by reformulating the problem as an online black-box optimization where slice-level schedulers (parameterized by a weight vector) combined with flow-level schedulers result in user/service level stochastic rewards representing performance fitness; the goal is to learn the best weight vector. We develop a bandit algorithm based on queueing cycles by building on Hierarchical Optimistic Optimization (HOO). The algorithm

[1]The content of this chapter is based on Song, Jianhan, et al. "Online learning for hierarchical scheduling to support network slicing in cellular networks," *Performance Evaluation*, vol. 152, p. 102237, 2021. The author, Jianhan Song, took on most of the responsibility for the problem formulation and the theoretical analysis, and conducted all of the simulations.

guides the system to improve the choice of the weight vector based on observed rewards. Theoretical analysis of our algorithm shows a sub-linear regret with respect to an omniscient genie. Finally through simulations, we show that the algorithm adaptively learns the optimal weight vectors when combined with opportunistic and/or utility-maximizing flow-level schedulers. This work is completed and has been published in [5].

3.1 Introduction

The increasing complexity of cellular wireless networks has led to network slicing as a popular paradigm for resource sharing [42]. Network slicing is a coarse resource allocation mechanism that partitions traffic flows into groups (slices) and allocates network resources (e.g. spectrum) to each of these slices. Network slicing typically operates hand-in-hand with a finer-grain resource manager (flow-level scheduler) that allocates resources among the flows within each slice. Slicing can be used for various reasons including isolating groups from each other in the presence of traffic load fluctuations, or grouping flows with similar Quality of Service (QoS) requirements so that flow-level schedulers can operate across groups of flows with roughly homogeneous requirements.

For example, a network operator might provision a slice for a Mobile Virtual Network Operator (MVNO), e.g., a cheaper cellular provider, or to an autonomous driving service (for map downloads to cars and over-the-air software upgrades). Each slice could then be virtually allocated network resources with some guarantees on availability and/or isolation from traffic fluctuations from other traffic sharing the network. As another example, a carrier might create a slice to support mobile users requiring real-time video flows for which the desired QoS metric is tied to meeting packet deadlines, and a slice for users carrying out file downloads for which QoS is better tied to mean delays and/or flow throughput. Since the QoS requirements within each slice are similar, such grouping allows simpler flow-level (within each slice) scheduling algorithms.

In this paper, we adopt a hierarchical online learning approach to network slicing that is driven by user feedback in the form of rewards. Given a collection of slices (each slice defined through a collection of flows, and with QoS and spectrum-share requirements), we develop a slice-level scheduler (top of the hierarchy) that dynamically allocates resources to each slice based on the observed rewards from mobile users within each slice. This slice-level scheduler allocates resources by dynamically selecting weights for each slice, with these weights specifying a share of spectrum for each slice through an allocation mechanism such as a *Generalized Processor Sharing (GPS)* scheduler [7].

Further, within each slice, a flow-level scheduler (such as the MaxWeight rule [13]) allocates channel resources to individual flows. Thus, the reward obtained from a slicing allocation depends both on the sharing of spectrum for each slice and the individual allocations within each slice. By treating the transformation from weight selection to reward accumulation[2] as a *blackbox*

[2]This transformation from weights to rewards occurs through a multi-step process: The weights define spectrum shared to each slice through the slice-level GPS scheduler. Within

function, we build on bandit-based blackbox optimization methods to develop adaptive slicing mechanisms. Our main contributions are as follows:

1. Hierarchical Scheduling through Blackbox Optimization: We consider a hierarchical scheduling framework in which a slice-level scheduler is parameterized by a weight vector $\boldsymbol{w}$. For any choice of the weight vector $\boldsymbol{w}$, the system observes a mean reward rate associated with users' performance/utility – this mapping from weights to reward rates is represented as a function $f(\boldsymbol{w})$. Due to the complexity and dynamics of such systems, $f(\boldsymbol{w})$ is analytically hard to optimize and the problem can be better studied as a blackbox optimization. Using a multi-armed bandit framework, where the (continuous-valued) weights correspond to the arms of the bandit and the corresponding arm-rewards accrue from (noisy) user feedback, we develop algorithms that explore the choice of weights to adaptively optimize the blackbox function.

2. CHOOC Algorithm and Analysis: We propose *Cycle-Based HOO with Clipping* (CHOOC) algorithm, a modified Hierarchical Optimistic Optimization (HOO) algorithm [2] to determine the best weight vector for our blackbox optimization problem. CHOOC operates at the time-scale of queueing cycles (idle + busy period); the queue dynamics and rewards are conditionally (given the weight parameter) independent over cycles under proper assumptions.[3] A

each slice, the flow-level scheduler opportunistically (based on the current channel realization) allocates channel resources to individual flows. The mobile nodes, upon receiving the packets from the flows generate rewards (e.g. meeting delay deadlines), which are aggregated at the slice level and then across slices to result in the instantaneous (noisy) reward for a weight selection.

[3]Note that naively applying HOO by *periodically* (regardless of the queueing backlogs)

single exploration sample of $f(\cdot)$ corresponds to selecting a weight vector $\boldsymbol{w}$, using these weights to allocate spectrum resources (to slices) via the associated slice-level scheduler, in turn, allow predetermined flow-level schedulers to assign resources to individual users, and finally collecting the aggregate reward from active users over a queueing cycle.

From a technical perspective, as compared to HOO we address two additional challenges: *(i) Ratio of Rewards:* Since the length of queueing cycles is random and depends on the action (the weight vector $\boldsymbol{w}$), our reward rate is described through a ratio of two random summations – reward accrued over cycles divided by the cumulative cycle lengths – thus, we need to control the associated uncertainty which does not directly fit the standard HOO model (because ratios of sums differ from the sum of ratios). *(ii) Sub-Exponential Rewards and Unstable Queues:* Unlike the sub-Gaussian reward setting of HOO, queueing cycle lengths are either sub-exponential (if $\boldsymbol{w}$ results in stable queues), or can be infinite if the queues become unstable. Thus, we need to clip cycles (i.e. truncate overly long cycles by dropping packets), but must do so with a negligible rate of clipping (to minimize drops). By properly addressing these issues, our theoretical analysis recovers a sub-linear regret, which is of the same order as HOO.

3. Empirical Evaluation: We simulate our algorithm in various wireless settings, which include different slice partitions and and heterogeneous per-

exploring new weights/collecting feedback will not work due to the lack of (conditional) independence of feedback.

formance metrics of user packets, such as mean delay, deadline requirement, and total throughput (for infinitely backlogged users). The experiments show our algorithm is able to locate the optimal weight after a reasonable amount of exploration, and in particular, demonstrate its potential to solve difficult problems, where simple heuristics are either hard to design or under-perform. The simulation examples exhibit the power of our framework in tackling the optimization of performance tradeoffs via hierarchical scheduling across slices, including scenarios where slices' flow-level opportunistic schedulers are designed to optimize the sum utility of infinitely backlogged users share resources with users that use queue-based opportunistic schedulers meet queue stability and/or deadline requirements.

3.1.1 Related Work

Wireless Scheduling and Network Slicing. Multi-user wireless scheduling has been studied for decades with numerous designs developed to meet various needs (see [7] for a survey). Meanwhile, network slicing has received substantial attention recently [43, 42], introducing new research topics regarding wireless scheduling [44, 45, 46]. In this paper, we consider a hierarchical scheduling model for network slicing, which is aligned most to the *network virtualization substrate (NVS)* architecture proposed in [44].

Not surprisingly, attempts at using machine learning (and in particular, reinforcement learning) techniques are made to address the network slicing/scheduling problem due to its complex nature. Some successes have been reported in several application settings, mostly from a practical perspective – see e.g. [47, 48, 49, 50, 51] for some recent developments.

Multi-Armed Bandits and Blackbox Optimization. Multi-armed bandits (MAB) are an online learning model with rich literature on theory and applications [26, 27]. In particular, MAB settings have been applied to the problem of *blackbox optimization*, where an algorithm optimizes a function f by sequentially selecting actions (aka inputs to f) and receiving feedback (aka function evaluations). Early works include Zooming [52], HOO [2], DOO [53], StoSOO [54], etc, with recent studies focusing on more refined theoretical guarantees and/or various applications, e.g., [55, 56, 57, 58].

Finally, bandit algorithms have also been studied in various wireless problems, such as [35, 34, 37, 38]. Furthermore, rewards in the form of ratios appear naturally in queueing settings [32]. This chapter uses clipping similar to Chapter 2 to truncate cycles to ensure the stability of queues (clipping was originally proposed in [32] for handling heavy tails to improve rewards in budgeted bandits with queueing), but our focus here is on weight-choice over a continuum (infinite arms) for the higher-level slicing problem; thus, our challenges and algorithmic approach are different (a more detailed comparison will be discussed in Section 3.3).

3.1.2 Notation

In this chapter, we use bold font characters for vectors and normal font for scalars. Unless stated otherwise, random variables are denoted by capital letters. We denote $\mathbb{1}$ as the $\{0,1\}$-indicator function and $[n]$ as the set $\{1, 2, \cdots, n\}$. Finally, we use "*i.i.d.*" for "independently and identically distributed" and "*a.s.*" for "almost surely".

3.2 System Model

In this section, we formally present a multi-armed bandit framework to address the parameterized network slicing problem. For convenience, the notation introduced in this section is summarized in Table 3.1.

3.2.1 Traffic and Service Model

We consider a queueing system with a single server (base station) and a set of u users, denoted by $\mathcal{U} = [u]$. The users are further grouped into s slices. For each $j \in [s]$, denote $\mathcal{U}_j$ as the set of user indices associated with slice j.

Each user has an associated packet queue. At any time t, the queue lengths of users in the system are denoted by a random vector $\boldsymbol{Q}[t] = (Q_i[t])_{i\in\mathcal{U}}$, where $Q_i[t]$ is the number of packets of the i-th user at the beginning of time slot t. Suppose the system may also include a set of users *with infinitely-backlogged queues*, denoted by $\mathcal{U}^{\mathsf{b}}$. Thus, for any $i \in \mathcal{U}^{\mathsf{b}}$, $Q_i[t] = \infty$ for all t. For users in $\mathcal{U}^{\mathsf{nb}} := \mathcal{U} \setminus \mathcal{U}^{\mathsf{b}}$, we assume $(Q_i[0])_{i\in\mathcal{U}^{\mathsf{nb}}} = \mathbf{0}$, and packet arrivals are modeled as a random process $(\boldsymbol{A}[t])_{t\geq 0}$. Here, $\boldsymbol{A}[t] = (A_i[t])_{i\in\mathcal{U}^{\mathsf{nb}}}$ where $A_i[t]$ has a bounded distribution for any $i \in \mathcal{U}^{\mathsf{nb}}$. We assume $(\boldsymbol{A}[t])_{t\geq 0}$ are *i.i.d.* across time and denote the expectation by $\boldsymbol{\lambda}$ ($\boldsymbol{\lambda} \in \mathbb{R}^{|\mathcal{U}^{\mathsf{nb}}|}$).

Wireless channels are time-varying. We model the service rates at t

Table 3.1: Notations used in Chapter 3.

$\mathcal{U}$	set of users in the system.
$\mathcal{U}_j, j \in [s]$	set of users in slice j, so $\mathcal{U} = \cup_{j\in[s]}\mathcal{U}_j$.
$\mathcal{U}^{\mathsf{b}}, \mathcal{U}^{\mathsf{nb}}$	sets of users with/without infinitely-backlogged queues.
$\boldsymbol{Q}[t] = (Q_i[t])_{i\in\mathcal{U}}$	queue vector denoting the queue lengths at the start of slot t.
$\boldsymbol{S}[t] = (S_i[t])_{i\in\mathcal{U}}$	service rate vector for time slot t.
$\boldsymbol{A}[t] = (A_i[t])_{i\in\mathcal{U}^{\mathsf{nb}}}$	packet arrival vector for time slot t.
$\boldsymbol{\lambda}$	mean arrival rate across non-backlogged queues ($\boldsymbol{\lambda} \in \mathbb{R}^{\lvert\mathcal{U}^{\mathsf{nb}}\rvert}$).
$\boldsymbol{w} = (w_j)_{j\in[s]}$	weight vector with w_j denoting the resource allocation of slice j.
$\mathsf{HS}(\boldsymbol{w})$	hierarchical scheduling policy with parameter choice $\boldsymbol{w}$.
$\mathcal{W}$	set of permissible weight parameters.
$\mathcal{W}(\boldsymbol{\lambda})$	subset of $\mathcal{W}$ in which any weight choice for $\mathsf{HS}(\boldsymbol{w})$ is able to stabilize the system with an arrival rate $\boldsymbol{\lambda}$.
$\mathcal{C}$	long-term capacity region such that any $\boldsymbol{\lambda} \in \mathcal{C}^\circ$ is stabilizable.
$\mathcal{C}_{\mathcal{W}}$	capacity region achieved by the class of schedulers $\{\mathsf{HS}(\boldsymbol{w}) : \boldsymbol{w} \in \mathcal{W}\}$ (we assume $\boldsymbol{\lambda} \in (\mathcal{C}_{\mathcal{W}})^\circ$).
$C^{(\boldsymbol{w})}(n), U^{(\boldsymbol{w})}(n)$	random variables denoting the cycle length and reward for the n-th cycle if $\mathsf{HS}(\boldsymbol{w})$ scheduler is selceted.
π	an adaptive policy to determine weight and clipping choices.
$\hat{C}^\pi_n, \hat{U}^\pi_n$	observed (possibly truncated) cycle length and reward of n-th cycle under policy π.
$f(\cdot)$	reward rate function defined for $\boldsymbol{w} \in \mathcal{W}$.
$\boldsymbol{w}^*$	unique best weight vector, i.e., $\boldsymbol{w}^* := \text{argmax}_{\boldsymbol{w}\in\mathcal{W}} f(\boldsymbol{w})$.
f^*	optimal reward rate, i.e., $f^* := f(\boldsymbol{w}^*)$.
$\mathcal{T}$	tree structure into which $\mathcal{W}$ is partitioned.
(h, i)	node of $\mathcal{T}$ at depth h with index $i \in [2^h]$.
$\mathcal{P}_{h,i}$	subset of $\mathcal{W}$ with which node (h, i) is associated.
$f^*_{h,i}$	supremum of reward rates in node (h, i), i.e, $f^*_{h,i} = \sup_{\boldsymbol{w}\in\mathcal{P}_{h,i}} f(\boldsymbol{w})$.
i^*_h	node index such that (h, i^*_h) contains $\boldsymbol{w}^*$.
$d(\nu, \rho)$	near-optimality dimension with respect to parameters (μ, ρ).

as a random vector $\boldsymbol{S}[t] = (S_i[t])_{i\in\mathcal{U}}$ where $S_i[t]$ denotes the service per plot available to the i-th user at t. We assume $(\boldsymbol{S}[t])_{t\geq 0}$ are *i.i.d.* over time and also independent of the queue lengths and the arrival process. Without loss of generality, we assume $\mathcal{U}^{\mathsf{nb}} \neq \phi$ throughout this chapter[4]. Let $\mathcal{C}$ ($\mathcal{C} \subset \mathbb{R}^{|\mathcal{U}^{\mathsf{nb}}|}$) denote the *long-term capacity* of the system (see [7]), induced by the distribution of $\boldsymbol{S}[t]$. This suggests for any arrival rate $\boldsymbol{\lambda} \in \mathcal{C}^{\circ}$ (the interior of $\mathcal{C}$), there exists at least one scheduling policy that stabilizes the system (i.e., the average queue lengths are finite).

3.2.2 Hierarchical Scheduler

We consider a hierarchical scheduler operated by the system, which consists of a slice-level scheduler and s flow-level schedulers, one for each slice. The slice-level scheduler determines how many resources (e.g., number of resource blocks) are allocated to each slice for each time slot, which is typically *not* channel-aware (i.e., opportunistic). The flow-level scheduler of each slice decides which user flows to serve within that slice using the resources allocated by the slice-level scheduler, and is typically opportunistic to time-varying channels.

In this chapter, we consider a parameterized ***S**lice-**L**evel **S**cheduler*, denoted by $\mathsf{SLS}(\boldsymbol{w})$. The vector $\boldsymbol{w} = (w_j)_{j\in[s]}$ indicates the weights of slices for resource allocation. Roughly speaking, w_j determines in some sense the

[4]If $\mathcal{U}^{\mathsf{nb}} = \phi$, the system is such that there are no queueing stability concerns, which makes the problem simpler.

Algorithm 3 GPS Slice-Level Scheduler

```
1: Parameters: weight vector $\boldsymbol{w}$, number of resource blocks per slot $\texttt{RB}$
2: for $t = 0, 1, 2, \cdots$ do
3:     For any slice $c \in [s]$ which is non-idle, allocate $(\gamma \cdot \texttt{RB})$ resource blocks (with proper discretization) where
           $$\gamma = \frac{w_c}{\sum_{c' \in [s]} \mathbb{1}\{c' \text{ is non-idle}\} w_{c'}}$$
4:     Do the previous step on remaining unused resource blocks (some slices may not exhaust allocated blocks), until all the blocks are used or all packets are transmitted.
```

proportion of resources that is to be allocated to j-th slice. A larger w_j implies users in $\mathcal{U}_j$ are allocated more resources. Here, we present a simple *Generalized Processor Sharing* (GPS) slice-level scheduler in Algorithm 1 as a driving example.

Isolating Slices through Share Guarantees: Without loss of generality, we assume $\boldsymbol{w} \in \mathcal{W} := \{\boldsymbol{w}' \in [0,1]^s : |\boldsymbol{w}'|_1 = 1, \boldsymbol{w}' \succeq \boldsymbol{\zeta}\}$ where $\boldsymbol{\zeta} \in [0,1]^s$ indicates a pre-specified system constraint on the lowest resource allocation for each class, and thus provides a natural way to encode protection (guarantees on resource share) to each slice.

Remark 7 (General Isolation Constraints for Slices)**.** *We let $\mathcal{W}$ to have a simplex shape under the constraint $\boldsymbol{w} \succeq \boldsymbol{\zeta}$. This is to simplify the tree-partitioning procedure which will be discussed in Section 3.2.6. In general, more complex isolation constraints (e.g. guarantees on the sum of weights of slices for a service controlling multiple slices) can be considered; we skip details for ease of exposition.*

We assume j-th slice ($j \in [s]$) pre-specifies its own ***F**low-**L**evel **S**cheduler*, denoted by FLS_j, for its specific needs. The scheduler FLS_j is channel- and queue-aware (for example, a MaxWeight scheduler).

Denote the parameterized ***H**ierarchical **S**cheduler* as $\mathsf{HS}(\boldsymbol{w})$, and $\mathsf{HS}(\boldsymbol{w})$ $:= (\mathsf{SLS}(\boldsymbol{w}), (\mathsf{FLS}_j)_{j\in[s]})$. The class of schedulers $\{\mathsf{HS}(\boldsymbol{w}) : \boldsymbol{w} \in \mathcal{W}\}$ induces a capacity region $\mathcal{C}_{\mathcal{W}} \subset \mathcal{C}$, i.e, for any $\boldsymbol{\lambda} \in (\mathcal{C}_{\mathcal{W}})^{\circ}$ there exists $\boldsymbol{w} \in \mathcal{W}$ such that $\mathsf{HS}(\boldsymbol{w})$ stabilizes the system. We assume a fixed $\boldsymbol{\lambda} \in (\mathcal{C}_{\mathcal{W}})^{\circ}$ for the rest of this chapter. The goal is to find the best parameter $\boldsymbol{w}$ that generates the largest rate of rewards, where the rewards are defined through regenerative cycles which will be discussed later. We denote $\mathcal{W}(\boldsymbol{\lambda}) \subset \mathcal{W}$ as the set of weights such that for any $\boldsymbol{w} \in \mathcal{W}(\boldsymbol{\lambda})$, $\mathsf{HS}(\boldsymbol{w})$ stabilizes the system.

Remark 8. *In this work, we fix the flow-level schedulers (which can be seen as good state-of-the-art approaches, or may be independently decided by the tenant "owning" the slice) but optimize the slice-level scheduling to reduce the learning complexity. We believe this is a practical approach to adopt. It is worth noting that the joint optimization of both slice- and flow-level schedulers is an interesting direction to further investigate. If the flow-level scheduler can be well parameterized as was done for the slice-level counterpart, one could theoretically still approach the problem based on the same bandit framework proposed in this chapter, but this would introduce a higher complexity for the parameter space and may result in longer convergence time. Other ideas for joint optimization can also be considered, e.g., multi-level bandit frameworks, but this is beyond the scope of this chapter.*

3.2.3 Reward Model and Regenerative Dynamics

A queueing system with stochastic arrivals and departures consists of alternate idle and busy periods, i.e., periods of the system without/with packets.[5] A consecutive (idle + busy period) is called a *cycle*.[6] Next, we describe the system dynamics through such cycles before introducing the adaptive scheduler model in the next sub-section. We mostly follow assumptions from Section II-B of [59].

If $\mathsf{HS}(\boldsymbol{w})$ is implemented for the n-th cycle, suppose that the system observes a *cycle length* denoted by $C^{(\boldsymbol{w})}(n)$ (possibly-infinite if $\mathsf{HS}(\boldsymbol{w})$ is unstable), and receives a sequence of *non-negative* rewards $(U^{(\boldsymbol{w})}(n,i))_{1\le i\le C^{(\boldsymbol{w})}(n)}$. Denote by $U^{(\boldsymbol{w})}(n)$ the *cycle reward* of the n-th cycle, i.e., $U^{(\boldsymbol{w})}(n)$ equals the sum of $U^{(\boldsymbol{w})}(n,i)$ for $1 \le i \le C^{(\boldsymbol{w})}(n)$. We make the following model assumptions on the process $((C^{(\boldsymbol{w})}(n), U^{(\boldsymbol{w})}(n)))_{n\ge 1}$:

(1) The cycle length variables $C^{(\boldsymbol{w})}(n)$ are *i.i.d.* over n. This is automatically true due to the *i.i.d.* arrival/service model in Section 3.2.1 if $\mathsf{HS}(\boldsymbol{w})$ is Markovian, i.e., making decisions solely based on current system states. If $\mathsf{HS}(\boldsymbol{w})$ leverages past information (e.g., a proportionally-fair flow-level scheduler), we require the past information is cleared before a new cycle – note that this is solely an implementation choice so as to ensure that the rewards of each

[5]To be precise, a system is idle when there are no packets *excluding infinitely-backlogged queues*. Without loss of generality, we assume arrivals occur at the start of a slot while departures occur at the end.

[6]A weaker notion will be introduced in Remark 13 (Section 3.4) for practical use in high-load systems, where the queues may not be strictly idle to restart a cycle.

cycle are $\boldsymbol{w}$-conditionally independent, otherwise a fair comparison of different weight choices would not be possible.

(2) The cycle rewards $U^{(\boldsymbol{w})}(n)$ are also *i.i.d.* for $n \geq 1$. In addition, we assume that for any $\boldsymbol{w} \in \mathcal{W}$, $0 \leq U^{(\boldsymbol{w})}(n) \leq \bar{r}C^{(\boldsymbol{w})}(n)$ for some $\bar{r} > 0$, i.e., the cumulative rewards generated by $\mathsf{HS}(\boldsymbol{w})$ do not grow faster than linearly over time.

Remark 9 (Reward Model)**.** *A wide range of user requirements or performance metrics can be captured with our reward model. For instance, suppose each packet is associated a bounded reward (e.g., over* $[0, 1]$*) upon reception, and the cycle reward* $U^{(\boldsymbol{w})}(n)$ *equals the sum of the packet rewards delivered within the n-th cycle under scheduler* $\mathsf{HS}(\boldsymbol{w})$*. For a latency-sensitive user with strict packet deadlines, each packet reward can be set as 1* ***only*** *if it meets the deadline; for a user that prefers a smaller mean packet delay, the per packet reward may be set as* $(1 - c \cdot \texttt{delay}, 0)^+$ *for a proper coefficient c. It is worth noting that the manner in which rewards are calculated/defined is not necessarily known by the base station in our model, which allows for very general (possibly user-customized) reward structures.*

Finally, we assume that for any stable weight $\boldsymbol{w} \in \mathcal{W}(\boldsymbol{\lambda})$, cycle length $C^{(\boldsymbol{w})}(n)$ and thereby $U^{(\boldsymbol{w})}(n)$ are sub-exponentially-distributed.

Assumption 4 (Cycle Length Distribution)**.** *If* $\boldsymbol{w} \in \mathcal{W}(\boldsymbol{\lambda})$*,* $C^{(\boldsymbol{w})}(n)$ *is a sub-exponential random variable. This implies that, there exist (*$\boldsymbol{\lambda}$*-dependent)* *parameters* $(\xi^2_{\boldsymbol{w}}, \alpha_{\boldsymbol{w}})$*, such that for all* $n \geq 1$*,*

$$\mathbb{P}(|C^{(\boldsymbol{w})}(n) - \mathbb{E}[C^{(\boldsymbol{w})}(n)]| \geq \varepsilon) \leq \begin{cases} 2e^{-\varepsilon^2/(2\xi^2_{\boldsymbol{w}})} & 0 < \varepsilon \leq \frac{\xi^2_{\boldsymbol{w}}}{\alpha_{\boldsymbol{w}}}, \\ 2e^{-\varepsilon/(2\alpha_{\boldsymbol{w}})} & \varepsilon > \frac{\xi^2_{\boldsymbol{w}}}{\alpha_{\boldsymbol{w}}}. \end{cases} \tag{3.1}$$

As pointed out in [59], this assumption holds true if the system has bounded arrival and channel distributions, and the policies considered are Markovian, which follows an argument of [39].

3.2.4 Adaptive Hierarchical Scheduler, Clipping and Feedback

We are interested in designing an online algorithm that learns the optimal weight parameter inducing the best rate of rewards. For this purpose, we model the problem as a multi-armed bandit (operating over cycles) and each choice of parameters as an arm (the arm set is continuous). At each cycle n, the bandit algorithm, referred to as the ***A****daptive* ***H****ierarchical* ***S****cheduler* (AHS), chooses a weight $\boldsymbol{w} \in \mathcal{W}$, (action of bandit from a continuum) based on past rewards and collects rewards using the Hierarchical Scheduler HS($\boldsymbol{w}$).

Clipping and Motivation: As in [59], the AHS makes decisions to change the weight $\boldsymbol{w} \in \mathcal{W}$ only at the end of queueing cycles (i.e. when the system is empty for users with finite backlogs). It can also choose to terminate a queueing cycle while it is progressing, meaning discard all packets and force the system to become empty. We refer to this operation as *clipping a cycle.* Clipping is used to ensure that a "bad" weight choice (e.g. when the resources allocated to a slice are too small and lead to queue instability within the slice) does not result in arbitrarily long cycles. Furthermore, rewards from a clipped cycle should be properly penalized, ensuring that bad weight choices are eliminated as the algorithm adapts over time.

We represent AHS by the sequence $\pi = (\pi_n)_{n\geq 1}$, where $\pi_n = (\boldsymbol{W}_n^\pi, L_n^\pi) \in \mathcal{W} \times (\mathbb{Z}_+ \cup \{+\infty\})$. A decision π_n consists of selecting both a weight parameter and a clipping threshold. In other words, in the n-th cycle, if $\pi_n = (\boldsymbol{w}, l)$, then AHS will use $\mathsf{HS}(\boldsymbol{w})$ to collect rewards, and will clip the cycle if its duration exceeds l time-slots. The stochastic feedback that is received by AHS at the end of the n-th cycle is represented by Z_n^π. Formally,

$$Z_n^\pi = (\hat{C}_n^\pi,\, \hat{U}_n^\pi,\, \mathbb{1}_{\{C^{(\boldsymbol{w})}(n)>l\}}),$$

where $\hat{C}_n^\pi$ and $\hat{U}_n^\pi$ denote the (random) observed length and reward for the n-th cycle, i.e., $\hat{C}_n^\pi = \min(C^{(\boldsymbol{w})}(n), l)$ and $\hat{U}_n^\pi = \sum_{j=1}^{\hat{C}_n^\pi} U^{(\boldsymbol{w})}(n,j)$.

Remark 10 (Soft Clipping)**.** *It is worth noting that the clipping discussed above is an implementation choice (rather than a necessary assumption) which we use to simplify the interpretation of the framework. For some applications it may be unacceptable to drop packets. In this case, it is possible to use an alternative soft clipping strategy as follows: whenever a cycle is clipped, instead of dropping packets, a default stabilizing policy* $\mathsf{HS}(\boldsymbol{w}_0)$ *is implemented until the system becomes idle again (we assume some* $\boldsymbol{w}_0 \in \mathcal{W}(\boldsymbol{\lambda})$ *is known a priori). By applying Theorem 2.3 of [39], it can be proven that the time taken by each queue-clearing process is polynomial to the total length of queues at clipping (which is at the same order of the clipping threshold). As we will show in the next section, the threshold only grows logarithmically, therefore, the extra* *reward loss resulting from this process is poly-logarithmic. Orderwise, this does not affect the main theoretical result in Section 3.3.4.*

3.2.5 Reward Rates, Optimal Weight and Regret

As discussed above, the goal of AHS is to locate the optimal weight parameter leading to the highest rate of rewards, which will be formally-defined in this section.

First, for AHS (denoted by) π, denote by $M_\pi(n)$ the total number of slots for first n (possibly-clipped) cycles and by $N_\pi[\tau]$ the number of *completed* cycles before time τ, i.e., $N_\pi[\tau] = \max(n : M_\pi(n) \le \tau)$. Let $\mathrm{Rew}_\pi[\tau]$ be the cumulative rewards collected under π over the first τ slots, i.e.,

$$\mathrm{Rew}_\pi[\tau] := \sum_{n=1}^{N_\pi[\tau]} \hat{U}_n^\pi + \tilde{U}(\tau),$$

where $\tilde{U}(\tau)$ is the shorthand notation for the sum of rewards of the first $\tau - M_\pi(N_\pi[\tau])$ slots of the $(N_\pi[\tau]{+}1)$-th cycle.

For any $\boldsymbol{w} \in \mathcal{W}$, we define a static non-clipping policy $\pi^{(\boldsymbol{w})}$ such that $\pi_n^{(\boldsymbol{w})} = (\boldsymbol{w}, \infty)$ for any $n \ge 1$. Then we can define a reward rate function f where

$$f(\boldsymbol{w}) = \limsup_{\tau\to\infty} \frac{1}{\tau}\mathbb{E}\big[\mathrm{Rew}_{\pi^{(\boldsymbol{w})}}[\tau]\big].$$

This function f indeed maps each weight choice to the rate of rewards generated by $\mathsf{HS}(\boldsymbol{w})$, and is to be optimized. Note that by Renewal Theory, for any stable weight $\boldsymbol{w} \in \mathcal{W}(\boldsymbol{\lambda})$,

$$f(\boldsymbol{w}) = \frac{\mathbb{E}[U^{(\boldsymbol{w})}(1)]}{\mathbb{E}[C^{(\boldsymbol{w})}(1)]}. \tag{3.2}$$

This motivates us to use (empirical reward / empirical length), or the empirical reward rate, as the estimator of f.

Denote the optimal weight by $\boldsymbol{w}^* := \operatorname{argmax}_{\boldsymbol{w} \in \mathcal{W}} f(\boldsymbol{w})$ and the optimal rate of rewards as $f^* := f(\boldsymbol{w}^*)$. For the analysis of our algorithm, we will assume $\boldsymbol{w}^*$ to be unique and lie in the stable weight region $\mathcal{W}(\boldsymbol{\lambda})$. In practice, an unstable system is typically unwelcome and should be penalized in rewards, which makes this assumption reasonable.

Finally, we define the cumulative regret of AHS π with respect to the best static non-clipping Hierarchical Scheduler, i.e., $\pi^{(\boldsymbol{w}^*)} = (\boldsymbol{w}^*, \infty)_{n \geq 1}$, as follows,

$$\mathrm{Reg}_\pi[\tau] = \mathbb{E}[\mathrm{Rew}_{\pi^{(\boldsymbol{w}^*)}}[\tau] - \mathrm{Rew}_\pi[\tau]].$$

3.2.6 Tree-based Partitioning and Structural Assumption

In this chapter we will present a tree search-type algorithm for the blackbox optimization of f. Let us first define a tree-based partitioning on $\mathcal{W}$ and a joint structural assumption on f and the partitions. This is a standard approach adopted in recent studies on bandit algorithms with a continuous arm set.

Assume that $\mathcal{W}$ is partitioned into an infinite binary tree $\mathcal{T}$, represented by $\mathcal{T} = \{(h, i)_{h \geq 0, 1 \leq i \leq 2^h}\}$. The index pair (h, i) represents the i-th node at the depth h of the tree, which is associated with a region $\mathcal{P}_{h,i} \subset \mathcal{W}$. The collection $(\mathcal{P}_{h,i})_{h \geq 0, 1 \leq i \leq 2^h}$ is referred to as a "tree of coverings", which satisfies $\mathcal{P}_{0,1} = \mathcal{W}$

Algorithm 4 Subroutines for Tree Partitioning

```
procedure SPLIT($\mathcal{P}_{h,i}$)
    $(\boldsymbol{v}_1, \cdots, \boldsymbol{v}_s) \leftarrow \mathcal{P}_{h,i}$          ▷ Retrieving vertices of $\mathcal{P}_{h,i}$
    $(j, k) \leftarrow \mathrm{argmax}_{(j',k')} ||\boldsymbol{v}_{j'} - \boldsymbol{v}_{k'}||_2$
    $\boldsymbol{v}' \leftarrow (\boldsymbol{v}_j + \boldsymbol{v}_k)/2$
    $\mathcal{P}_{h+1,2i-1} \leftarrow$ Replace vertex $\boldsymbol{v}_j$ of $\mathcal{P}_{h,i}$ by $\boldsymbol{v}'$
    $\mathcal{P}_{h+1,2i} \leftarrow$ Replace vertex $\boldsymbol{v}_k$ of $\mathcal{P}_{h,i}$ by $\boldsymbol{v}'$
    return $\mathcal{P}_{h+1,2i-1}, \mathcal{P}_{h+1,2i}$
procedure SELECT($\mathcal{P}_{h,i}$)
    $(\boldsymbol{v}_1, \cdots, \boldsymbol{v}_s) \leftarrow \mathcal{P}_{h,i}$
    return $\boldsymbol{w}_{h,i} := (1/s) \sum_{j=1}^{s} \boldsymbol{v}_s$
```

and $\mathcal{P}_{h,i} = \mathcal{P}_{h+1,2i-1} \cup \mathcal{P}_{h+1,2i}$ for all $(h, i) \in \mathcal{T}$. In this chapter, we will use a "normal" partition: The coverings $(\mathcal{P}_{h,i})_{h\geq 0, 1\leq i\leq 2^h}$ is determined by sub-routine SPLIT exhibited in Algorithm 4, i.e., starting from the root $\mathcal{W}$, iteratively halving each node simplex into two child simplices.

Let $f^*_{h,i} = \sup_{\boldsymbol{w}\in\mathcal{P}_{h,i}} f(\boldsymbol{w})$. The sub-optimality of node (h, i) is denoted by $\Delta_{h,i} := f^* - f^*_{h,i}$. The unique node at depth h that contains $\boldsymbol{w}^*$ is denoted by (h, i^*_h), i.e., $f^*_{h,i^*_h} = f^*$. For purposes of performing regret analysis, we will make the following assumption on function smoothness of f with respect to $\mathcal{W}$. This assumption follows that used in [56].

Assumption 5 (Function Smoothness)**.** *There exists $\nu > 0$ and $\rho \in (0, 1)$ such that the following holds: If a node (h, i) is such that $\Delta_{h,i} \leq c\nu\rho^h$ for some constant $c \geq 0$, then $f(\boldsymbol{w}) > f^* - \max\{2c, c + 1\}\nu\rho^h$ for all $\boldsymbol{w} \in \mathcal{P}_{h,i}$.*

This assumption implies that, for any optimal node (h, i^*_h), we have that $f^* - \inf_{\boldsymbol{w}\in\mathcal{P}_{h,i^*_h}} f(\boldsymbol{w}) < \nu\rho^h$ for some μ and ρ,, i.e., the worst loss in reward rates of an optimal node contracts geometrically with h. A related notion is *near-optimality dimension* with respect to (ν, ρ). Here we use the definition originally stated in [55].

Definition 5. *The near-optimality dimension of function f with respect to parameters (ν, ρ) is given by $d(\nu, \rho) := \inf\{d' \in \mathbb{R}_+ : \exists \mathsf{C}(\nu, \rho), \forall h \geq 0, \mathcal{N}_h(2\nu\rho^h) \leq \mathsf{C}(\nu, \rho)\rho^{-d'h}\}$, where $\mathcal{N}_h(\epsilon)$ is the number of nodes (h, i) such that $f^*_{h,i} \geq f^* - \epsilon$.*

Roughly speaking, $d(\nu, \rho)$ measures the difficulty of the problem: The larger the dimension is, the larger is the number of "near-optimal" nodes which are hard to distinguish from the optimal node, implying that more exploration is needed. This notion will be used in regret analysis.

3.3 Algorithm Design and Analysis

In this section, we introduce our main result – ***C****ycle-based* ***HOO*** *with* ***C****lipping (CHOOC)*, an HOO-type bandit algorithm to determine the optimal weight parameter $\boldsymbol{w}^*$. Compared with the original HOO in [2], we need to address two main challenges. First, each action (arm decision) is fixed over an entire (stochastic) cycle time, and the function to be optimized is a ratio of the cumulative reward across cycles divided by the cumulative cycle time. Thus, the exploration bonus used for the upper confidence bound has to account for the double stochasticity in both rewards and durations. Second, the distributions of cycle variables cannot be described by homogeneous sub-Gaussian parameters; instead, the length (and reward accordingly) of a cycle can be as large as infinite for some weight choices, suggesting that a proper cycle clipping rule is necessary (which does not drop an excessive number of packets).

As discussed in the *Related Work*, a similar bandit problem setting was tackled in Chapter 2, but with a finite (discrete) set of arms. In that setting, the arms that generate 'short' cycles (whose lengths are exponentially distributed under specific parameters) can be differentiated from those generating 'long' cycles by designing an appropriate exploration bonus, and thus the regrets can simply be computed separately in the analysis. In our work, as we have an infinite collection of arms (more precisely a continuum of arms), the discretization of arms via a tree-like structure complicates the design of an exploration bonus (due to the continuity of the reward ratio) and the aforementioned separation in regret analysis (since there is no clear border between stable and unstable weights). Thus, additional effort is needed to ensure that the algorithm still yields a logarithmic regret with cycles and clipping. In the following, we will present our proposed algorithm and introduce several key assumptions and design elements.

3.3.1 Hyper-parameters and Algorithm Design

Let us first discuss necessary hyper-parameters to be used in our algorithm. Before that, denote $\mathcal{W}(\boldsymbol{\lambda};\xi^2,\alpha) \subset \mathcal{W}(\boldsymbol{\lambda})$ as the largest set of arms such that for any $\boldsymbol{w} \in \mathcal{W}(\boldsymbol{\lambda};\xi^2,\alpha)$, variables $C^{(\boldsymbol{w})}(1)$ and $U^{(\boldsymbol{w})}(1)$ are both (ξ^2,α)-sub-exponentially distributed.

Assumption 6 (Hyper-parameters)**.** *We assume that the hyper-parameters* *are chosen to satisfy the following conditions:*

(a) (ν, ρ) *that satisfies the condition in Assumption 5.*

(b) (ξ^2, α) *and* $z \geq 1$ *such that* $\mathcal{P}_{z,i_z^*} \subset \mathcal{W}(\boldsymbol{\lambda}; \xi^2, \alpha)$*,*

(c) $\mu_{\min}, \mu_{\max}$ *such that for any* $\boldsymbol{w} \in \mathcal{P}_{z,i_z^*}$*,* $\mu_{\min} \leq \mathbb{E}[C^{(\boldsymbol{w})}(1)] \leq \mu_{\max}$*,*

(d) $r_{\max}$ *such that* $\mathbb{E}[U^{(\boldsymbol{w})}(1)|C^{(\boldsymbol{w})}(1) = l] \leq r_{\max}\, l$ *for all* $\boldsymbol{w} \in \mathcal{P}_{z,i_z^*}$ *and* $l \geq 1$*.*

Remark 11. *We assume a depth* z *is known such that the optimal node* (z, i_z^*) *and its descendants are purely* (ξ^2, α)*-sub-exponential (in terms of corresponding cycle length/reward variables). A "pure" node provides necessary concentration properties such that the exploration bonus, a term we will formally discuss later, sufficiently compensates the empirical rewards used to approximate* f*, which may be under-performing due to the stochasticity of feedback. This condition can be met by setting* z *to be large enough.*

With the exception of z, the other parameters described above are used for defining the exploration bonus. Parameters of type (a) and (b) are standard in HOO-type algorithms. Parameters similar to those in (c) and (d) are commonly used in bandit algorithms where each action costs non-unit amount of resources (cycle time in our model), e.g., [60, 32]. Note that $f^* \leq r_{\max} \leq \bar{r}$ (see Section 3.2.3 and Eq. (3.2)). While Assumption 6 is needed for our regret analysis, it is worth noting that none of the assumptions are "hard constraints"; indeed empirically selecting those parameters (instead of formally verifying that the conditions are satisfied) is sufficient in practice, as we observe good robust performance in our simulations. We refer to Section 3.4 (in particular, Remark 12) for details.

The full algorithm is presented in Algorithm 5. In the sequel, the term "explore node (h, i)" means "select a representative weight $\boldsymbol{w}_{h,i}$ inside $\mathcal{P}_{h,i}$ and run $\mathsf{HS}(\boldsymbol{w}_{h,i})$", where $\boldsymbol{w}_{h,i}$ is determined by subroutine Select described in Algorithm 4. Each node is associated with an *upper confidence bound* (or "B-value"), which is an optimistic estimate of $f^*_{h,i}$ that equals the corresponding empirical reward rate plus an exploration bonus, both to be defined later.

The AHS first partitions $\mathcal{W}$ into $(\mathcal{P}_{z,i})_{1\leq i\leq 2^z}$, and initializes the B-values of depth-z's nodes as ∞ (line 3-8). The algorithm starts by exploring nodes at depth z. This is to guarantee that the selected weights for the optimal nodes always induce (ξ^2, α)-sub-exponential cycles and thus, the optimal nodes (of depth $\geq z$) are associated with well-behaved B-values as commented in Remark 11, which is essential to the theoretical success of the algorithm.

The AHS starts with no explored nodes (except the virtual parent $(z-1, *)$) in $\mathcal{T}_{\mathsf{exp}}$ (the "explored tree"). Before each cycle (line 10-18), the AHS traverses $\mathcal{T}_{\mathsf{exp}}$ along a path P in which each node has the better B-value among its siblings. This procedure ends until an unexplored node is reached. This node will be explored for this cycle with $\mathcal{T}_{\mathsf{exp}}$ expanded accordingly (line 19-20).

After a cycle ends, the AHS collects feedback and updates the empirical reward and cycle length for each node in P (line 23-25). The empirical reward rate is the quotient of the two (line 26), aligned with Eq. (3.2). Finally

Algorithm 5 Cycle-Based HOO with Clipping (CHOOC)

```
1: **Inputs:** Schedulers $\mathsf{HS}(\cdot)$, Set of Available Weights $\mathcal{W}$, Tree Partitioning
   Subroutines SPLIT and SELECT
2: **Hyper-parameters:** Smoothness Parameters: $(\nu, \rho)$, Sub-Exponential
   Parameters: $(\xi^2, \alpha)$, Initial Depth $z$, Other Parameters: $\mu_{\min}, r_{\max}, \beta, \kappa$ $(\beta > \max(2\mu_{\max}, \mu_{\max}+\xi^2/\alpha), \kappa > 4\alpha)$
3: **Initialization:** $\mathcal{P}_{0,1} \leftarrow \mathcal{W}$, then partition the arm space hierarchically
   until depth $z$:
4:     **for** $0 \leq h \leq z-1$ **do**
5:         **for** $1 \leq i \leq 2^h$ **do**
6:             $\mathcal{P}_{h+1,2i-1}, \mathcal{P}_{h+1,2i} \leftarrow \text{SPLIT}(\mathcal{P}_{h,i})$
7:     Establish a node $(z-1,*)$ as the parent of nodes $\{(z,i) : 1 \leq i \leq 2^z\}$
8:     $\mathcal{T}_{\mathsf{exp}} \leftarrow \{(z-1,*)\}$, $B_{z,i} \leftarrow \infty, T_{z,i} \leftarrow 0\, \forall 1 \leq i \leq 2^z$
9: **for** cycle index $n = 1, 2, \cdots$ **do**
10:    $(h,i) \leftarrow (z-1,*), P \leftarrow \phi$
11:    **while** $(h,i) \in \mathcal{T}_{\mathsf{exp}}$ **do**
12:        **if** $h = z-1$ **then**
13:            $j \leftarrow \text{argmax}_{1 \leq j \leq 2^z} B_{z,j}$ (with a tie-breaker)
14:            $(h,i) \leftarrow (z,j)$
15:        **else**
16:            $j \leftarrow \text{argmax}_{j \in \{0,1\}} B_{h+1,2i-j}$ (w/ tie-breaker)
17:            $(h,i) \leftarrow (h+1, 2i-j)$
18:        $P \leftarrow P \cup \{(h,i)\}$
19:    $(H,I) \leftarrow (h,i), \mathcal{T}_{\mathsf{exp}} \leftarrow \mathcal{T}_{\mathsf{exp}} \cup \{(H,I)\}$
20:    $\boldsymbol{w} \leftarrow \text{SELECT}(\mathcal{P}_{H,I}), l \leftarrow \beta + \kappa \log n$
21:    Run $n$-th cycle using $\mathsf{HS}(\boldsymbol{w})$ with threshold $l$.
22:    Observe cycle length/reward $\hat{C}$ and $\hat{U}$.
23:    **for** all $(h,i) \in P$ **do**
24:        $T_{h,i} \leftarrow T_{h,i} + 1$
25:        $\hat{\mu}^{\mathtt{C}}_{h,i} \leftarrow (1 - 1/T_{h,i})\hat{\mu}^{\mathtt{C}}_{h,i} + \hat{C}/T_{h,i}$,   $\hat{\mu}^{\mathtt{U}}_{h,i} \leftarrow (1 - 1/T_{h,i})\hat{\mu}^{\mathtt{U}}_{h,i} + \hat{U}/T_{h,i}$
26:        $\hat{R}_{h,i} \leftarrow \hat{\mu}^{\mathtt{U}}_{h,i}/\hat{\mu}^{\mathtt{C}}_{h,i}$
27:    $\mathcal{P}_{H+1,2I-1}, \mathcal{P}_{H+1,2I} \leftarrow \text{SPLIT}(\mathcal{P}_{H,I})$
28:    For $j = \{0,1\}$, $B_{H+1,2I-j} \leftarrow \infty, T_{H+1,2I-j} \leftarrow 0$
29:    **Backward update** from leaves to the root in $\mathcal{T}_{\mathsf{exp}}$:
30:        $\Phi_{h,i} \leftarrow$ Compute according to (3.3)
                                        ▷ specially-designed exploration bonus
31:        $B_{h,i} \leftarrow \hat{R}_{h,i} + \Phi_{h,i} + \nu\rho^h$
32:        $B_{h,i} \leftarrow \min(B_{h,i}, \max(B_{h+1,2i-1}, B_{h+1,2i}))$
```

(line 29-32), for all nodes (h,i) in $\mathcal{T}_{\mathsf{exp}}$, new exploration bonuses $(\Phi_{h,i}+\nu\rho^h)$ are computed and B-values are updated, which can be further refined using the fact that the optimistic estimate of a node is no larger than the maximal estimate of its children.

3.3.2 Clipping

It is essential to design a proper clipping rule such that all the "bad" weights leading to queue instability are penalized without blocking the learning process while "good" weights are preserved with little clipping. In our algorithm, we apply the clipping threshold $l_n = \beta + \kappa \log n$ for n-th cycle, where we require $\kappa > 4\alpha$ and β reasonably large. This rule guarantees that any unstable cycle is clipped while cycles induced by $\boldsymbol{w}^*$ (and its neighborhood) are rarely affected, implying negligible performance loss. Indeed, by the property in (3.1), we have that for any $\boldsymbol{w} \in \mathcal{W}(\boldsymbol{\lambda}; \xi^2, \alpha)$,

$$\mathbb{E}\big[\sum_{n=1}^{\infty} \mathbb{1}_{\{C^{(\boldsymbol{w})}(n) > l_n\}}\big] \le \mathsf{C}_0 \text{ (a constant).}$$

Instead, a constant threshold rule inevitably results in a linear number of clippings.

Denote by $\hat{C}^{(\boldsymbol{w},l)}(n)$, $\hat{U}^{(\boldsymbol{w},l)}(n)$ the *observed* cycle variables given $\pi_n = (\boldsymbol{w}, l)$. For the purpose of regret analysis, we set the initial threshold β to be large enough: $\beta > \max(2\mu_{\max}, \mu_{\max} + \xi^2/\alpha)$. In addition, we make the following technical assumptions on cycle clipping.

Assumption 7 (Technical Assumptions)**.** *(a) For any $\boldsymbol{w} \in \mathcal{W} \setminus \mathcal{W}(\boldsymbol{\lambda}; \xi^2, \alpha)$, $\sup_{l \geq \beta} \frac{\mathbb{E}[\hat{U}^{(\boldsymbol{w},l)}(1)]}{\mathbb{E}[\hat{C}^{(\boldsymbol{w},l)}(1)]} \leq f^* - \delta$ for some $\delta > 0$.*

(b) For any $\boldsymbol{w} \in \mathcal{P}_{z,i_z^*}$, the l-interrupted cycle reward $\hat{U}^{(\boldsymbol{w},l)}(1)$ is (ξ^2, α)-sub-exponential for all $l \geq \beta$.

Assumption 7(a) says that for any $\boldsymbol{w} \in \mathcal{W} \setminus \mathcal{W}(\boldsymbol{\lambda}; \xi^2, \alpha)$, where the effect of clipping is *not* negligible, clipping does not significantly boost the reward rate to exceed f^*. The underlying intuition is that an unstable system typically receives much poorer rewards[7]. Assumption 7(b) requires that the clipping does not hurt the sub-exponential property of reward variables for $\boldsymbol{w} \in \mathcal{P}_{z,i_z^*}$. Both (a),(b) can be met if the initial threshold β is sufficiently large, since as $l \to \infty$, we have $\mathbb{E}[\hat{U}^{(\boldsymbol{w},l)}(1)]/\mathbb{E}[\hat{C}^{(\boldsymbol{w},l)}(1)] \to f(\boldsymbol{w})$ and $\hat{U}^{(\boldsymbol{w},l)}(1) \to \hat{U}^{(\boldsymbol{w})}(1)$ *a.s.*

3.3.3 Upper Confidence Bound

For a better description, denote (H_n, I_n), $(\hat{C}_n, \hat{U}_n)$, (C_n, U_n) as the selected node, the observed length/reward and the unclipped length/reward for cycle n. Let $\mathcal{D}(h,i)$ be the descendants of (h,i) including itself. The number of samples of (h,i) for first n cycles is thereby $T_{h,i}(n) = \sum_{t=1}^{n} \mathbb{1}_{\{(H_t,I_t) \in \mathcal{D}(h,i)\}}$. The empirical reward rate of (h,i), once explored, is given by

$$\hat{R}_{h,i}(n) := \frac{\hat{\mu}^{\mathtt{U}}_{h,i}(n)}{\hat{\mu}^{\mathtt{C}}_{h,i}(n)} = \frac{(1/T_{h,i}(n))\sum_{t=1}^{n} \hat{U}_t \mathbb{1}_{\{(H_t,I_t) \in \mathcal{D}(h,i)\}}}{(1/T_{h,i}(n))\sum_{t=1}^{n} \hat{C}_t \mathbb{1}_{\{(H_t,I_t) \in \mathcal{D}(h,i)\}}}.$$

To ensure sufficient exploration of various weights, the upper confidence bound used for comparing nodes' performance is defined as $B_{h,i}(n) := \hat{R}_{h,i}(n) +$

[7]In practice, one can simply set the reward for any clipped cycle to be 0.

$\nu\rho^h + \Phi_{h,i}(n)$,[8] where

$$\Phi_{h,i}(n) = \frac{(1+r_{\max})\epsilon_{h,i}(n) + \epsilon'_{h,i}(n)}{\mu_{\min} + \epsilon_{h,i}(n)}, \tag{3.3}$$

and

$$\epsilon_{h,i}(n) = \begin{cases} \sqrt{\frac{8\xi^2 \log n}{T_{h,i}(n)}} & \sqrt{\frac{8\xi^2 \log n}{T_{h,i}(n)}} \le \frac{\xi^2}{\alpha}, \\ \frac{8\alpha \log n}{T_{h,i}(n)} & \text{otherwise}, \end{cases}$$

$$\epsilon'_{h,i}(n) = \frac{\pi^2}{6} r_{\max} \frac{(\beta + 2\alpha + \kappa \log T_{h,i}(n) + 1)}{T_{h,i}(n)} e^{-\beta/4\alpha}.$$

The term $(\nu\rho^h + \Phi_{h,i}(n))$ is referred to as the exploration bonus. Paralleling the HOO algorithm, the exploration bonus is designed such that the optimal nodes of any depth (almost) always have a sufficiently optimistic estimate, which is a key to the success of UCB-type algorithms. Unlike HOO, however, additional effort is exerted to account for the fact that $\hat{R}_{h,i}(n)$ is a ratio of two random variables (i.e., to deal with the concentration of the empirical reward rate rather than the reward itself) as well as the clipping error. We formally present the discussion in the Lemma below.

Lemma 2. *For all optimal nodes (h, i^*_h), $h \ge z$, and all $n \ge 1$, we have that $\mathbb{P}\left(B_{h,i^*_h}(n) \le f^*\right) \le 2n^{-3}$.*

Proof Sketch. Let i^* be the shorthand of i^*_h when there is no ambiguity.

The first part of the bonus, $\nu\rho^h$, compensates for the gap of $f^* - \inf_{\boldsymbol{w} \in \mathcal{P}_{h,i^*}} f(\boldsymbol{w})$. Suppose there is no stochasticity of feedback or clipping error,

[8]We do not consider the last-step refinement of $B_{h,i}(n)$, since it does not affect the correctness of the regret analysis.

then $\hat{R}_{h,i}(n)$ is equivalent to:

$$\frac{\mu^{\mathtt{U},\dagger}_{h,i}(n)}{\mu^{\mathtt{C},\dagger}_{h,i}(n)} := \frac{(1/T_{h,i}(n))\sum_{t=1}^{n}\mathbb{E}[U_t]\mathbb{1}_{\{(H_t,I_t)\in\mathcal{D}(h,i)\}}}{(1/T_{h,i}(n))\sum_{t=1}^{n}\mathbb{E}[C_t]\mathbb{1}_{\{(H_t,I_t)\in\mathcal{D}(h,i)\}}},$$

i.e., replacing $\hat{U}_t, \hat{C}_t$ as in the original $\hat{R}_{h,i}(t)$ by $\mathbb{E}[U_t], \mathbb{E}[C_t]$. By (3.2) and the remark of Assumption 5, for any optimal node (h, i^*), we have that $\mu^{\mathtt{U},\dagger}_{h,i^*}(n)/\mu^{\mathtt{C},\dagger}_{h,i^*}(n) + \nu\rho^h > f^*$.

The term $\Phi_{h,i^*}(n)$ is used to compensate for stochasticity of cycle rewards and lengths as well as clipping, i.e., the gap between $\hat{\mu}^{\mathtt{U}}_{h,i^*}(n)/\hat{\mu}^{\mathtt{C}}_{h,i^*}(n)$ and $\mu^{\mathtt{U},\dagger}_{h,i^*}(n)/\mu^{\mathtt{C},\dagger}_{h,i^*}(n)$. By manipulating terms, we have that

$$\frac{\mu^{\mathtt{U},\dagger}_{h,i^*}(n)}{\mu^{\mathtt{C},\dagger}_{h,i^*}(n)} - \frac{\mu^{\mathtt{U},\dagger}_{h,i^*}(n) - (\epsilon_{h,i^*}(n) + \epsilon'_{h,i^*}(n))}{\mu^{\mathtt{C},\dagger}_{h,i^*}(n) + \epsilon_{h,i^*}(n)} \le \Phi_{h,i^*}(n).$$

Therefore, combining the discussions above, it follows that

$$\begin{aligned}&\mathbb{P}\left(\hat{R}_{h,i^*}(n) + \Phi_{h,i^*}(n) + \nu\rho^h \le f^*\right)\\ &\quad\le \mathbb{P}\left(\mu^{\mathtt{U},\dagger}_{h,i^*}(n) - \epsilon'_{h,i^*}(n) - \epsilon_{h,i^*}(n) \ge \hat{\mu}^{\mathtt{U}}_{h,i^*}(n)\right) + \mathbb{P}\left(\mu^{\mathtt{C},\dagger}_{h,i^*}(n) + \epsilon_{h,i^*}(n) \le \mu^{\mathtt{C}}_{h,i^*}(n)\right)\end{aligned}$$

Here, term $\epsilon'_{h,i^*}(n)$ is used to rectify the clipping-induced gap $\mu^{\mathtt{U},\dagger}_{h,i^*}(n) - \hat{\mu}^{\mathtt{U},\dagger}_{h,i^*}(n)$.[9] By algebra, we can show that

$$\mu^{\mathtt{U},\dagger}_{h,i^*}(n) - \hat{\mu}^{\mathtt{U},\dagger}_{h,i^*}(n) \le \epsilon'_{h,i^*}(n),$$

given that $\beta > \max(2\mu_{\max}, \mu_{\max} + \xi^2/\alpha)$. The term $\epsilon_{h,i^*}(n)$ serves as the high-probability bound of $|\hat{\mu}^{\mathtt{U}}_{h,i^*}(n) - \hat{\mu}^{\mathtt{U},\dagger}_{h,i^*}(n)|$ and its cycle length counterpart, following Azuma-Hoeffding inequality for Martingale differences (sub-exponential version), Assumption 6(b) and 7(b). This concludes the proof. $\square$

[9]The term $\hat{\mu}^{\mathtt{U},\dagger}_{h,i}(n)$ is defined as $\mu^{\mathtt{U},\dagger}_{h,i}(n)$ with U_t replaced by $\hat{U}_t$.

3.3.4 Main Theoretical Result

Theorem 2. *Given that the hyper-parameters satisfy Assumption 6 and 7, the cumulative regret of π induced by Algorithm 5 has the following upper bound:*

$$\mathrm{Reg}_\pi[\tau] \le \mathsf{C}^{\mathrm{Reg}} \left(\mathsf{C}(\nu,\rho)\right)^{\frac{1}{d(\nu,\rho)+2}} \tau^{\frac{d(\nu,\rho)+1}{d(\nu,\rho)+2}} (\log \tau)^{\frac{1}{d(\nu,\rho)+2}} + \mathcal{O}(\log^4 \tau)$$

for a universal constant $\mathsf{C}^{\mathrm{Reg}}$.

This result is of the same order of HOO in spite of the cyclic behavior of the algorithm and clipping. Before giving a proof sketch on the main theorem, let us introduce another key lemma, stating that the number of visits to "not-near-optimal" nodes is sub-linear. First, we define a modified reward rate function $\tilde{f}$ where

$$\tilde{f}(\boldsymbol{w}) = \begin{cases} f(\boldsymbol{w}) & \boldsymbol{w} \in (\mathcal{W}(\boldsymbol{\lambda};\xi^2,\alpha))^\circ, \\ \sup\limits_{l \ge \beta} \frac{\mathbb{E}[\hat{U}^{(\boldsymbol{w},l)}(1)]}{\mathbb{E}[\hat{C}^{(\boldsymbol{w},l)}(1)]} & \text{otherwise.} \end{cases}$$

The function value for $\boldsymbol{w} \notin (\mathcal{W}(\boldsymbol{\lambda};\xi^2,\alpha))^\circ$ represents an upper bound on the reward rate *under clipping* (see Assumption 7(a)). The gap $\tilde{\Delta}_{h,i}$ is defined accordingly.

Lemma 3. *For all suboptimal nodes (h,i) such that $\tilde{\Delta}_{h,i} > \nu\rho^h$, we have that for all $n \ge 1$,*

$$\mathbb{E}[T_{h,i}(n)] = \begin{cases} \mathcal{O}(\log n/(\tilde{\Delta}_{h,i} - \nu\rho^h)^2) & \text{if } (h,i) \in \mathcal{D}(z, i_z^*), \\ \mathcal{O}(\log^3 n/(\tilde{\Delta}_{h,i} - \nu\rho^h)^2) & \text{otherwise.} \end{cases}$$

Proof Sketch. By Lemma 14 of [2], we have that for any $u \geq 1$,

$$\mathbb{E}[T_{h,i}(n)] \leq u + \sum_{t=u+1}^{n} \mathbb{P}\big(\underbrace{\bigcup\nolimits_{s=1}^{t} \{B_{s,i_s^*}(t) \leq f^*\}}_{\mathcal{E}_1(t)} \bigcup \underbrace{\{B_{h,i}(t) > f^*\} \cap \{T_{h,i}(t) > u\}}_{\mathcal{E}_2(t)}\big).$$

By Lemma 2, we have that $\sum_{t=1}^{n} \mathcal{E}_1(t) \leq \pi^2/6$. It suffices to show when $T_{h,i}(t) > u$ for a reasonably large u, the event $\{\hat{R}_{h,i}(t) + \Phi_{h,i}(t) > f^* - \nu\rho^h\}$ is unlikely to happen. For $(h,i) \in \mathcal{D}(z, i_z^*)$, the observed cycle length/reward variables are (ξ^2, α)-sub-exponential by Assumption 6(b). Accordingly, we can find a high-probability upper bound $f_{h,i}^* + \Phi'_{h,i}(t)$ for $\hat{R}_{h,i}(t)$, where the term $\Phi'_{h,i}(t)$ is similar to $\Phi_{h,i}(t)$ with a slight adjustment on the clipping, and both $\Phi'_{h,i}(t)$ and $\Phi_{h,i}(t)$ are of the order $\mathcal{O}(\sqrt{\log t / T_{h,i}^{(t)}})$. This suggests that it suffices to find u such that $T_{h,i}(t) > u$ and $\Phi'_{h,i}(t) + \Phi_{h,i}(t) \leq f^* - f_{h,i}^* - \nu\rho^h = \Delta_{h,i} - \nu\rho^h$, which implies the constant $u \sim \mathcal{O}(\log t / (\Delta_{h,i} - \nu\rho^h)^2)$.

For $(h,i) \notin \mathcal{D}(z, i_z^*)$, a looser bound for $\Phi'_{h,i}(t)$ can be shown in the order of $\mathcal{O}(\sqrt{\log t / T_{h,i}^{(t)}} \cdot \log t)$. This concentration is given by the boundedness (instead of sub-exponentiality) of clipped cycles, where the bounds grow logarithmically, which accounts for the extra $\log t$. □

Proof Sketch of Theorem 2. Denote $\mathcal{T}^{(z)}$ as the collection of nodes in $\mathcal{T}$ with depth *greater than or equal to* z. Let $\mathcal{T}^{(z)} = \mathcal{T}_1 \cup \mathcal{T}_2$ where $\mathcal{T}_1 = \mathcal{D}(z, i_z^*)$ and $\mathcal{T}_2 = \mathcal{T}^{(z)} \setminus \mathcal{T}_1$.

It is simple to show that $\mathrm{Rew}_{\pi(\boldsymbol{w}^*)}[\tau] = f^*\tau + \mathcal{O}(1)$. Hence,

$$\mathrm{Reg}_\pi[\tau] \leq f^*\tau - \mathbb{E}[\sum_{n=1}^{N_\pi[\tau]} \hat{U}_n] + \mathcal{O}(1)$$

$$
\begin{aligned}
&\leq \mathbb{E}\Big[\sum_{n=1}^{N_\pi[\tau]} f^* \hat{C}_n - \hat{U}_n\Big] + \mathcal{O}(\log \tau) \\
&\leq \sum_{j=1}^{2} \underbrace{\mathbb{E}\Big[\sum_{n=1}^{N_\pi[\tau]} (f^* \hat{C}_n - \hat{U}_n) \mathbb{1}_{\{(H_n, I_n) \in \mathcal{T}_j\}}\Big]}_{\mathsf{Reg}_{\pi,j}[\tau]} + \mathcal{O}(\log \tau).
\end{aligned}
$$

We can prove that for $\mathcal{T}_1$,

$$
\mathsf{Reg}_{\pi,1}[\tau] \leq \mu_{\max} \mathbb{E}\Big[\sum_{n=1}^{\tau} \mathbb{1}_{\{(H_n, I_n) \in \mathcal{T}_1\}} (f^* - f(\boldsymbol{W}_n^\pi))\Big] + \mathcal{O}(\log \tau).
$$

Applying Lemma 2 and techniques used in the original HOO analysis [2], we show that

$$
\mathsf{Reg}_{\pi,1}[\tau] = \mathsf{C}^{\mathsf{Reg}} \left(\mathsf{C}(\nu, \rho)\right)^{\frac{1}{d(\nu,\rho)+2}} \tau^{\frac{d(\nu,\rho)+1}{d(\nu,\rho)+2}} (\log \tau)^{\frac{1}{d(\nu,\rho)+2}}
$$

for a universal constant $\mathsf{C}^{\mathsf{Reg}}$. This involves further partitioning $\mathcal{T}_1$ in terms of whether or not a node is "near-optimal" with respect to its depth and smoothness parameters (ν, ρ), and handling subsets of $\mathcal{T}_1$ respectively.

For $\mathcal{T}_2$, we have that

$$
\mathsf{Reg}_{\pi,2}[\tau] \leq l_\tau \mathbb{E}\Big[\sum_{n=1}^{N_\pi[\tau]} \mathbb{1}_{\{(H_n, I_n) \in \mathcal{T}_2\}}\Big] f^* + \mathcal{O}(\log n).
$$

Note that any cycle before time τ is bounded by l_τ. The expected number of visits to $\mathcal{T}_2$ is at the order of $\mathcal{O}(\log^3 n)$ as a result of Assumption 7(a), implying that the number of "near-optimal" nodes is finite, and Lemma 3. $\square$

Full proofs of the lemmas and the main theorem can be found in Appendix.

3.4 Performance Evaluation

In this section, we evaluate CHOOC from two experimental perspectives: (1) how well the algorithm converges (i.e., correctness and speed) in different settings, and (2) a series of useful scenarios the system might benefit from the CHOOC framework.

Basic Model Settings for Simulation: For experiments introduced in this section, we first model a simplified cellular wireless network to emulate stochastic channel and arrival processes. Suppose the CHOOC AHS is deployed at a Base Station (BS), located at the center of a circular cell of radius 250 m, which serves 10-20 active users. We assume the total bandwidth ($\mathtt{BW}$) is 10 MHz which can be subdivided into 200 resource blocks. Each time slot lasts 0.5 ms.

For the arrival process of each user, we assume $A_i[t]$ has a *binomial* distribution. Each packet has a fixed size 5 kb. The *signal-to-interference ratio (SIR)* of user i at time t is modeled as $\mathtt{SIR}_i[t] = P_b g_i[t]/I_i[t]$, where $P_b, g_i[t], I_i[t]$ denote the transmit power of the BS, channel gain, and interference level respectively. We set $P_b = 47$ dBm, and consider the channel gain as a combined effect of path loss, Rayleigh fast fading and an antenna gain of 17 dBi. Denote the user distance to BS by $\mathtt{dist}_i$, the path loss is then modeled as $39.1 \log_{10}(\mathtt{dist}_i) + 13.5 + 20 \log_{10}(f_c)$ where $f_c = 2.0$ GHz.[10] For simplicity, we assume the interference level is homogeneous to all users in the cell at -56 dBm.

[10]The SIR-related parameters stated above follow [41].

Table 3.2: Service rate vs User distance in our simulation system.

Distance to BS (m)	50	100	150	200	250
Mean Rate (packets/slot)	9.16	5.34	3.28	2.04	1.28

The service rate of user i is given by $S_i[t] = \mathtt{BW} \times \log_2(1 + 10^{0.1(\mathtt{SIR}_i[t] - \mathtt{L})})$ bps where $\mathtt{L} = 3\,\mathrm{dB}$ describes a gap relative to the ideal Shannon capacity. In the following simulations, we adjust the user distance to the BS to vary the channel distribution seen by users. Table 3.2 exhibits the mapping of a user's distance to mean service rates.

Suppose each transmitted packet is associated a reward, which is determined by the user type. Denote `delay` as the packet delay in slots for any non-backlogged users' packet. We consider users of the following types:

(a) Mean-Delay Type (`MD`): The reward of each packet is given by $(1 - \mathtt{delay} * 0.1)^+$.

(b) Deadline-(t) Type ($\mathtt{DDL}(t)$): The packet reward equals the value of $\mathbb{1}_{\{\mathtt{delay} < t\}}$, i.e., each packet has a deadline t.

(c) Backlogged Type (`BL`): The reward for each transmitted packet of an infinitely-backlogged queue is 1.

The cycle reward is defined as the sum of packet rewards of all users within a cycle, which is further normalized such that the cycle length/reward variables are of the same order. We set the cycle reward to be 0 if a cycle is clipped in order to penalize unstable systems. We use GPS as the slice-level scheduler throughout the simulation.

3.4.1 Convergence Behavior of CHOOC

We explore the convergence behavior of CHOOC in our first experiments. We have observed that CHOOC always locates the *neighborhood of the best weight* within a reasonable time (note that one can only expect the algorithm to reach a certain level of optimality in limited time), accompanied with logarithmically-growing regret, which validates the main theorem of the last section. Below we consider convergence in the context of a 2-slice and a 3-slice setting.

3.4.1.1 A 2-slice system

We start with a simple 2-slice system, where each slice is associated with 6 users whose distances to the BS equal $50, 80, \cdots, 200$ m. Let $\mathcal{U}_1$ be $\texttt{MD}$ users while $\mathcal{U}_2$ be $\texttt{DDL}(7)$ users. The arrival process $A_i[t]$ is identical for all users, and is $\texttt{Binomial}(3, 0.1)$, i.e., $\lambda_i = 0.3$. The Log-Rule [1] is implemented as the flow-level scheduler for each slice. Let $\mathcal{W} = \{\boldsymbol{w} : |\boldsymbol{w}|_1 = 1, \boldsymbol{w} \succeq \mathbf{0}\}$.

To implement CHOOC, we set $\alpha = 4, \xi^2 = 1, \nu = 0.1, \rho = 0.5, \beta = 300, \kappa = 50, \mu_{\min} = 10, r_{\max} = 1$ and $z = 1$. Note that these parameter choices may be overly aggressive (to favor exploitation over exploration) to the extent that Assumption 3 may not hold. However, this is a common practice in UCB-type algorithms in order to see good convergence rate.

The top-left panel in Figure 3.1 shows the (Monte Carlo-simulated) reward rate $f(\boldsymbol{w})$ and the number of clippings after simulating each of $\{\mathsf{HS}(\boldsymbol{w}) : w_1 = 0.05, \cdots, 0.95\}$ for 10k cycles under our proposed clipping rule $l_n =$

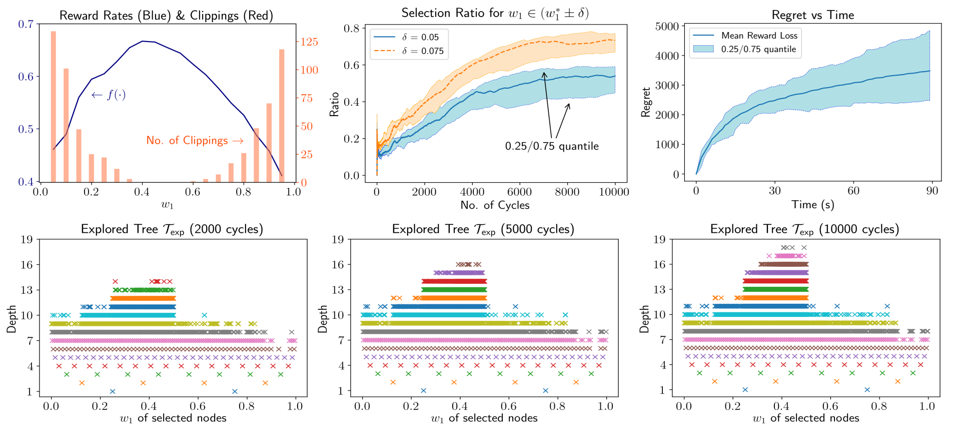


Figure 3.1: Simulation results on experiments in Section 3.4.1. *(Top-Left)* Reward rate function $f(\boldsymbol{w})$ (left axis) and the number of clippings over 10000 cycles for the set of schedulers $\{\mathsf{HS}(\boldsymbol{w}) : w_1 = 0.05, \cdots, 0.95\}$ (right axis). *(Top-Middle)* Selection ratio for $w_1 \in (w_1^* \pm \delta)$ over the number of cycles: median, 0.25/0.75 quantile are shown after simulating CHOOC 20 times. *(Top-Right)* Total regret over time (20 simulation runs). *(Bottom)* Explored tree $\mathcal{T}_{\mathsf{exp}}$ after running CHOOC for 2k, 5k and 10k cycles respectively. Each dot in the scatter plots represents a weight selection at the corresponding depth of the tree.

$\beta + \kappa \log n$. The optimum is roughly $w_1 = 0.42$. We then run CHOOC for 10k cycles. In the bottom panels of Figure 3.1, we show how the explored tree $\mathcal{T}_{\mathsf{exp}}$ evolves from cycle index $n = 2$k to 10k, where each dot in the scatter plots represents a weight selection at the corresponding depth. As expected, the tree grows deeper around the optimum, implying that CHOOC is focusing on exploring near-optimal weights.

Next, we repeatedly simulate the same setting for 20 times and plot the median "near-optimal selection ratio" over cycles (exhibited at the top-right of Figure 3.1), where the ratio at cycle index n is defined as the fraction of w_1 selections lying in $(w_1^* - \delta, w_1^* + \delta)$ for the first n cycles. We set $\delta = 0.05$ or

0.075. The AHS consistently finds the optimum's neighborhood. (Note that since f is relatively flat around $\boldsymbol{w}^*$, it is hard to concentrate the selections exactly at $\boldsymbol{w}^*$.) Convergence is further verified by the time-vs-regret plot in Figure 3.1 (Top-Right), which shows a logarithmic growth. Recall that 1 second is equivalent to 2000 time slots.

Remark 12 (Hyper-Parameter Choices)**.** *The convergence behavior is faster but more error-prone, if the parameters are set more "aggressively" to favor exploitation. For example, setting relatively-small μ and ρ (to accelerate contraction of the exploration bonus) has the benefit of "faster pruning", i.e., dropping out "obviously bad" regions more quickly, but risks ending up with sub-optimal solutions. In practice, one can use Assumption 3 as a (conservative) baseline when picking parameters and then adjust them properly.*

Remark 13 (Convergence in a High-Load System)**.** *In the above simulation, we simulated a moderately-loaded system (the load was 3.6 packets/slot vs a mean service rate ~4.8) and the cycles were short (~10 ms). With higher loads, the set of system-stabilizing weights $\mathcal{W}(\boldsymbol{\lambda})$ contracts while the cycle time grows. The former effect helps the algorithm converge faster, since unstable weights induce much lower rewards (either intrinsically or penalized by clipping) and get eliminated sooner **in terms of the number of cycles used**. The latter effect delays the exploration of new weights, however, in practice it is possible to weaken the definition of cycles to further shorten the convergence time. For instance, one can require a cycle to terminate at time t such that $t = \arg\min_{t'}\{t' \geq t_0 + \tau_0 : \boldsymbol{Q}[t'] \in \mathcal{B}\}$, where t_0 is the cycle start time, τ_0 is*

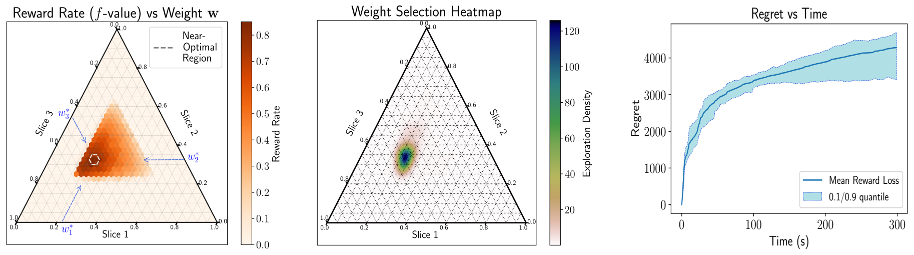


Figure 3.2: Simulation results on experiments in Section 3.4.1.2. *(Left)* Reward rate f. *(Middle)* Heatmap for CHOOC's weight selections (6k cycles). *(Right)* Regret vs Time (averaging over 20 simulation runs).

a constant and $\mathcal{B}$ *is a pre-specified bounded set in the queueing space. This means we only require the queues to return to a bounded region (rather than to the idle state) to finish a cycle, and an ideally-small* $\mathcal{B}$ *with a proper* τ_0 *(both serving as hyper-parameters) ensures that rewards are only weakly correlated across cycles. Later in Section 3.4.2.1, we will conduct an experiment using this relaxation and exhibit its performance.*

3.4.1.2 A 3-slice system

In this part, we set up a 3-slice system to test the convergence of CHOOC in a more general weight space. Suppose each slice contains a mix of `MD`, `DDL` and `BL` type users (5 per slice) but the distributions are non-identical. The BS-to-user distance ranges from 50 to 200 m and the arrival rate is 0.25 packets/slot per user. We set $\mathcal{W} = \{\boldsymbol{w} : |\boldsymbol{w}|_1 = 1, w_i \geq 0.15, \forall i\}$.

The simulation-estimated function f is exhibited in Figure 3.2 (Left). The optimal weight is close to $[0.22, 0.33, 0.45]$, and the peripheral area with 0 reward rate indicates unstable weights. We run CHOOC (following the hyper-parameters in Section 3.4.1.1) for 6k cycles and draw a heatmap for its weight choices. As is shown in Figure 3.2 (Middle), CHOOC faithfully concentrates on the optimal region, and those several layers around the optimum reflect the tree-based exploration process.

The regret over time is plotted as in Figure 3.2 (Right). As a side note, there is no direct relation between the number of slices and CHOOC's convergence rate. The hardness of learning is determined explicitly by smoothness of f, which is measured by the near-optimality dimension $d(\mu, \rho)$ and its corresponding constant $\mathsf{C}(\mu, \rho)$. Roughly speaking, however, a higher dimension of $\mathcal{W}$ tends to induce a larger $\mathsf{C}(\mu, \rho)$, provided that the "curvature" of the function near its optimum (i.e., $d(\mu, \rho)$) is the same. This contributes to a larger coefficient term in the regret (see the main theorem).

3.4.2 Further Motivating Application Scenarios

In this section, we showcase several scenarios where the best weight is hard to pre-determine due to the heterogeneity of user traffic, reward types or slicing structures, but all of which can be easily be optimized by our CHOOC framework. These experiments further exhibit the potential of CHOOC for learning the best weights for hierarchical scheduling in complicated environments.

Table 3.3: Simulation results on experiments in Section 3.4.2.1. For each setting, we list the f values for 4 weights: CHOOC-estimated weights for $\texttt{S1}, \texttt{S2}, \texttt{S3}$, and $\boldsymbol{w}_0 = [0.25, 0.25, 0.25, 0.25]$ (naive partition).

Grouping Strategy	CHOOC-Estimated Weight $\tilde{\boldsymbol{w}}^*$	f^*	Reward Rates $f(\cdot)$			
			$\tilde{\boldsymbol{w}}^*(\texttt{S1})$	$\tilde{\boldsymbol{w}}^*(\texttt{S2})$	$\tilde{\boldsymbol{w}}^*(\texttt{S3})$	$\boldsymbol{w}_0$
S1	$[0.33, 0.27, 0.28, 0.12]$	*0.772*	**0.771**	0.752	0.768	0.755
S2	$[0.18, 0.28, 0.29, 0.25]$	*0.686*	0.635	**0.686**	0.680	0.664
S3	$[0.20, 0.33, 0.28, 0.19]$	*0.689*	0.650	0.659	**0.687**	0.661

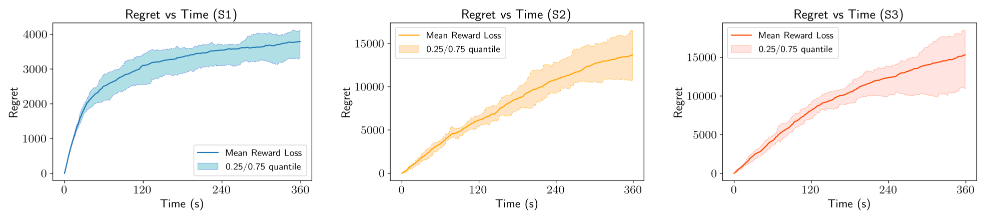


Figure 3.3: Regret plots on experiments in Section 3.4.2.1.

3.4.2.1 Different Grouping Strategies

Sometimes users are grouped into slices based on similar traits, but the criteria for grouping may vary. In the following, we set up a 4-slice, 16-user system and apply CHOOC over varied grouping strategies of heterogeneous users into slice partitions.

There are 4 types of users: $\texttt{DDL(2)}, \texttt{DDL(3)}, \texttt{DDL(4)}, \texttt{DDL(5)}$. Associated with each type there are 4 users whose distances to the BS are $50, 80, 110, 140$ m respectively. The arrival process is homogeneous for each user i and $\lambda_i = 0.32$. Since this system contains more users with higher loads, we adopt the weakened cycle notion proposed in Remark 13 to ensure faster convergence, and set $\mathcal{B} = \{\boldsymbol{Q} : \max_i Q_i \leq 2\}$ and $\tau_0 = 40$. (As a result, cycles under the best weights take ~ 50 ms.)

When setting up the hierarchical scheduler, we consider three grouping strategies for slice partitioning: (`S1`) users with the same distance to BS are grouped together; (`S2`) users with the same deadline requirement are grouped together; (`S3`) mixed – each slice contains users with 4 different distances/deadlines. We set $\mathcal{W} = \{\boldsymbol{w} : w_i \geq 0.1, \forall i\}$, providing a level of isolation among slices.

We first estimate the best reward rate f^* of each scenario by grid searching the simplex $\mathcal{W}$ and Monte-Carlo simulations. Then we run CHOOC in the three settings respectively for 720k time slots (equivalent to 360 s), and we define the *CHOOC-estimated weight* $\tilde{\boldsymbol{w}}^*$ as the mean of weight parameters for last 2k cycles.[11] Finally, we compute $f(\tilde{\boldsymbol{w}}^*)$ by numerical simulation to validate the correctness of CHOOC. The results of $f^*, \tilde{\boldsymbol{w}}^*$ and $f(\tilde{\boldsymbol{w}}^*)$ are summarized in Table 3.3.

As one can expect, the best weight (or say the set of near-optimal weights), despite being easily learned by the CHOOC algorithm, is highly unpredictable due to the complicated trade-offs among users with different service rates and requirements. For example in `S1`, the optimal weight allocation

[11]When approximating the f function of this 4-slice system, we observe that for each setting there exists a subset of weights that all correspond to "near-optimal" reward rates, and it is computationally-hard to estimate the *exact* best weight $\boldsymbol{w}^*$ by numerical simulation due to stochastic error and the complexity of f (compared to the previous 2-slice system where f is a simple concave 1D function). For the same reason, the CHOOC-estimated weight indeed varies over simulation runs (within a near-optimal region), and here in Table 3.3 we only display one representation.

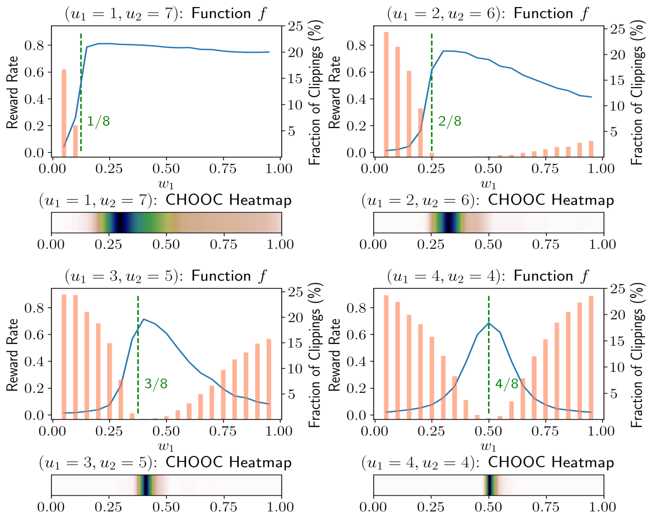


Figure 3.4: Simulation results on experiments in Section 3.4.2.2. For each scenario, the reward rate function and the corresponding weight density heatmap selected by CHOOC are displayed. Note that the simple heuristic $|\mathcal{U}_1|/|\mathcal{U}|$ (dashed line) does not always match the optimal weight.

indeed sacrifices users with the worst service rates (Slice 4) so as to achieve a higher rate of sum rewards. To further validate this observation, for each setting, we compare f values of all three CHOOC-estimated weights as well as a naive choice $\boldsymbol{w}_0 = [0.25, 0.25, 0.25, 0.25]$. It turns out the naive partition is far from ideal and the best weights cannot be transferred across settings.

For each setting, we plot the regret over time (averaged over 20 CHOOC simulation runs) as shown in Figure 3.3. It is worth noting that `S1` has a much better regret mainly due to the fact that the stable set $\mathcal{W}(\boldsymbol{\lambda})$ for `S1` is larger (i.e., lower chance to explore low-reward unstable weights inducing long cycles).

3.4.2.2 Imbalanced Number of Users across Slices

In the next set of simulation, we test the performance of CHOOC under variations in the number of users in each slice. We set up an 8-user, 2 slice system. All users are of type `MD` with arrival rates being 0.58 packets/slot and the distances to BS equaling 120 m. Both slices implement a Log-Rule scheduler. We assign the number of users to the two slices as being $(1, 7), (2, 6), (3, 5), (4, 4)$. In Figure 3.4, we plot the simulation-approximated f values for each scenario. We observe that, except in the last scenario, the optimal weight w_1^* is not $|\mathcal{U}_1|/|\mathcal{U}|$ (proportional to the number of users) as one might suggest as a reasonable heuristic. This is due to the joint effect of the asymmetry in the number of users per slice and opportunistic scheduling within slices. [12]

For each case, we run CHOOC for 10k cycles and the convergence is shown by a density heatmap (see Figure 3.4). As one can imagine, the selected weights are less concentrated around $\boldsymbol{w}^*$ for the first case, since f is flatter.

3.4.2.3 Tradeoffs between Backlogged and Non-backlogged Users

In this set of simulations, we set up a scenario exhibiting the ability of CHOOC to realize tradeoffs among slices of backlogged and non-backlogged users. We include these results in Appendix B.4.1.

[12]Note that a GPS scheduler will re-assign unused resources by $\mathcal{U}_1$ to $\mathcal{U}_2$. Thus, $w_1 = 1$ does not mean that $\mathcal{U}_2$ receives no resources; instead, it implies $\mathcal{U}_1$ is given the full priority. This explains why in the top left panel in Fig. 3.4, f is almost flat for $w_1 > 0.2$ as $\mathcal{U}_1$ already receives sufficient resources.

3.4.2.4 Different Scheduler Choices in CHOOC Framework.

We investigate the performance of CHOOC under various scheduler implementations. These results are included in Appendix B.4.2.

3.5 Conclusion

In this chapter, we study a hierarchical scheduling model for network slicing where the optimal resource allocation among slices, parameterized by a weight vector, is to be determined. To address the complexities of the problem (diversity in user traffic or service requirements), we formulate this through an MAB blackbox optimization framework. We propose the CHOOC algorithm (an adaptive hierarchical scheduler) that builds on the classical HOO with algorithmic/theoretic modifications to account for cycles with clippings and thus is applicable in a queueing-based slicing/scheduling wireless system. Our simulations show that our algorithm has the ability to find the best weight in various scenarios. Although a GPS-based hierarchical scheduler is considered in this chapter, we note that the CHOOC algorithm can be generalized to other parameterized scheduling models with cycles. It is thus of interest to explore its benefits in other related settings.

Chapter 4

Online Learning for Multi-Agent Based Resource Allocation in Weakly Coupled Wireless Systems

In this chapter, we propose and evaluate a learning-based framework to address multi-agent resource allocation in coupled wireless systems. [1] In particular, we consider multiple agents (e.g., base stations, access points, etc.) that choose amongst a set of resource allocation options towards achieving their own performance objective/requirements, and where the performance observed at each agent is further coupled with the actions chosen by the other agents, e.g., through interference, channel leakage, etc. The challenge is to find the best collective action. To that end we propose a Multi-Armed Bandit (MAB) framework wherein the best actions (aka arms) are adaptively learned through online reward feedback. Our focus is on systems which are "weakly-coupled" wherein the best arm of each agent is invariant to others' arm selection the *majority* of the time – this majority structure enables one to develop light weight

[1]The content of this chapter is based on Song, Jianhan, et al. "Online learning for multi-agent based resource allocation in weakly coupled wireless systems," in *Proceedings of the Twenty-Third International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, 2022, pp. 111–120. The author, Jianhan Song, took on most of the responsibility for the problem formulation and the theoretical analysis and conducted all of the simulations.

efficient algorithms. This structure is commonly found in many wireless settings such as channel selection and power control. We develop a bandit algorithm based on the Track-and-Stop strategy, which shows a logarithmic regret with respect to a genie. Finally through simulation, we exhibit the potential use of our model and algorithm in several wireless application scenarios. This work is completed and has been published in [6].

4.1 Introduction

Dynamic resource allocation, including the allocation of time slots, frequency sub-channels, power, etc., is a key part of the design of wireless systems. In a multi-cell setting, resource allocation is especially challenging due to the triad of (i) heterogeneity and uncertainty of the network environment (e.g., time-varying loads, channel states, interference, etc.), (ii) distributed decision-making (separate controller/agent in each base-station), and (iii) availability of only partial state information at each agent (e.g., only local channel states). In such settings, if each agent selects their own allocation strategy/action without consideration of other agents' decisions, the collective can suffer a significant loss in total utility.

We can view the multi-agent resource allocation problem through the following abstraction. Each agent is allowed an action from among a collection of actions (e.g., choice of frequency sub-band in the channel selection problem). Its choice of action has two consequences: (a) the agent accrues a *reward* for itself (e.g., average throughput/delay for users in its cell), and (b) the action induces an *environment* that affects all other agents (e.g., transmitting on a frequency sub-band generates strong interference to other agents in that frequency sub-band, and weaker interference in nearby frequency sub-bands). In a multi-agent setting, the goal then is to find an action for each of the agents (equivalently, a collective of actions across agents), which in turn induces a collective of environments, such that the utility of the collective is maximized.

The immediate search-based solution to this problem – attempting every action at each agent for a sufficiently long duration, empirically estimating the collective reward, and choosing the collective that has the highest utility – can scale poorly due to the super-linear growth in search space. Indeed, even with two users, the number of environments scales as k^2 if each user has k possible actions, making it computationally impractical to learn the best actions within a reasonable time. In general, with no assumptions about the actions and the resulting environments, it is not hard to see that such complexity is unavoidable.

However, in many resource allocation settings that we are interested in, there are additional properties of the overall system that can be used to reduce this complexity. Specifically in this chapter, we focus on systems that are *weakly coupled*. We say a system is weakly-coupled if it satisfies the *majority condition*: we suppose that the optimal action of an agent is also the best action in a majority of environments, where each environment corresponds to a distinct action tuple that can be chosen by the other agents. The intuition is that under moderate interference levels, *most of the time*, the performance of one agent's action does not fluctuate much when actions taken by nearby agents are changed. The majority condition holds in several wireless settings. For example, in the channel selection problem, once an agent selects a frequency sub-band, only a small set of adjacent channels will be significantly interfered with due to channel leakage. Another example is one where each base-station can choose a *scheduler* from among a candidate set [4]. Different schedulers trade-off for different performance metrics within the cell (e.g. MaxWeight for stability, vs. round robin for jitter); however, they have different impact on neighboring cells. In this setting, good schedulers tend to incur low interference to nearby agents (cells) since they typically schedule opportunistically and use channels more efficiently (therefore, the majority condition holds provided that most of the schedulers are "good").

The majority condition is especially useful for algorithm design because we show that it has the following three properties. *(1) Local greedy property:* For each agent, it suffices to learn its best action in each of the environments and choose the "majority best" as its overall best action; *(2) Avoiding hard environments:* Identifying each agent's best action can be cast as k separate multi-armed bandit best-arm identification instances, where k is the number of possible environments. Some of these environments might be especially hard, e.g., strong interference/poor channel quality, thus all actions of the agent in this "hard" environment have low reward, making this best arm identification instance difficult. Crucially however, the majority conditions enable one to avoid solving such hard environments, once the best actions from the easiest $k/2$

environments have been learned; *(3) Sub-sampling property:* When the number of environments k is large, it is possible to sample a subset of environments and still learn the best action (with high probability).

Building on these properties, we develop a decentralized algorithm for multi-agent resource allocation with bandit feedback. The algorithm proceeds episodically with each episode consisting of an exploration and an exploitation phase. During the exploration phase, one agent runs a collection of best-arm-identification subroutines to learn the optimal arm (aka action) in each environment based on local reward feedback, while the other agents cycle over actions from a randomly chosen subset (of all the actions) in a round-robin fashion until the first agent learns the "majority best arm" with a *fixed confidence* (crucially, not all environments have to be "solved"). Once each agent learns the best collective arm using the above procedure, it is applied in the exploitation phase. As the episode index grows, the confidence level is made increasingly tight as the increment of regret converges to zero. We build on Track-and-Stop [3], which is designed for best arm identification with fixed confidence, as the subroutines used in the main algorithm. Track-and-Stop focuses on exploring arms with good rewards and is known to be asymptotically optimal in terms of the number of plays needed to determine the best arm. This further accelerates the exploration and improves the overall performance (in particular, compared to the vanilla Explore-Then-Commit (ETC) approach).

Our main contributions are summarized as follows:

1. **Weakly Coupled Systems under the Majority Condition:** We

develop a multi-armed bandit framework to address the multi-agent resource allocation problem for weakly coupled systems. In these systems, the best arm of each agent is invariant to other agents' arm choices in the majority of scenarios. We believe this assumption is reasonable in many wireless applications, and allows the design of an algorithm with manageable computational and communication costs.

2. Track-and-Stop Based Decentralized Algorithm: We develop a decentralized bandit algorithm using Track-and-Stop as a building block. For systems satisfying the majority condition, this algorithm has two main advantages over classical bandit algorithms: (1) Low communication cost: the decision-making is decentralized as no reward/action information is exchanged and the only coordination needed is when one agent signaling others the end of a Track-and-Stop subroutine. Note that for centralized algorithms such as UCB or the vanilla Track-and-Stop (i.e., best arm identification among all the collective arms), a central controller who has access to all the reward feedback has to be introduced to determine the action for each agent. (2) Efficient with a logarithmic regret: it can be shown that with high probability the regret scales as $O((m-1)k \log k \log T)$ where T is the time horizon, m is the number of agents and k is the (max) number of arms of each agent — this is much improved compared to any classical algorithm which equally views all the k^m collective arms in implementation, with the regret scaling as $O(k^m \log T)$.

3. Empirical Evaluation: We simulate the algorithm in two wireless applications to show the potential usage of our model: (1) channel selection with power leakage and (2) best scheduler selection for wireless queueing systems. In both cases, we show the systems are indeed weakly-coupled such that our algorithm can be applied. Furthermore, our simulations show that the agents can correctly learn the best collective action in reasonable time with a sub-linear regret.

4.1.1 Related Work

Multi-Agent Resource Allocation in Wireless Settings. Many well-studied wireless applications are by nature multi-agent resource allocation problems, such as power control and cognitive spectrum access. A classical theoretic approach is to study the problems through a game theory perspective, e.g., [61, 62] on power control, [63, 64] on dynamic spectral access and cognitive radio, [65, 66, 67] on wireless sensor networks, [68] on edge computing, etc. Moreover, due to the complexity of the problem, machine learning/reinforcement learning techniques have recently be proposed to address related problems, see e.g., [69, 70, 71].

Decentralized Multi-Agent MABs. Multi-agent decision making has been formulated as decentralized multi-armed bandit problems, where multiple players simultaneously pull their arms at each round. In a collaborative setting, the agents learn the same stochastic bandit instance in a decentralized manner, and the goal is to minimize individual regret via information sharing, see, e.g., [72, 73, 74]. Recent works [75, 76] further consider the tradeoffs between regret minimization and communication cost.

More aligned with this chapter is the study on multi-agent MABs with *collision*. In those problems, agents receive normal reward feedback only if other agents do not choose the same arm ("collision") — otherwise, zero rewards are observed by colliding agents. Several settings have been studied in this line of work, including [77, 78, 79] on the homogeneous reward setting (agents observe the same reward distributions on the same arm), and [80, 81, 82] on the heterogeneous reward setting. A recent work [83] further explores the scenario when agents observe non-zero rewards on collisions. Compared to these works, our model is more general regarding the impact of interference on rewards — we do not restrict to a collision-based model and the reward distribution of an arm may be different when nearby agents change to any arm (not necessarily the "colliding arm"). Instead, we explore a special arm-reward structure, i.e., weakly coupled systems under the majority condition, and develop efficient decentralized bandit algorithms.

4.1.2 Notation

Throughout this chapter, we use $[n]$ to denote the set $\{1, 2, \cdots, n\}$, and $\mathbb{1}$ for the $\{0, 1\}$ indicator function. The symbols $\lceil a \rceil$ and $\lfloor a \rfloor$ represent the ceiling and floor function over the value a.

4.2 Problem Formulation

4.2.1 Two-Agent Weakly Coupled Systems

For simplicity let us first focus on a 2-agent system and introduce the notion of weak coupling. Here on, we use the standard bandit terminology of 'arm' to denote an agent's action. In this system, Agent 1 and Agent 2 can choose one over k_1 and k_2 arms respectively for each play (round). We call any pair of arms (i_1, i_2) a collective arm. The joint rewards for two agents choosing $(i_1, i_2) \in [k_1] \times [k_2]$ are independently and identically distributed over multiple plays, and the average rewards are denoted as $(\mu^{(1)}_{i_1,i_2}, \mu^{(2)}_{i_1,i_2})$. Note that the rewards are "coupled" and changing either arm of (i_1, i_2) might affect both rewards $(\mu^{(1)}_{i_1,i_2}, \mu^{(2)}_{i_1,i_2})$.

As usual the goal of a bandit framework is to find the best (collective) arm. Let the arm pair (i_1^*, i_2^*) satisfy that $\mu^{(1)}_{i_1^*,i_2^*} + \mu^{(2)}_{i_1^*,i_2^*} > \mu^{(1)}_{i_1,i_2} + \mu^{(2)}_{i_1,i_2}$ for all $(i_1, i_2) \in [k_1] \times [k_2]$. Simply applying classical bandit algorithms (such as UCB) in this problem can be challenging and problematic, since it requires a centralized controller observing rewards from both agents and exploring all k_1k_2 arms, leading to high communication and computational cost.

Therefore, in this paper, we consider *weakly coupled systems*, which have a special arm-reward structure such that only minimal communication between agents is needed — in particular, no reward/action information is required to be shared — and that fewer arm pairs are necessarily explored to locate the best collective arm. Before formally define the condition regarding weak coupling, we introduce several notations as follows: Denote $i_1^*(j)$ as the best arm for Agent 1 when Agent 2 plays arm j for any $j \in [k_2]$, i.e., $i_1^*(j) = \arg\max_{i'} \mu_{i',j}^{(1)}$. Similarly, let $i_2^*(j) = \arg\max_{i'} \mu_{j,i'}^{(2)}$ for any $j \in [k_1]$. Let $c_1(i) = \sum_{j\in[k_2]} \mathbb{1}_{\{i=i_1^*(j)\}}$ (i.e., the number of Agent 2 choices, aka "environments", resulting in arm i being the best arm for Agent 1) and similarly, $c_2(i) = \sum_{j\in[k_1]} \mathbb{1}_{\{i=i_2^*(j)\}}$.

We call a system weakly coupled if it satisfies the following *majority condition*:

Assumption 8 (Majority Condition)**.** *Suppose there exist an arm pair* $(i_1^{\mathsf{M}}, i_2^{\mathsf{M}}) \in [k_1] \times [k_2]$ *such that* $c_1(i_1^{\mathsf{M}}) \geq (1+\gamma)k_2/2$*, and* $c_2(i_2^{\mathsf{M}}) \geq (1+\gamma)k_1/2$ *for some* $0 < \gamma \leq 1$*. Furthermore, assume that* $\mu_{i_1^{\mathsf{M}},i_2^{\mathsf{M}}}^{(1)} > \mu_{i_1,i_2}^{(1)}$ *and* $\mu_{i_1^{\mathsf{M}},i_2^{\mathsf{M}}}^{(2)} > \mu_{i_1,i_2}^{(2)}$ *for any* $(i_1, i_2) \in [k_1] \times [k_2]$.

Arm i_1^{M} of Agent 1 is the best choice for him for majority of Agent 2's selections, and analogously for i_2^{M} (an illustration is given in Figure 4.1.). We call $i_1^{\mathsf{M}}, i_2^{\mathsf{M}}$ the *majority arms* of both agents (hence the notation). Clearly, the majority arm pair is the optimal, i.e., (i_1^*, i_2^*), if the condition holds.

To understand the intuition of this condition, first consider the case when there is no coupling, i.e, $\mu_{i_1,i_2}^{(1)}$ is a constant for any $i_2 \in [k_2]$ when fixing i_1 (similar for $\mu_{i_1,i_2}^{(2)}$ when fixing i_2). The majority condition holds with $\gamma = 1$. In this case, each agent can locate the best arm solely based on the observed rewards itself.

With more coupling, the mean rewards observed by one agent are no longer constant as the other agent changes arms — however, in a weakly-coupled system, we assume the change of arm at the other agent will not affect the

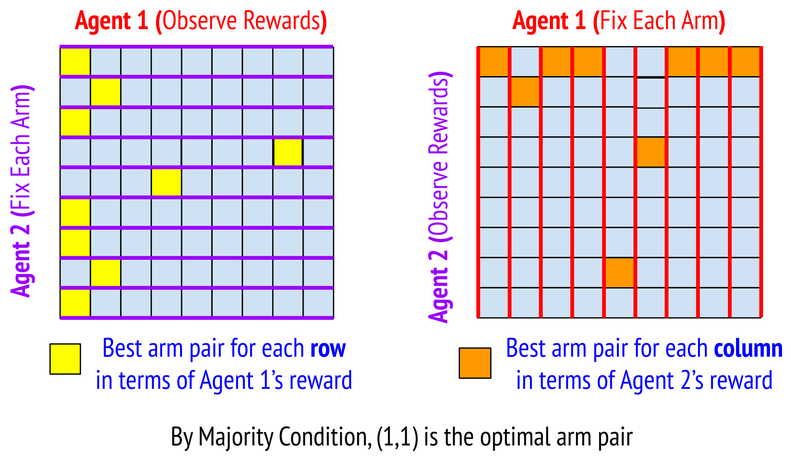

Agent 1 (Observe Rewards)
Agent 2 (Fix Each Arm)
Best arm pair for each row in terms of Agent 1's reward
Agent 1 (Fix Each Arm)
Agent 2 (Observe Rewards)
Best arm pair for each column in terms of Agent 2's reward
By Majority Condition, (1,1) is the optimal arm pair

Figure 4.1: An illustration of the majority condition.

best arm *majority of time*. In other words, only a small number of actions by the other agent make a significantly negative impact on the best arm (actually a stronger condition would be that only a few arm pairs lead to significant reward degradation, but we focus on the best arm exclusively). As we will see, with more robustness this arm-reward structure still preserves the property that local reward feedback is sufficient for the best arm identification of each agent.

Remark 14 (Weakly Coupled Wireless Systems)**.** *Weak coupling can be found in several wireless settings. Two examples that we consider in this paper are: (i) channel selection across multiple base stations, with coupling due to interference leakage across adjacent channels, and (ii) scheduler selection at multiple base stations, with coupling due to the out-of-cell interference resulting from the transmission patterns induced by the chosen scheduler. We study both these settings in Section 4.4, where we discuss the nature of coupling, as well as the* *efficiency benefits of our approach.*

4.2.2 An Alternative Condition, Regret

Condition 8 naturally captures the weak-coupling nature of some applications. In Condition 8, both agents are "symmetric". Here, we introduce a non-symmetric, weaker notion as follows.

Assumption 9 (Weaker Majority Condition)**.** *Suppose there exist an arm i_1^{M} such that $c_1(i_1^{\mathsf{M}}) \geq (1+\gamma)k_2/2$. Furthermore, assume that $\mu^{(1)}_{i_1^{\mathsf{M}}, i_2^*(i_1^{\mathsf{M}})} + \mu^{(2)}_{i_1^{\mathsf{M}}, i_2^*(i_1^{\mathsf{M}})} \geq \mu^{(1)}_{i_1^*, i_2^*} + \mu^{(2)}_{i_1^*, i_2^*}$.*

Note that Condition 8 strictly implies Condition 9 since i_2^{M} must be $i_2^*(i_1^{\mathsf{M}})$ under Condition 8 — therefore, it is better to adopt a more general notion. Consider the channel selection example: with some small probability, the majority arms of both agents might happen to "collide" with each other (i.e., being adjacent channels). Then it is preferred to aim at finding the arm pair $(i_1^{\mathsf{M}}, i_2^*(i_1^{\mathsf{M}}))$ rather than $(i_1^{\mathsf{M}}, i_2^{\mathsf{M}})$ when we design an algorithm so as to avoid the collision (when there is no collision, Condition 9 becomes Condition 8). In practice, even when Condition 9 is not held, the pair $(i_1^{\mathsf{M}}, i_2^*(i_1^{\mathsf{M}}))$ still gives acceptable "near-optimal" rewards for similar settings which involve collision avoiding.

The goal is to develop an efficient and communication-light bandit algorithm to minimize regret. We define the regret as the loss of rewards with respect to the arm pair $(i_1^{\mathsf{M}}, i_2^*(i_1^{\mathsf{M}}))$ in accordance with Condition 2.[2] Let $(I_1(t), I_2(t))$ denote the arm pair selected by the two users at time t. The regret with horizon T is defined as

$$\mathsf{Regret}_T = \mathbb{E}\left[\sum_{t=1}^{T}(\mu^{(1)}_{i_1^{\mathsf{M}}, i_2^*(i_1^{\mathsf{M}})} - \mu^{(1)}_{I_1(t), I_2(t)}) + \sum_{t=1}^{T}(\mu^{(2)}_{i_1^{\mathsf{M}}, i_2^*(i_1^{\mathsf{M}})} - \mu^{(2)}_{I_1(t), I_2(t)})\right].$$

When Condition 2 holds, the regret expression above reduces to the normal definition (i.e., with respect to the best pair (i_1^*, i_2^*)).

4.2.3 Generalization to Multi-Agent Systems

The model described above can be generalized to systems with more than 2 agents. For notation simplicity we consider a 3-agent system in this subsection. Let $i_1^*(\cdot, i_2, i_3) \in [k_1]$ be the best arm for Agent 1 when Agent 2 and Agent 3 play $i_2 \in [k_2]$ and $i_3 \in [k_3]$ respectively. (Arm $i_2^*(i_1, \cdot, i_3)$ and arm $i_3^*(i_1, i_2, \cdot)$ are analogously defined.) The majority condition is stated as follows:

Assumption 10 (Majority Condition: 3-Agent System)**.** *Suppose there exist an arm $i_1^{\mathsf{M}} \in [k_1]$ such that*

$$\sum_{(i_2, i_3) \in [k_2] \times [k_3]} \mathbb{1}_{\{i_1^{\mathsf{M}} = i_1^*(\cdot, i_2, i_3)\}} \geq (1+\gamma)(k_2 k_3)/2,$$

and an arm $i_2^{\mathsf{M}, i_1 = i_1^{\mathsf{M}}} \in [k_2]$ such that

$$\sum_{i_3 \in [k_3]} \mathbb{1}_{\{i_2^{\mathsf{M}, i_1 = i_1^{\mathsf{M}}} = i_2^*(i_1^{\mathsf{M}}, \cdot, i_3)\}} \geq (1+\gamma)k_3/2.$$

[2]Indeed, with slight modification our algorithm can minimize a regret that is defined with respect to $(i_1^{\mathsf{M}}, i_2^{\mathsf{M}})$. We use the current definition in accordance with Condition 9 for the benefit previously discussed.

Furthermore, assume that $(i_1^{\mathsf{M}}, i_2^{\mathsf{M}, i_1 = i_1^{\mathsf{M}}}, i_3^*(i_1^{\mathsf{M}}, i_2^{\mathsf{M}, i_1 = i_1^{\mathsf{M}}}, \cdot))$ *is the best collective arm in terms of sum (mean) rewards.*

Note that we follow the non-symmetric pattern of the alternative condition in Section 4.2.2. Accordingly, the regret is defined as the loss with respect to the triplet $(i_1^{\mathsf{M}}, i_2^{\mathsf{M}, i_1 = i_1^{\mathsf{M}}}, i_3^*(i_1^{\mathsf{M}}, i_2^{\mathsf{M}, i_1 = i_1^{\mathsf{M}}}, \cdot))$.

4.3 Algorithm Design and Analysis

4.3.1 Building Block: Track-and-Stop

Our algorithm applies the *Track-and-Stop (T-a-S)* algorithm [3] as subroutines to locate the best arm in each environment. Track-and-Stop is a single-agent bandit algorithm for the purpose of *best arm identification* — the goal is to learn the best arm with a fixed confidence δ using the least number of plays. The T-a-S agent explores arms and collects feedback until a certain criterion is met, and outputs a "recommended" arm such that it is the best arm *w.p.* $1 - \delta$. In each round, the agent computes the "optimal proportion" of arms needed for exploration based on observed mean rewards[3], and chooses the arm which better matches ("tracks") the proportion.

Compared to the "pure exploration" approach (i.e., exploring the arms in a round-robin fashion), a T-a-S agent spends more effort on exploring arms with better reward feedback, which is significantly more efficient. Indeed, for

[3]For example, if an arm shows much worse reward feedback than others after some initial exploration, the proportion assigned to this arm should be lower.

some structured bandit environments, it has been shown that Track-and-Stop is asymptotically optimal in terms of the number of explorations needed for the fixed-confidence best arm identification problem. We present the following result (taken from [27]) which will be used in our regret analysis.

Let $\mathcal{E}_k$ be the set of k-armed Gaussian bandit environments. For any $\nu \in \mathcal{E}_k$, denote ν_i as the reward distribution of arm $i \in [k]$ (which is normally distributed) and μ_i as its mean. We denote $\mathcal{E}_{k,\mathsf{alt}(\nu)}$ as the set of bandits whose best arms are different from the one in ν, i.e., $\mathcal{E}_{k,\mathsf{alt}(\nu)} = \{\nu' \in \mathcal{E}_k : i^*(\nu) \cap i^*(\nu') = \phi\}$ where $i^*(\nu) = \arg\max_{i \in [k]} \mu_i(\nu)$.

Lemma 4 ([27], Theorem 33.6)**.** *For any bandit environment $\nu \in \mathcal{E}$, the stopping time of a Track-and-Stop instance with a confidence parameter δ, $\tau(\delta;\nu)$, satisfies that*

$$\lim_{\delta \to 0} \frac{\mathbb{E}[\tau(\delta;\nu)]}{\log(1/\delta)} = \rho^*(\nu) := \sup_{\alpha \in \mathcal{P}_{k-1}} \left(\inf_{\nu' \in \mathcal{E}_{k,\mathsf{alt}(\nu)}} \left(\sum_{i=1}^{k} \alpha_i d(\nu_i, \nu_i') \right) \right)$$

where $\mathcal{P}_{k-1}$ is the $(k-1)$-probability simplex and $d(\cdot,\cdot)$ denotes the Kullback-Leibler divergence of two distributions.[4]

Note that the value $\rho^*(\nu)$ is the asymptotic lower bound.

4.3.2 Algorithm for Weakly Coupled Systems

In this section, we introduce the main result — a decentralized bandit algorithm for weakly coupled systems using Track-and-Stop as a building block.

[4]A similar result on exponential family bandits is given in [3] (Theorem 14).

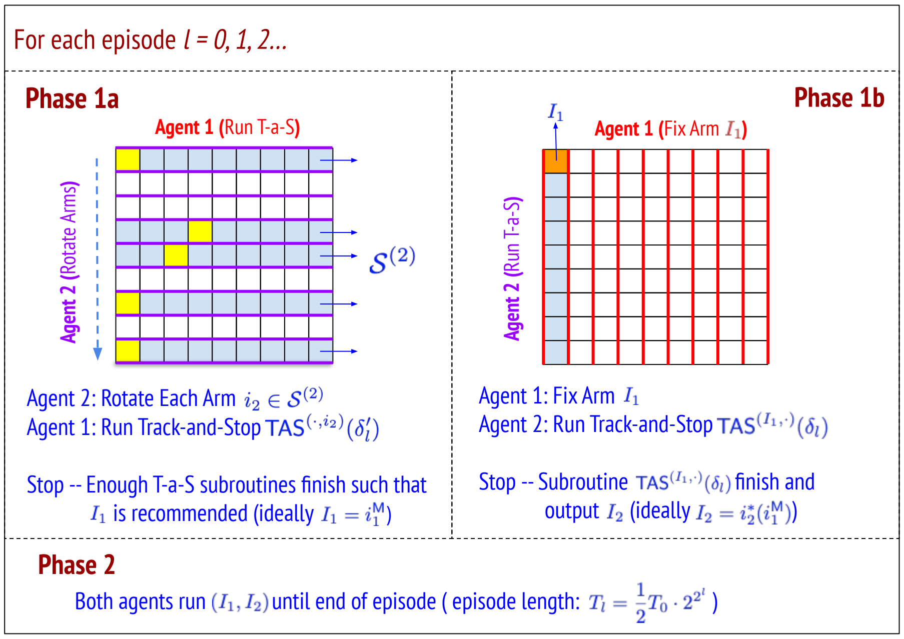

For each episode $l = 0, 1, 2...$
Phase 1a
Agent 1 (Run T-a-S)
Agent 2 (Rotate Arms)
$\mathcal{S}^{(2)}$
Agent 2: Rotate Each Arm $i_2 \in \mathcal{S}^{(2)}$
Agent 1: Run Track-and-Stop $\mathsf{TAS}^{(\cdot, i_2)}(\delta_l')$
Stop -- Enough T-a-S subroutines finish such that I_1 is recommended (ideally $I_1 = i_1^{\mathsf{M}}$)
Phase 1b
I_1
Agent 1 (Fix Arm I_1)
Agent 2 (Run T-a-S)
Agent 1: Fix Arm I_1
Agent 2: Run Track-and-Stop $\mathsf{TAS}^{(I_1, \cdot)}(\delta_l)$
Stop -- Subroutine $\mathsf{TAS}^{(I_1, \cdot)}(\delta_l)$ finish and output I_2 (ideally $I_2 = i_2^*(i_1^{\mathsf{M}})$)
Phase 2
Both agents run (I_1, I_2) until end of episode (episode length: $T_l = \frac{1}{2} T_0 \cdot 2^{2^l}$)

Figure 4.2: An illustration of the main algorithm.

As we will see, our algorithm exploits three properties: (1) *Local greedy property,* where there is no sample sharing across agents and decision-making is based on local majority; (2) *Avoiding hard environments,* where the T-a-S algorithm is initially deployed on a larger set of environments, but is stopped early on those environments that are hard (meaning the gap between the means of the best and second-best arms is small), and (3) *Sub-sampling property,* where only a limited set of environments are ever explored by any agent. The complete algorithm is presented in Algorithm 6, and an illustrative figure is exhibited in Figure 4.2.

Let $\mathsf{TAS}^{(i,\cdot)}(\delta)$ denote a sub-routine as follows: Agent 1 plays arm i repeatedly; Agent 2 implements T-a-S with respect to the confidence parameter

δ based on her own feedback. The sub-routine $\mathsf{TAS}^{(\cdot,i)}(\delta)$ is defined analogously. Before implementation, let Agent 2 randomly choose a sample set of arms $\mathcal{S}^{(2)} \subset [k_2]$ such that $|\mathcal{S}^{(2)}| = s(k_2)$, where $s(k_2)$ is a global constant which is known to both users. By choosing each of the arms in $\mathcal{S}^{(2)}$, Agent 2 will generate $s(k_2)$ environments for Agent 1 where Agent 1 can learn its "majority best" arm i_1^{M}, i.e., which maximizes its local rewards in a majority of the environments. Sampling is important when k_2 is large — we will discuss its impact in the analysis section later.

The algorithm proceeds by episodes. Each episode l lasts $T_l := \frac{1}{2}T_0 \cdot 2^{2^l}$ rounds (arm pulls), and is split into two phases: the exploration phase which consists of phase (1a) and (1b), and the exploitation phase (phase (2)).

In phase (1a), Agent 2 selects $i_2 \in \mathcal{S}^{(2)}$ in a round-robin fashion while Agent 1 runs Track-and-Stop instances (with respect to corresponding Agent 2's arms) under a confidence parameter $\delta_l' = h(\delta_l)$ where $\delta_l := 2\delta_0 \cdot (\frac{1}{2})^{2^l}$. The definition of h will be discussed later in Lemma 5. Once $\mathsf{TAS}^{(\cdot,j)}(\delta_l')$ stops (i.e., Agent 1 outputs an arm recommendation $D^{(\cdot,j)}$), Agent 1 will inform Agent 2 to skip choosing j in the following rounds. Phase (1a) stops when 1) Agent 1 learns $D^{(\cdot,j)}$ for all $j \in \mathcal{S}^{(2)}$ or when 2) more than $(1-\gamma)|\mathcal{S}^{(1)}|/2$ Track-and-Stop instances output the same (non-ϕ) arm recommendation. Note that this latter step corresponds to *avoiding hard environments* that we discussed earlier. Phase (1a) ends with Agent 1 choosing an arm I_1 which is most frequently recommended (and ideally i_1^{M}). In Phase (1b), Agent 1 chooses I_1 while Agent 2 runs subroutine $\mathsf{TAS}^{(I_1,\cdot)}(\delta_l)$ until Agent 2 outputs a recommended arm I_2

(ideally $i_2^*(i_1^{\mathsf{M}})$).

Finally, in phase (2), Agent 1 and Agent 2 select I_1 and I_2 respectively for the remaining time slots in episode l. Note that there is possibility that phase (1a) or (1b) is not finished by the end of episode l — in this case, we start a new episode nevertheless. In practice, this scenario could be avoided by properly choosing parameters T_0 and δ_0.

Note that the constant γ is pre-selected as a hyper-parameter to reflect the degree of coupling of the system — the less coupling there is (as one assumes), the larger γ can be set, and the less exploration is needed. In an extreme case, when $\gamma = 1$ (i.e., i_1^{M} is the best with respect to any arm choice of Agent 2), only one sample is needed in $[k_2]$ for the exploration of arm I_1 in phase (1a).

Remark 15 (Communication Cost)**.** *In this algorithm, communication occurs when one agent signals the other the end of a Track-and-Stop instance or the end of phase (1a) or (1b), and no other information requires exchange. Furthermore, in phase (1a) and (1b), the agent who implements the Track-and-Stop instance does not need to know which arm the other agent selects since the other agent chooses arms in a round-robin manner — the only knowledge needed is $s(k_2)$ for Agent 1. (In Algorithm 6, for notation simplicity we use $D^{(\cdot,j)}$ as Agent 1's local variables to denote the outputs of Track-and-Stop subroutines, although the exact indices j are not needed.)*

Remark 16 (Non-Weakly Coupled Systems)**.** *When the system is not weakly-coupled, the recommended arms (I_1, I_2) can be suboptimal — in some settings,*

Algorithm 6 Decentralized Bandit for Weakly Coupled Systems

```
**Initialization**: Agent 2 randomly select $\mathcal{S}^{(2)} \subset [k_2]$, such that $|\mathcal{S}^{(2)}| = s(k_2)$.
for $l = 0, 1, 2, \cdots$ do
    Global clock $t \leftarrow 1$
    $T_l \leftarrow \frac{1}{2} T_0 \cdot 2^{2^l}, \delta_l \leftarrow 2\delta_0 \cdot (\frac{1}{2})^{2^l}, \delta'_l \leftarrow h(\delta_l)$
    [Phase 1a]
    [Agent 1] Set local variables: $D^{(\cdot,j)} \leftarrow \phi$ for all $j \in \mathcal{S}^{(2)}$
    [Agent 2] Set local variable: $i_2 \leftarrow$ lowest index in $\mathcal{S}^{(2)}$
    while NOT phase_1a_stop AND $t \le T_l := \frac{1}{2} T_0 \cdot 2^{2^l}$ do
        Proceed $\mathsf{TAS}^{(\cdot,i_2)}(\delta'_l)$ by one time slot
        if $\mathsf{TAS}^{(\cdot,i_2)}(\delta'_l)$ stops (observed by Agent 1) then
            [Agent 1] $D^{(\cdot,i_2)} \leftarrow$ Output of $\mathsf{TAS}^{(\cdot,i_2)}(\delta'_l)$
            Agent 1 informs Agent 2 that $D^{(\cdot,i_2)} \neq \phi$
        [Agent 2] $i_2 \leftarrow$ the next arm (in a round-robin fashion) in $\mathcal{S}^{(2)}$ where $D^{(\cdot,i_2)} = \phi$
        $t \leftarrow t + 1$
    [Agent 1] $I_1 \leftarrow \mathsf{Mode}((D^{(\cdot,i_2)})_{i_2 \in \mathcal{S}^{(2)}})$
    [Phase 1b]
    [Agent 2] Set local variable: $D^{(I_1,\cdot)}$
    while NOT phase_1b_stop AND $t \le T_l := \frac{1}{2} T_0 \cdot 2^{2^l}$ do
        Proceed $\mathsf{TAS}^{(I_1,\cdot)}(\delta_l)$ by one time slot
        if $\mathsf{TAS}^{(I_1,\cdot)}(\delta_l)$ stops (Observed by Agent 2) then
            [Agent 2] $D^{(I_1,\cdot)} \leftarrow$ Output of $\mathsf{TAS}^{(I_1,\cdot)}(\delta_l)$
            Agent 2 informs Agent 1 that $D^{(I_1,\cdot)} \neq \phi$
        $t \leftarrow t + 1$
    [Agent 2] $I_2 \leftarrow D^{(I_1,\cdot)}$
    [Phase 2]
    Agent 1 and Agent 2 choose $(I_1, I_2)$ repeatedly until $t = T_l$

Definition (Phase Stopping Criteria):
phase_1a_stop = $\{\exists j \in \mathcal{S}^{(1)}$ such that $\sum_{i_2 \in \mathcal{S}^{(2)}} \mathbb{1}\{D^{(\cdot,i_2)} = j\} > (1 - \gamma)\frac{|\mathcal{S}^{(2)}|}{2}\}$ or $\{D^{(\cdot,i_2)} \neq \phi, \forall i_2 \in \mathcal{S}^{(2)}\}$.
phase_1b_stop = $\{D^{(I_1,\cdot)} \neq \phi\}$.
```

"greedy choices" may have a negative impact on each other. When this happens (e.g., the rewards observed in phase (2) are much smaller than expected), one solution is for both agents to switch to a pre-agreed arm pair or a centralized bandit algorithm. For instance, an Explore-Then-Commit (ETC) approach is a reasonable centralized algorithm — all of the $k_1 \times k_2$ arm pairs are selected in a round-robin fashion for a fixed length of time, and the best arm pair (after exchanging the information regarding mean rewards) is used in the exploitation phase.

Remark 17 (Extension to 3-Agent Systems)**.** *This algorithm can be easily extended to systems with more than 2 agents. For example, when there are are 3 agents, phase (1) is split into 3 sub-phases: in phase (1a), Agent 2 and Agent 3 first select a subset of arms in $[k_2] \times [k_3]$ and rotate arms accordingly, while Agent 1 runs Track-and-Stop subroutines to identify the majority arm I_1 (ideally i_1^{M}); in phase (1b), Agent 1 fixes his arm choice I_1, while Agent 2 and Agent 3 follow the procedure as in the original phase (1a) of Algorithm 6 such that Agent 2 learns I_2 (ideally $i_2^{\mathsf{M}, i_1 = i_1^{\mathsf{M}}}$ as defined in Section 4.2.3); finally, in phase (1c), Agent 3 learns the recommended arm I_3 when Agent 1 and Agent 2 play (I_1, I_2). The exploitation phase remains the same.*

4.3.3 Regret Analysis

In this section, we present the regret analysis of our main algorithm. For simplicity, we assume the distribution of rewards (for each arm) observed by each agent is Gaussian, such that the theoretical guarantee of Track-and-Stop can be applied.

4.3.3.1 Soundness of Phase (1)

The first result states the soundness of the exploration phase, i.e., the best collective arm is identified with high probability. We will focus on the soundness of phase (1a) since the result for phase (1b) is straightforward (as only one Track-and-Stop instance is involved).

Lemma 5. *Assume that* $\sum_{j\in\mathcal{S}^{(2)}} \mathbb{1}_{\{i_1^{\mathsf{M}}=i_1^*(j)\}} \geq \frac{s(k_2)}{2}$*, i.e., the sample set* $\mathcal{S}^{(2)}$ *preserves the majority condition. Let* $\tau_l^{(1a)}$ *denote the stopping time of phase (1a) in episode* l *(and suppose episode* l *can run indefinitely). It satisfies that*

$$\mathbb{P}(\{\tau_l^{(1a)} < \infty\} \cap \{I_1 = i_1^{\mathsf{M}}\}) \geq 1 - \delta_l$$

provided that

$$h^{-1}(\delta) = 1 - \sum_{n=\lfloor(1-\gamma)s(k_1)/2\rfloor+1}^{\lceil s(k_2)/2\rceil} \binom{\lceil s(k_2)/2\rceil}{n} (1-\delta)^n \delta^{\lceil s(k_2)/2\rceil - n}.$$

Proof. The intuition is as follows: Let Event A be "more than $(1-\gamma)s(k_2)/2$ Track-and-Stop instances will *eventually* choose Arm i_1^{M} if running indefinitely". Let Event B be "the majority-stopping criterion `phase_1a_stop` is reached, and Arm i_1^{M} is chosen when `phase_1a_stop` is reached". Clearly we have that A implies B. Thus, it suffices to compute the error probability of event A to get the error bound for Event B.

Let $D_j[t]$ be the random variable denoting the decision ("output") for instance $\mathsf{TAS}^{(\cdot,j)}(\delta_l')$ at time t. Thus, $D_j[t] \in \{\phi\} \cup [k_1]$. Let $\tilde{\mathcal{S}}^{(1)} \subset \mathcal{S}^{(1)}$ be the set of "good" arms for Agent 2 such that $i_1^{\mathsf{M}} = i_1^*(j)$ for all $j \in \tilde{\mathbb{S}}^{(2)}$ (by assumption, $|\tilde{\mathbb{S}}^{(2)}| \geq \lceil s(k_2)/2 \rceil \geq \lfloor (1-\gamma)s(k_2)/2 \rfloor + 1$). Therefore, by the soundness of Track-and-Stop, for any $j \in \tilde{\mathbb{S}}^{(2)}$, $\lim_{t\to\infty} \mathbb{1}_{\{D_j[t]=i_1^{\mathsf{M}}\}} = Y_j$, *a.s.* where Y_j is a $\mathsf{Bern}(1-\delta_l')$ random variable. Furthermore, since $\{D_i[t]\}_{t\geq 0}$ and $\{D_j[t]\}_{t\geq 0}$ are independent from each other for any $i \neq j$, we have that Y_i are independent for all $i \in \tilde{\mathbb{S}}^{(2)}$.

Let $N[t] = \sum_{i\in\mathbb{S}^{(2)}} \mathbb{1}_{\{D_i[t]=i_1^{\mathsf{M}}\}}$ and $\tilde{N}[t] = \sum_{i\in\tilde{\mathbb{S}}^{(2)}} \mathbb{1}_{\{D_i[t]=i_1^{\mathsf{M}}\}}$. Note that phase (1a) is good if and only if there exists $t > 0$ such that $N[t] > (1-\gamma)s(k_2)/2$. Now observe that,

$$\begin{aligned}
\{\forall t, N[t] \leq (1-\gamma)s(k_2)/2\} &\implies \{\lim_{t\to\infty} N[t] \leq (1-\gamma)s(k_2)/2\} \\
&\implies \{\lim_{t\to\infty} \tilde{N}[t] \leq (1-\gamma)s(k_2)/2\}.
\end{aligned}$$

Note that $\lim_{t\to\infty} N[t]$ and $\lim_{t\to\infty} \tilde{N}[t]$ both exist due to monotonicity and $N[t] \geq \tilde{N}[t]$ for all $t \geq 0$.

Now observe that $\lim_{t\to\infty} \tilde{N}[t] = \sum_{i\in\tilde{\mathbb{S}}^{(2)}} Y_i$. Therefore, we have that

$$\begin{aligned}
&\mathbb{P}(\{\tau_l^{(1a)} = \infty\} \cup \{I_1 \neq i_1^{\mathsf{M}}\}) \leq \mathbb{P}(\lim_{t\to\infty} \tilde{N}[t] \leq (1-\gamma)s(k_2)/2) \\
&\quad \leq 1 - \sum_{n=\lfloor(1-\gamma)s(k_2)/2\rfloor+1}^{|\tilde{\mathbb{S}}^{(2)}|} \binom{|\tilde{\mathbb{S}}^{(2)}|}{n} (1-\delta_l')^n \delta_l'^{|\tilde{\mathbb{S}}^{(2)}|-n} \\
&\quad \leq h^{-1}(\delta_l') = \delta_l.
\end{aligned}$$

The last inequality holds true for any possible $\tilde{\mathbb{S}}^{(2)}$, considering $|\tilde{\mathbb{S}}^{(2)}| \geq s(k_2)/2$ by assumption. $\square$

4.3.3.2 Regret

To compute the cumulative regret, let us first give a bound on the expected length of the exploration phase. Note that for any fixed arm $j \in \mathcal{S}^{(2)}$, Agent 1 operates a k_1-armed bandit in instance $\mathsf{TAS}^{(\cdot,j)}(\delta'_l)$. Let $\nu^{(\cdot,j)} \in \mathcal{E}_{k_1}$ denote the corresponding environment in which Agent 1 plays. We have the following result.

Lemma 6. *Rank the elements in $\mathcal{S}^{(2)}$ as $(j_1, j_2, \cdots, j_{|\mathcal{S}^{(2)}|})$, such that $\rho^*(\nu^{(\cdot,j_1)}) \leq \rho^*(\nu^{(\cdot,j_2)}) \leq \cdots \leq \rho^*(\nu^{(\cdot,j_{|\mathcal{S}^{(2)}|})})$. For any $\epsilon > 0$, there exists δ_0 such that for all $l \geq 0$,*

$$\mathbb{E}[\tau_l^{(1a)}] \leq (1+\epsilon)(\log \frac{1}{\delta_0} + 2^l \log 2)\mathsf{C}^{(1a)}$$

where

$$\mathsf{C}^{(1a)} = \left(\sum_{m=1}^{\tilde{s}} \rho^*(\nu^{(\cdot,j_m)}) + (s(k_2) - \tilde{s}) \cdot \rho^*(\nu^{(\cdot,j_{\tilde{s}})}) \right)$$

and $\tilde{s} = \lfloor (1-\gamma)\frac{s(k_2)}{2} \rfloor + 1$.

Remind that $\delta_l = 2\delta_0 \cdot (\frac{1}{2})^{2^l}$. The above is a direct result from Lemma 4 and the phase stopping criterion. Analogously, we can derive a similar (and simpler) result for phase (1b) using another constant $\mathsf{C}^{(1b)}$, which can be defined as $\mathsf{C}^{(1b)} = \max_{j \in [k_1]} \rho^*(\nu^{(j,\cdot)})$.

We have the following regret bound for the main algorithm.

Theorem 3. *Provided that $\sum_{j \in \mathcal{S}^{(2)}} \mathbb{1}_{\{i_1^{\mathsf{M}} = i_1^*(j)\}} \geq s(k_2)/2$, we have that for any $\epsilon > 0$, there exists δ_0 such that*

$$\mathsf{Regret}_T \leq 4(1+\epsilon)\Delta_{\max}(\mathsf{C}^{(1a)} + \mathsf{C}^{(1b)}) \cdot \max(\log((T/T_0) \cdot 2), 1) + o(\log T)$$

where $\Delta_{\max} = \max_{(i_1,i_2)} \left((\mu^{(1)}_{i_1^*,i_2^*} + \mu^{(2)}_{i_1^*,i_2^*}) - (\mu^{(1)}_{i_1,i_2} + \mu^{(2)}_{i_1,i_2}) \right)$, *and the constants* $\mathsf{C}^{(1a)}$ *and* $\mathsf{C}^{(1b)}$ *are defined as in Lemma 6.*

Proof. Let $l(T)$ be the index of the episode at the horizon T. By observation we have that

$$\begin{aligned} & l(T) = 0 \text{ or } (T_0/2) \cdot 2^{2^{l(T)-1}} \leq T \\ \implies & l(T) \leq \max(0, \log_2 \log_2((T/T_0) \cdot 2)) + 1. \end{aligned}$$

The regret can be split into two parts: the loss of rewards due to the exploration in phase (1a) and (1b), and the loss of rewards due to the "wrong" recommendation in phase (2). Therefore, using the bound on $l(T)$, we have that

$$\begin{aligned} \mathsf{Regret}_T &\leq \sum_{l=0}^{l(T)} (\mathbb{E}[\tau_l^{(1a)} + \tau_l^{(1b)}]\Delta_{\max} + (2\delta_l) \cdot T_l) \\ &\leq \sum_{l=0}^{l(T)} \left((1+\epsilon)(\mathsf{C}^{(1a)} + \mathsf{C}^{(1b)})\Delta_{\max}(\log \frac{1}{\delta_0} + 2^l \log 2) + 2\delta_0 T_0 \right) \\ &\leq 4(1+\epsilon)\Delta_{\max}(\mathsf{C}^{(1a)} + \mathsf{C}^{(1b)}) \cdot \max(\log(\frac{2T}{T_0}), 1) + o(\log T). \end{aligned}$$

Note that the first inequality applies the soundness guarantee given in Lemma 5.

□

4.3.3.3 Sampling

Note that the constant $\mathsf{C}^{(1a)}$ in the above theorem scales as $O(s(k_2))$. Thus, when $\mathcal{S}^{(2)} = [k_2]$ (no sampling), we have that $\mathsf{Regret}_T = O(k_1 k_2 \log T)$.

When the number of arms is large, sampling is typically needed to reduce the computational cost. Using standard concentration techniques which bound the probability of the event $\{\sum_{j\in \mathcal{S}^{(2)}} \mathbb{1}_{\{i_1^{\mathsf{M}}=i_1^*(j)\}} \geq s(k_2)/2\}$, we have the following corollary.

Corollary 2. *When Condition 9 holds, there exist* $\beta_2 > 0$ *such that when the sample size* $s(k_2) = \beta_2 \log k_2$*, the regret satisfies that* $\mathsf{Regret}_T = O(k_1 \log k_2 \log T)$ *with probability* $1/k_2$ *for sufficiently large* k_2*.*

Corollary 2 can be extended to systems with more than two agents. Suppose there are m agents, each with k arms. Using the procedure discussed in Remark 17, and letting the size of sample sets in phase (1) scaling as $O(\log k)$, we have that the regret scales as $O((m-1)k \log k \log T)$ *w.p.* $O(1/k)$. Note that the term $(m-1)$ stems from the number of sub-phases needed in phase (1).

4.4 Performance Evaluation

In this section, we discuss the potential applications of our main algorithm and evaluate its performance. We consider two wireless scenarios: channel selection and best scheduler selection.

4.4.1 Multi-Channel Selection

4.4.1.1 A Two-AP Example

We first consider an application example in the wireless channel selection problem. Suppose two access points (APs) are located within a close range, each serving a nearby mobile user. The APs decide amongst a set of channels (frequency bands) which one to use to serve the respective users. Due to some environment effects (such as shadowing or exogenous interference), the best channels are unknown to the APs. Furthermore, the decisions of each AP will interfere with the rewards received by the other due to *channel leakage*. However, since the leakage mainly affects a small number of adjacent channels, it is reasonable to believe that the majority condition holds, and one can apply our algorithm to locate the best channel at each AP.

Experiment Settings: Suppose each AP chooses among (the same) n frequency bands which are indexed $1, 2, \cdots, n$ for each time slot (which lasts 0.5 ms). We set $n = 13$. The Signal-to-Interference ratio (SIR) at time slot t is modeled as $\mathtt{SIR}[t] = P_a g[t]/\mathtt{I}[t]$ where P_a is the transmit power of the AP, $g[t]$ denotes the channel gain at the user and $\mathtt{I}[t]$ denotes the interference level. We set $P_a = 23$ dBm for both APs. The channel gain is determined through the path loss, fading properties (Rayleigh fast fading) and a channel-dependent shadowing gain. We assume the channel-dependent gains (in dB) are drawn from Gaussian distribution $\mathcal{N}(0, 6)$ and constant through the simulation.

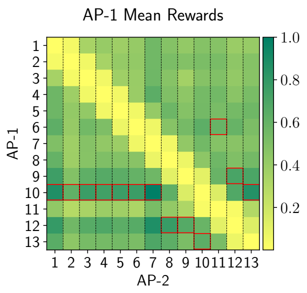
Assume that the interference $\mathtt{I}[t]$ is exclusively caused by channel leakage from the nearby AP. We adopt the following simplified power leakage model for AP s ($s = 1, 2$): when channel $i \in [n]$ is chosen, the relative power leakage (in dB) in channel $j \in [n]$ equals $\min(0, \max(-\beta_s|i - j|, -\varphi_s) + \gamma_{i,j})$. This reflects the nature of common channel leakage, i.e., adjacent channels experience significantly higher interference (subject to channel-dependent "noise" $\gamma_{i,j}$). In our simulation, we set $\beta_s = 33$ dB, $\varphi_s = 90$ dB and $\gamma_{i,j}$ is randomly chosen from Gaussian distribution $\mathcal{N}(0, 4)$. Note that we do not assume the agents (APs) have any prior knowledge of the leakage model, and our algorithm can easily address more complicated models (e.g., with APs using different sets of channels or abnormal non-adjacent channel leakage). Furthermore, we assume each mobile user is closer to its corresponding AP, and the relative gain due to path loss is 20 dB for both users.

Suppose the users have infinitely backlogged queues and the rewards received by each user equal the number of packets transmitted, i.e., the instantaneous service rate. For any time slot t, the rate at any user i is given by

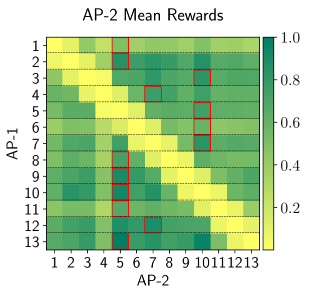
$$S_i[t] = \mathtt{BW} \times \log_2(1 + 10^{0.1(\mathtt{SIR}_i[t] - \mathtt{L})}) \quad \text{bps} \tag{4.1}$$

where $\mathtt{BW}$ is set to be 20 MHz and the parameter $\mathtt{L} = 3$dB describes a loss to Shannon capacity.

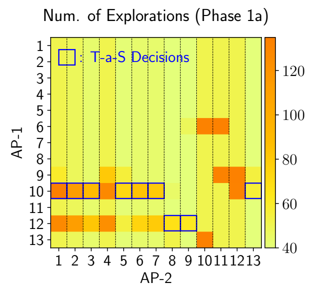
Results: We first run Monte-Carlo simulations to compute mean rewards received by both APs under different (i_1, i_2) pairs. Figure 4.3a and 4.3b show a typical realization of channel rewards (note that the results vary by simulations due to randomness), where the adjacent channel leakage negatively affects the rewards in the diagonal squares. In Figure 4.3a, we use red boxes to denote the best AP-1 arm under each AP-2 arm choice i_2, and AP-1 arm 10 is the majority arm i_1^{M} (under this simulation). Similarly, AP-2 arm 5 is shown as the majority arm i_2^{M} by Figure 4.3b (which is also $i_2^*(i_1^{\mathsf{M}})$).



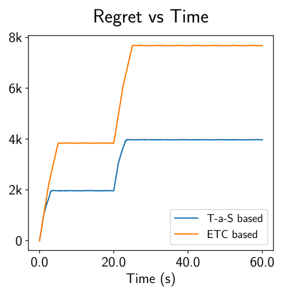
(a) AP-1 Mean Rewards



(b) AP-2 Mean Rewards



(c) Number of Explorations for Each Arm Pair in Algorithm Phase 1a



(d) Regret vs Time

Figure 4.3: Simulation results on experiments in Section 4.4.1.1.

We then run the main algorithm to find the best collective arm. For simplicity, we apply a computationally-efficient version of Track-and-Stop, assuming the rewards are normally distributed (see [3], Section 2). In particular, we are interested in the TAS sub-routines in phase (1a) of the algorithm. Figure 4.3c shows the number of explorations for each arm pair in phase (1a) in the first episode. As expected, for each sub-routine $\mathsf{TAS}^{(\cdot,i_2)}(\delta_l')$ (corresponding to each column), typically only the top two arms are heavily explored to determine the best one — this shows a major advantage of Track-and-Stop compared to a naive round-robin exploration. In addition, we use blue boxes to denote decisions made by each sub-routine. The larger is the reward gap between the top two arms, the less exploration is required (e.g., when $i_2 = 13$). Note that phase (1a) stops once the stopping criterion has been met, and not all of the sub-routines are needed to output a recommendation.

Finally, in Figure 4.3d, we exhibit the cumulative regret over time, which grows logarithmically with an episodic behavior. We compare our algorithm to the classical *explore-and-commit (ETC)* algorithm, which utilizes round-robin exploration. Our algorithm exhibits a much-improved regret in the exploration phase, suggesting that the majority-based algorithm with Track-and-Stop subroutines better exploits the structure of the system.

4.4.1.2 A Three-AP Example

In this experiment, we extend our multi-channel selection example to a 3-AP setting. We follow the channel leakage model introduced in the previous section, and set $\beta_s = 33, 39, 45$ dB for $s = 1, 2, 3$ respectively. The relative path loss gain ranges from 20 to 40 dB among different pairs of users. Other parameters remain the same.

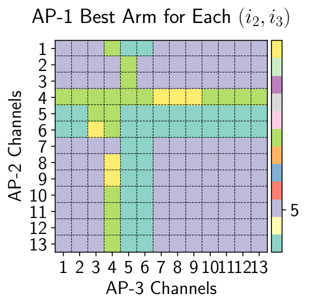
Figure 4.4a shows the best arm for AP-1 when each (i_2, i_3) pair is selected (for one realization of the channel model). For the simulation we present here, the majority arm $i_1^{\mathsf{M}} = 5$. Figure 4.4b further exhibits the reward gap between the majority arm and the second-best in each environment (we set the gap as 0 when the majority arm 5 is not the best arm in that environment).

When implementing phase (1a) of the algorithm, we sample 20 (i_2, i_3) pairs (otherwise, the number of sub-routines needed significantly grows with more APs). The total number of explorations of each sampled sub-routine in the first episode is shown in Figure 4.4c, with orange boxes denoting the sub-routines that output arm recommendations before phase (1a) finishes — as expected, "easy" sub-routines with larger reward gap complete faster. A regret plot is presented in Figure 4.4d for completeness. The regret is computed with respect to the best collective action $(5, 11, 2)$.

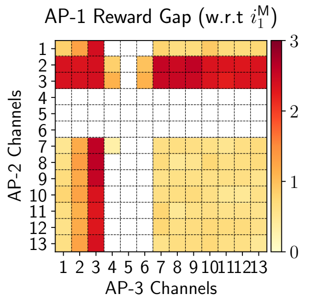
4.4.2 Best Scheduler Selection

In this section, we explore a potential application of our algorithm for best scheduler selection in wireless queueing systems, which is first proposed as the "meta-scheduling" problem in [4, 5] (i.e., Chapter 2 and Chapter 3). Wireless scheduling with queues is a challenging task — many schedulers are developed (e.g., MaxWeight, Log Rule, Exp Rule, etc.) for specific settings/goals, however, there lacks a systematic approach to find a good scheduler across diverse performance metrics and deployment scenarios. A "meta-scheduler" is introduced as a multi-armed bandit framework which selects the best scheduler from a set of predefined policies through users' feedback evaluating the performance. This is a flexible model which allows complicated and user-customized reward schemes to be considered.

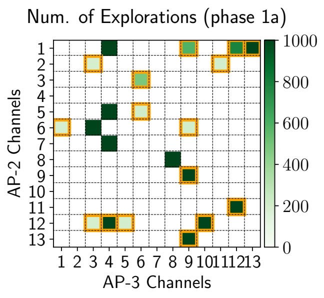
The algorithm proposed in Chapter 2 is designed for single-agent scenarios. When there are multiple nearby base stations, reward feedback at each



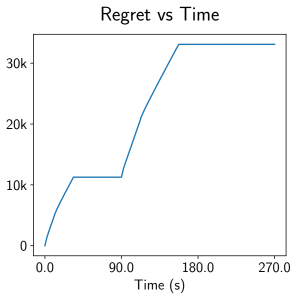
(a) AP-1's Best Arm for Each (i_2, i_3) Pair Selected by AP-2 and AP-3



(b) Reward Gap between i_1^{M} and Second Best for Each (i_2, i_3) Pair



(c) Phase (1a) Exploration Heatmap



(d) Regret vs Time

Figure 4.4: Simulation results on experiments in Section 4.4.1.2.

agent is coupled with decisions from other agents due to signal interference. Furthermore, different scheduler combinations might lead to heterogeneous interfering behaviors. Therefore, it can be problematic to run a single-agent bandit algorithm individually at each station without effective coordination.

Under reasonable interference levels and typical reward schemes, we believe the systems are weakly coupled. The intuition is that effective scheduling policies tend to schedule users opportunistically (i.e., making use of good channels to improve transmission efficiency), and as a byproduct incur less interference to other agents (since less power/time is needed to transmit the same users' packet flows). Therefore, the majority condition should hold if the candidate set consists of mostly "good" schedulers. In the following, we will set up a simple downlink scheduling system and showcase the usage of our algorithm.

Experiment Settings: Suppose there exists 2 base stations (BS-1 and BS-2), each serving 4 downlink users. For each base station, we follow a packet transmission model used in [4]: The instantaneous SINR of user i at time t equals $P_b g_i[t]/(\sigma^2 + I_i[t])$, where the transmit power of BS P_b is set to be 47 dBm and the noise level $\sigma^2 = -104$. The channel gain $g_i[t]$ is a combination of path loss and Rayleigh fast fading, and the path loss (in dB) is computed as $39.1 \log_{10}(\texttt{dist}) + 13.5 + 20 \log_{10}(f_c)$ where $f_c = 2.0$ GHz and $\texttt{dist}$ denotes the user distance. The interference level $I_i[t]$ is a result of packet transmission of the nearby base station, and $I_i[t] = 0$ when the other base station is idle. The instantaneous service rate of each user is computed according to (4.1) with $\texttt{BW} = 10$ MHz. Each time slot lasts 0.5 ms and each packet has a fixed size 5 kb.

Let the 4 users served by BS-2 close to their base station (subject to small interference) with a light load — the arrival rate for each user is set as 0.3 packets/slot. For the 4 users served by BS-1, we set the arrival rate as 0.6 packets/slot and focus on two scenarios: ($\texttt{S1}$) For each user, the distance to BS-1 ($\texttt{dist}_1$) equals 150 m and the distance to BS-1 ($\texttt{dist}_2$) ranges from 300 ± 10

Table 4.1: Mean rewards observed by BS-1 and BS-2 in Scenario (S1) of Section 3.2. The best policy in each environment (row or column) is highlighted in bold font. The best collective arm is (C, C).

			BS-1 Arms (Policies)					
BS-1 Mean Rewards			A)	B)	C)	D)	E)	F)
	BS-2	A)	0.773	0.549	**0.785**	0.311	0.768	0.760
		B)	0.769	0.508	**0.781**	0.264	0.764	0.755
		C)	0.776	0.571	**0.787**	0.337	0.771	0.763
		D)	0.745	0.046	**0.757**	0.019	0.739	0.724
		E)	0.773	0.541	**0.784**	0.303	0.768	0.759
		F)	0.749	0.117	**0.761**	0.027	0.744	0.730
BS-2 Mean Rewards			BS-1 Arms (Policies)					
			A)	B)	C)	D)	E)	F)
	BS-2	A)	0.945	0.945	0.944	0.945	0.941	0.943
		B)	0.937	0.936	0.933	0.932	0.932	0.930
		C)	**0.948**	**0.952**	**0.953**	**0.952**	**0.952**	**0.951**
		D)	0.632	0.682	0.633	0.636	0.640	0.634
		E)	0.940	0.936	0.941	0.940	0.938	0.941
		F)	0.878	0.861	0.871	0.872	0.862	0.875

m. (S2) For each user, $\texttt{dist}_1 = 150$ m and $\texttt{dist}_2 \in 250 \pm 10$ m. The second scenario sees a higher interference level. Each agent chooses over 6 scheduling policies: A) MaxWeight, B) Max-Queue, C) Max-Rate, D) Round-Robin, E) Log-Rule, F) Exp-Rule, and collect reward feedback every 200 time slots (aka one "round"). Packets not transmitted at the end of each round are dropped to ensure the reward feedback are conditionally independent.[5] We define the

[5]Note that if there is no packet drop, then a "bad" non-stable policy resulting in long queues will skew the reward feedback for the next round, even if a "good" queue-stabilizing policy is chosen. Ideally, only good policies are selected after some initial exploration, and thus the impact of packet drop is minimal. A detailed discussion on this issue is given in [4], which introduces a queueing cycle-based algorithm to avoid packet drop; adapting it to our multi-agent setting is of future interest.

Table 4.2: Mean rewards observed by BS-1 and BS-2 in Scenario (S2) of Section 3.2. The best policy in each environment (row or column) is highlighted in bold font. The best collective arm is (C, C).

			BS-1 Arms (Policies)					
			A)	B)	C)	D)	E)	F)
BS-1 Mean Rewards	BS-2	A)	0.569	0.183	**0.612**	0.051	0.596	0.505
		B)	0.544	0.125	**0.587**	0.030	0.575	0.474
		C)	0.588	0.221	**0.630**	0.074	0.611	0.530
		D)	0.265	0.010	0.267	0.009	**0.372**	0.050
		E)	0.560	0.168	**0.604**	0.046	0.588	0.493
		F)	0.339	0.010	0.376	0.009	**0.417**	0.158
			BS-1 Arms (Policies)					
			A)	B)	C)	D)	E)	F)
BS-2 Mean Rewards	BS-2	A)	0.936	0.941	0.945	0.942	0.942	0.939
		B)	0.934	0.933	0.936	0.932	0.932	0.931
		C)	**0.955**	**0.953**	**0.951**	**0.951**	**0.952**	**0.952**
		D)	0.709	0.737	0.700	0.696	0.737	0.671
		E)	0.932	0.930	0.938	0.932	0.939	0.932
		F)	0.873	0.878	0.876	0.866	0.874	0.871

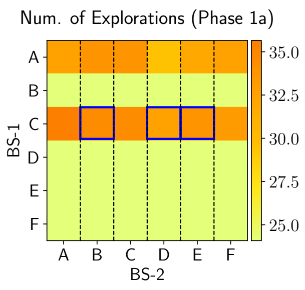
reward of each packet as $1 - \tanh(0.04 * \texttt{delay})$ and the reward feedback of one round is the sum of all packet rewards.

Results: We first compute the mean rewards observed by both base stations under different policy pairs using Monte-Carlo simulations, which is presented in Table 4.1 and 4.2 (the rewards are normalized by the episode length and packet loads). In both Scenario (S1) and Scenario (S2), the best arm for BS-2 is Max-Rate and the mean rewards do not vary much when BS-1 changes policies due to the low load and negligible interference.

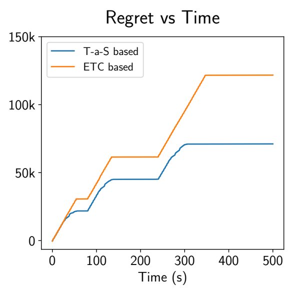
Now let us focus on the rewards observed by BS-1. In Scenario (S1), it turns out Max-Rate is the best arm of BS-1 no matter what policy BS-2 selects (due to the relatively low interference level compared to Scenario (S2)). This can be expected since in our simulation settings, all the users are almost symmetric (in terms of load and service rates) — the Max-Rate policy, which greedily serves the user with the best service rate, is proved to be efficient in minimizing packet delays for symmetric moderate-load scenarios. By contrast, in Scenario (S2), as the interference level increases (with service rates degrading), for some choices of BS-2 (Round-Robin and Exp-Rule), Max-Rate performs badly — instead, the Log-Rule policy which has a better queue-stabilizing property prevails in these cases.[6] However, the majority condition still holds for Scenario (S2), showing the robustness of our model, and our algorithm can indeed be applied to find the best collective policy.

Finally, we run the main algorithm for both scenarios. The simulation results are exhibited in Figure 4.5. For each scenario, we show the exploration heatmap for the algorithm phase (1a) in the first episode — the best policy is identified with most of the explorations focusing on good performing policies. Moreover, not all Track-and-Stop sub-routines are needed to complete, and we use blue boxes to denote finished sub-routines. As a result, our algorithm has a lower regret than ETC as shown in Figure 4.5b and Figure 4.5d.

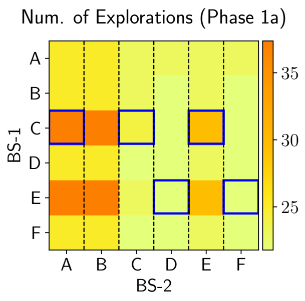
[6]To be precise, the Max-Rate policy, unlike Log-Rule or MaxWeight, is not *throughput-optimal* and has a smaller capacity region. In this setting, when BS-2 chooses Round-Robin or Exp-Rule which turns out incurring more interference, Max-Rate no longer stabilizes the load and tends to result in long queues, thus worsening the delay metric.



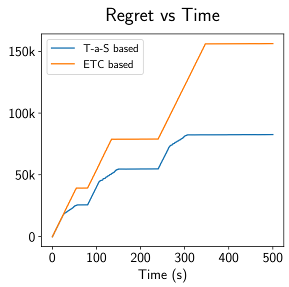
(a) S1 — Exploration Heatmap



(b) S1 — Regret vs Time



(c) S2 — Exploration Heatmap



(d) S2 — Regret vs Time

Figure 4.5: Simulation results on experiments in Section 4.4.2.

4.5 Conclusion

In this chapter we study an online learning framework for the multi-agent resource allocation problem. In particular, we focus on so-called weakly coupled systems with a special arm-reward structure — the majority condition, which states that most of the time the best arm of each agent is invariant to other agents' arm selection. When this condition holds, the optimal arms can be learned with local signals (reward feedback) with proper coordination from other agents, therefore allowing the design of less demanding algorithms compared to classical methods which simply examine all the collective actions.

Furthermore, we develop an efficient decentralized bandit algorithm with minimal communication overheads. Through simulation, we validate the usefulness of our model and algorithm in two wireless settings: channel selection among nearby APs, and best scheduling policy selection by interfering base stations. We believe weak coupling is a reasonable abstraction for several wireless applications, and it is of great interest to explore its benefits in other related settings.

Chapter 5

Conclusion

The main focus of this dissertation is exploring the applications of multi-armed bandit algorithms in wireless systems so as to improve the adaptability and robustness of wireless scheduling/resource allocation.

We first studied the multi-user scheduling problem for the wireless downlink with instantaneous channel rate and queue information. We introduced the notion of “meta-scheduling” which formulates the selection of a good wireless scheduler as a bandit problem, then propose a UCB-type bandit algorithm that adapts to the queueing dynamics (e.g., dealing with the randomness of cycles). Then, we extended the meta-scheduling idea and studied a model of hierarchical scheduling in the context of network slicing, in which the base station learns the optimal option among infinitely many arms. We formulated the problem as a blackbox optimization and addressed it with CHOOC, an HOO-type bandit algorithm adaptive to random queueing cycles. Lastly, we moved to a multi-agent setting, where neighboring learning agents' decisions are coupled with each other through interference. We identified a low-complexity structure — the weakly-coupled system — and proposed a decentralized bandit algorithm to learn the best collective actions. For each segment, we both provided rigorous theoretical proof to show that the proposed algorithm gives a desired sub-linear regret compared to a genie and demonstrated the effectiveness of the algorithm through a series of experiments using simulation.

The results show that through online learning with bandit feedback, agent(s) are capable of adjusting their strategies/policies to improve the overall performance at a manageable cost, which supports our main statement:

Wireless resource allocation problems can be approached by multi-armed bandit algorithms to systematically address the complexity of systems and improve the robustness to changing environments.

In the following sections, we provide a summary of our contributions and discuss potential research directions for future work.

5.1 Summary of Contributions

Chapter 2: In this chapter, we studied a multi-armed bandit framework for multi-user scheduling in wireless downlink. We called the framework "meta-scheduling", where an overlay algorithm, the meta-scheduler, dynamically selects the best scheduler amongst a candidate set. We proposed a UCB-based meta-scheduling algorithm with a cycle interruption mechanism adapting to queueing system dynamics. Through mathematical analysis, we showed that the algorithm gives a logarithmic regret. Finally, through simulation, we exhibited the algorithm's learning ability in choosing the optimal schedulers under different traffic/reward conditions.

Chapter 3: In this chapter, we delved into an online learning-based hierarchical scheduling framework in support of network slicing applications. In this framework, the scheduler is parameterized by a weight vector used for slice-level resource allocation. Compared to the model in Chapter 2, the focus is on selecting weights across a continuum with an infinite number of options (arms), which is aligned with a blackbox optimization problem. We designed a modified HOO algorithm, CHOOC, tackling the technical challenges of random queueing cycles and clipping. The theoretical analysis of CHOOC showed a sub-linear regret at the same order of the original HOO. Finally, we conducted an empirical evaluation of the algorithm in various wireless settings to demonstrate its adaptability in handling performance tradeoffs across slices.

Chapter 4: In this chapter, we extend our work to a multi-agent bandit setting. We consider scenarios where multiple agents select from a range of scheduling options, and the performance experienced by one agent is influenced by decisions made by other agents. We focused on weak coupling, a low-complexity arm-reward structure that states that most of the time the optimal arm for each agent remains unchanged despite the arm selections by other agents. Accordingly, we proposed a decentralized and efficient bandit algorithm, using Track-and-Stop as a building block. We validated the efficacy of our model and algorithm through both theoretical regret analysis and empirical studies.

5.2 Future Directions

In this section, we discuss some ideas for future research that could potentially be explored.

5.2.1 Empirical Study in Practical Settings

In this dissertation, we explored various applications of multi-armed bandit algorithms in wireless/queueing system settings. The simulation results showed that the online learning approach improves the adaptability and robustness of a scheduling agent to complicated environments. However, the empirical studies were conducted using synthetic data emulating queueing/channel dynamics of a wireless system based on simplified assumptions. It would be interesting to see how the algorithms perform in more practical settings. With real-world data, it can be imagined that some of the assumptions may not be always fulfilled (e.g., the *i.i.d.* assumptions of queueing cycles and rewards), and some practical workarounds might be needed – this can lead to additional algorithmic/theoretical interest as well. For example, if a system observes periodic fluctuation of traffic, it could be helpful to set up heuristic rules to cluster the traffic "contexts" based on side information (e.g., number of users within a cell), and run separate bandit instances for each context to possibly accelerate convergence and improve the overall performance — the design and corresponding benefits of such practices depend on the traffic pattern and are better to be validated using practical data.

5.2.2 Extension to Multi-Agent Settings

In Chapter 4, we formulated a multi-agent bandit setting where bandits were coupled and focused on a low complexity structure — weak coupling under the majority condition. It would be interesting to explore other structures akin to such coupled multi-agent settings. One possible direction could be exploiting the structure of the coupling topology. Recall that in Chapter 4, when there are more than two agents, we simply consider a complete graph to describe the mutual coupling effects (i.e., each agent can be influenced by any other agent in the system). In practice, however, one agent may only be coupled with neighboring agents, and agents are arranged under some simple geographic patterns (e.g., grids). There is potential to leverage this type of structure to design efficient bandit algorithms so as to learn the optimal collective action.

5.2.3 Meta-Scheduling in Non-Stationary Settings

In this dissertation, we typically assume the stochastic processes describing the system (e.g., rewards, traffic flows) to be stationary within the running time of the algorithm. When the environment drastically changes, an easy practical fix could be re-running the algorithm as discussed in Chapter 2. Indeed, there could be significant practical/theoretical interest in studying meta-scheduling in a non-stationary setting. We could possibly leverage ideas in the literature of *non-stationary bandits* (e.g., using a sliding window for recent feedback when computing empirical averages, see Chapter 31 in [27]) to design new meta-scheduling algorithms, which are capable of "cleverly" capturing the change of environments as well as adapt to wireless/queuing system settings.

Appendices

Appendix A

Appendix for Chapter 2

A.1 Derivation of Equation 2.18

When $\beta > (1+\gamma)(\mu_{\max}+\nu^2/\alpha)$ and $\kappa > 4\alpha$, one can show that

$$\begin{aligned}
&\mathbb{E}[U_1^{(*)}] - \frac{1}{s}\sum_{i=1}^{s}\mathbb{E}[\hat{U}_i^{(*,f_i)}] \\
\le& \frac{1}{s}\sum_{i=1}^{s}\sum_{l=\lceil f_i \rceil}^{\infty}\mathbb{E}[U^{(*)}(1)|C^{(*)}(1)=l]\mathbb{P}(C^{(*)}(1)=l) \\
\le& \frac{1}{s}\sum_{i=1}^{s}\sum_{l=\lceil f_i \rceil}^{\infty} r_{\max}\, l\, \mathbb{P}(C^{(*)}(1)=l) \\
\le& \frac{1}{s}\sum_{i=1}^{s} r_{\max}\left(f_i\mathbb{P}(C^{(*)}(1)\ge f_i) + \sum_{l=\lceil f_i \rceil}^{\infty}\mathbb{P}(C^{(*)}(1)\ge l)\right) \\
\le& \frac{1}{s}\sum_{i=1}^{s}\frac{r_{\max}}{i^{\kappa/2\alpha}}(\beta+2\alpha+\kappa\log i+1)e^{-\beta\gamma/2(1+\gamma)\alpha} && \text{(A.1)} \\
\le& \frac{1}{s} r_{\max}\frac{\pi^2}{6}(\beta+2\alpha+\kappa\log s+1)e^{-\beta\gamma/2(1+\gamma)\alpha}. && \text{(A.2)}
\end{aligned}$$

Equation (A.1) gives the term $\epsilon'_{n,s}$ in (2.18), which is further bounded by (A.2). Hence, $\epsilon'_{n,s} \sim O(\log s/s)$.

A.2 Proof of Lemma 1

Proof. We apply a technique commonly used in classical UCB proofs [27]: Let $H^{(k)}(s)$ denote the cycle index when the arm k is first selected s times, i.e.

$$H^{(k)}(s) = \min\{n : T_n^{(k)} = s\}. \tag{A.3}$$

If there exists a positive non-decreasing function $\mathsf{g}^{(k)}(n)$ and an event $\mathcal{E}_n$ for each n such that

$$\{A_n = k\} \subset \mathcal{E}_n \cup \{T_{n-1}^{(k)} < \mathsf{g}^{(k)}(n)\}, \tag{A.4}$$

then we have that

$$\begin{aligned}\mathbb{E}[T_n^{(k)}] =& \mathbb{E}\big[\sum_{m=1}^{n} \mathbb{1}_{\{A_m=k\}}\big]\\ =& \mathbb{E}\big[\sum_{m=1}^{n} \mathbb{1}_{\{m\le H^{(k)}(\mathsf{g}^{(k)}(n))\}}\mathbb{1}_{\{A_m=k\}}\big]\\ &+ \mathbb{E}\big[\sum_{m=1}^{n} \mathbb{1}_{\{m> H^{(k)}(\mathsf{g}^{(k)}(n))\}}\mathbb{1}_{\{A_m=k\}}\big]\\ \le& \mathsf{g}^{(k)}(n) + \mathbb{E}\big[\sum_{m=n_0+1}^{n} \mathbb{1}_{\mathcal{E}_m}\mathbb{1}_{\{A_m=k\}}\big],\end{aligned} \tag{A.5}$$

where $n_0 := \min(H^{(k)}(\mathsf{g}^{(k)}(n)), n-1)$. Note that the last inequality holds since

$$\begin{aligned} m > H^{(k)}(\mathsf{g}^{(k)}(n)) &\implies T_{m-1}^{(k)} \ge \mathsf{g}^{(k)}(n) \ge \mathsf{g}^{(k)}(m)\\ &\implies \{A_m = k\} \subset \mathcal{E}_m \cap \{A_m = k\}.\end{aligned}$$

Therefore, for any $k \neq k^*$, it suffices to find a function $\mathsf{g}^{(k)}(n)$ and some $\mathcal{E}_n$ such that (A.4) holds and both $\mathsf{g}^{(k)}(n)$ and $\mathbb{E}[\sum_{m=n_0+1}^{n} \mathbb{1}_{\mathcal{E}_m}\mathbb{1}_{\{A_m=k\}}]$ are sub-linear (or even finitely bounded). In plain language, this means finding a suitable $\mathsf{g}^{(k)}(n)$ such that when a suboptimal arm k has been played sufficiently many times (larger than $\mathsf{g}^{(k)}(n)$), the probability of selecting k for n-th cycle is negligible.

The lemma is proved for $k \in \mathcal{A}^u(\boldsymbol{\lambda})$ and $k \in \mathcal{A}^s(\boldsymbol{\lambda})\backslash\{k^*\}$ respectively.

Unstable Arms: Recall that $p_k = \mathbb{P}(C_1^{(k)}{=}\infty)$ and $p_k > 0$ for unstable arms. Let $Y_s^{(k)} = \mathbb{1}_{\{C_s^{(k)} > f_s\}}$. Observe that if $A_n = k \in \mathcal{A}^u(\boldsymbol{\lambda})$, one of the following events must occur for $n > |\mathcal{A}|$ (excluding the initialization round):

$$\mathcal{E}_{1,n} = \{\sum_{i=1}^{T_{n-1}^{(k)}} Y_i^{(k)} - p_k T_{n-1}^{(k)} \leq -\sqrt{2T_{n-1}^{(k)} \log n}\}$$
$$\mathcal{E}_{2,n} = \{p_k T_{n-1}^{(k)} < 2\sqrt{2T_{n-1}^{(k)} \log n} + \frac{\pi^2}{6}\}$$

Otherwise, $-\sum_{i=1}^{T_{n-1}^{(k)}} Y_i^{(k)} + \frac{\pi^2}{6} + \sqrt{2T_{n-1}^{(k)} \log n} > 0$ and the stability indicator will eliminate arm k for this decision.

If $\mathcal{E}_{2,n}$ is true, $T_{n-1}^{(k)} < (18/{p_k}^2)\log n$. The probability of $\mathcal{E}_{1,n}$ is bounded by Bernstein's inequality. Note that

$$\mathbb{P}\left(\sum_{i=1}^{s} Y_i^{(k)} - p_k s < -\sqrt{2s\log n}\right) < \frac{1}{n^4} \quad \forall 1 \leq s \leq n.$$

The value $\mathbb{P}(\mathcal{E}_{2,n})$ is bounded by taking a union bound over all possible s. Therefore, applying the reasoning in (A.5), for all $k \in \mathcal{A}^u(\boldsymbol{\lambda})$,

$$\begin{aligned}\mathbb{E}[T_n^{(k)}] &\leq 1 + \lceil \frac{18}{{p_k}^2} \log n \rceil + \sum_{m=|\mathcal{A}|+1}^{n} \mathbb{E}[\mathbb{1}_{\mathcal{E}_{1,m}} \mathbb{1}_{\{A_m = k\}}] \\ &\leq 1 + \lceil \frac{18}{{p_k}^2} \log n \rceil + \sum_{m=1}^{\infty} \sum_{s=1}^{m} \frac{1}{m^4} \leq \lceil \frac{18}{{p_k}^2} \log n \rceil + \frac{\pi^2}{6} + 1.\end{aligned}$$

The "plus 1" term corresponds to the first exploration in the initialization round.

Stable Arms: For all $n > |\mathcal{A}|$, if $A_n = k \in \mathcal{A}^s(\boldsymbol{\lambda}) \setminus \{k^*\}$, one of the following events must happen:

$$\begin{aligned}
\mathcal{E}_{3,n} &= \{-\sum_{i=1}^{T_{n-1}^{(*)}} Y_i^{(*)} + \frac{\pi^2}{6} + \sqrt{2T_{n-1}^{(*)} \log n} \le 0\},\\
\mathcal{E}_{4,n} &= \{\hat{R}^{(k)}(n-1) - \bar{R}^{(k)}(n-1) > \frac{d^{(k)}}{4}\},\\
\mathcal{E}_{5,n} &= \{\bar{R}^{(k)}(n-1) \ge r^{(k)} + \Delta'(\hat{\epsilon}_n^{(k)})\},\\
\mathcal{E}_{6,n} &= \{\hat{R}^{(*)}(n-1) \le r^{(*)} - \Delta(\epsilon_n^{(*)}, \epsilon_n'^{(*)})\},\\
\mathcal{E}_{7,n} &= \{\Delta'(\hat{\epsilon}_n^{(k)}) + \Delta(\epsilon_n^{(k)}, \epsilon_n'^{(k)}) > \frac{3d^{(k)}}{4}\}.
\end{aligned}$$

The variables $\bar{R}^{(k)}(n)$ and $\Delta'(\hat{\epsilon}_n^{(k)})$ will be defined later. We argue by contradiction. Assume that the five events are all false, then we have

$$\begin{gathered}
-\sum_{i=1}^{T_{n-1}^{(*)}} Y_i^{(*)} + \frac{\pi^2}{6} + \sqrt{2T_{n-1}^{(*)} \log n} > 0, \quad \text{and}\\
\hat{R}^{(k)}(n-1) + \Delta(\epsilon_n^{(k)}, \epsilon_n'^{(k)}) < \hat{R}^{(*)}(n-1) + \Delta(\epsilon_n^{(*)}, \epsilon_n'^{(*)}),
\end{gathered}$$

and thus we reach a contradiction that $A_n \ne k$.

Next, we transform the event $\bigcup_{j=3}^{7} \mathcal{E}_{j,m}$ into the form exhibited in (A.4) to investigate bounds on $\mathbb{E}[T_n^{(k)}]$.

$\boldsymbol{\mathcal{E}_{3,n}}$ and $\boldsymbol{\mathcal{E}_{6,n}}$: These two events correspond to the design of the stability indicator and the exploration bonus respectively. By McDiarmid's inequality, we observe that

$$\mathbb{P}\left(\sum_{i=1}^{s} Y_i^{(*)} - \frac{\pi^2}{6} \geq \sqrt{2s\log n}\right) < \frac{1}{n^4}, \quad \forall s \leq n.$$

By the concentration property stated in Assumption 2 and Eqs. (2.17)–(2.19), we have that

$$\begin{aligned}
&\mathbb{P}\left(\hat{R}_s^{(*)} + \Delta(\epsilon_{n,s}, \epsilon'_{n,s}) \leq r^{(*)}\right) \\
&\leq \mathbb{P}\left(\frac{1}{s}\sum_{i=1}^{s} \hat{C}_i^{(*,f_i)} > \mathbb{E}[C_1^{(*)}] + \epsilon_{n,s}\right) + \\
&\qquad \mathbb{P}\left(\frac{1}{s}\sum_{i=1}^{s} \hat{U}_i^{(*,f_i)} \leq \mathbb{E}[U_1^{(*)}] - \epsilon'_{n,s} - \epsilon_{n,s}\right) \\
&\leq \frac{2}{n^3}, \quad \forall s \leq n.
\end{aligned}$$

Note that the term $(1/s)\sum_{i=1}^{s}\hat{U}_i^{(*,f_i)}$ is $(\nu^2/s, \alpha/s)$-sub-exponential by the technical assumption in item (3) of Assumption 3.

Therefore, we conclude that

$$\begin{aligned}
&\sum_{m=|\mathcal{A}|+1}^{n} \sum_{j=\{3,6\}} \mathbb{E}[\mathbb{1}_{\mathcal{E}_{j,m}} \mathbb{1}_{\{A_m=k\}}] \\
&\leq \sum_{m=|\mathcal{A}|+1}^{n} \left(\mathbb{P}\left(\mathcal{E}_{3,n}\right) + \mathbb{P}\left(\mathcal{E}_{6,n}\right)\right) \\
&\leq \sum_{m=1}^{n}\sum_{s=1}^{m}\left(\frac{1}{m^3} + \frac{2}{m^3}\right) \leq \frac{\pi^2}{2}.
\end{aligned} \tag{A.6}$$

$\boldsymbol{\mathcal{E}_{4,n}}$: Now we define the "non-truncated" empirical rate of arm k after s samples as $\bar{R}_s^{(k)}$, where for all $s \geq 1$,

$$\bar{R}_s^{(k)} = \frac{\sum_{i=1}^{s} U_i^{(k)}}{\sum_{i=1}^{s} C_i^{(k)}}.$$

Therefore, $\mathcal{E}_{4,n}$ describes the event that the observed average $\hat{R}^{(k)}(n)$ is significantly higher than the non-truncated empirical average $\bar{R}^{(k)}(n)$, which is possible if a substantial number of arm k's samples are interrupted. We will show that this will not occur *w.h.p.* after the sub-optimal arm k is played a sufficient number of times.

Observe that, for any $s \leq n$

$$\begin{aligned}
&\mathbb{P}\left(\hat{R}_s^{(k)} - \bar{R}_s^{(k)} > \frac{d^{(k)}}{4}\right) \\
\leq&\mathbb{P}\left(\frac{\sum_{i=1}^{s} U_i^{(k)}}{\sum_{i=1}^{s} C_i^{(k)}} \cdot \frac{\left(\sum_{i=1}^{s} C_i^{(k)} - \sum_{i=1}^{s} \hat{C}_i^{(k,f_i)}\right)}{\sum_{i=1}^{s} \hat{C}_i^{(k,f_i)}} > \frac{d^{(k)}}{4}\right) \\
\leq&\mathbb{P}\left(\bar{r} \cdot \frac{\sum_{i=1}^{s} C_i^{(k)} \mathbb{1}_{\{C_i^{(k)} > f_i\}}}{s} > \frac{d^{(k)}}{4}\right).
\end{aligned}$$

First consider the case when $k \in \mathcal{A}_0$. Then $\kappa/\alpha_k = \kappa/\alpha > 4$ by the algorithm setting. Therefore, by Assumption 2 and 3, we have that $\mathbb{E}[\sum_{i=1}^{s} \mathbb{1}_{\{C_i^{(k)} > f_i\}}] \leq \pi^2/6$. Let $\mathsf{M}_1^{(k)}$ be a constant such that

$$\bar{r} \cdot \frac{(\mathbb{E}[C_1^{(k)}] + 6\alpha_k \log \mathsf{M}_1^{(k)})(\frac{\pi^2}{6} + \sqrt{\mathsf{M}_1^{(k)} \log \mathsf{M}_1^{(k)}})}{\mathsf{M}_1^{(k)}} \leq \frac{d^{(k)}}{4}.$$

Then we apply proof by contrapositive to see that

$$\mathcal{E}_{4,n} \subset \mathcal{E}_{4,n,(1)} \cup \mathcal{E}_{4,n,(2)} \cup \mathcal{E}_{4,n,(3)}$$

where

$$\mathcal{E}_{4,n,(1)} = \{T_{n-1}^{(k)} < \mathsf{M}_1^{(k)}\},$$

$$\mathcal{E}_{4,n,(2)} = \bigcup_{i=1}^{T_{n-1}^{(k)}} \{C_i^{(k)} > \mathbb{E}[C_1^{(k)}] + 6\alpha_k \log T_{n-1}^{(k)}\},$$

$$\mathcal{E}_{4,n,(3)} = \{\sum_{i=1}^{T_{n-1}^{(k)}} \mathbb{1}_{\{C_i^{(k)} > f_i\}} > \frac{\pi^2}{6} + \sqrt{T_{n-1}^{(k)} \log T_{n-1}^{(k)}}\}.$$

We can derive that

$$\begin{aligned}
&\sum_{m=|\mathcal{A}|+1}^{n} \sum_{j=\{2,3\}} \mathbb{E}[\mathbb{1}_{\mathcal{E}_{4,m,(j)}} \mathbb{1}_{\{A_m=k\}}] \\
&\quad \leq \sum_{s=1}^{\infty} \mathbb{P}(\bigcup_{i=1}^{s} \{C_i^{(k)} > \mathbb{E}[C_1^{(k)}] + 6\alpha_k \log s) \\
&\qquad + \sum_{s=1}^{\infty} \mathbb{P}(\sum_{i=1}^{s} \mathbb{1}_{\{C_i^{(k)} > f_i\}} > \frac{\pi^2}{6} + \sqrt{s \log s}) \\
&\quad \leq \sum_{s=1}^{\infty} (\sum_{i=1}^{s} \frac{1}{s^3}) + \frac{1}{s^2} \leq \frac{\pi^2}{3}
\end{aligned} \tag{A.7}$$

In the first inequality we move from the "global" cycle index m to the "local" index s (denoting the number of cycles selecting arm k), since $\mathcal{E}_{4,m,(2)}$ and $\mathcal{E}_{4,m,(3)}$ only depend on the number of samples $T_{m-1}^{(k)}$ (rather than m). The second inequality is given by Assumption 2 and McDiarmid's inequality respectively.

When $k \notin \mathcal{A}_0$, however, the value $\sum_{i=1}^{s} \mathbb{1}_{\{C_i^{(k)} > f_i\}}$ does not satisfies the concentration property, since it is possible that $\alpha_k > \alpha$ and $\kappa/\alpha_k < 4$. That means this stable arm k can still be interrupted relatively frequently due to our underestimating of the distribution's tails.

We note that due to the design of the stability indicator, if $A_n = k$, there is a natural bound on $\sum_{i=1}^{T_{n-1}^{(k)}} \mathbb{1}_{\{C_i^{(k)} > f_i\}}$ *w.h.p.* given by,

$$\sum_{i=1}^{T_{n-1}^{(k)}} \mathbb{1}_{\{C_i^{(k)} > f_i\}} \leq \pi^2/6 + \sqrt{2T_{n-1}^{(k)} \log n}, \tag{A.8}$$

i.e., the stability indicator of arm k at n-th decision is true. If this bound does not hold, then the indicator of every arm must be all false (including the best arm) such that the random tie-breaker has a chance to select k, which implies that $\mathcal{E}_{3,n}$ must happen.

Now let $\mathsf{J}_1^{(k)}$ be a constant such that for any $s \geq (\mathsf{J}_1^{(k)} \log^{1+\delta} n \vee \chi)$ for some $\delta > 0$ and $\chi > 1$,

$$\bar{r} \cdot \frac{(\mathbb{E}[C_1^{(k)}] + 6\alpha_k \log s)(\frac{\pi^2}{6} + \sqrt{2s \log n} + 1)}{s} \leq \frac{d^{(k)}}{4}.$$

One choice of $\mathsf{J}_1^{(k)}$ can be such that for any $\chi > 1$,

$$\mathsf{J}_1^{(k)} \geq \frac{4}{d^{(k)}} \bar{r} \frac{\left(\mathbb{E}[C_1^{(k)}] + 6\alpha_k \log(\mathsf{J}_1^{(k)} \log^{1+\delta} \chi)\right)(\frac{\pi^2}{6} + \sqrt{2\mathsf{J}_1^{(k)} \log^{2+\delta} \chi} + 1)}{\log^{1+\delta} \chi}.$$

Following a similar argument as in the last case, we have that

$$\mathcal{E}_{4,n} \subset \tilde{\mathcal{E}}_{4,n,(1)} \cup \mathcal{E}_{4,n,(2)} \cup \tilde{\mathcal{E}}_{4,n,(3)},$$

where

$$\tilde{\mathcal{E}}_{4,n,(1)} = \{T_{n-1}^{(k)} < \mathsf{J}_1^{(k)} \log^{1+\delta} n \vee \chi\},$$
$$\tilde{\mathcal{E}}_{4,n,(3)} = \{\sum_{i=1}^{T_{n-1}^{(k)}} \mathbb{1}_{\{C_i^{(k)} > f_i\}} > \frac{\pi^2}{6} + \sqrt{2T_{n-1}^{(k)} \log n}\}.$$

Note that $\{A_n = k\} \cap \tilde{\mathcal{E}}_{4,n,(3)} \subset \mathcal{E}_{3,n}$ as discussed earlier and we can apply the similar bound as in (A.7).

To interpret the result, we note that when the number of samples selecting arm k ($k \notin \mathcal{A}_0$) grows slightly faster than $O(\log n)$, there must exist enough samples of k that are not interrupted (due to the stability indicator), which helps to guarantee $\hat{R}_s^{(k)} \approx \bar{R}_s^{(k)}$.

$\boldsymbol{\mathcal{E}_{5,n}}$: This event describes that the empirical reward rates of sub-optimal arms show a nice concentration around $r^{(k)}$, which contributes to Assumption 2.

Recall that $\bar{\Delta}(\epsilon)$ defined in (2.15) such that $\bar{R}^{(*)}(n) \le r^{(*)} - \bar{\Delta}(\epsilon_n^{(*)})$ *w.h.p.* Similarly, define that

$$\Delta'(\epsilon) := \frac{\epsilon(1 + \mathbb{E}[U^{(k)}(1)]/\mathbb{E}[C^{(k)}(1)])}{\max(\mathbb{E}[C^{(k)}(1)] - \epsilon, 1)}$$

Then we have

$$\{\bar{R}_s^{(k)} \ge r^{(k)} + \Delta'(\epsilon)\} \subset \{\frac{1}{s}\sum_{i=1}^{s} C_i^{(k)} \le \max(\mathbb{E}[C_1^{(k)}] - \epsilon, 1)\}$$
$$\bigcup \{\frac{1}{s}\sum_{i=1}^{s} U_i^{(k)} \ge \mathbb{E}[U_1^{(k)}] + \epsilon\}.$$

Let $\hat{\epsilon}_{n,s}$ be defined as $\epsilon_{n,s}$ in (2.17) except that (α, ν^2) is replaced by (α_k, ν_k^2). And let $\hat{\epsilon}_n^{(k)} = \hat{\epsilon}_{n,T_{n-1}^{(k)}}$. Then,

$$\begin{aligned}&\mathbb{P}(\bar{R}^{(k)}(n) \ge r^{(k)} + \Delta'(\hat{\epsilon}_n^{(k)})) \\ &\qquad \le \sum_{s=1}^{n} \mathbb{P}(\bar{R}_s^{(k)} \ge r^{(k)} + \Delta'(\hat{\epsilon}_{n,s})) \le \frac{1}{n^2},\end{aligned}$$

and thus

$$\sum_{m=|\mathcal{A}|+1}^{n} \mathbb{E}[\mathbb{1}_{\mathcal{E}_{5,m}} \mathbb{1}_{\{A_m = k\}}] \le \sum_{m=|\mathcal{A}|+1}^{n} \mathbb{P}(\mathcal{E}_{5,m}) \le \frac{\pi^2}{6}. \tag{A.9}$$

$\boldsymbol{\mathcal{E}_{7,n}}$: Now we show that $\mathcal{E}_{7,n}$ is always false when $T_{n-1}^{(k)}$ is large enough. Let $\rho := \frac{\mu_{\min}}{\mu_{\min}+1}$. First observe that

$$\begin{aligned} &\{\Delta(\epsilon_n^{(k)}, \epsilon_n'^{(k)}) + \Delta'(\hat{\epsilon}_n^{(k)}) \leq \frac{3d^{(k)}}{4}\} \\ \supset\ &\underbrace{\{\frac{\epsilon_n^{(k)}(1+r_{\max})}{\mu_{\min}} \leq \frac{d^{(k)}}{2}(1-\rho)\}}_{\mathsf{E1}} \cup \underbrace{\{\frac{\epsilon_n'^{(k)}}{\mu_{\min}} \leq \frac{d^{(k)}}{4}\}}_{\mathsf{E2}} \\ &\cup \underbrace{\{\hat{\epsilon}_n^{(k)}(1+r_{\max}) \leq \frac{d^{(k)}}{2}\rho\}}_{\mathsf{E3}}. \end{aligned}$$

The first and the third set imply that

$$\mathsf{E1} \subset \{T_{n-1}^{(k)} \geq \underbrace{(\frac{24(1+r_{\max})^2\nu^2}{(d^{(k)})^2\rho^2} \vee \frac{12(1+r_{\max})\alpha}{d^{(k)}\rho})}_{\mathsf{K}_2^{(k)}} \cdot \log n\},$$

$$\mathsf{E3} \subset \{T_{n-1}^{(k)} \geq \underbrace{(\frac{24(1+r_{\max})^2\nu_k^2}{(d^{(k)})^2\rho^2} \vee \frac{12(1+r_{\max})\alpha_k}{d^{(k)}\rho})}_{\mathsf{K}_3^{(k)}} \cdot \log n\}.$$

The second set $\mathsf{E2}$ implies $T_{n-1}^{(k)} \geq \mathsf{M}_2^{(k)}$ where $\mathsf{M}_2^{(k)}$ is defined as the smallest integer such that

$$\frac{r_{\max}}{\mathsf{M}_2^{(k)}} \cdot \frac{\pi^2}{6} e^{-\beta/4\alpha}(\beta + 2\alpha + \kappa \log \mathsf{M}_2^{(k)} + 1) < \frac{d^{(k)}}{4}\mu_{\min}.$$

These three results indicate that when $\mathcal{E}_{7,n}$ occurs, $T_{n-1}^{(k)} < (\mathsf{K}_2^{(k)} \vee \mathsf{K}_3^{(k)}) \log n \vee \mathsf{M}_2^{(k)}$.

Combining the analysis of the five events, we can conclude that for $k \in \mathcal{A}_0$,

$$\mathsf{g}^{(k)}(n) = \lceil (\mathsf{K}_2^{(k)} \vee \mathsf{K}_3^{(k)}) \log n \vee \mathsf{M}_1^{(k)} \vee \mathsf{M}_2^{(k)} \rceil + 1,$$

$$\mathcal{E}_n = \mathcal{E}_{3,n} \cup \mathcal{E}_{4,n,(2)} \cup \mathcal{E}_{4,n,(3)} \cup \mathcal{E}_{5,n} \cup \mathcal{E}_{6,n}.$$

as indicated in (A.4). Note that $\mathsf{K}_2^{(k)} \vee \mathsf{K}_3^{(k)} = \mathsf{K}_2^{(k)}$ since $\nu_k^2 \leq \nu^2, \alpha_k \leq \alpha$ for $k \in \mathcal{A}_0$, and $\mathbb{E}[\sum_{m=|\mathcal{A}|+1}^{n} \mathbb{1}_{\mathcal{E}_m} \mathbb{1}_{\{A_m=k\}}]$ is bounded by π^2 (see Eqs. (A.6), (A.7) and (A.9)). The case of $k \notin \mathcal{A}_0$ follows the same reasoning. This finishes the proof.

□

A.3 Proof of Theorem 1

Proof. **Claim (1)**: Note that $N_\pi[\tau] \leq \tau$. Clearly, we have that

$$\mathbb{E}[D[\tau]] \leq \left(\mathbb{E}\Big[\sum_{k\in\mathcal{A}} \sum_{n=1}^{\tau} \mathbb{1}_{\{A_n=k\}} \mathbb{1}_{\{C^{(k)}(n) > f_{T_n^{(k)}}\}}\Big] \right) \bar{a} u f_\tau,$$

where the term in parentheses bounds the number of interruptions in total while $\bar{a}uf_\tau$ is the maximal number of packets that can occur in a cycle before time τ (since f_τ is the largest possible length of a cycle).

Note that the expected number of interruptions of arm k is either bounded by $\pi^2/6$ if $k \in \mathcal{A}_0$ or (trivially) bounded by $\mathbb{E}[T_\tau^{(k)}]$ otherwise. Indeed, a better bound can be found for $k \notin \mathcal{A}_0$. Take $k \in \mathcal{A}^u(\boldsymbol{\lambda})$ as an example, we observe that

$$\begin{aligned}
&\mathbb{E}\Big[\sum_{n=1}^{\tau} \mathbb{1}_{\{A_n=k\}} \mathbb{1}_{\{C^{(k)}(n) > f_{T_n^{(k)}}\}}\Big] \\
\leq &\sum_{n=1}^{\tau} \mathbb{E}\big[\mathbb{1}_{\{A_n=k\}} \mathbb{1}_{\{T_{n-1}^{(k)} < \mathsf{g}_1^{(k)}(\tau)\}} \mathbb{1}_{\{C^{(k)}(n) > f_{T_n^{(k)}}\}}\big]
\end{aligned}$$

$$
\begin{aligned}
&+\sum_{n=1}^{\tau}\mathbb{E}[\mathbb{1}_{\{A_n=k\}}\mathbb{1}_{\{T^{(k)}_{n-1}\geq \mathsf{g}^{(k)}_1(\tau)\}}],\\
\leq&\mathbb{E}\Big[\sum_{s=1}^{\mathsf{g}^{(k)}_1(\tau)\wedge T^{(k)}_\tau}\mathbb{1}_{\{C^{(k)}_s>f_s\}}\Big]+\sum_{n=1}^{\tau}\mathbb{E}[\mathbb{1}_{\{A_n=k\}}\mathbb{1}_{\mathcal{E}_{1,n}}],\\
\leq&\mathbb{E}\Big[\sum_{s=1}^{\mathsf{g}^{(k)}_1(\tau)\wedge T^{(k)}_\tau}\mathbb{1}_{\{C^{(k)}_s>f_s\}}\Big]+\frac{\pi^2}{6}.
\end{aligned}
$$

Recall that $\mathcal{E}_{1,n}$ is defined in the proof of Lemma 1.

Now we define for any integer random variable $N \geq |\mathcal{A}|+1$ (*a.s.*),

$$
\mathcal{E}_{3,N} = \bigcup_{n\geq|\mathcal{A}|+1}\{N=n\}\cap\mathcal{E}_{3,n}.
$$

Then events $\mathcal{E}_{3,H^{(k)}(s)}$ are well-defined[1] for any $s\geq 2$, where $H^{(k)}(s)$ is defined in (A.3). Observe that, on *anywhere but* $\mathcal{E}_{3,H^{(k)}(2\wedge \mathsf{g}^{(k)}_1(\tau)\wedge T^{(k)}_\tau)}\cap\{\mathsf{g}^{(k)}_1(\tau)\wedge T^{(k)}_\tau\geq 2\}$, we have

$$
\sum_{s=1}^{\mathsf{g}^{(k)}_1(\tau)\wedge T^{(k)}_\tau-1}\mathbb{1}_{\{C^{(k)}_s>f_s\}}\leq\frac{\pi^2}{6}+\sqrt{2\mathsf{g}^{(k)}_1(\tau)\log\tau}\quad a.s.
$$

Otherwise the $(\mathsf{g}^{(k)}_1(\tau)\wedge T^{(k)}_\tau)$-th sample will not be chosen due to the stability indicator (a similar argument is given in (A.8)). Note that $H(\mathsf{g}^{(k)}_1(\tau)\wedge T^{(k)}_\tau)\leq\tau$ (*a.s.*).

Therefore,

$$
\mathbb{E}\Big[\sum_{s=1}^{\mathsf{g}^{(k)}_1(\tau)\wedge T^{(k)}_\tau}\mathbb{1}_{\{C^{(k)}_s>f_s\}}\Big]
$$

[1]Events $\mathcal{E}_{3,H^{(k)}(1)}$ are not well-defined, since $H^{(k)}(1)\leq|\mathcal{A}|$.

$$\begin{aligned}
&\le (\frac{\pi^2}{6} + \sqrt{2\mathsf{g}_1^{(k)}(\tau)\log\tau} + 1) \\
&\qquad\qquad + \mathbb{E}\big[(\mathsf{g}_1^{(k)}(\tau) \wedge T_\tau^{(k)})\mathbb{1}_{\mathcal{E}_{3,H^{(k)}(2\wedge \mathsf{g}_1^{(k)}(\tau)\wedge T_\tau^{(k)})}}\big], \\
&\le \frac{\pi^2}{6} + \sqrt{2\mathsf{g}_1^{(k)}(\tau)\log\tau} + 2.
\end{aligned}$$

The last inequality is given by the fact that $\mathbb{P}(\mathcal{E}_{3,n}) \le 1/n^3$ (see (A.6)) and $H^{(k)}(s) \ge s$ (*a.s.*) for all s.

Finally, the claim is proved by combining all the bounds discussed above.

Claim (2): We follow a similar approach as developed in [32]. First we claim that the expected cumulative reward of π^{opt} over a time horizon τ is bounded as follows:

$$\mathbb{E}[\text{Rew}_{\pi^{\text{opt}}}[\tau]] \le r^{(*)}\tau + r^{(*)}\frac{\mathbb{E}[(C^{(*)}(1))^2]}{\mathbb{E}[C^{(*)}(1)]}.$$

To prove this claim, observe that,

$$\begin{aligned}
&\mathbb{E}[\text{Rew}_{\pi^{\text{opt}}}[\tau]] \le \mathbb{E}\big[\sum_{n=1}^{N_{\pi^{\text{opt}}}[\tau]+1} U^{(*)}(n)\big], \\
&= \mathbb{E}\big[\sum_{n=1}^{\infty} \mathbb{1}_{\{S_{n-1}^{\pi^{\text{opt}}} \le \tau\}}\mathbb{E}[U^{(*)}(n)]\big], \\
&\le r^{(*)}\mathbb{E}\big[\sum_{n=1}^{\infty} \mathbb{1}_{\{S_{n-1}^{\pi^{\text{opt}}} \le \tau\}}\mathbb{E}[C^{(*)}(n)]\big], \\
&= r^{(*)}\mathbb{E}\big[\sum_{n=1}^{N_{\pi^{\text{opt}}}[\tau]+1} C^{(*)}(n)\big], \\
&\le r^{(*)}(\tau + \frac{\mathbb{E}[(C^{(*)}(1))^2]}{\mathbb{E}[C^{(*)}(1)]}).
\end{aligned}$$

Note that the second and fourth equations hold true due to the independence of $S_{n-1}^{\pi^{\mathrm{opt}}}$ and $(C^{(*)}(n), U^{(*)}(n))$. The last line follows Lorden's inequality for overshoot.

We note that τ can be further bounded with respect to meta-policy π as follows:

$$\tau \leq \mathbb{E}\big[\sum_{n=1}^{N_\pi[\tau]} \sum_{k\in\mathcal{A}} \mathbb{1}_{\{A_n=k\}} \hat{C}^{(k, f_{T_n^{(k)}})}(n)\big] + f_\tau.$$

Thus, we have the following upper bound:

$$\begin{aligned}\mathbb{E}[\mathrm{Rew}_{\pi^{\mathrm{opt}}}[\tau]] \leq\, & r^{(*)}(\mathbb{E}\big[\sum_{n=1}^{N_\pi[\tau]} \sum_{k\in\mathcal{A}} \mathbb{1}_{\{A_n=k\}} \hat{C}^{(k, f_{T_{n-1}^{(k)}})}(n)\big]) \\ & + r^{(*)} f_\tau + r^{(*)} \frac{\mathbb{E}[(C^{(*)}(1))^2]}{\mathbb{E}[C^{(*)}(1)]}.\end{aligned}$$

The lower bound of $\mathbb{E}[\mathrm{Rew}_\pi[\tau]]$ is given by

$$\mathbb{E}[\mathrm{Rew}_\pi[\tau]] \geq \mathbb{E}\big[\sum_{n=1}^{N_\pi[\tau]} \sum_{k\in\mathcal{A}} \mathbb{1}_{\{A_n=k\}} \hat{U}^{(k, f_{T_n^{(k)}})}(n)\big].$$

Therefore, the regret is bounded by

$$\mathbb{E}[\mathrm{Rew}_{\pi^{\mathrm{opt}}}[\tau] - \mathrm{Rew}_\pi[\tau]] \leq \mathsf{T1} + \mathsf{T2} + r^{(*)} f_\tau + O(1) \qquad \text{(A.10)}$$

where

$$\begin{aligned}\mathsf{T1} &= \sum_{k\in\mathcal{A}\setminus\mathcal{A}_0} \mathbb{E}\big[\sum_{n=1}^{N_\pi[\tau]} \mathbb{1}_{\{A_n=k\}} \times (r^{(*)} \hat{C}^{(k, f_{T_n^{(k)}})}(n) - \hat{U}^{(k, f_{T_n^{(k)}})}(n))\big], \\ \mathsf{T2} &= \sum_{k\in\mathcal{A}_0} \mathbb{E}\big[\sum_{n=1}^{N_\pi[\tau]} \mathbb{1}_{\{A_n=k\}} \times (r^{(*)} \hat{C}^{(k, f_{T_n^{(k)}})}(n) - \hat{U}^{(k, f_{T_n^{(k)}})}(n))\big].\end{aligned}$$

Here the term $\mathsf{T1}$ denotes the regret induced by the arms that have frequent interruptions, which includes unstable arms and stable arms that are not in $\mathcal{A}_0$ with respect to the hyper-parameters. Observe that

$$\begin{aligned}
\mathsf{T1} &\leq \sum_{k\in\mathcal{A}\setminus\mathcal{A}_0} \mathbb{E}\big[\sum_{n=1}^{\infty} \mathbb{1}_{\{S_{n-1}^{\pi}\leq\tau\}}\mathbb{1}_{\{A_n=k\}} \times (r^{(*)}\mathbb{E}[\hat{C}^{(k,f_{T_n^{(k)}})}(n)] - \mathbb{E}[\hat{U}^{(k,f_{T_n^{(k)}})}(n)])\big], \\
&\leq \sum_{k\in\mathcal{A}\setminus\mathcal{A}_0} \mathbb{E}\big[\sum_{n=1}^{\infty} \mathbb{1}_{\{S_{n-1}^{\pi}\leq\tau\}}\mathbb{1}_{\{A_n=k\}} \times (r^{(*)} - \tilde{r}^{(k)})\mathbb{E}[\hat{C}^{(k,f_{T_n^{(k)}})}(n)]\big], \\
&\leq \sum_{k\in\mathcal{A}\setminus\mathcal{A}_0} (r^{(*)} - \tilde{r}^{(k)})\mathbb{E}[T_\tau^{(k)}]f_\tau. \qquad \text{(A.11)}
\end{aligned}$$

The first inequality holds true since S_{n-1}^{π} and A_n are independent of $\hat{C}^{(k,l)}(n), \hat{U}^{(k,l)}(n)$ for any l.

For the term $\mathsf{T2}$, which indicates the regret induced by the arms with few interruptions (i.e., arms in $\mathcal{A}_0$), we can derive the following bound.

$$\begin{aligned}
\mathsf{T2} &\leq \sum_{k\in\mathcal{A}_0}\Bigg(\mathbb{E}\big[\sum_{n=1}^{N_\pi[\tau]} \mathbb{1}_{\{A_n=k\}}(r^{(*)}C^{(k)}(n) - U^{(k)}(n))\big] \\
&\qquad\qquad + \mathbb{E}\big[\sum_{n=1}^{N_\pi[\tau]} \mathbb{1}_{\{A_n=k\}}(U^{(k)}(n) - \hat{U}^{(k,f_{T_n^{(k)}})}(n))\big]\Bigg), \\
&\leq \sum_{k\in\mathcal{A}_0}\Bigg(\mathbb{E}\big[\sum_{n=1}^{N_\pi[\tau]} \mathbb{1}_{\{A_n=k\}}(r^{(*)} - r^{(k)})\mathbb{E}[C^{(k)}(n)]\big] + \mathbb{E}\big[\sum_{i=0}^{\infty}(U_i^{(k)} - \hat{U}_i^{(k,f_i)})\big]\Bigg), \\
&\leq \sum_{k\in\mathcal{A}_0}\Bigg((r^{(*)} - r^{(k)})\mathbb{E}[T_\tau^{(k)}]\mu_{\max} + \mathbb{E}\big[\sum_{i=0}^{\infty} \mathbb{1}_{\{C_i^{(k)}>f_i\}}U_i^{(k)}\big]\Bigg), \\
&\leq \sum_{k\in\mathcal{A}_0}\Bigg((r^{(*)} - r^{(k)})\mathbb{E}[T_\tau^{(k)}]\mu_{\max} + r_{\max}\frac{\pi^2}{6}e^{-\beta/4\alpha}(f_\tau+2\alpha+1)\Bigg). \qquad \text{(A.12)}
\end{aligned}$$

Note that the value $\mathbb{E}[\sum_{i=0}^{\infty} \mathbb{1}_{\{C_i^{(k)}>f_i\}}U_i^{(k)}]$ is bounded using a similar derivation from Appendix A.1.

In conclusion, by combining Eqs. (A.10)–(A.12), we have that

$$\mathbb{E}[\mathrm{Rew}_{\pi^{\mathrm{opt}}}[\tau] - \mathrm{Rew}_{\pi}[\tau]] \leq f_\tau \sum_{k\in\mathcal{A}\setminus\mathcal{A}_0} (r^{(*)} - \tilde{r}^{(k)})\mathbb{E}[T_\tau^{(k)}]$$
$$+ \mu_{\max} \sum_{k\in\mathcal{A}_0} (r^{(*)} - r^{(k)})\mathbb{E}[T_\tau^{(k)}] + O(\log\tau).$$

Then the main claim is proved by applying Lemma 1.

□

A.4 Proof of Corollary 1

Proof. To show this result, we first present the following lemma.

Lemma 7. *Consider a queueing system with an initial state $\boldsymbol{Q}[0]$, in which the number of packet arrivals each slot is bounded. Suppose a stable policy is applied such that there exist a Lyapunov function $\phi(\cdot)$, a bounded set $\mathcal{Q}$ and constants $\psi_{\max}$, $\iota > 0$ such that*

$$\mathbb{E}\big[\phi(\boldsymbol{Q}[k+1]) - \phi(\boldsymbol{Q}[k]) \mid \boldsymbol{Q}[k]\big] \leq \begin{cases} \psi_{\max}, & \text{if } \boldsymbol{Q}[k] \in \mathcal{Q}, \\ -\iota, & \text{if } \boldsymbol{Q}[k] \notin \mathcal{Q}, \end{cases}$$

then the mean hitting time $\mathbb{E}[\omega_b|\boldsymbol{Q}[0]]$, where $\omega_b := \min\{k \geq 0 : \phi(\boldsymbol{Q}[k]) < \max_{\boldsymbol{Q}\in\mathcal{Q}} \phi(\boldsymbol{Q})\}$, is bounded by $\mathsf{C}_1\phi(\boldsymbol{Q}[\mathbf{0}])$ for some constant C_1.

Proof. Consider the case when $\boldsymbol{Q}_0 \notin \mathcal{Q}$ (since the result is trivial otherwise). According to Theorem 2.3 in [39], when the queueing system has bounded arrivals and satisfies the Lyapunov stability condition, there exists a constant c such that for any $\eta < \iota/c$ and $\rho = 1 - \iota\eta + c\eta^2$, the following inequality holds:

$$\mathbb{P}\left(\omega_b > k \mid \boldsymbol{Q}[0]\right) < e^{\eta(\phi(\boldsymbol{Q}[\mathbf{0}])-b)}\rho^k,$$

where $b := \max_{\boldsymbol{Q}\in\mathcal{Q}} \phi(\boldsymbol{Q})$. Therefore, we have that

$$\begin{aligned}\mathbb{E}\big[\omega_b \mid \boldsymbol{Q}[0]\big] &\leq \sum_{k=0}^{\infty} \mathbb{P}\left(\omega_b > k \mid \boldsymbol{Q}[0]\right)\\ &\leq e^{\eta(\phi(\boldsymbol{Q}[\boldsymbol{0}])-b)}\frac{\rho}{1-\rho}\\ &\leq e^{\eta(\phi(\boldsymbol{Q}[\boldsymbol{0}])-b)}\frac{2}{\iota\eta}, \text{ when } \eta \leq \frac{\iota}{2c}.\end{aligned}$$

Take $\eta = \min(\frac{1}{\phi(\boldsymbol{Q}[\boldsymbol{0}])-b}, \frac{\iota}{2c})$, then it follows that

$$\mathbb{E}\big[\omega_b \mid \boldsymbol{Q}[0]\big] \leq \frac{2e}{\iota}(\phi(\boldsymbol{Q}[\boldsymbol{0}]) - b)$$

and the claim is proved. □

Note that the average hitting time from any queueing state within a compact (finite) set $\mathcal{Q}$ to the idle state (i.e., $\boldsymbol{Q} = \boldsymbol{0}$) is bounded by a ($\mathcal{Q}$-dependent) constant, since the Markov chain of the queueing system is positive recurrent under a stable policy. This implies that when a cycle is interrupted with initial state $\boldsymbol{Q}[0]$, it takes $\mathsf{C}_1\phi(\boldsymbol{Q}[0])$ time (for some C_1) for a stable policy like MaxWeight to return the system to the idle state.

Note that $\phi(\boldsymbol{Q}) = \|\boldsymbol{Q}\|_2^2$ is the Lyapunov function used to show the stability of MaxWeight [7]. By our model settings (in particular, bounded arrivals per time slot) and the interruption rule, there exists a constant C_2 such that the longest queue (after an interrupted cycle) is bounded by $\mathsf{C}_2 \log\tau$ for any interruption before horizon τ. This leads to an extra regret bounded by $r_{\max}|\mathcal{U}|\mathsf{C}_1\mathsf{C}_2^2\log^2\tau$ per interruption. Furthermore, since the average number of interruptions is logarithmic as bounded by Lemma 1, the claim of this corollary is proved. □

Appendix B

Appendix for Chapter 3

B.1 Proof of Lemma 2

As a shorthand notation, we define an event $\mathcal{G}_n(h,i)$ for any $(h,i) \in \mathcal{T}$ and $n \geq 1$ where

$$\mathcal{G}_n(h,i) := [(H_n, I_n) \in \mathcal{D}(h,i)].$$

Recall that we denote (H_n, I_n), $(\hat{C}_n, \hat{U}_n)$, (C_n, U_n) as the (random) selected node, observed length/reward, unclipped length/reward for cycle n under CHOOC, and $\mathcal{D}(h,i)$ as the descendants of node (h,i) including itself. Let i^* be the shorthand of i_h^* when there is no ambiguity.

Proof of Lemma 2. Note that $B_{h,i^*}(n) = \infty$ when $T_{h,i^*}^{(n)} = 0$. Hence, it suffices to assume $T_{h,i^*}^{(n)} \geq 1$.

By Assumption 5, we have $f^* - f(\boldsymbol{w}) \leq \nu\rho^h$ for all $\boldsymbol{w} \in \mathcal{P}_{h,i^*}$. Observe that

$$\frac{\sum_{t=1}^n \mathbb{E}[U_t] \mathbb{1}_{\mathcal{G}_t(h,i^*)}}{\sum_{t=1}^n \mathbb{E}[C_t] \mathbb{1}_{\mathcal{G}_t(h,i^*)}} \geq \inf_{\boldsymbol{w} \in \mathcal{P}_{h,i^*}} f(\boldsymbol{w}).$$

Thus, we have that

$$f^* - \nu\rho^h - \frac{\sum_{t=1}^n \mathbb{E}[U_t] \mathbb{1}_{\mathcal{G}_t(h,i^*)}}{\sum_{t=1}^n \mathbb{E}[C_t] \mathbb{1}_{\mathcal{G}_t(h,i^*)}} \leq 0.$$

Then we have the following equations:

$$\begin{aligned}
&\mathbb{P}\left(B_{h,i^*}(n) \le f^*\right) \\
&\quad = \mathbb{P}\left(\hat{R}_{h,i^*}(n) + \Phi_{h,i^*}(n) + \nu\rho^h \le f^*\right), \\
&\quad = \mathbb{P}\Bigg((f^* - \nu\rho^h - \frac{\sum_{t=1}^n \mathbb{E}[U_t]\mathbb{1}_{\mathcal{G}_t(h,i^*)}}{\sum_{t=1}^n \mathbb{E}[C_t]\mathbb{1}_{\mathcal{G}_t(h,i^*)}}) \\
&\qquad\qquad + \frac{\sum_{t=1}^n \mathbb{E}[U_t]\mathbb{1}_{\mathcal{G}_t(h,i^*)}}{\sum_{t=1}^n \mathbb{E}[C_t]\mathbb{1}_{\mathcal{G}_t(h,i^*)}} - \hat{R}_{h,i^*}(n) \ge \Phi_{h,i^*}(n) \Bigg), \\
&\quad \le \mathbb{P}\left(\frac{\sum_{t=1}^n \mathbb{E}[U_t]\mathbb{1}_{\mathcal{G}_t(h,i^*)}}{\sum_{t=1}^n \mathbb{E}[C_t]\mathbb{1}_{\mathcal{G}_t(h,i^*)}} - \hat{R}_{h,i^*}(n) \ge \Phi_{h,i^*}(n) \right), \\
&\quad = \mathbb{P}\left(\frac{\sum_{t=1}^n \mathbb{E}[U_t]\mathbb{1}_{\mathcal{G}_t(h,i^*)}}{\sum_{t=1}^n \mathbb{E}[C_t]\mathbb{1}_{\mathcal{G}_t(h,i^*)}} - \frac{\sum_{t=1}^n \hat{U}_t\mathbb{1}_{\mathcal{G}_t(h,i^*)}}{\sum_{t=1}^n \hat{C}_t\mathbb{1}_{\mathcal{G}_t(h,i^*)}} \ge \Phi_{h,i^*}(n) \right). \qquad \text{(B.1)}
\end{aligned}$$

Now observe that

$$\Phi_{h,i}(n) = \tilde{\Phi}(\epsilon_{h,i}(n), \epsilon'_{h,i}(n))$$

where

$$\begin{aligned}
\tilde{\Phi}(\epsilon, \epsilon') &:= \frac{\epsilon(1 + r_{\max}) + \epsilon'}{\mu_{\min} + \epsilon} \\
&\ge \frac{\sum_{t=1}^n \mathbb{E}[U_t]\mathbb{1}_{\mathcal{G}_t(h,i^*)}}{\sum_{t=1}^n \mathbb{E}[C_t]\mathbb{1}_{\mathcal{G}_t(h,i^*)}} - \frac{\sum_{t=1}^n \mathbb{E}[U_t]\mathbb{1}_{\mathcal{G}_t(h,i^*)} - (\epsilon + \epsilon')T_{h,i^*}(n)}{\sum_{t=1}^n \mathbb{E}[C_t]\mathbb{1}_{\mathcal{G}_t(h,i^*)} + \epsilon T_{h,i^*}(n)}.
\end{aligned}$$

Hence, we can continue to find an upper bound of Equation (B.1) as follows:

$$\begin{aligned}
&\text{Eq. (B.1)} \\
&\le \mathbb{P}\left(\frac{\sum_{t=1}^n \mathbb{E}[U_t]\mathbb{1}_{\mathcal{G}_t(h,i^*)} - (\epsilon_{h,i^*}(n) + \epsilon'_{h,i^*}(n))T_{h,i^*}(n)}{\sum_{t=1}^n \mathbb{E}[C_t]\mathbb{1}_{\mathcal{G}_t(h,i^*)} + \epsilon_{h,i^*}(n)T_{h,i^*}(n)} \ge \frac{\sum_{t=1}^n \hat{U}_t\mathbb{1}_{\mathcal{G}_t(h,i^*)}}{\sum_{t=1}^n \hat{C}_t\mathbb{1}_{\mathcal{G}_t(h,i^*)}} \right),
\end{aligned}$$

$$
\begin{aligned}
\leq & \mathbb{P}\left(\sum_{t=1}^{n} \mathbb{E}[U_t] \mathbb{1}_{\mathcal{G}_t(h,i^*)} - (\epsilon_{h,i^*}(n) + \epsilon'_{h,i^*}(n)) T_{h,i^*}(n) \geq \sum_{t=1}^{n} \hat{U}_t \mathbb{1}_{\mathcal{G}_t(h,i^*)}\right) \\
& + \mathbb{P}\left(\sum_{t=1}^{n} \mathbb{E}[C_t] \mathbb{1}_{\mathcal{G}_t(h,i^*)} + \epsilon_{h,i^*}(n) T_{h,i^*}(n) \leq \sum_{t=1}^{n} \hat{C}_t \mathbb{1}_{\mathcal{G}_t(h,i^*)}\right), \\
\leq & \mathbb{P}\left(\sum_{t=1}^{n} \mathbb{E}[\hat{U}_t] \mathbb{1}_{\mathcal{G}_t(h,i^*)} - \epsilon_{h,i^*}(n) T_{h,i^*}(n) \geq \sum_{t=1}^{n} \hat{U}_t \mathbb{1}_{\mathcal{G}_t(h,i^*)}\right) \\
& + \mathbb{P}\left(\sum_{t=1}^{n} \mathbb{E}[C_t] \mathbb{1}_{\mathcal{G}_t(h,i^*)} + \epsilon_{h,i^*}(n) T_{h,i^*}(n) \leq \sum_{t=1}^{n} C_t \mathbb{1}_{\mathcal{G}_t(h,i^*)}\right), \qquad \text{(B.2)} \\
\leq & \frac{2}{n^3}. \qquad \text{(B.3)}
\end{aligned}
$$

Equation (B.2) is given by the fact that for any $(h, i) \in \mathcal{D}(z, i^*)$, from standard calculation,

$$
\sum_{t=1}^{n} \mathbb{E}[\hat{U}_t] \mathbb{1}_{\mathcal{G}_t(h,i)} + \epsilon'_{h,i}(n) T_{h,i}(n) \geq \sum_{t=1}^{n} \mathbb{E}[U_t] \mathbb{1}_{\mathcal{G}_t(h,i)}. \qquad \text{(B.4)}
$$

Equation (B.3) follows Azuma-Hoeffding inequality for Martingale differences (sub-exponential version), Assumption 6(b), Assumption 7(b) and union bounds. This concludes the proof. □

B.2 Proof of Lemma 3

In this section, we will break down the proof of Lemma 3 into several lemmas. Lemma 3 is a direct result of combining Lemma 10 and Lemma 12. Recall that we define event $\mathcal{G}_n(h, i) := [(H_n, I_n) \in \mathcal{D}(h, i)]$ for any $(h, i) \in \mathcal{T}$ and $n \geq 1$.

Lemma 8. *For any suboptimal node (h, i), let $z \leq k \leq h - 1$ be the largest depth such that (k, i_k^*) is an ancestor of (h, i). Then for any integer $u \geq 0$, we* *have*

$$\mathbb{E}[T_{h,i}(n)] \leq u + \sum_{t=u+1}^{n} \mathbb{P}(\mathcal{E}_1(t) \cup \mathcal{E}_2(t))$$

where

$$\mathcal{E}_1(t) = \bigcup_{s=k+1}^{t} [B_{s,i_s^*}(t) \leq f^*],$$
$$\mathcal{E}_2(t;u) = [B_{h,i}(t) > f^*] \cap [T_{h,i}(t) > u].$$

Proof. This is a modified restatement of Lemma 14 in [2]. □

Lemma 9. *For all suboptimal nodes $(h,i) \in \mathcal{D}(z, i^*)$ and $n \geq 1$,*

$$\mathbb{P}\left(\bar{R}_{h,i}(n) > f_{h,i}^* + \frac{(1+r_{\max})\epsilon_{h,i}(n)}{\min(\mu_{\min}-\epsilon_{h,i}(n), 1)}, T_{h,i}(n) \geq 1\right) \leq 2n^{-3}$$

where given $T_{h,i}(n) \geq 1$,

$$\bar{R}_{h,i}(n) := \frac{\sum_{t=1}^{n} U_t \mathbb{1}_{\mathcal{G}_t(h,i)}}{\sum_{t=1}^{n} C_t \mathbb{1}_{\mathcal{G}_t(h,i)}}.$$

Proof. Note that $\bar{R}_{h,i}(n)$ is the untruncated version of the empirical average. We find it more straightforward to prove the concentration of $\bar{R}_{h,i}(n)$. In Lemma 10, we will show that $\bar{R}_{h,i}(n) \approx R_{h,i}(n)$ when (h,i) is sufficiently explored.

Observe that

$$\frac{\sum_{t=1}^{n} \mathbb{E}[U_t] \mathbb{1}_{\mathcal{G}_t(h,i)}}{\sum_{t=1}^{n} \mathbb{E}[C_t] \mathbb{1}_{\mathcal{G}_t(h,i)}} \leq f_{h,i}^* := \sup_{\boldsymbol{w} \in \mathcal{P}_{h,i}} f(\boldsymbol{w}),$$

and that for any $\epsilon \leq \mu_{\min} - 1$,[1]

$$\frac{(1+r_{\max})\epsilon}{\mu_{\min}-\epsilon} \geq \frac{\sum_{t=1}^{n} \mathbb{E}[U_t]\mathbb{1}_{\mathcal{G}_t(h,i)} + \epsilon T_{h,i}(n)}{\sum_{t=1}^{n} \mathbb{E}[C_t]\mathbb{1}_{\mathcal{G}_t(h,i)} - \epsilon T_{h,i}(n)} - \frac{\sum_{t=1}^{n} \mathbb{E}[U_t]\mathbb{1}_{\mathcal{G}_t(h,i)}}{\sum_{t=1}^{n} \mathbb{E}[C_t]\mathbb{1}_{\mathcal{G}_t(h,i)}}.$$

Then the proof follows the same arguments as in Lemma 2. □

Lemma 10. *For all suboptimal nodes $(h,i) \in \mathcal{D}(z,i^*)$ such that $\Delta_{h,i} > \nu\rho^h$, we have that for all $n \geq 1$,*

$$\mathbb{E}[T_{h,i}(n)] \leq ((\mathsf{K}_1^{(h,i)} \log n) \vee \mathsf{S}^{(h,i)}(n) \vee \mathsf{M}_1^{(h,i)}) + 6,$$

where

$$\mathsf{K}_1^{(h,i)} = (\frac{32(1+r_{\max})^2\xi^2(1+\mu_{\min})^2}{(\Delta_{h,i} - \nu\rho^h)^2\mu_{\min}^2} \vee \frac{16(1+r_{\max})\alpha(1+\mu_{\min})}{(\Delta_{h,i} - \nu\rho^h)\mu_{\min}})$$

and $\mathsf{M}_1^{(h,i)}, \mathsf{S}^{(h,i)}(n)$ are functions of $h, i(,n)$ such that

$$\mathsf{M}_1^{(h,i)} \sim O(1/(\Delta_{h,i} - \nu\rho^h)^2),\ \mathsf{S}^{(h,i)}(n) \sim O(\log n/(\Delta_{h,i} - \nu\rho^h)^2).$$

Proof. We will follow the recipe given in Lemma 8 to show this result. First we show that

$$\sum_{t=u+1}^{n} \mathbb{P}(\mathcal{E}_1(t)) \leq \sum_{t=1}^{n} t \cdot \frac{2}{t^3} \leq \frac{\pi^2}{3},$$

which follows Lemma 2 and a union bound (see the definition of $\mathcal{E}_1(t)$ in Lemma 8).

[1]It is sufficient to consider $\mu_{\min}-\epsilon \geq 1$ since $C_t \geq 1$ almost surely.

Then we observe that (on the event $[T_{h,i}(t) \geq 1]$ such that all is well-defined)

$$[B_{h,i}(t) > f^*] \subset \mathsf{E}_1(t) \cup \mathsf{E}_2(t) \cup \mathsf{E}_3(t)$$

where

$$\begin{aligned}
\mathsf{E}_1(t) &= [\bar{R}_{h,i}(t) > f^*_{h,i} + \frac{(1{+}r_{\max})\epsilon_{h,i}(n)}{\min(\mu_{\min}{-}\epsilon_{h,i}(n), 1)}],\\
\mathsf{E}_2(t) &= [\Phi_{h,i}(t) + \frac{(1{+}r_{\max})\epsilon_{h,i}(t)}{\min(\mu_{\min}{-}\epsilon_{h,i}(t), 1)} > \frac{3}{4}(\Delta_{h,i} - \nu\rho^h)],\\
\mathsf{E}_3(t) &= [\hat{R}_{h,i}(t) - \bar{R}_{h,i}(t) > \frac{\Delta_{h,i} - \nu\rho^h}{4}].
\end{aligned}$$

Recall that $\bar{R}_{h,i}(n)$ is defined as in Lemma 9. To see this, if none of the three events happen, $B_{h,i}(t) \leq f^*$.

Then we show that the probability of each event above is negligible given $T_{h,i}(t)$ is sufficiently large.

(i) First we have that by Lemma 9,

$$\mathbb{P}\left(\mathsf{E}_1(t), T_{h,i}(t) \geq 1\right) \leq 2t^{-3}.$$

(ii) Event $\mathsf{E}_2(t)$ can be further partitioned as follows:

$$\mathsf{E}_2(t) \subset [\frac{1{+}\mu_{\min}}{\mu_{\min}}(1{+}r_{\max})\epsilon_{h,i}(t) > \frac{\Delta_{h,i} - \nu\rho^h}{2}] \cup [\frac{\epsilon'_{h,i}(t)}{\mu_{\min}} > \frac{\Delta_{h,i} - \nu\rho^h}{4}] \quad \text{(B.5)}$$

The first event in RHS of (B.5) implies that (by algebra)

$$T_{h,i}(t) < \mathsf{K}_1^{(h,i)} \cdot \log t.$$

The second event in RHS of (B.5) implies that $T_{h,i}(t) \leq \mathsf{M}_1^{(h,i)}$ where $\mathsf{M}_1^{(h,i)}$ is defined as the largest integer such that

$$\frac{r_{\max}}{\mathsf{M}_1^{(h,i)}} \cdot \frac{\pi^2}{6} e^{-\beta/4\alpha}(\beta + 2\alpha + \kappa \log \mathsf{M}_1^{(h,i)} + 1) > \frac{(\Delta_{h,i} - \nu\rho^h)}{4} \mu_{\min}.$$

Note that $\mathsf{M}_1^{(h,i)} \sim O(1/(\Delta_{h,i} - \nu\rho^h)^2)$.

Therefore, we have that for $t \leq n$,

$$\mathbb{P}\left(\mathsf{E}_2(t), T_{h,i}(t) > \max(\mathsf{K}_1^{(h,i)} \log n, \mathsf{M}_1^{(h,i)})\right) = 0.$$

(iii) Let $\mathsf{S}^{(h,i)}(n)$ be the smallest integer (with respect to a fixed $n \geq 1$) such that for any $s \geq \mathsf{S}^{(h,i)}(n)$,

$$s > \frac{4\bar{r}}{\Delta_{h,i} - \nu\rho^h}\left(\frac{\pi^2}{6} e^{-\beta/2\alpha}(2\alpha{+}1) + \sqrt{2\alpha s \log n}\right). \tag{B.6}$$

Note that $\mathsf{S}^{(h,i)}(n) \sim O(\log n/(\Delta_{h,i} - \nu\rho^h)^2)$. For any $t \leq n$,

$$\begin{aligned}
&\mathbb{P}\left(\mathsf{E}_3(t), T_{h,i}(t) \geq \mathsf{S}^{(h,i)}(n)\right) \\
&\leq \mathbb{P}\Bigg(\underbrace{\frac{\sum_{s=1}^t U_s \mathbb{1}_{\mathcal{G}_s(h,i)}}{\sum_{s=1}^t C_s \mathbb{1}_{\mathcal{G}_s(h,i)}}}_{\leq \bar{r}} \cdot \underbrace{\frac{\sum_{s=1}^t (C_s - \hat{C}_s) \mathbb{1}_{\mathcal{G}_s(h,i)}}{\sum_{s=1}^t \hat{C}_s \mathbb{1}_{\mathcal{G}_s(h,i)}}}_{\text{denom.}\geq T_{h,i}^{(t)}} > \frac{\Delta_{h,i} - \nu\rho^h}{4}, T_{h,i}(t) \geq \mathsf{S}^{(h,i)}(n) \Bigg), \\
&\leq \mathbb{P}\left(\sum_{s=1}^t (C_s - \hat{C}_s) \mathbb{1}_{\mathcal{G}_s(h,i)} \geq (\pi^2/6) e^{-\beta/2\alpha}(2\alpha{+}1) + \sqrt{2\alpha T_{h,i}(t) \log n} \right). \qquad \text{(B.7)} \\
&\leq \frac{1}{n}. \qquad \text{(B.8)}
\end{aligned}$$

Equation (B.7) is given by (B.6) since $T_{h,i}(t) \geq \mathsf{S}^{(h,i)}(n)$. For (B.8), first we can derive that (by algebra computation),

$$\sum_{s=1}^t \mathbb{E}[C_s - \hat{C}_s] \mathbb{1}_{\mathcal{G}_s(h,i)} \leq (\pi^2/6) e^{-\beta/2\alpha}(2\alpha{+}1).$$

Next, observe that when $\boldsymbol{w}_s \in \mathcal{P}_{z,i^*}$, since $l_s \geq \beta > \mathbb{E}[C_s] + \xi^2/\alpha$, we have for all s

$$(C_s - \hat{C}_s) - \mathbb{E}[C_s - \hat{C}_s] \leq C_s - \mathbb{E}[C_s] - \xi^2/\alpha \ \ a.s.$$

We can then apply Azuma-Hoeffding and the sub-exponentiality of C_s as in previous lemmas to show (B.8) holds.

Finally, we apply Lemma 8 to conclude the proof. When $u = (\mathsf{K}_1^{(h,i)} \log n) \vee \mathsf{M}_1^{(h,i)} \vee \mathsf{S}^{(h,i)}(n)$, we have

$$\mathbb{E}[T_{h,i}(n)] \leq u + \frac{\pi^2}{3} + \sum_{t=u+1}^{n} \mathbb{P}(\mathcal{E}_2(t;u)) \leq u + \frac{\pi^2}{3} + \sum_{t=1}^{n} (\frac{2}{t^3} + \frac{1}{n}) \leq u + 6.$$

□

Before introducing the following two lemmas, recall that we have defined a modified reward function $\tilde{f}$:

$$\tilde{f}(\boldsymbol{w}) = \begin{cases} f(\boldsymbol{w}) & \boldsymbol{w} \in (\mathcal{W}(\boldsymbol{\lambda}; \xi^2, \alpha))^\circ, \\ \sup\limits_{l \geq \beta} \frac{\mathbb{E}[\hat{U}^{(\boldsymbol{w},l)}(1)]}{\mathbb{E}[\hat{C}^{(\boldsymbol{w},l)}(1)]} & \boldsymbol{w} \in \mathcal{W} \setminus (\mathcal{W}(\boldsymbol{\lambda}; \xi^2, \alpha))^\circ. \end{cases}$$

The function values for $\boldsymbol{w} \notin (\mathcal{W}(\boldsymbol{\lambda}; \xi^2, \alpha))^\circ$ are compatible with Assumption 7(a). $\tilde{f}^*$, $\tilde{f}^*_{h,i}$ and $\tilde{\Delta}_{h,i}$ are defined accordingly. Note that $f^* = \tilde{f}^*$.

Lemma 11. *For all suboptimal nodes $(h,i) \notin \mathcal{D}(z, i^*)$ and $n \geq 1$,*

$$\mathbb{P}\left(\bar{R}_{h,i}(n) > \tilde{f}^*_{h,i} + \frac{(1+r_{\max})\tilde{\epsilon}_{h,i}(n)}{\min(\mu_{\min} - \tilde{\epsilon}_{h,i}(n), 1)}\right) \leq 2n^{-3}$$

where

$$\bar{R}_{h,i}(n) = \frac{\sum_{t=1}^{n} U_t \mathbb{1}_{\mathcal{G}_t^{(1)}(h,i)} + \hat{U}_t \mathbb{1}_{\mathcal{G}_t^{(2)}(h,i)}}{\sum_{t=1}^{n} C_t \mathbb{1}_{\mathcal{G}_t^{(1)}(h,i)} + \hat{C}_t \mathbb{1}_{\mathcal{G}_t^{(2)}(h,i)}},$$

$$\mathcal{G}_t^{(1)}(h,i) = \mathcal{G}_t(h,i) \cap [\boldsymbol{w}_t \in (\mathcal{W}(\boldsymbol{\lambda};\xi^2,\alpha))^\circ],$$
$$\mathcal{G}_t^{(2)}(h,i) = \mathcal{G}_t(h,i) \cap [\boldsymbol{w}_t \notin (\mathcal{W}(\boldsymbol{\lambda};\xi^2,\alpha))^\circ],$$
$$\tilde{\epsilon}_{h,i}(n) = \max(\epsilon_{h,i}(n), \sqrt{\frac{2(1\vee\bar{r})^2 l_n^2 \log n}{T_{h,i}(n)}}).$$

Proof. This lemma is analog to Lemma 9. Since when $(h,i) \notin \mathcal{D}(z,i^*)$, $\boldsymbol{w}_t$ may or may nor be in the region $(\mathcal{W}(\boldsymbol{\lambda};\xi^2,\alpha))^\circ$, we therefore redefine $\bar{R}_{h,i}(n)$ accordingly. By Assumption 7, we have that

$$\frac{\sum_{t=1}^n \mathbb{E}[U_t]\mathbb{1}_{\mathcal{G}_t^{(1)}(h,i)} + \mathbb{E}[\hat{U}_t]\mathbb{1}_{\mathcal{G}_t^{(2)}(h,i)}}{\sum_{t=1}^n \mathbb{E}[C_t]\mathbb{1}_{\mathcal{G}_t^{(1)}(h,i)} + \mathbb{E}[\hat{C}_t]\mathbb{1}_{\mathcal{G}_t^{(2)}(h,i)}} \le \tilde{f}_{h,i}^*.$$

For $\boldsymbol{w} \notin (\mathcal{W}(\boldsymbol{\lambda};\xi^2,\alpha))^\circ$, the sub-exponential type concentration is invalid. Thus, we additionally apply Azuma-Hoeffding for bounded differences by the fact that $\max(\hat{C}_t,\hat{U}_t) \le (1\vee\bar{r})l_n$ (*a.s.*), and the final upper bound is the maximum of the two bound types as given in $\tilde{\epsilon}_{h,i}(n)$. Then the proof follows the same argument as in Lemma 9. □

Lemma 12. *For all suboptimal nodes $(h,i) \notin \mathcal{D}(z,i^*)$ such that $\tilde{\Delta}_{h,i} > \nu\rho^h$, we have that for all $n \ge 1$,*

$$\mathbb{E}[T_{h,i}(n)] \le ((\mathsf{K}_2^{(h,i)} l_n^2 \log n) \vee (\mathsf{K}_1^{(h,i)} \log n) \vee \mathsf{S}^{(h,i)}(n) \vee \mathsf{M}_1^{(h,i)}) + 6,$$

where

$$\mathsf{K}_2^{(h,i)} = \frac{8(1+r_{\max})^2(1\vee\bar{r})^2(1+\mu_{\min})^2}{(\Delta_{h,i}-\nu\rho^h)^2\mu_{\min}^2},$$

and $\mathsf{K}_1^{(h,i)}, \mathsf{M}_1^{(h,i)}, \mathsf{S}^{(h,i)}(n)$ are defined as in Lemma 10.

Proof. This is a redo of the proof of Lemma 10, except that the step using Lemma 9 is replaced by Lemma 11. □

B.3 Proof of Theorem 1

Lemma 13. *Let π^{opt} be the optimal static policy without clipping, i.e., $\pi_n^{\text{opt}} = (\boldsymbol{w}^*, \infty)$ for all $n \geq 1$. The expected cumulative reward of π^{opt} over a time horizon τ is bounded as*

$$\mathbb{E}[\text{Rew}_{\pi^{\text{opt}}}[\tau]] \leq r^{(*)}\tau + r^{(*)}\frac{\mathbb{E}[(C^{(*)}(1))^2]}{\mathbb{E}[C^{(*)}(1)]}.$$

Proof. By the definition of $\mathbb{E}[\text{Rew}_{\pi^{\text{opt}}}[\tau]]$, we have the following upper bound:

$$\begin{aligned}
\mathbb{E}[\text{Rew}_{\pi^{\text{opt}}}[\tau]] &\leq \mathbb{E}\big[\sum_{n=1}^{N_{\pi^{\text{opt}}}[\tau]+1} U^{(*)}(n)\big], \\
&= \mathbb{E}\big[\sum_{n=1}^{\infty} \mathbb{1}_{\{S_{n-1}^{\pi^{\text{opt}}} \leq \tau\}} \mathbb{E}[U^{(*)}(n)]\big], \\
&\leq r^{(*)}\mathbb{E}\big[\sum_{n=1}^{\infty} \mathbb{1}_{\{S_{n-1}^{\pi^{\text{opt}}} \leq \tau\}} \mathbb{E}[C^{(*)}(n)]\big], \\
&= r^{(*)}\mathbb{E}\big[\sum_{n=1}^{N_{\pi^{\text{opt}}}[\tau]+1} C^{(*)}(n)\big], \\
&\leq r^{(*)}(\tau + \frac{\mathbb{E}[(C^{(*)}(1))^2]}{\mathbb{E}[C^{(*)}(1)]}).
\end{aligned}$$

Note that the second and fourth equations hold true due to the independence of $S_{n-1}^{\pi^{\text{opt}}}$ and $(C^{(*)}(n), U^{(*)}(n))$. The last line follows Lorden's inequality for overshoot [84]. $\square$

Proof of Theorem 1. This proof follows a similar approach as in [2]. First we define $\mathcal{I}_h = \{(h,i) : \Delta_{h,i} \leq 2\nu\rho^h\}$, i.e., the set of nodes at depth h that are $2\nu\rho^h$-optimal. Let $\mathcal{J}_h$ be the set of nodes that (i) are located at a depth h, (ii) are not $2\nu\rho^h$-optimal, and (iii) whose parent belongs to the set $\mathcal{I}_{h-1}$.

Denote $\mathcal{T}(z)$ as the collection of nodes in $\mathcal{T}$ with depth *greater or equal to* z. For a fixed depth $H (H \geq z)$, which will be determined later, the collection $\mathcal{T}(z)$ consists of four types of nodes as follows:

(1) $\mathcal{T}_1$: $\mathcal{I}_H \cap \mathcal{D}(z, i^*)$ and their descendants,

(2) $\mathcal{T}_2$: $\bigcup_{z \leq h \leq H-1} \mathcal{I}_h \cap \mathcal{D}(z, i^*)$,

(3) $\mathcal{T}_3$: $\bigcup_{z+1 \leq h \leq H} \mathcal{J}_h \cap \mathcal{D}(z, i^*)$ and their descendants,

(4) $\mathcal{T}_4$: $(\mathcal{D}(z, i^*))^{\complement} := \mathcal{T}(z) \setminus \mathcal{D}(z, i^*)$.

To find an upper bound of regret $\mathrm{Reg}_\pi[\tau]$, first we observe that

$$\begin{aligned}
\mathrm{Reg}_\pi[\tau] &\leq (r^{(*)}\tau + \bar{r}\frac{\mathbb{E}[(C^{(*)}(1))^2]}{\mathbb{E}[C^{(*)}(1)]}) - \mathbb{E}[\sum_{n=1}^{N_\pi[\tau]} \hat{U}_n], \\
&\leq r^{(*)}(\mathbb{E}[\sum_{n=1}^{N_\pi[\tau]} \hat{C}_n] + l_\tau) + \bar{r}\frac{\mathbb{E}[(C^{(*)}(1))^2]}{\mathbb{E}[C^{(*)}(1)]} - \mathbb{E}[\sum_{n=1}^{N_\pi[\tau]} \hat{U}_n], \\
&= \mathbb{E}[\sum_{n=1}^{N_\pi[\tau]} r^{(*)}\hat{C}_n - \hat{U}_n] + O(\log \tau).
\end{aligned}$$

The cumulative regret can be considered as induced by nodes of different types respectively. Denote that for $1 \leq j \leq 4$,

$$\mathrm{Reg}_j[\tau] = \mathbb{E}\big[\sum_{n=1}^{N_\pi[\tau]} (r^{(*)}\hat{C}_n - \hat{U}_n)\mathbb{1}_{\{(H_n, I_n) \in \mathcal{T}_j\}}\big].$$

Hence,

$$\mathrm{Reg}[\tau] = \sum_{j=1}^{4} \mathrm{Reg}_j[\tau] + O(\log \tau),$$

and we will tackle $\mathrm{Reg}_j[\tau]$ separately in the following paragraphs.

Part A. Observe that for $j = 1, 2, 3$,

$$\begin{aligned}
\text{Reg}_j[\tau] &\leq \mathbb{E}\big[\sum_{n=1}^{N_\pi[\tau]} \mathbb{1}_{\{(H_n,I_n)\in\mathcal{T}_j\}}(r^{(*)}C_n - U_n)\big] + \underbrace{\mathbb{E}\big[\sum_{n=1}^{N_\pi[\tau]} \mathbb{1}_{\{(H_n,I_n)\in\mathcal{T}_j\}}(U_n - \hat{U}_n)\big]}_{\mathsf{T1}}, \\
&\leq \mathbb{E}\big[\sum_{n=1}^{N_\pi[\tau]} \mathbb{1}_{\{(H_n,I_n)\in\mathcal{T}_j\}}(r^{(*)} - r^{(\boldsymbol{w}_n)})\mathbb{E}[C_n|H_n, I_n]\big] + \mathsf{T1}, \\
&\leq \mu_{\max}\mathbb{E}\big[\sum_{n=1}^{N_\pi[\tau]} \mathbb{1}_{\{(H_n,I_n)\in\mathcal{T}_j\}}(f^* - f(\boldsymbol{w}_n))\big] + \mathsf{T1}. \qquad \text{(B.9)}
\end{aligned}$$

Note that $\mathsf{T1} \sim O(\log n)$, which has been shown in Lemma 2 (see (B.4)).

Let $\widetilde{\text{Reg}}_j[\tau] := \mathbb{E}[\sum_{n=1}^{N_\pi[\tau]} \mathbb{1}_{\{(H_n,I_n)\in\mathcal{T}_j\}}(f^* - f(\boldsymbol{w}_n))]$. We can follow a similar analysis as in [2].

(i) For any node $(h, i) \in \mathcal{I}_H$, $f^* - f^*_{h,i} \leq 2\nu\rho^H$. By Assumption 5, this implies $f^* - \inf_{\boldsymbol{w}\in\mathcal{P}_{h,i}} f(\boldsymbol{w}) \geq 4\nu\rho^H$. Hence,

$$\widetilde{\text{Reg}}_1[\tau] \leq 4\nu\rho^H\tau.$$

(ii) By Definition (near-optimality dimension), $|\mathcal{I}_h| \leq \mathsf{C}(\nu, \rho)\rho^{-d(\nu,\rho)h}$. Therefore,

$$\widetilde{\text{Reg}}_2[\tau] \leq \sum_{h=z}^{H-1} 4\nu\rho^h|\mathcal{I}_h| \leq 4\nu\mathsf{C}(\nu, \rho)\sum_{h=z}^{H-1}\rho^{(1-d(\nu,\rho))h}.$$

(iii) The parent of any node in $\mathcal{J}_h$ lies in $\mathcal{I}_{h-1}$, which implies any weight selected in nodes of $\mathcal{J}_h$ is $4\nu\rho^{h-1}$-optimal. Hence,

$$\widetilde{\text{Reg}}_3[\tau] \leq \sum_{h=z+1}^{H} 4\nu\rho^{h-1} \sum_{i:(h,i)\in\mathcal{J}_h} \mathbb{E}[T_{h,i}(\tau)],$$

$$\leq \sum_{h=z+1}^{H} 4\nu\rho^{h-1}|\mathcal{J}_h|\mathbb{E}[T_{h,i}(\tau)],$$
$$\leq \sum_{h=z+1}^{H} 8\nu\rho^{h-1}\mathsf{C}(\nu,\rho)\rho^{-d(\nu,\rho)(h-1)}\mathbb{E}[T_{h,i}(\tau)].$$

Note that when $\Delta_{h,i} > 2\nu\rho^h$, $\mathbb{E}[T_{h,i}(\tau)] = O((\xi^2 \vee \alpha)\log\tau/\rho^{2h})$ according to Lemma 10.

Combining the analysis in **(i)**-**(iii)**, we have that for some universal constants γ_1, γ_2,

$$\sum_{j=1}^{3} \widetilde{\mathrm{Reg}}_j[\tau] = \gamma_1\rho^H\tau + \gamma_2\mathsf{C}(\nu,\rho)\rho^{-H(d(\nu,\rho)+1)}(\xi^2 \vee \alpha)\log\tau.$$

Choosing H such that the two terms above have the same order in τ, implying ρ^H in the order of $(\tau/\ln\tau)^{-1/(\mathsf{C}(\nu,\rho)+2)}$, we have that for some universal constant $\mathsf{C}^{\mathrm{Reg}}$,

$$\sum_{j=1}^{3} \widetilde{\mathrm{Reg}}_j[\tau] = \mathsf{C}^{\mathrm{Reg}}\left(\mathsf{C}(\nu,\rho)(\xi^2 \vee \alpha)\right)^{\frac{1}{d(\nu,\rho)+2}}\tau^{\frac{d(\nu,\rho)+1}{d(\nu,\rho)+2}}(\log\tau)^{\frac{1}{d(\nu,\rho)+2}}. \tag{B.10}$$

Part B. For $j = 4$, by a similar derivation as in (B.9),

$$\mathrm{Reg}_j[\tau] \leq l_\tau\mathbb{E}\big[\sum_{n=1}^{N_\pi[\tau]} \mathbb{1}_{\{(H_n,I_n)\in\mathcal{T}_j\}}f^*\big] + \mathsf{T1}. \tag{B.11}$$

Note that $\mathbb{E}[C_n|H_n, I_n]$ is (trivially) bounded by l_τ almost surely when $(H_n, I_n) \notin \mathcal{D}(z, i^*)$.

Let $\tilde{z}$ be the smallest integer such that $\tilde{\Delta}_{\tilde{z},i} > 2\nu\rho^{\tilde{z}}$ for all $i : (\tilde{z}, i) \notin \mathcal{D}(z, i^*)$. Such an integer exists since by assumption $f^* - \sup_{\boldsymbol{w}\notin\mathcal{P}_{z,i^*}} \tilde{f}(\boldsymbol{w}) \geq \delta$

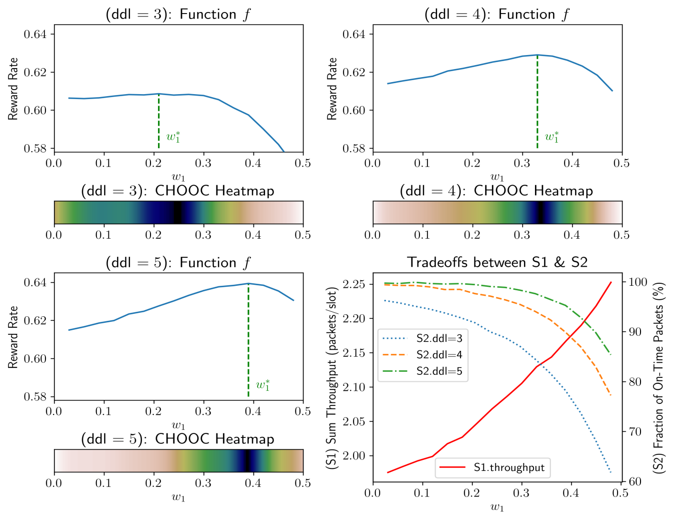


Figure B.1: Simulation results on experiments in Section 3.4.2.3. For `ddl` $= 3, 4, 5$, the reward rate function and the corresponding weight heatmap (first 10k cycles) are exhibited. We show the tradeoffs of rewards for two slices in the bottom-right panel.

for some $\delta > 0$. This gives that,

$$\begin{aligned}\mathbb{E}\Big[\sum_{n=1}^{N_\pi[\tau]} \mathbb{1}_{\{(H_n,I_n)\in\mathcal{T}_j\}} f^*\Big] &\le \Big(\big|\{(h,i)\in(\mathcal{D}(z,i^*))^{\complement} : h<\tilde{z}\}\big| \\ &\qquad + \sum_{i:(\tilde{z},i)\in(\mathcal{D}(z,i^*))^{\complement}} \mathbb{E}[T_{\tilde{z},i}(\tau)]\Big) f^*, \\ &\le (2^{\tilde{z}} + 2^{\tilde{z}}\gamma_3 \log^3 \tau) f^* \end{aligned} \tag{B.12}$$

for some universal constant γ_3. The last inequality holds as stated by Lemma 12.

Finally, the theorem is proved by combining Eqs. (B.9)–(B.12).

□

B.4 Additional Experiments

B.4.1 Tradeoffs between Backlogged and Non-backlogged Users

In this experiment, we set up a scenario exhibiting the ability of CHOOC to realize tradeoffs among slices of backlogged and non-backlogged users. Suppose the first slice consists of 9 `BL` users and the second slice has 3 $\texttt{DDL}(t)$ users, and we consider $t = 3, 4, 5$ respectively. The user distance to BS (m) equals $100 + 50(i \bmod 3)$ for i-th user and the arrival rate is set identically at 0.45 packets/slot for all non-backlogged users.

Suppose a proportionally-fair (PF) scheduler is implemented for flows in Slice 1 and a Log-Rule scheduler for those in Slice 2. As discussed in Section 3.2.3, we reset the moving average of PF for each cycle. To avoid short cycles where the moving average does not converge, we require that a cycle must exceed 60 slots (i.e., a cycle ends when it becomes idle the first time after 60 slots).

A larger weight to Slice 2 gives priority to packets with deadline requirements, but negatively affects the throughput of Slice 1 (see the bottom-right panel of Figure B.1). Thus, it is unclear beforehand how much priority should be given to Slice 2. In Figure B.1, we show the f values for each case, with the optimal weights highlighted accordingly. As expected, CHOOC correctly locates the best weight as is shown in the heatmaps.

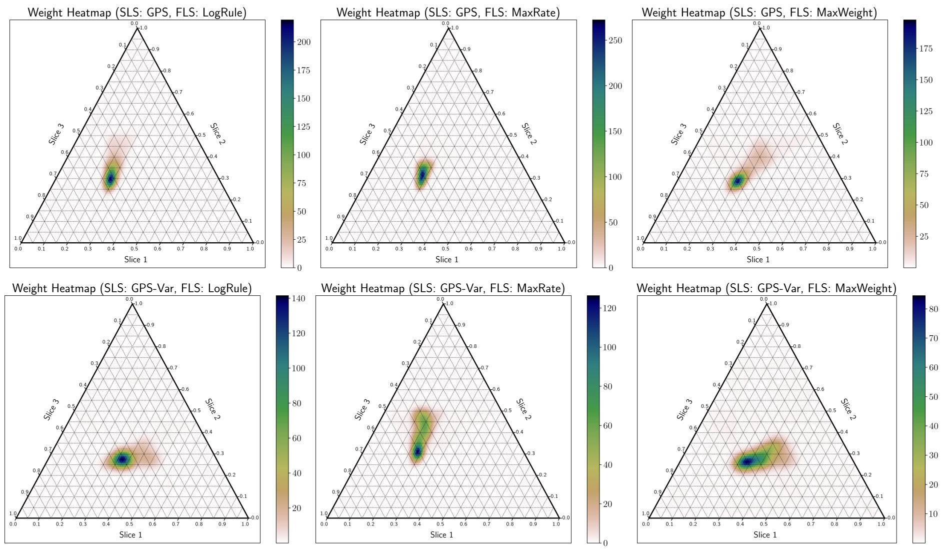


Figure B.2: Weight selection heatmaps (6k cycles) on experiments in Section B.4.2-A.

B.4.2 CHOOC under different slice- and flow-level schedulers

In this section, we investigate the impact of different choices for the slice- and flow-level schedulers on the CHOOC framework.

A. We follow the 3-slice system settings introduced in Section 3.4.1.2 in terms of user type and traffic load. At the slice level, we successively applied two weight-based scheduling implementations. Besides the plain GPS algorithm given in Algorithm 3, we simulated a variant of GPS (GPS-Var), where for each time slot only one slice can be scheduled, which is randomly chosen according to the weight allocation vector. For the flow-level schedulers, we applied Log-Rule, MaxWeight and Max-Rate respectively (for all of the slices in each case).

The weight selection heatmaps are shown in Figure B.2 for all of the 6 slice- and flow-level combinations. As expected, CHOOC locates the optimal weight allocation in each scenario, despite that they are slightly varied. This experiment also shows that the optimal slicing is affected by the associated schedulers (besides other factors such as traffic load and user utility) in a sophisticated manner.

B. To further demonstrate how CHOOC learns the trade-offs among slices under different flow-level schedulers, we set up another 3-slice system with homogeneous slices (in terms of user traffic and utility). In this system, each slice consists of 5 `DDL(3)` users. The BS-to-user distances are $50, 80, 110, 140, 170$ m in each slice, and the arrival rate is set to be 0.3 packets/slot for every user.

We first tested CHOOC on a system where all three slices implement Log-Rule as flow-level schedulers, then we changed the scheduler at Slice 1 to be Max-Rate. The results are shown in Figure B.3. As can be seen, CHOOC correctly locates $[1/3, 1/3, 1/3]$ as the best weight under the first scenario. When asymmetry is introduced in the second simulation, CHOOC allocates more resources to Slice 1 to maximize the utility as the best weight is close to $[0.4, 0.3, 0.3]$. It turns out at this traffic load level, Max-Rate (compared to Log-Rule) is favored to facilitate more packets to meet the tight deadlines. This is validated by further numerical tests which show that the optimal slicing in the second system increases the overall reward rate by 13% compared to the equal slicing for the first system.

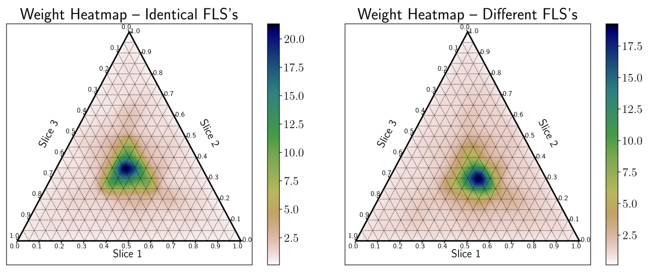


Figure B.3: Simulation results on experiments in Section B.4.2-B. *(Left)* CHOOC's weight selection heatmap (10k cycles) when all three slices apply Log-Rule as flow-level schedulers. *(Right)* Weight selection heatmap when the three slices apply Max-Rate, Log-Rule and Log-Rule as flow-level schedulers respectively.

Bibliography

[1] B. Sadiq, S. J. Baek, and G. De Veciana, "Delay-optimal opportunistic scheduling and approximations: The log rule," *IEEE/ACM Transactions on Networking (TON)*, vol. 19, no. 2, pp. 405–418, 2011.

[2] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári, "X-armed bandits." *Journal of Machine Learning Research*, vol. 12, no. 5, 2011.

[3] A. Garivier and E. Kaufmann, "Optimal best arm identification with fixed confidence," in *Conference on Learning Theory.* PMLR, 2016, pp. 998–1027.

[4] J. Song, G. de Veciana, and S. Shakkottai, "Meta-scheduling for the wireless downlink through learning with bandit feedback," *IEEE/ACM Transactions on Networking*, 2021.

[5] ——, "Online learning for hierarchical scheduling to support network slicing in cellular networks," *Performance Evaluation*, vol. 152, p. 102237, 2021.

[6] ——, "Online learning for multi-agent based resource allocation in weakly coupled wireless systems," in *Proceedings of the Twenty-Third International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, 2022, pp. 111–120.

[7] R. Srikant and L. Ying, *Communication networks: an optimization, control, and stochastic networks perspective.* Cambridge University Press, 2013.

[8] P. Viswanath, D. N. C. Tse, and R. Laroia, "Opportunistic beamforming using dumb antennas," *IEEE transactions on information theory*, vol. 48, no. 6, pp. 1277–1294, 2002.

[9] X. Liu, E. K. P. Chong, and N. B. Shroff, "Opportunistic transmission scheduling with resource-sharing constraints in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 10, pp. 2053–2064, 2001.

[10] A. L. Stolyar, "On the asymptotic optimality of the gradient scheduling algorithm for multiuser throughput allocation," *Operations research*, vol. 53, no. 1, pp. 12–25, 2005.

[11] I.-H. Hou and P. R. Kumar, "Packets with deadlines: A framework for real-time wireless networks," *Synthesis Lectures on Communication Networks*, vol. 6, no. 1, pp. 1–116, 2013.

[12] I.-H. Hou, "Scheduling heterogeneous real-time traffic over fading wireless channels," *IEEE/ACM Transactions on Networking*, vol. 22, no. 5, pp. 1631–1644, 2013.

[13] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, R. Vijayakumar, and P. Whiting, "Scheduling in a queuing system with asynchronously varying

service rates," *Probability in the Engineering and Informational Sciences*, vol. 18, no. 2, pp. 191–217, 2004.

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

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

[16] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, 2016, pp. 50–56.

[17] E. C. 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.

[18] 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 *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[19] H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep reinforcement learning based resource allocation for v2v communications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3163–3173, 2019.

[20] R. Balakrishnan, K. Sankhe, V. S. Somayazulu, R. Vannithamby, and J. Sydir, "Deep reinforcement learning based traffic-and channel-aware ofdma resource allocation," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[21] B. Liu, Q. Xie, and E. Modiano, "Reinforcement learning for optimal control of queueing systems," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2019, pp. 663–670.

[22] D. Shah, Q. Xie, and Z. Xu, "Stable reinforcement learning with unbounded state space," *arXiv preprint arXiv:2006.04353*, 2020.

[23] B. Liu, Q. Xie, and E. Modiano, "Rl-qn: a reinforcement learning framework for optimal control of queueing systems," *arXiv preprint arXiv:2011.07401*, 2020.

[24] L. Huang, X. Liu, and X. Hao, "The power of online learning in stochastic network optimization," in *The 2014 ACM international conference on Measurement and modeling of computer systems*, 2014, pp. 153–165.

[25] T. Chen, A. Mokhtari, X. Wang, A. Ribeiro, and G. B. Giannakis, "Stochastic averaging for constrained optimization with application to online resource allocation," *IEEE Transactions on Signal Processing*, vol. 65, no. 12, pp. 3078–3093, 2017.

[26] S. Bubeck, N. Cesa-Bianchi *et al.*, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Foundations and Trends® in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.

[27] T. Lattimore and C. Szepesvári, *Bandit algorithms.* Cambridge University Press, 2020.

[28] A. Badanidiyuru, R. Kleinberg, and A. Slivkins, "Bandits with knapsacks," in *2013 IEEE 54th Annual Symposium on Foundations of Computer Science.* IEEE, 2013, pp. 207–216.

[29] L. Tran-Thanh, A. Chapman, A. Rogers, and N. R. Jennings, "Knapsack based optimal policies for budget–limited multi–armed bandits," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[30] Y. Xia, H. Li, T. Qin, N. Yu, and T.-Y. Liu, "Thompson sampling for budgeted multi-armed bandits," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[31] S. Agrawal and N. Devanur, "Linear contextual bandits with knapsacks," in *Advances in Neural Information Processing Systems*, 2016, pp. 3450–3458.

[32] S. Cayci, A. Eryilmaz, and R. Srikant, "Learning to control renewal processes with bandit feedback," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, no. 2, p. 43, 2019.

[33] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, no. 47, pp. 235 – 256, 2002.

[34] Y. Gai, B. Krishnamachari, and R. 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).* IEEE, 2010, pp. 1–9.

[35] S. H. A. Ahmad and M. 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).* IEEE, 2009, pp. 1361–1368.

[36] Y.-H. Kao, K. Wright, B. Krishnamachari, and F. Bai, "Online learning for wireless distributed computing," *arXiv preprint arXiv:1611.02830*, 2016.

[37] Y. Sun, X. Guo, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Learning-based task offloading for vehicular cloud computing systems," in *2018 IEEE International Conference on Communications (ICC).* IEEE, 2018, pp. 1–7.

[38] I. Tariq, R. Sen, G. de Veciana, and S. Shakkottai, "Online channel-state clustering and multiuser capacity learning for wireless scheduling," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications.* IEEE, 2019, pp. 136–144.

[39] B. Hajek, "Hitting-time and occupation-time bounds implied by drift analysis with applications," *Advances in Applied probability*, vol. 14, no. 3, pp. 502–525, 1982.

[40] K. Liu and Q. Zhao, "Extended ucb policy for multi-armed bandit with light-tailed reward distributions," *arXiv preprint arXiv:1112.1768*, 2011.

[41] M. Series, "Guidelines for evaluation of radio interface technologies for imt-advanced," *Report ITU*, vol. 638, 2009.

[42] O. Sallent, J. Perez-Romero, R. Ferrus, and R. Agusti, "On radio access network slicing from a radio resource management perspective," *IEEE Wireless Communications*, vol. 24, no. 5, pp. 166–174, 2017.

[43] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5g: Survey and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[44] X. Costa-Pérez, J. Swetina, T. Guo, R. Mahindra, and S. Rangarajan, "Radio access network virtualization for future mobile carrier networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 27–35, 2013.

[45] T. Guo and A. Suárez, "Enabling 5g ran slicing with edf slice scheduling," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2865–2877, 2019.

[46] J. Zheng, P. Caballero, G. De Veciana, S. J. Baek, and A. Banchs, “Statistical multiplexing and traffic shaping games for network slicing,” *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2528–2541, 2018.

[47] V. Sciancalepore, X. Costa-Perez, and A. Banchs, “Rl-nsb: Reinforcement learning-based 5g network slice broker,” *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1543–1557, 2019.

[48] F. Wei, G. Feng, Y. Sun, Y. Wang, S. Qin, and Y.-C. Liang, “Network slice reconfiguration by exploiting deep reinforcement learning with large action space,” *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2197–2211, 2020.

[49] R. Li, C. Wang, Z. Zhao, R. Guo, and H. Zhang, “The lstm-based advantage actor-critic learning for resource management in network slicing with user mobility,” *IEEE Communications Letters*, vol. 24, no. 9, pp. 2005–2009, 2020.

[50] X. Shen, J. Gao, W. Wu, K. Lyu, M. Li, W. Zhuang, X. Li, and J. Rao, “Ai-assisted network-slicing based next-generation wireless networks,” *IEEE Open Journal of Vehicular Technology*, vol. 1, pp. 45–66, 2020.

[51] S. K. Singh, M. M. Salim, J. Cha, Y. Pan, and J. H. Park, “Machine learning-based network sub-slicing framework in a sustainable 5g environment,” *Sustainability*, vol. 12, no. 15, p. 6250, 2020.

[52] R. Kleinberg, A. Slivkins, and E. Upfal, “Multi-armed bandits in metric spaces,” in *Proceedings of the fortieth annual ACM symposium on Theory of computing*, 2008, pp. 681–690.

[53] R. Munos, “Optimistic optimization of a deterministic function without the knowledge of its smoothness,” in *Advances in neural information processing systems*, 2011, pp. 783–791.

[54] M. Valko, A. Carpentier, and R. Munos, “Stochastic simultaneous optimistic optimization,” in *International Conference on Machine Learning*, 2013, pp. 19–27.

[55] J.-B. Grill, M. Valko, and R. Munos, “Black-box optimization of noisy functions with unknown smoothness,” in *Advances in Neural Information Processing Systems*, 2015, pp. 667–675.

[56] R. Sen, K. Kandasamy, and S. Shakkottai, “Noisy blackbox optimization using multi-fidelity queries: A tree search approach,” in *The 22nd international conference on artificial intelligence and statistics*, 2019, pp. 2096–2105.

[57] P. L. Bartlett, V. Gabillon, and M. Valko, “A simple parameter-free and adaptive approach to optimization under a minimal local smoothness assumption,” in *Algorithmic Learning Theory*, 2019, pp. 184–206.

[58] C. Fiegel, V. Gabillon, and M. Valko, “Adaptive multi-fidelity optimization with fast learning rates,” in *International Conference on Artificial*

Intelligence and Statistics, 2020, pp. 3493–3502.

[59] J. Song, G. de Veciana, and S. Shakkottai, “Meta-scheduling for the wireless downlink through learning with bandit feedback,” in *Proc. IEEE WIOPT Workshop on Machine Learning in Wireless Communications (WMLC)*, June 2020, pp. 1–7, invited paper.

[60] W. Ding, T. Qin, X.-D. Zhang, and T.-Y. Liu, “Multi-armed bandit with budget constraint and variable costs,” in *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[61] J. Huang, R. A. Berry, and M. L. Honig, “A game theoretic analysis of distributed power control for spread spectrum ad hoc networks,” in *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.* IEEE, 2005, pp. 685–689.

[62] E. J. Hong, S. Y. Yun, and D.-H. Cho, “Decentralized power control scheme in femtocell networks: A game theoretic approach,” in *2009 IEEE 20th international symposium on personal, indoor and mobile radio communications.* IEEE, 2009, pp. 415–419.

[63] B. Wang, Y. Wu, and K. R. Liu, “Game theory for cognitive radio networks: An overview,” *Computer networks*, vol. 54, no. 14, pp. 2537–2561, 2010.

[64] W. Wang, A. Kwasinski, D. Niyato, and Z. Han, “A survey on applications of model-free strategy learning in cognitive wireless networks,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1717–1757, 2016.

[65] H.-Y. Shi, W.-L. Wang, N.-M. Kwok, and S.-Y. Chen, "Game theory for wireless sensor networks: a survey," *Sensors*, vol. 12, no. 7, pp. 9055–9097, 2012.

[66] W. Z. Guo, J. Y. Chen, G. L. Chen, and H. F. Zheng, "Trust dynamic task allocation algorithm with nash equilibrium for heterogeneous wireless sensor network," *Security and Communication Networks*, vol. 8, no. 10, pp. 1865–1877, 2015.

[67] R. Casado-Vara, F. Prieto-Castrillo, and J. M. Corchado, "A game theory approach for cooperative control to improve data quality and false data detection in wsn," *International Journal of Robust and Nonlinear Control*, vol. 28, no. 16, pp. 5087–5102, 2018.

[68] J. Moura and D. Hutchison, "Game theory for multi-access edge computing: Survey, use cases, and future trends," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 260–288, 2018.

[69] I. Althamary, C.-W. Huang, and P. Lin, "A survey on multi-agent reinforcement learning methods for vehicular networks," in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE, 2019, pp. 1154–1159.

[70] F. Wilhelmi, B. Bellalta, C. Cano, and A. Jonsson, "Implications of decentralized q-learning resource allocation in wireless networks," in *2017 ieee 28th annual international symposium on personal, indoor, and mobile radio communications (pimrc)*. IEEE, 2017, pp. 1–5.

[71] Z. Wang, M. Eisen, and A. Ribeiro, “Decentralized wireless resource allocation with graph neural networks,” in *2020 54th Asilomar Conference on Signals, Systems, and Computers.* IEEE, 2020, pp. 299–303.

[72] P. Landgren, V. Srivastava, and N. E. Leonard, “Distributed cooperative decision-making in multiarmed bandits: Frequentist and bayesian algorithms,” in *2016 IEEE 55th Conference on Decision and Control (CDC).* IEEE, 2016, pp. 167–172.

[73] R. K. Kolla, K. Jagannathan, and A. Gopalan, “Collaborative learning of stochastic bandits over a social network,” *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1782–1795, 2018.

[74] M. Chakraborty, K. Y. P. Chua, S. Das, and B. Juba, “Coordinated versus decentralized exploration in multi-agent multi-armed bandits.” in *IJCAI*, 2017, pp. 164–170.

[75] N. Korda, B. Szorenyi, and S. Li, “Distributed clustering of linear bandits in peer to peer networks,” in *International conference on machine learning.* PMLR, 2016, pp. 1301–1309.

[76] R. Chawla, A. Sankararaman, A. Ganesh, and S. Shakkottai, “The gossiping insert-eliminate algorithm for multi-agent bandits,” in *International Conference on Artificial Intelligence and Statistics.* PMLR, 2020, pp. 3471–3481.

[77] O. Avner and S. Mannor, "Concurrent bandits and cognitive radio networks," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases.* Springer, 2014, pp. 66–81.

[78] J. Rosenski, O. Shamir, and L. Szlak, "Multi-player bandits–a musical chairs approach," in *International Conference on Machine Learning.* PMLR, 2016, pp. 155–163.

[79] L. Besson and E. Kaufmann, "Multi-player bandits revisited," in *Algorithmic Learning Theory.* PMLR, 2018, pp. 56–92.

[80] A. Magesh and V. V. Veeravalli, "Multi-user mabs with user dependent rewards for uncoordinated spectrum access," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers.* IEEE, 2019, pp. 969–972.

[81] A. Mehrabian, E. Boursier, E. Kaufmann, and V. Perchet, "A practical algorithm for multiplayer bandits when arm means vary among players," in *International Conference on Artificial Intelligence and Statistics.* PMLR, 2020, pp. 1211–1221.

[82] I. Bistritz and A. Leshem, "Game of thrones: Fully distributed learning for multiplayer bandits," *Mathematics of Operations Research*, vol. 46, no. 1, pp. 159–178, 2021.

[83] A. Magesh and V. V. Veeravalli, "Decentralized heterogeneous multi-player multi-armed bandits with non-zero rewards on collisions," *IEEE Transactions on Information Theory*, 2021.

[84] G. Lorden, "On excess over the boundary," *The Annals of Mathematical Statistics*, pp. 520–527, 1970.

Vita

Jianhan Song received his B.S. degree in Information Engineering from The Chinese University of Hong Kong in 2017. He is currently working towards his Ph.D. degree in the Department of Electrical and Computer Engineering at The University of Texas at Austin, Austin, TX, USA. He also joined the Wireless Networking and Communications Group (WNCG) at the University of Texas at Austin, as a Graduate Research Assistant. His research interests include queuing systems, wireless networking, and multi-armed bandit problems.

Permanent address: huntersong572@gmail.com

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.